Wasserstein Adversarial Mixture for Deep Generative Modeling and Clustering

Warith Harchaoui **Pierre-Alexandre Mattei**

Andrés Almansa **Charles Bouveyron**

Abstract

Unsupervised learning, and in particular clustering, is probably the most central problem in learning theory nowadays. This work focuses on the clustering of complex data by introducing a deep generative approach for both modeling and clustering. The proposed approach, called hereafter Wasserstein Adversarial Mixture Clustering (WAMiC), is based on an auto-encoder and a latent mixture model living in a low-dimensional space. We achieve this clustering task through an adversarial optimization of the Wasserstein distance between the real and generated data distributions. The proposed approach allows in turn both dimensionality reduction and model selection. We achieve competitive results that are on par with recent deep clustering approaches — and substantially outperform non-neural-networks-based approaches — on diverse datasets like images, sparse and dense data.

1 Introduction

Clustering (Duda et al., 2012) 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 (Ponce and Forsyth, 2011) and natural language processing (Goldberg, 2017).

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

In this paper, the clustering is achieved from a new space through a combination of a Wasserstein Generative Adversarial Network (Goodfellow et al., 2014; Arjovsky et al., 2017; Gulrajani et al., 2017) and a mixture model (Fraley and Raftery, 2002) optimized in a stochastic gradient fashion. We propose an algorithm to perform clustering within this framework that we call "WAMiC" for Wasserstein Adversarial Mixture Clustering. We postulate that a generative approach, namely a tuned 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 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 (Duda et al., 2012) and Spectral Clustering (Ng et al., 2001; Zelnik-Manor and Perona, 2004; Von Luxburg, 2007) 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 (Yang et al., 2016b) 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 (1983), 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 (Bengio et al., 2013; LeCun et al., 2015). Principal Components Analysis (PCA) and auto-encoders (AE) which can be seen as non-linear extension of PCA (Baldi and Hornik, 1989) have been used for representing faces (Turk and Pentland, 1991) or to produce a hierarchy of features (Chan et al., 2015). Other techniques have been used such as sparse coding (Mairal et al., 2008) where the representation of one image is a linear combination of a few elements in a dictionary of features. More recently Bojanowski and Joulin (2017) 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 there is no clustering result 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 (Vincent et al., 2010). 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 (Alain and Bengio, 2014; Sonoda and Murata, 2016).

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. (2014) and more recently, but independently, Yang et al. (2016a). 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 a mixture model. 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 the mixture model diminishes that effect thanks to low density/probability values for far points. In a similar spirit, Xie et al. (2015) embrace the t-SNE framework (Maaten and Hinton, 2008) 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. (2018).

In our preliminary experiments, a (Gaussian) Mixture Model trained with Expectation-Maximization (Dempster et al., 1977) on codes coming from a vanilla Multi-Layered Perceptron (MLP) auto-encoder without convolutions is able to reach approximately 80% of unsupervised clustering accuracy on the famous digits MNIST images dataset 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.

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. (2017). On the other hand, Kingma and Welling (2013) 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 (Dilokthanakul et al., 2016) and (Jiang et al., 2016) who used a GMM prior. Finally, GAN approaches minimize the Jensen-Shannon divergence in the original paper of GAN (Goodfellow et al., 2014) and more recently the Wasserstein distance (Gulrajani et al., 2017) through the Wasserstein GAN (WGAN). Inspired by recent advances in Optimal Transport theory (Villani, 2008), Cuturi (2013) has developped an efficient and differentiable way to compute a regularized version of the Wasserstein distance between two empirical distributions (Genevay et al., 2017) 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

Unsupervised Learning consists in describing hidden structure from unlabeled data. Clustering is one way to do that with groups.

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 (see e.g. Vincent et al. 2010). 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

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

corresponds to a categorical (multinomial) random variable for clustering with prior 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 $\mathbf{z}$:

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

with proportions $\boldsymbol{\pi}$, probability distribution functions of the components $(g_k)_{k=1}^K$ and their parameters ξ_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

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

Although any parametric density functions can be used for each mixture component g_k, we restrict ourselves in this work to densities allowing the use of the reparameterization trick (Kingma and Welling, 2013) which has a central role in adversarial optimization. In the following, we will illustrate our methodology using a mixture of Gaussians, *i.e.* we suppose $g_k(.;\xi_k) = \mathcal{N}(.;\mathbf{m}_k, \boldsymbol{\Sigma}_k)$ (a Gaussian distribution with mean $\mathbf{m}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$) where $(\pi_k)_{k=1,...,K}$, $(\mathbf{m}_k)_{k=1,...,K}$, and $(\boldsymbol{\Sigma}_k)_{k=1,...,K}$ are the mixture parameters, stored in $\theta_{\mathcal{M}}$. Note that, beyond the GMM prior that we consider here, our approach could be extended to any mixture of reparametrizable distributions: one might for example consider a mixture of von Mises as the prior distribution, in order to obtain interesting visualizations on a hyper-sphere, such as the ones of Davidson et al. (2018).

3.2 Inference

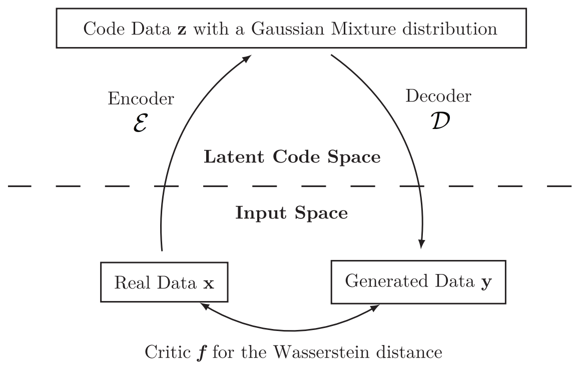

Figure 1: WAMiC Optimization Scheme.

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. Figure 1 summarizes the modeling proposed here.

The posterior probability $p(\mathbf{z}|\mathbf{x})$ in Eq. (3) is hard to compute because of the nonlinearity of the decoder. But, as in the work of Kingma and Welling (2013), we can approximate it using an inference network $q(\mathbf{z}|\mathbf{x})$ built according to the encoder $\mathcal{E}$. As emphasized by Kingma and Welling (2013), 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 the maximum-a-posteriori (MAP) rule in code space:

$$P(c = k|\mathbf{x}) \approx \frac{\pi_k g_k(\mathcal{E}(\mathbf{x}); \xi_k)}{\sum_{k'=1}^{K} \pi_{k'} g_{k'}(\mathcal{E}(\mathbf{x}); \xi_k)}. \tag{4}$$

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 (Arjovsky et al., 2017; Gulrajani et al., 2017) 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 (Gulrajani et al., 2017) 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. (2017), 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 (2013), we need to specify for each cluster k a differentiable reparametrization. Note that, beyond the Gaussian mixture which is used hereafter to illustrate the proposed methodology, this trick can be extended to any mixture of reparametrisable distributions – for a recent overview of reparametrizable distributions, see Figurnov et al. (2018).

Assuming that the codes $\mathbf{z}_k$ come from Gaussians with full covariance matrices $\mathbf{\Sigma}_k$, $\mathbf{z}_k$ follows:

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

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:

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

is the regularized version of Eq. (9) for unconstrained optimization purposes (Gulrajani et al., 2017).

Following the work of Gulrajani et al. (2017) 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:

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

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

Step 2: EM-based Mixture Initialization We fit a Gaussian Mixture Model $\mathcal{M}$ with the Expectation-Maximization algorithm (Dempster et al., 1977) on the encoded data:

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

Step 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).

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

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

Steps 1 and 2 are just intuitive initializations 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. (Dumoulin et al., 2017; Tolstikhin et al., 2018)). At the end, we use the Maximum a Posteriori rule (MAP) in the code space in order to compute the posterior probabilities and to finally cluster the data points which makes our simple approach particularly fit for clustering.

3.4 Model Selection

A key element of supervised problems is the choice of the unavoidable hyper-parameters of the modeling. The advantage of generative modeling is that the statistical litterature is particularly rich in solution for this issue, ranging from cross-validated likelihood to Bayesian approaches. Here, we are in particular interested in choosing the number of clusters. We could also select the bottleneck dimensionality d of the latent space in the same way.

To address the problem of model selection within WAMiC, we propose to rely on the Wasserstein distance which is minimized in step 3 of Section 3.3 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 criterion. Once trained, we estimate the Wasserstein distance between generated data and some held-out validation data to check under/over-fitting and pick the number of classes minimizing that criterion. Traditionally, model selection is accomplished according to the likelihood (or completed likelihood) criterion based on the Kullback-Leibler divergence (see Schwarz (1978) for example). One of the originalities of our approach is that our model selection technique is done according to an other divergence which is 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. This model selection procedure can be viewed as a cross-validated Wasserstein criterion, in analogy with the cross-validated likelihood criterion (Smyth, 2000).

4 Numerical Experiments

4.1 Implementation details

We did our experiments in Python by using the PyTorch (Paszke et al., 2017) and Scikit-learn (Pedregosa et al., 2011) libraries with the same learning rate of 10^{-5} for the Adam optimization strategy (Kingma and Ba, 2014) everywhere. As in (Gulrajani et al., 2017), we chose $\lambda = 10$ for the gradient penalty in Eq. (10) and Eq. (12).

Since several spurious local minima are possible, careful initialization is crucial. This well-established fact for people using Neural Networks and people using Expectation-Maximization led us to the use of: (i) the so-called Xavier initialization (Glorot and Bengio, 2010) for step 1 (Auto-Encoder Initialization), and (ii) k-means++ (Arthur and Vassilvitskii, 2006) with k-means before GMM for step 2 (EM-based Mixture Initialization). Without Xavier and k-means++, the "AE + GMM" baseline (just the first 2 steps of our algorithm) performs poorly in our preliminary experiments. Thus, all the experiments we report in this work actually use these two initialization techniques.

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 (Kuhn, 1955; Stephens, 2000) on the confusion matrix. This criterion is also used in order to be comparable with other existing works in Table 1.

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 Figure 2. To this end, we consider a simple auto-encoder whose encoder architecture is D-64-32-16-8-d ($D = 2$ and $d = 1$ for input space and bottleneck dimensionalities) with tanh nonlinearities except in the last layers of the encoder and the decoder. 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.

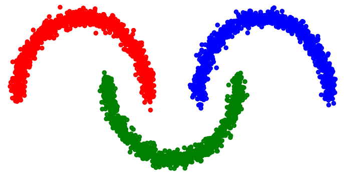

Figure 2: Samples from the Three-Moons dataset.

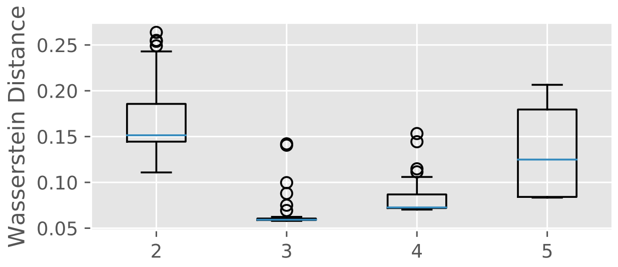

Figure 3: Wasserstein model selection on the three moons dataset for the number of clusters on a validation dataset, (on 30 runs for each number of clusters).

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 Figure 3 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): (i) *MNIST*, 70 000 handwritten digits images dataset living in dimension 784 (for 28×28 pixels); (ii) *Reuters*, English news stories labeled with a category tree Lewis et al. (2004). Following DEC Xie et al. (2015), 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; (iii) *Reuters-10k*, a random subset of Reuters with only 10000 examples (selected with the same random seed as DEC); (iv) *HHAR*, The Heterogeneity Human Activity Recognition (HHAR) dataset Stisen et al. (2015) 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. (2015) 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 (Jiang et al., 2016).

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 Figure 4, 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.

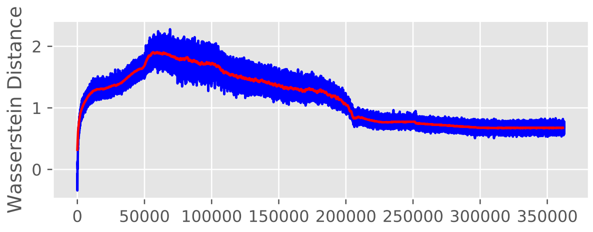

Figure 4: Mini-Batch Wasserstein Distance Estimated on Reuters-10k through Iterations (with rolling mean in red).

Furthermore, on Figure 5, we have the principal component analysis (PCA) representation of the code space at the end of our WAMiC optimization. It suggests the approximated *Gaussianity* of our classes forced by the adversarial training.

On MNIST, in Figure 6, 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. (2015). Furthermore, there is a supplementary significant adversarial improvement for WAMiC. For MNIST, these good results are comparable to those of supervised non-convolutional

Datasets	MNIST	Reuters	Reuters-10k	HHAR
WAMiC (ours)	97.26 ± 0.74	79.15 ± 1.02	79.84 ± 2.06	82.54 ± 3.37
VaDE (Jiang et al., 2016)	94.06	79.38	79.83	84.46
DEC (Xie et al., 2015)	84.30	75.63	72.17	79.82
AE + GMM (full covariance)	82.56	70.98	70.12	78.48
IMSAT (Hu et al., 2017)	98.40	–	71.00	–
GAR (Kilinc and Uysal, 2018)	98.32	–	–	–
DEPICT (Dizaji et al., 2017)	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 taken from DEC (Xie et al., 2015). The other lines do not correspond to the same architecture. The results of the first line, corresponding to our approach, are averaged over 10 runs (means and standard deviations are provided).

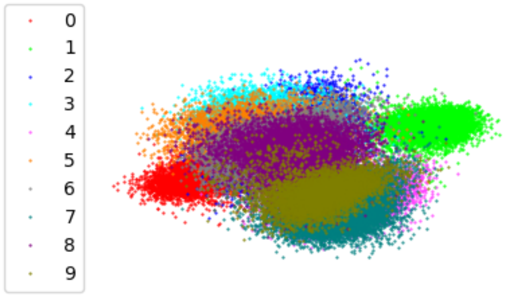

Figure 5: Principal Component Analysis rendering of the code space for MNIST at the end of the WAMiC optimizations, with colors indicating the ground truth labels (best seen in color).

networks with just a few layers of the 1990s (see `http://yann.lecun.com/exdb/mnist` for a complete MNIST benchmark). Indeed, it is worth noticing our unsupervised neural network is almost competitive with its supervised counterparts.

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. (Arjovsky et al., 2017)). In that regard, our algorithm inherits the strenghts of WGAN and models image data much more faithfully.

5 Conclusion and Future Work

We presented WAMiC: a mixture-based Deep Generative Model for Clustering that combines WGAN and Variational Inference. WAMiC learns a representation of the data together with a clustering-aware deep generative model. Empirically, we observed the emergence of some symbiosis between the tasks of clustering and learning a non-linear embedding. Interestingly, WAMiC achieves clustering results actually competitive with supervised classification ones, which give hopes for self-supervised learning method. Although WAMiC outperforms other algorithms on most datasets, the "AE + GMM" baseline works surprisingly well on average. A principled mathematical theory is missing to explain this empirical finding: *Why would a simple auto-encoder consistently map data in separated Gaussian groups?* In real-world settings, the "AE + GMM" baseline combines both ease of implementation and good results.

The flexibility of neural networks raises the question of the relevance of using a mixture prior. Indeed, both in theory and practice, using a unimodal Gaussian prior can model arbitrarily complex distributions (e.g. Dinh et al., 2017). However, the good empirical results of both our model and VaDE suggest that using a mixture prior is beneficial for deep generative modelling. Such mixture priors have also recently proven efficient not in clustering contexts, such as missing data imputation (Nazabal et al., 2018), or single-cell gene expression data modelling (Grønbech et al., 2018). One explanation of these phenomena might be that our algorithm detects the space zones where the density mass is occupied thanks to the probabilistic model inherent to the mixture model. The trade-off between the flexibility of the neural network and the rigidity of the mixture model is handled thanks to the adversarial optimization. Intuitively, a problematic case would consist in a dataset with non-separable classes: the system would not know precisely when to *activate the low-density threshold* in order to optimally put a class frontier.

Regarding future work, an interesting direction would be to design a end-to-end trainable deep generative model fit for clustering, similar in spirit to the approach of Dumoulin et al. (2017). We also plan to revisit our

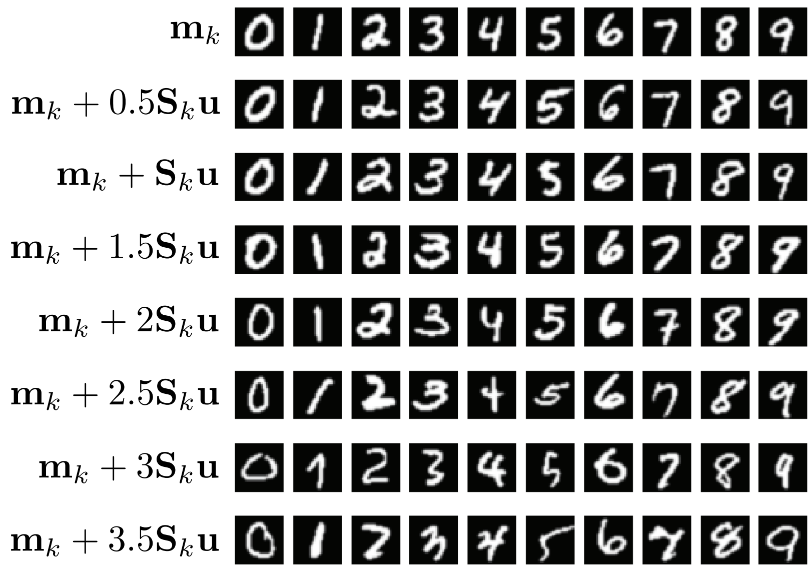
$\mathbf{m}_k$
$\mathbf{m}_k + 0.5\mathbf{S}_k\mathbf{u}$
$\mathbf{m}_k + \mathbf{S}_k\mathbf{u}$
$\mathbf{m}_k + 1.5\mathbf{S}_k\mathbf{u}$
$\mathbf{m}_k + 2\mathbf{S}_k\mathbf{u}$
$\mathbf{m}_k + 2.5\mathbf{S}_k\mathbf{u}$
$\mathbf{m}_k + 3\mathbf{S}_k\mathbf{u}$
$\mathbf{m}_k + 3.5\mathbf{S}_k\mathbf{u}$

Figure 6: 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.

work for other distances (away from the traditional ℓ_1 or ℓ_2): we wish to minimize the k-Wasserstein distance for the worst metric k in a wide but constrained (bounded) class of metrics in a way similar to the one of Genevay et al. (2017) and Li et al. (2017). We mentioned earlier that we chose the Wasserstein distance in lieu of Kullback-Leibler divergence. One of the reason of this choice is that it could be possible to investigate the geometric interpretation of a learned metric k that pushes data analysis further.

References

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

Aljalbout, E., Golkov, V., Siddiqui, Y., and Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., 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. PMLR.

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bojanowski, P. and Joulin, A. (2017). Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310.*

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. (2015). Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032.

Chang, W.-C. (1983). On using principal components before separating a mixture of two multivariate normal distributions. *Applied Statistics*, pages 267–275.

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

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and

Tomczak, J. M. (2018). Hyperspherical variational auto-encoders. *Uncertainty in Artificial Intelligence*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real nvp. *International Conference on Learning Representations*.

Dizaji, K. G., Herandi, A., Deng, C., Cai, W., and Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5747–5756. IEEE.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2017). Adversarially learned inference. In *International Conference on Learning Representations*.

Figurnov, M., Mohamed, S., and Mnih, A. (2018). Implicit reparameterization gradients. *Advances in Neural Information Processing Signals*.

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

Genevay, A., Peyré, G., and Cuturi, M. (2017). Learning Generative Models with Sinkhorn Divergences. (2017-83).

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Grønbech, C. H., Vording, M. F., Timshel, P. N., Sønderby, C. K., Pers, T. H., and Winther, O. (2018). scVAE: Variational auto-encoders for single-cell gene expression data. *bioRxiv*, page 318295.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1558–1567. PMLR.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2016). Variational deep embedding: A generative approach to clustering.

Kilinc, O. and Uysal, I. (2018). Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization. In *International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

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

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017). Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Mairal, J., Elad, M., and Sapiro, G. (2008). Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69.

Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. (2018). Handling incomplete heterogeneous data using VAEs. *arXiv preprint arXiv:1807.03653*.

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2001). On spectral clustering: Analysis and an algorithm. 14(2):849–856.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

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

Ponce, J. and Forsyth, D. (2011). *Computer vision: a modern approach.*

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Smyth, P. (2000). Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and computing*, 10(1):63–72.

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

Sonoda, S. and Murata, N. (2016). Decoding stacked denoising autoencoders. *arXiv preprint arXiv:1605.02832.*

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

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. (2015). Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. pages 127–140.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2018). Wasserstein auto-encoders. In *International Conference on Learning Representations.*

Turk, M. A. and Pentland, A. P. (1991). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

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(Dec):3371–3408.

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

Xie, J., Girshick, R., and Farhadi, A. (2015). Unsupervised deep embedding for clustering analysis. *arXiv preprint arXiv:1511.06335.*

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. (2016a). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint arXiv:1610.04794.*

Yang, J., Parikh, D., and Batra, D. (2016b). Joint Unsupervised Learning of Deep Representations and Image Clusters.

Zelnik-Manor, L. and Perona, P. (2004). Self-tuning spectral clustering. *Advances in neural information processing systems*, 17(1601-1608):16.