Wasserstein Adversarial Mixture Clustering

Warith Harchaoui, Andrés Almansa, Pierre-Alexandre Mattei, Charles Bouveyron

Wasserstein Adversarial Mixture Clustering

Warith Harchaoui
Research and Development
Oscaro.com
Paris, France
warith.harchaoui@oscaro.com

Pierre-Alexandre Mattei
Department of Computer Science
IT University of Copenhagen
Copenhagen, Denmark
pima@itu.dk

Andrés Alamansa
MAP5, UMR CNRS 8145
Universités Paris Descartes
Paris France
andres.almansa@parisdescartes.fr

Charles Bouveyron
Laboratoire J.A. Dieudonné
Université de Nice - Sophia Antipolis
Nice, France
charles.bouveyron@math.cnrs.fr

Abstract

Clustering complex data is a key element of unsupervised learning which is still a challenging problem. In this work, we introduce a deep approach for unsupervised clustering based on a latent mixture living in a low-dimensional space. We achieve this clustering task through adversarial optimization of the Wasserstein distance between the real and generated data distributions. The proposed approach also allows both dimensionality reduction and model selection. We achieve competitive results on difficult datasets made of images, sparse and dense data.

1 Introduction

Clustering [14] is the task of making groups without the need of any manual annotations. Dimensionality reduction give clues to the underlying structure of the data. Along with dimensionality reduction, clustering is a desirable goal in data analysis, visualization and is often a preliminary step in many algorithms for example in computer vision [34] and natural language processing [18].

In this paper, the clustering is achieved from a new space through a combination of a Wasserstein Generative Adversarial Network [3, 19, 20] and a Gaussian mixture model [16] optimized in a stochastic gradient fashion. We propose an algorithm to perform unsupervised classification (a.k.a. clustering) within this framework that we call "WAMiC" for Wasserstein Adversarial Mixture Clustering. We postulate that a generative approach, namely a tuned Gaussian mixture model, can capture an explicit latent model that is the cause of the observed data, in the original space through a latent embedding.

2 Related Work

The purpose of this research is to build a linear-complexity algorithm that uses a non-linear embedding into a code space. Indeed, in the clustering literature, one can distinguish two kinds of clustering algorithms with respect to their speed and memory complexity in the number of examples. On one side, we have linear algorithms such as k-means (KM) and

Preprint. Work in progress.

Gaussian Mixture Models (GMM), which usually work directly on the data (*i.e.* without any medium such as embeddings). On the other side, we also have quadratic and cubic algorithms such as Hierarchical Clustering [14] and Spectral Clustering [31, 48, 44] that use pairwise similarities to emphasize the latent clustering structure lying on the data. The proposed approach belongs to the first category although similar approaches to ours such as [47] fall in the second category. Our proposed WAMiC algorithm is meant to work in a scalable fashion with any cluster shapes thanks to a latent space equipped with a tunable mixture distribution.

2.1 Clustering-aware representation learning

The importance of finding a suitable representation for clustering was first highlighted by Chang [8], who showed that embeddings based on principal component analysis were often unfit for clustering purposes.

The problem of learning representations from data in an unsupervised manner is a long-standing problem in machine learning [5, 27]. Principal Components Analysis (PCA) and auto-encoders (AE) which can be seen as non-linear extension of PCA [4] have been used for representing faces [41] or to produce a hierarchy of features [7]. Other techniques have been used such as sparse coding [30] where the representation of one image is a linear combination of a few elements in a dictionary of features. More recently Bojanowski and Joulin [6] learned features unsupervisedly by a procedure that consists in mapping a large collection of images to noise vectors through a deep convolutional neural networks. Their work has a clustering objective but they do not report clustering results as their real goal is unsupervised feature learning.

When it comes to compressing data while limiting loss of reconstruction information, auto-encoders have proved efficient [43]. Briefly, an auto-encoder is a neural network made of two parts: (i) the *encoder* maps the data in a low-dimension space, (ii) the *decoder* maps them back to the original space. An auto-encoder is usually trained to reconstruct the data in the original space in a least squares fashion. At the end, if the reconstruction error is low, one can consider that the code resulting from the encoder has compressed the data without loosing too much information. The assumption is that the input data space of high dimensionality contains structure that could be successfully embedded in a lower-dimensionality manifold [1, 37].

In our work, we try to accomplish representation learning for clustering. The first work we saw doing clustering in the code space of an auto-encoder is the one of Song et al. [36] and more recently, but independently, Yang et al. [46]. More precisely, they considered a KM-regularized auto-encoder loss to get a code space that is more easily clustered with KM namely their loss is the sum of the reconstruction and the KM residual. This philosophy is the one adopted for our own approach but with GMM. In our experiments, we found that optimizing the KM objective (online) when doing joint clustering and feature learning did not work well. We believe this is because it creates high magnitude gradients for points that are far away from cluster centers and there are sharp discontinuities at cluster boundaries whereas GMM diminishes that effect thanks to low density/probability values for far points. In a similar spirit, Xie et al. [45] embrace the t-SNE framework [29] in a clustering context through an auto-encoder in a non-model-based fashion. All these works tend to show that simultaneous representation learning and clustering do help each other. The reader will find an excellent review in the work of Aljalbout et al. [2].

In our preliminary experiments, a Gaussian Mixture Model trained with Expectation-Maximization [10] on codes coming from a vanilla MLP[1] auto-encoder without convolutions is able to reach approximately 80% of unsupervised clustering accuracy on the famous digits MNIST images dataset[2] directly on raw pixels. The idea of our work is based on the fact that clustering and compression (dimensionality reduction) seem to be closely related tasks that auto-encoders accomplish well. Our approach is an attempt to take advantage of that empirical fact.

[1] Multi-Layered Perceptron

[2] http://yann.lecun.com/exdb/mnist/

2.2 Deep Generative Models

In the recent literature, we observe three kinds of inference techniques for Deep Generative Models (DGM) based on likelihood, variational auto-encoders and GANs. The goal of the likelihood-based approach is to minimize the Kullback-Leibler divergence between the original data distribution and the generated data distribution. A recent example of that kind is the work of Dinh et al. [12]. On the other hand, Kingma and Welling [25] allow to directly specify a prior distribution over the code space of a variational auto-encoder (VAE). Inference is done using stochastic gradient variational Bayes (SGVB), a method based on a reparametrization of the variational lower bound. Deep generative models for clustering may be built using a mixture model as prior distribution. This approach was recently explored by [11] and [22] who used a GMM prior. Finally, GAN approaches minimize the Jensen-Shannon divergence in the original paper of GAN [19] and more recently the Wasserstein distance [20] through the Wasserstein GAN (WGAN). Inspired by recent advances in Optimal Transport theory [42], Cuturi [9] has developped an efficient and differentiable way to compute a regularized version of the Wasserstein distance between two empirical distributions [17] called the Sinkhorn algorithm. For this current work, we have chosen the WGAN approach because of the remarkable quality of the data generation.

3 Wasserstein Adversarial Mixture Clustering

Clustering is the machine learning task of inferring a function to describe hidden structure from unlabeled data. We aim at clustering a dataset of N points $\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N$ samples of the random variable $\mathbf{x}$ living in a space $\mathcal{X}$ (say a D-dimensional space or even more complex). At the heart of our model, there is an auto-encoder made of: (i) an encoder network $\mathcal{E}$ (parametrized by $\theta_{\mathcal{E}}$) and (ii) a decoder network $\mathcal{D}$ (parametrized by $\theta_{\mathcal{D}}$). That auto-encoder plays the role of a bridge between the data space and a latent space more akin to clustering.

Concretely, clustering is the task of gathering the data in K homogeneous groups. Most of the time even the number K of groups can be found through model selection. This model selection step is discussed at the end of this section and briefly illustrated on synthetic data in section 3.4.

3.1 Model

We build a generative model based on two assumptions. First, we believe the data are living on a low-dimensional manifold which is a common assumption in the Machine Learning literature. Thus, there exists a latent code space $\mathcal{Z}$ of a low dimension d (say $d = 10$) such that there is a mapping $\mathcal{D}$ from $\mathcal{Z}$ to $\mathcal{X}$ connecting the random variable $\mathbf{x}$ in $\mathcal{X}$ to its latent counterpart $\mathbf{z}$ in $\mathcal{Z}$. Second, we also assume that $\mathbf{z}$ follows a mixture distribution.

In details, our model consists in saying that the data have been generated as follows. First the clustering variable c

$$c \sim \text{Cat}(\boldsymbol{\pi}) \tag{1}$$

corresponds to a categorical (multinomial) random variable for clustering with proportions defined in vector $\boldsymbol{\pi}$. In this generative process, once the cluster k is chosen, one can generate a code in $\mathcal{Z}$ which implies a mixture marginal for $\mathbf{z}$:

$$\mathbf{z}|c = k \sim g_k \qquad\qquad \mathbf{z} \sim \sum_{k=1}^{K} \boldsymbol{\pi}_k g_k$$

with proportions $\boldsymbol{\pi}$ and components $(g_k)_{k=1}^K$. Ultimately a point in $\mathcal{X}$ is generated:

$$\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathcal{D}(\mathbf{z}), \sigma^2 \mathbf{I}_p) \tag{2}$$

To translate the assumption that the code data lie extremely close to a low-dimensional manifold, we further assume that $\sigma \to 0$. In this context, the posterior probability needed to cluster $\mathbf{x}$ is given by

$$p(c = k|\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})}[p(c = k|\mathbf{z})] \tag{3}$$

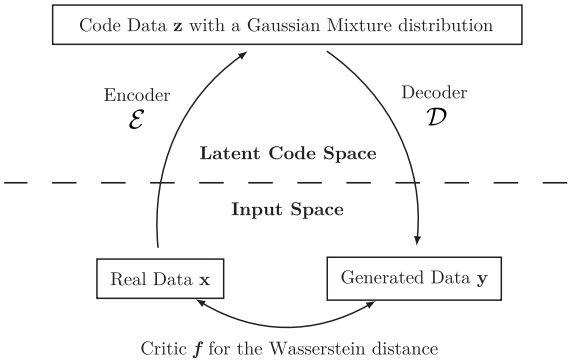


Figure 1: WAMiC Optimization Scheme

Although any parametric density functions can be used for each mixture component g_k, we restrict ourselves in this work to Gaussian mixtures and in the following, we impose $g_k = \mathcal{N}(\mathbf{m}_k, \mathbf{\Sigma}_k)$ (a Gaussian distribution with mean $\mathbf{m}_k$ and covariance matrix $\mathbf{\Sigma}_k$) where $(\pi_k)_{k=1,\dots,K}$, $(\mathbf{m}_k)_{k=1,\dots,K}$, and $(\mathbf{\Sigma}_k)_{k=1,\dots,K}$ are the mixture parameters, stored in $\theta_{\mathcal{M}}$.

3.2 Inference

In order to fit our generative model, we build a random variable $\mathbf{y}$ to match $\mathbf{x}$ in terms of Wasserstein distance in an adversarial fashion. We only have access to samples of $\mathbf{x}$ (through the dataset) and $\mathbf{y}$ (through a random generator and $\mathcal{D}$), thus we use a WGAN to fit our model. Fig. 1 summarizes the modeling proposed here.

The posterior probability $p(\mathbf{z}|\mathbf{x})$ is intractable in Eq. (3) because of its decoder part. But, as in the work of Kingma and Welling [25], we can approximate it using a variational inference network $q(\mathbf{z}|\mathbf{x})$ built according to the encoder $\mathcal{E}$. As emphasized by Kingma and Welling [25], minimizing the Kullback-Leibler divergence between the true posterior and $q(\mathbf{z}|\mathbf{x})$ leads to minimizing a penalized quadratic auto-encoder loss. Since $\sigma \to 0$, the dominating term in this loss will precisely be the loss of a vanilla auto-encoder which is what we do in practice for the sake of simplicity. Eventually, we can compute an approximation of $p(c = k|\mathbf{x})$ by simply replacing the true posterior by the approximation, which leads to using the maximum-a-posteriori (MAP) rule in code space:

$$p(c = k|\mathbf{x}) \approx \frac{\pi_k \mathcal{N}(\mathbf{m}_k, \mathbf{\Sigma}_k)(\mathcal{E}(\mathbf{x}))}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(\mathbf{m}_{k'}, \mathbf{\Sigma}_{k'})(\mathcal{E}(\mathbf{x}))}. \tag{4}$$

Given a real data distribution $p_{\mathbf{x}}$ (generating variable $\mathbf{x}$) and a generated data distibution $p_{\mathbf{y}}$ (generating variable $\mathbf{y}$), a WGAN [3, 20] minimizes the Kantorovich-Rubinstein duality of the Wasserstein distance between these two distributions:

$$\max_{\|\nabla f\| \leq 1} \mathbb{E}_{\mathbf{x}}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{y}}\left[f(\mathbf{y})\right] \tag{5}$$

with f called the critic and implemented in practice by a neural network of parameters θ_f and constrained by an augmented Lagrangian (see [20] for details):

$$\max_{\theta_f} \mathbb{E}_{\mathbf{x}}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{y}}\left[f(\mathbf{y})\right] - \lambda \mathbb{E}_{\tilde{\mathbf{x}}}\left[\left(\|\nabla f(\tilde{\mathbf{x}})\| - 1\right)^2\right] \tag{6}$$

where $\tilde{\mathbf{x}}$ is sampled from segments between $\mathbf{x}$ and $\mathbf{y}$.

In the work of Gulrajani et al. [20], a WGAN usually uses a simple fixed distribution (Gaussian or uniform) for the random noise generator that is transformed by a neural network called the generator to fit the data distribution in the Wasserstein sense. We use a tunable mixture distribution instead for clustering purposes.

Inspired by the work of Kingma and Welling [25], we need to specify for each cluster k a differentiable reparametrization. The codes $\mathbf{z}_k$ coming from each Gaussian with full covariance matrices $\mathbf{\Sigma}_k$ follow:

$$\mathbf{z}_k = \mathbf{S}_k \mathbf{e} + \mathbf{m}_k \tag{7}$$

and

$$\mathbf{y}_k = \mathcal{D}(\mathbf{z}_k) \tag{8}$$

are our generated data for each cluster k where: $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{S}_k$ is a Cholesky decomposition (with only non-zeros in the lower-triangular part) of the full covariance $\mathbf{\Sigma}_k$ (with positive diagonal entries in $\mathbf{S}_k$ parametrized with exponentials). The Kantorovich-Rubinstein duality Eq. (5) becomes:

$$\max_{\|\nabla f\| \leq 1} \left(\mathbb{E}_{\mathbf{x}} \left[f(\mathbf{x}) \right] - \sum_{k=1}^{K} \pi_k \mathbb{E}_{\mathbf{y_k}} \left[f(\mathbf{y_k}) \right] \right) \tag{9}$$

and:

$$\max_{\theta_f} \mathbb{E}_{\mathbf{x}} \left[f(\mathbf{x}) \right] - \sum_{k=1}^{K} \pi_k \mathbb{E}_{\mathbf{y_k}} \left[f(\mathbf{y_k}) \right] - \lambda \mathbb{E}_{\tilde{\mathbf{x}}} \left[\left(\|\nabla f(\tilde{\mathbf{x}})\| - 1 \right)^2 \right] \tag{10}$$

is the regularized version of Eq. (9) for unconstrained optimization purposes.

Following the work of Gulrajani et al. [20] for WGAN, we are trying to make our generated data "indistinguishable" from the real data in the Wasserstein distance sense. In our case, the source of generated noise is coming from our mixture $\mathcal{M}$ through the generator/decoder $\mathcal{D}$. Thus, we optimize:

$$\min_{\theta_{\mathcal{D}}, \theta_{\mathcal{M}}} \mathcal{L}(\theta_{\mathcal{D}}, \theta_{\mathcal{M}}) \tag{11}$$

where:

$$\mathcal{L}(\theta_{\mathcal{D}}, \theta_{\mathcal{M}}) = \max_{\theta_f} \left(\mathbb{E}_{\mathbf{x}} \left[f(\mathbf{x}) \right] - \sum_{k=1}^{K} \pi_k \mathbb{E}_{\mathbf{y_k}} \left[f(\mathbf{y_k}) \right] - \lambda \mathbb{E}_{\tilde{\mathbf{x}}} \left[\left(\|\nabla f(\tilde{\mathbf{x}})\| - 1 \right)^2 \right] \right) \tag{12}$$

3.3 Algorithm

Our WAMiC algorithm can be decomposed in four different steps:

1. **Auto-Encoder Initialization**: We train a classic auto-encoder $(\mathcal{E}, \mathcal{D})$ (for an encoder $\mathcal{E}$ and decoder $\mathcal{D}$):

$$(\theta^0_{\mathcal{E}}, \theta^0_{\mathcal{D}}) = \arg\min_{\theta_{\mathcal{E}}, \theta_{\mathcal{D}}} \mathbb{E}_{\mathbf{x} \sim \text{Data}} \left[\|\mathcal{D} \circ \mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2^2 \right] \tag{13}$$

2. **EM-based Gaussian Mixture Initialization**: We fit a Gaussian Mixture Model $\mathcal{M}$ with the Expectation-Maximization algorithm [10] on the encoded data:

$$\theta^0_{\mathcal{M}} = \arg\max_{\theta_{\mathcal{M}}} \mathbb{E}_{\mathbf{x} \sim \text{Data}} \left[\log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{m}_k, \Sigma_k) \big(\mathcal{E}^0(\mathbf{x}) \big) \right) \right] \tag{14}$$

3. **Clustered Data Generation**: We optimize the mixture $\mathcal{M}$ and the decoder $\mathcal{D}$ (that now plays the role of a generator) to minimize the regularized Wasserstein distance presented above with the initialization coming from the two previous steps;

$$(\hat{\theta}_{\mathcal{D}}, \hat{\theta}_{\mathcal{M}}) = \arg\min_{\theta_{\mathcal{D}}, \theta_{\mathcal{M}}} \mathcal{L} \tag{15}$$

   from Eq. (12)

4. **Clustering Decoding**: We train a new encoder $\mathcal{E}$ with an auto-encoder quadratic loss with the decoder $\mathcal{D}$ just found maintained fixed.

$$\hat{\theta}_{\mathcal{E}} = \arg\min_{\theta_{\mathcal{E}}} \mathbb{E}_{\mathbf{x} \sim \text{Data}} \left[\|\hat{\mathcal{D}} \circ \mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2^2 \right] \tag{16}$$

Steps 1 and 2 are just intuitive initialization of step 3. Step 4 can be seen as an *ad hoc* way of inverting the decoder $\mathcal{D}$. Several other strategies have been proposed for that purpose (e.g. [15, 40]). We use the MAP rule in the code space in order to finally cluster the data points which makes our simple approach particularly fit for clustering.

3.4 Model Selection

Step 3 of section 3.3 minimizes an estimated Wasserstein distance between the real and generated data distribution. That step's role is to generate clustered data close to the real data in terms of distributions for the Wasserstein criteria. Once trained, we can measure the Wasserstein distance between generated data and some held-out validation data to check under/over-fitting and ultimately choose the number of classes.

Traditionally, model selection is accomplished according to the likelihood (or completed likelihood) itself related to the Kullback-Leibler divergence (see [35] for example). One of the originality of our approach is that our model selection technique is done according to an other divergence which is the Wasserstein distance. This parallel allows us to use the historical likelihood-based literature by replacing the well-known Kullback-Leibler divergence by the Wasserstein distance.

More concretely, to choose the number of clusters, we first train step 1 of section 3.3 once for all, then for each number of clusters, we do steps 2 and 3 in parallel. At the end, one can take the best Wasserstein distance and do step 4 to get the best clustering according to these models. For our numerical optimizations, we estimate the Wasserstein distance on batches only. So, the Wasserstein score that we use for model selection is simply the mean of our batch Wasserstein estimations once our f function is finally trained.

4 Numerical Experiments

4.1 Implementation details

We did our experiments in Python by using the PyTorch [32] and Scikit-learn [33] libraries. with the same learning rate of 10^{-5} with the Adam optimization strategy [24] everywhere.

Given a clustering result and the ground truth, one can measure the quality of the clustering result according to what we call the "accuracy" computed with the Hungarian Method [26, 38] on the confusion matrix.

4.2 Synthetic Experiment

To illustrate our approach, we first work on a toy dataset that we call "Three Moons" with 1000 points for each of the 3 classes in 2 dimensions as presented in Fig. 2(a). With a code space of dimension 1, even though the clusters are not linearly separable in the original data space, our system is able to cluster them with 100% of accuracy.

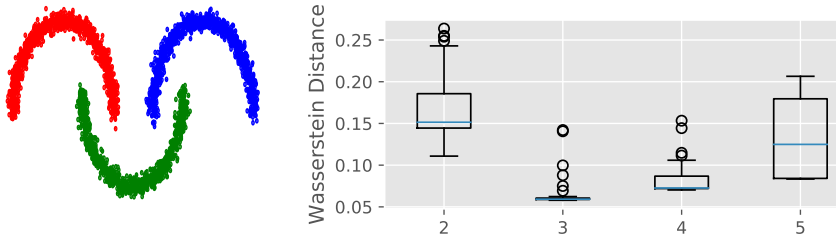
For that simple toy dataset, most of the clustering work is done by the vanilla auto-encoder. Indeed, once the auto-encoder is trained, we observed that the 1D codes are already separated according to the cluster labels. Here, we just wanted to see if model selection was plausible in a simple scenario.

For the number of clusters, without labels in our unsupervised context, while a classification-score-based cross-validation is not an option, one can still measure an estimate of the Wasserstein distance (our loss) but on a validation set. The intuition behind is that if we did not overfit, our loss function will still be satisfactory on that validation set (that was not seen during training). After running our four steps algorithm, we can train a new neural network f to measure the Wasserstein distance between a held-out validation dataset and some generated data empirical distributions. More precisely, we find the results presented in Fig. 2(b) actually selecting 3 clusters which is satisfactory.

4.3 Real-world Experiments

For the real-world experiments, we were interested in 4 datasets that are different in nature (images, sparse and dense data):

- MNIST: 70 000 handwritten digits images dataset living in dimension 784 (for 28×28 pixels);



(a) Three Moons Dataset (b) Model selection

Figure 2: Wasserstein model selection on the three moons dataset for the number of clusters on a validation dataset, (on 30 runs for each number of clusters)

- Reuters: English news stories labeled with a category tree [28]. Following DEC [45], we used 4 root categories: corporate/industrial, government/social, markets, and economics as labels and discarded all documents with multiple labels. We computed tf-idf features on the 2000 most frequent words to represent all articles;
- Reuters-10k: a random subset of Reuters with only 10 000 examples (selected with the same random seed as DEC);
- HHAR: The Heterogeneity Human Activity Recognition (HHAR) dataset [39] contains 10299 sensor records from smart phones and smart watches. All samples are partitioned into 6 categories of human activities and each sample is of 561 dimensions.

Throughout the experiments, we used the same architecture from Xie et al. [45] D-500-500-2000-d (D is the dimensionality of the input space *e.g.* 784 for MNIST and $d = 10$ is the dimensionality of the code space) for the encoder for fair comparisons with VaDE [22].

The most difficult and longest part of the optimization is step 3 of section 3.3. Indeed, this is a min-max optimization that is inherent to WGANs. As we can see in Fig. (3), at the beginning, the Wasserstein distance is not correctly estimated and is even negative: one has to wait for a good critic in the first 50k iterations before seeing the loss actually decreasing meaningfully.

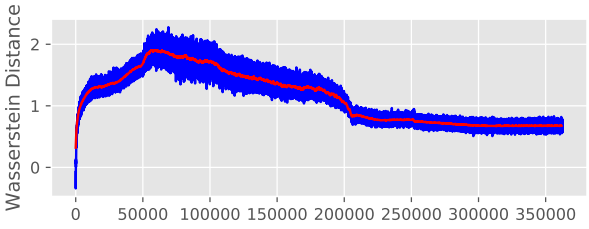


Figure 3: Mini-Batch Wasserstein Distance Estimated on Reuters-10k through Iterations (with rolling mean in red)

On MNIST, in Fig. (4), we generated data further and further in random directions from the centroids: we see that the digits get *fancier* away from the centroids. The good quality of the generation is comparable to those of regular GANs but in a cluster-wise fashion.

4.3.1 Results

The results of our approach compare favorably to the deep clustering state-of-the-art in Table. (1). First, we observe that there is an improvement over standard baseline algorithms (GMM and KM) when fed with the output of an AE which is coherent with the results of Xie et al. [45]. Furthermore, there is a supplementary significant adversarial improvement for WAMiC . These good results are comparable to those of supervised non-convolutional

Datasets	MNIST	Reuters	Reuters-10k	HHAR
WAMiC (ours)	**98.42**	79.24	**79.87**	81.42
VaDE [22]	94.06	**79.38**	79.83	**84.46**
DEC [45]	84.30	75.63	72.17	79.82
AE + GMM (full covariance)	82.56	70.98	70.12	78.48
IMSAT ([21], different architecture)	98.40	–	71.00	–
GAR ([23], convolutional net)	98.32	–	–	–
DEPICT ([13], convolutional net)	96.50	–	–	–
GMM (diagonal covariance)	53.73	55.81	54.72	60.34
KM	53.47	53.29	54.04	59.98

Table 1: Experimental accuracy results (%, the higher, the better) based on the Hungarian method. The top four lines correspond to the same architecture.

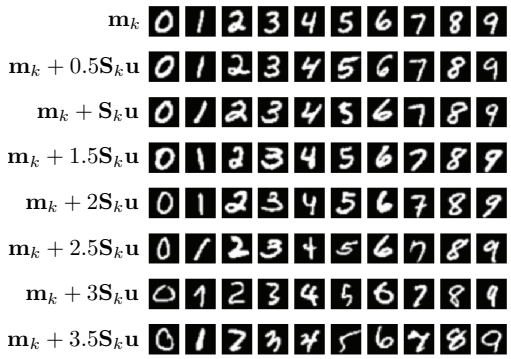


Figure 4: Generated digits images. From left to right, we have the ten classes found by WAMiC and ordered thanks to the Hungarian algorithm. From top to bottom, we go further and further in random directions from the centroids (the first row being the decoded centroids). More specifically, $\mathbf{u}$ is sampled from the uniform random density on the unit hypersphere in the code space.

networks with just a few layers of the 1990s[3]. On MNIST, our unsupervised neural network is almost competitive with its supervised counterparts which is surprising and tells us this dataset is not anymore a difficult one even in the non-convolutional unsupervised context.

WAMiC compares favorably to the other frameworks that share the same network architecture (VaDE and DEC). On MNIST, the important performance gap between VaDE and GMAC can be explained by the fact that variational auto-encoders have difficulties with data that lie extremely close to low-dimensional manifolds, like images (see e.g. [3]). In that regard, our algorithm inherits the strenghts of WGAN and models image data much more faithfully.

5 Conclusion

We present WAMiC a mixture-based Deep Generative Model for Clustering by combining WGAN and variational inference. Empirically, we observe some symbiosis operating between clustering and non-linear embedding. For future work, we want to explore a great advantage of our generative approach: Wasserstein model selection on real data for selecting the number of clusters.

[3]see `http://yann.lecun.com/exdb/mnist` for a complete MNIST benchmark

Acknowledgments

The authors would like to thank Vincent Delaitre from Deepomatic.com, Olivier Grisel from INRIA, Georges Oppenheim from Université de Paris Sud and Université Paris Est and Pierre Roussillon from Université Paris Saclay for fruitful discussions and valuable inputs. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. Finally, we thank Oscaro.com for the funding and support.

References

[1] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

[2] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/arjovsky17a.html`.

[4] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2013.50.

[6] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310*, 2017.

[7] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032, 2015.

[8] Wei-Chien Chang. On using principal components before separating a mixture of two multivariate normal distributions. *Applied Statistics*, pages 267–275, 1983.

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2292–2300. 2013.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[11] Nat Dilokthanakul, Pedro A.M. Mediano, Marta Garnelo, Matthew C.H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *International Conference on Learning Representations*, 2017.

[13] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5747–5756. IEEE, 2017.

[14] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[15] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *International Conference on Learning Representations*, 2017.

[16] Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[17] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning Generative Models with Sinkhorn Divergences. (2017-83), October 2017. URL `https://ideas.repec.org/p/crs/wpaper/2017-83.html`.

[18] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[21] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1558–1567. PMLR, 06–11 Aug 2017.

[22] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: A generative approach to clustering. 2016. URL `http://arxiv.org/abs/1611.05148`.

[23] Ozsel Kilinc and Ismail Uysal. Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkMvEOlAb`.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, number 2014, 2013.

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

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[28] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr): 361–397, 2004.

[29] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[30] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69, 2008.

[31] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. 14(2):849–856, 2001.

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Jean Ponce and David Forsyth. *Computer vision: a modern approach*. 2011.

[35] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6 (2):461–464, 1978.

[36] Chunfeng Song, Yongzhen Huang, Feng Liu, Zhenyu Wang, and Liang Wang. Deep auto-encoder based clustering. *Intelligent Data Analysis*, 18(6S):S65–S76, 2014.

[37] Sho Sonoda and Noboru Murata. Decoding stacked denoising autoencoders. *arXiv preprint arXiv:1605.02832*, 2016.

[38] Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.

[39] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. pages 127–140, 2015.

[40] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkL7n1-0b`.

[41] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[42] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[43] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11 (Dec):3371–3408, 2010.

[44] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

[45] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. *arXiv preprint arXiv:1511.06335*, 2015.

[46] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint arXiv:1610.04794*, 2016.

[47] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. 2016. URL `http://arxiv.org/abs/1604.03628`.

[48] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *Advances in neural information processing systems*, 17(1601-1608):16, 2004.