Wasserstein Clustering

Warith HARCHAOUI

October 2020

Man Gave Names to All the Animals

Bob Dylan *in* Slow Train Coming, 1979

Abstract

Clustering is partitionning the data into groups. Deep approaches for clustering are promising for extending the success of neural networks beyond the limits of supervised classification. In this chapter, clustering is tackled at the crossroad of several literatures: auto-encoders, generative adversarial networks (GANs) for optimal transport, statistical mixture models. Two criteria are studied: (i) the first generatively minimizes the Wasserstein distance between data and cluster-separated generated data inspired by the GANs success and (ii) the second discriminatively maximizes over all partitions the Wasserstein distances between the associated groups.

These two mechanisms are compatible with model selection according to a Wasserstein criterion measured on held-out validation data. Competitive results are achieved on benchmark datasets such as images, sparse and dense data, with the benefits of selecting the number of groups which promises interesting further research.

Keywords

Clustering, Neural Networks, Wasserstein Generative Adversarial Networks, Mixture Model, Deep Generative Models, Optimal Transport, Discriminative Clustering, Generative Clustering

Contents

1 Introduction

In a new environment, the first thing a human being (even as a child) does is mentally grouping the elements of the surroundings and putting names on those groups. The ambition of this work is to tackle the problem of clustering within the deep learning framework. While the vast majority of the model-based clustering approaches focused on the Kullback-Leibler divergence (closely related to maximum likelihood up to an additional constant term [Bishop, 2006]), we investigate in this chapter an attempt to operate with the Wasserstein distance in lieu of Kullback-Leibler divergence for clustering with mixture distributions.

In the first part of this chapter, we take some *generative* ingredients coming from the machine learning literature to build our clustering technique as chronologically, we got inspired by Generative Adversarial Networks (GAN) remarkable results [Goodfellow, 2016]. In the second part of this chapter, on the contrary, we choose more *discriminative* ingredients while still using some of the statistical and algorithmical tools empowered by the experience gained previously. Recently and independently from us, we have seen that Mukherjee et al. [2019] successfully built a better generative clustering work than us but our approach is slightly different: Jensen-Shanon divergence from the original GAN [Goodfellow et al., 2014] whereas we used the Wasserstein distance in our case thanks to the work of Arjovsky et al. [2017].

In the discriminative clustering context, Bouveyron and Brunet [2011] revisited an ancient approach from Fisher [1936]: separating clusters with a maximum Kullback-Leibler divergence inter-cluster hence the discriminative point of view. Likewise and with the same spirit, we propose to maximally separate data into clusters which is exactly adopting a discriminative angle to clustering instead of generating closest possible clusters to data which corresponds to a generative angle. Here, we explore new ways to do clustering with inter-cluster Wasserstein distances maximization moving away from the conventional literature by borrowing ideas from generative model-based approaches and discriminative methods.

1.1 Related Work

Authors such as Flamary et al. [2018] actually did go down that appealing research road for Wasserstein-distance-based discriminative clustering with simultaneous linear dimensionality reduction. Indeed, Flamary et al. [2018] adapted the old latent discriminant analysis of Fisher [1936] but with the Wasserstein distance instead of a sum of distances approach that is closer to a *Kullback-Leibler mindset* while still being both discriminative and model-based. The choice of divergence (even beyond Kullback-Leibler and Wasserstein) and the choice between generative and discriminative approaches are scientifically intruiguing and produce different algorithms in practice as far as clustering is concerned.

The ambition of this work is to tackle clustering within the deep learning framework because of its large success in supervised learning that we want to inherit in unsupervised classification (a. k. a. clustering). More precisely, our motivation does not come from the popularity of the so-called *deep learning* approaches but is rather nurtured by the functional expressivity that neural networks and the fact that the associated scientific community is profuse in free high-quality toolboxes, which is, of course, impossible to distangle from the deep learning tremendous popularity.

To achieve these goals, we take advantage of the recent Wasserstein Generative Adversarial Networks research to estimate these aforementioned generative and discriminative criteria. The smoothness implied by the deep learning optimization gradual procedures made us think that soft memberships probabilities should be prefered over discrete categorization output. Handling the Wasserstein distance meant dealing with transport plans which are uneasy objects intuitively in terms of software programming. Thanks to the Kantorovich-Rubinstein formulation, we end up with uncoupled lossed for the distributions which is easier but at the price of an adversarial min-max optimization for the generative case which is notoriously difficult to monitor in practice (although much research attention simplified it). For the discriminative case, maximizing the Wasserstein distance estimated by the Kantorovich-Runbinstein maximization duality is especially suitable for optimization stability. Clearly, we do benefit from algorithmic tools from the adversarial neural networks for the generative part of this work but we still do benefit from this tools for the discriminative easier algorithm construction, especially for critics (or potential) lipchitzian functions. Indeed, for the discriminative algorithm, we surprisingly avoid undergoing the difficulty and instability of a min-max (adversarial) optimization with just an overall maximization instead.

At the very start of this thesis in 2016, in our preliminary experiments, a Gaussian mixture model trained with Expectation-Maximization [Dempster et al., 1977] on codes coming from a vanilla

MLP[1] auto-encoder without convolutions is able to reach above 80% of unsupervised clustering accuracy on the famous digits MNIST images dataset[2] directly on raw pixels. Encouraged by this surprisingly good result, we pursue our efforts towards an attempt to take advantage of that empirical fact in a sound framework.

While supervised classification has been a long-standing problem for many decades until recently thanks to computational hardware dramatic improvements and a considerable research effort towards statistical tools and algorithms, unsupervised classification (a. k. a. clustering) is still a difficult area but taking advantage of the supervised findings proved efficient in terms of research. In fact, although we can only confirm that clustering is an ill-posed problem as explained earlier due to the Kleinberg's impossibility theorem [Kleinberg, 2003], we still find it desirable with the same issues that the supervised classification research had to solve in easier settings because supervised classification has a clearer objective. Beyond ill-posed problems, the way we tackle model-based clustering in this work has at least two issues, namely model selection and mixture parametrization that we brielfy describe here before presenting two techniques, one generative which we improved thanks into a discriminative one.

1.2 Model Selection

In supervised techniques, after training different models with some different hyper-parameters, we can find th best set of hyper-parameters thanks to (cross-) validation: measuring the different associated accuracy scores on a labeled held-out dataset. Unfortunately, this is not a solution in our unsupervised clustering task where no labels exist (never: neither at training, nor validation and of course not at testing stages). Nevertheless, we have two ways to circumvent this problem [Bouveyron et al., 2019]:

with a held-out and unlabeled dataset our model-based techniques compare distributions within a statistically meaningful quantity (divergence or distance minimization or maximization): one representing the data and another one representing a fitted model. A natural way to select some hyper-parameters sets among several trained models is to measure the same training quantity and same model but with different data corresponding to the validation dataset. In this work, we develop two techniques: DiWaC (which corresponds to a discriminative approach) and GeWaC (which corresponds to a generative one) and these objectives measure a model-data fitting Wasserstein score and allow us to measure a model-data fitting score on unseen and held-out data. In supervised classification, the validation accuracy in measured from the comparison between pre-annotated labels and predictions from a model optimized with training data. In our unsupervised clustering case, we allow ourselves to loosely adapt the *held-out validation score* expression because we simply measure some Wasserstein distances between model distributions and validation empirical data distributions (without labels otherwise this would not be realistic) to check under- and over-fitting phenomena;

withtout any other dataset than the initial training set a vast literature exists with Kullback-Leibler divergences and Likelihood Maximization for selecting a good model among many [Schwarz et al., 1978, Biernacki et al., 2000] using the number of model parameters following an Occam's razor for Bayesian Machine Learning to compensate for extra data validation but to the best of our knowledge, we do not know non-likelihood-based techniques (here we use Wasserstein distances instead).

We clearly chose the first approach with some held-out unlabeled data with our Wasserstein distances. Now we explore how we manipulate mixtures distributions thanks to the *reparametrization trick*.

1.3 Reparametrization Trick for a Mixture

In order to handle distributions and more precisely being able to differentiate our objective functions with respect to the parameters of these distributions for learning purposes, we adapt the *Reparametrization Trick* from Kingma et al. [2015] first used in a variational context[3]. Indeed, we have chosen a probabilistic framework from which probability distribution parameters are estimated. Usually, in deep learning, first, the optimization process finds functional parameters for output prediction thanks to the minimization (or maximization) of an objective funcion and the notion

[1] Multi-Layered Perceptron

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

[3] see this scientific blog for details: `https://gregorygundersen.com/blog/2018/04/29/reparameterization`

of gradient gets easy under mild (sub)-differentiability conditions towards training optimization. In our case, this is different: we do not optimize functional parameters for an output prediction but we do find density parameters for a distribution which plays the role of the output prediction in a probabilistic fashion. In less than a decade, the variational (see the surveys accomplished by Kingma et al. [2019]) and optimal transport (see the book of Peyré et al. [2019]) literatures in machine learning got much attention for that estimation scenario with probability distribution parameters. Instead of using some sophisticated mathematical tools for differentiating some objective over some distribution parameters, many researchers used the *Reparametrization Trick* thanks to Kingma et al. [2015]: transforming a known pseudo-random noise as an estimator of the theorereticall random variable for optimization reasons which makes differentiation easier. Graves [2016] gives a more comprehensive overview of the problem.

Indeed, Kingma et al. [2015] allow to directly specify a prior distribution over the code space of a variational auto-encoder (VAE). Inference is done using a stochastic gradient variational bayesian (SGVB) method, based on a reparametrization of the variational lower bound. In this work, we will revisit and adapt this technique called the *Reparametrization Trick* for our distributions settings. 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 Gaussian mixture prior.

In practice, the Reparametrization Trick consists in manipulating a random variable $\mathbf{z}$ coming from a parametrized distribution (say Gaussian of mean $\mathbf{a}$ and covariance matrix $\mathbf{B} = \mathbf{C}\mathbf{C}^\top$ such that $\mathbf{z} \sim \mathcal{N}(\mathbf{a}, \mathbf{B})$) thanks to a default random generator (here $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$) that is transformed through ($\mathbf{z} = \mathbf{a} + \mathbf{C}\boldsymbol{\epsilon}$ which simulates $\mathbf{z} \sim \mathcal{N}(\mathbf{a}, \mathbf{B})$) in order to get a differentiable version of the random variable we needed. This technique can used in many different ways according to the literature [Graves, 2016, Doersch, 2016, Blei et al., 2017].

2 Generative Wasserstein Clustering

Contemporary to a frenetic rythm of papers about Generative Adversarial Networks (GAN) since their spectacular birth [Goodfellow et al., 2014], we admit we have been deeply influenced especially by the Wasserstein-based approaches [Arjovsky et al., 2017] on the one hand and the French aura associated to optimal transport since the acclaimed work of Villani [2008] on the other hand. Indeed, Arjovsky et al. [2017] proposed a praiseworthy attempt to outline this particularly fast-growing scientific landscape of GANs initiated by Goodfellow [2016] by insisting on the mathematical quantities minimized between real and generated beyond the detective-forger metaphor describing the revisited min-max adversarial optimization method (the forger being the generator function and the detective being the discriminator or critic function).

At the same time, we wanted to accomplish some model-based clustering contributions thanks to the statistical legacy summarized by Bouveyron and Brunet-Saumard [2014b] and later in the commendable book of Bouveyron et al. [2019]. As clustering and data imitation are both unsupervised tasks, we wondered since 2016, how was it possible to accomplish clustering with data imitation tools such as GANs. Indeed, we investigate in this part a generative approach by mimicking data that matches best real data with respect to the Wasserstein distance.

2.1 Deep Generative Models for Clustering

In the recent statistical learning literature, there is a significant trend towards better deep generative models (DGM) based on different inference procedures: (i) likelihood, (ii) GANs.

First, the goal of the likelihood-based approach is to minimize the Kullback-Leibler divergence between the original data distribution and the parametrized model data distribution (which is equivalent to maximizing the likelihood Duda et al. [2012]). A recent example of that kind is the work of Dinh et al. [2017] that had to introduce neural networks bijections in order to apply a revisited *change of variable formula* on the likelihood mainly because the Kullback-Leibler divergence requires same distribution supports and thus a bijection is almost mandatory to parse the entire data space (including the vast empty data zones exactly like Gaussian mixture where each Gaussian component support is the entire space).

Second, inverting the problem of accessing to the parametrized model data distribution is to sample generated data from it which what Goodfellow et al. [2014] astutely proposed with GANs: they minimize a divergence between real and generated data over the generator function parameters implemented by a neural network thanks to a critic function that evaluates that divergence. Since then, the Wasserstein distance [Gulrajani et al., 2017] through the Wasserstein GAN (WGAN) was

also proposed among other divergences summarized by a survey done by Goodfellow [2016] gives a glimpse of what is offered by that thriving on-going scientific literature.

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 $\mathbb{R}^D$) into K homogeneous groups. We suppose there exists a latent code space $\mathcal{Z}$ of a low dimension d (say $d = 10$ which is low compared to the original data dimensionality D: $d \ll D$) such that there is a mapping $\mathcal{D}$ between $\mathcal{Z}$ to $\mathcal{X}$ connecting the random variable $\mathbf{x}$ in $\mathcal{X}$ and its latent counterpart $\mathbf{z}$ in $\mathcal{Z}$. We also assume that $\mathbf{z}$ follows a mixture $\mathcal{M}$ of rather common distributions as components (say Gaussian).

At the heart of our Generative Wasserstein Clustering (GeWaC) 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 two-ways bridge between the data space and a code (or latent) space which more akin to clustering alleviating the curse of dimensionality thanks to its lower dimensionality (d instead of D).

Our model consists in saying that the data have been generated as follows. The clustering variable c

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

corresponds to a categorical random variable among $K \geq 2$ clusters with prior proportions defined in vector $\boldsymbol{\pi}$ (of K positive scalars which sums to 1). In this generative process, once the cluster k is chosen, one can generate a code in $\mathcal{Z}$ which implies a mixture marginal for $\mathbf{z}$:

$$\mathbf{z}|c = k \sim g_k(.; \xi_k) \qquad\qquad \mathbf{z} \sim \sum_{k=1}^{K} \boldsymbol{\pi}_k g_k(.; \xi_k) \tag{2}$$

for the probability distributions $(g_k)_{k=1}^K$ of each of the K components parametrized by ξ_k. Ultimately a point in $\mathcal{X}$ is generated:

$$\mathbf{x}|\mathbf{z} \sim \delta_{\mathcal{D}(\mathbf{z})} \tag{3}$$

To translate into statistics the assumption that the code data lie extremely close to a low-dimensional manifold, we should have written $\mathcal{N}(\mathcal{D}(\mathbf{z}), \sigma^2 \mathbf{I}_p)$ instead of a Dirac $\delta_{\mathcal{D}(\mathbf{z})}$ located on a decoded data point $\mathcal{D}(\mathbf{z})$ and them further assume that $\sigma \to 0$. In this context, the posterior probability needed to cluster $\mathbf{x}$ is given by

$$\mathbb{P}(c = k|\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})}[P(c = k|\mathbf{z})] \tag{4}$$

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, Kingma et al., 2015] which has been described above and has a central role in our adversarial optimization as we will see later.

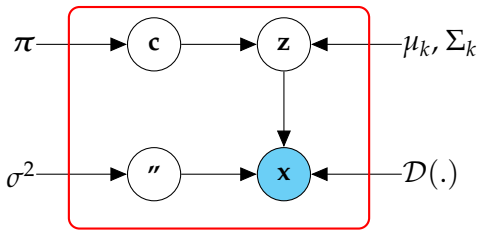

Figure 1: Graphical Model for Data Generation

In the following, we will illustrate our methodology using a mixture of Gaussians in the code space, *i.e.* we choose:

$$g_k(.; \xi_k) = \mathcal{N}(.; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{5}$$

(a Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$) where $(\boldsymbol{\pi}_k)_{k=1,...,K}$, $(\boldsymbol{\mu}_k)_{k=1,...,K}$, and $(\boldsymbol{\Sigma}_k)_{k=1,...,K}$ are the mixture parameters stored in $\theta_{\mathcal{M}}$. Note that, beyond the Gaussian mixture 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]. Our graphical model displayed in Fig. 1 summarizes the chosen data generation modelling.

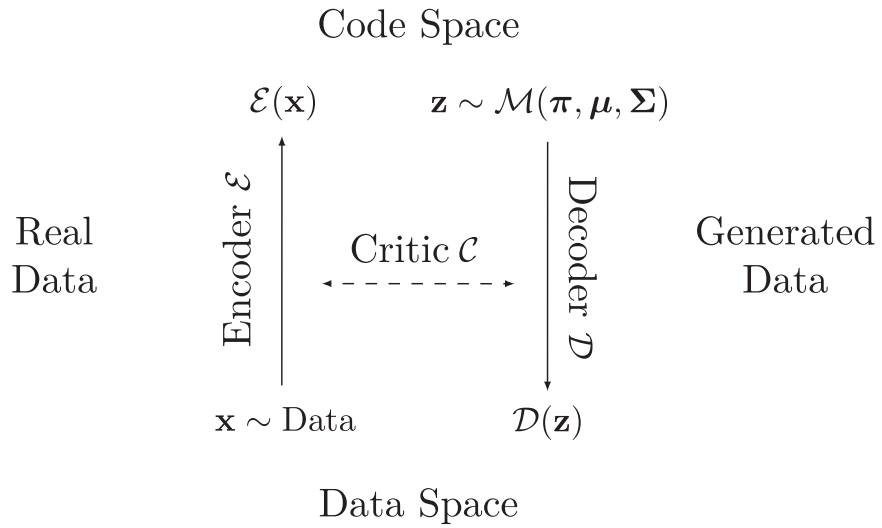

Figure 2: GeWaC Optimization Scheme

In order to fit our generative model, we generate a random variable $\mathcal{D}(\mathbf{z})$ 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 $\mathcal{D}(\mathbf{z})$ (through a pseudo-random generator for $\mathbf{z}$ and the decoder or generator $\mathcal{D}$), which is a scenario where the WGANs proved successful as Fig. 2 summarizes the modeling proposed here.

The posterior probability $p(\mathbf{z}|\mathbf{x})$ in Eq. (4) 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 et al. [2015], 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:

$$P(c = k|\mathbf{x}) \simeq \frac{\boldsymbol{\pi}_k g_k(\mathcal{E}(\mathbf{x}); \xi_k)}{\sum_{k'=1}^{K} \boldsymbol{\pi}_{k'} g_{k'}(\mathcal{E}(\mathbf{x}); \xi_k)} \tag{6}$$

following the Bayes formula.

After several attempts, we came up with two fruitful strategies to make our system work: first, the *Concatenation Trick* from Dumoulin et al. [2017] to ensure some consistency between code and data spaces and second, the *Reparametrization Trick* from Kingma et al. [2015] to handle parametrized mixture distributions described earlier.

2.2 Concatenation Trick

Our concatenation approach consists in operating a clustering that is consistent both in the embedding code and input data spaces as described in Fig. 2:

- on the one hand, there is one space (called "code space") with real encoded data (random variable $\mathcal{E}(\mathbf{x})$) next to a generated mixture distribution $\mathbf{z} \sim \mathcal{M}$;
- on the other hand, there is an other space (called "input space") of real data at hand (random variable $\mathbf{x}$) coexisting with generated decoded signal $\mathcal{D}(\mathbf{z})$.

This mechanism allows our GAN technique to bring real and generated data distributions together. We use a small dimensionality code space to benefit from the natural probabilistic interpretation of mixture (e. g. Gaussian) for clustering in which each code mixture component corresponds to a cluster. To guarantee a minimum level of consistency for our encoder/decoder system bridging between our two aforementioned spaces, we use a technique that we dub the "concatenation trick" proposed by Dumoulin et al. [2017]. This is necessary to make sure the encoder $\mathcal{E}$ and decoder $\mathcal{D}$ functions are reciprocal mathematically *almost everywhere* on the data and code manifolds at hand (see Donahue et al. [2016] for proof).

On the data side, if we only bring together the distributions of $\mathbf{x}$ and $\mathcal{D}(\mathbf{z})$, then this would not have been enough because clustering would take place for $\mathcal{E}(\mathbf{x})$ and $\mathbf{z}$. On the code side, if we only bring together the distributions of $\mathcal{E}(\mathbf{x})$ and $\mathbf{z}$, then this would not have been enough either because true data $\mathbf{x}$ would not be even concerned anymore which would be equivalent to disconnecting the codes from data. Concatenating data and codes solves these two problems. Fortunately, in other settings (data imitation and compression), the concatenation trick formulation of Dumoulin et al. [2017] nicely fits ours, thanks to the idea of bringing together the distributions of

- $\tilde{\mathbf{x}} = a(\mathbf{x}) = \left[\mathbf{x}^\top, \mathcal{E}(\mathbf{x})^\top\right]^\top \sim \tilde{p}$ considered as *real* with $\mathbf{x}$ representing data;
- $\tilde{\mathbf{y}} = b(\mathbf{z}) = \left[\mathcal{D}(\mathbf{z})^\top, \mathbf{z}^\top\right]^\top \sim \tilde{q}$ considered as *generated* with $\mathbf{z}$ sampled over a parametrized mixture $q = \mathcal{M}(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

by minimizing the Wasserstein distance between $\tilde{p}$ and $\tilde{q}$ over all parameters (mixture, encoder and decoder). This way, we elegantly get the *almost everywhere reciprocity* between the encoder $\mathcal{E}$ and decoder $\mathcal{D}$ on the data manifold adapting the work of Dumoulin et al. [2017] and also Donahue et al. [2016].

The nature of the chosen GAN (Wasserstein GAN or other) is not crucial here as Arjovsky et al. [2017] explained that the original GAN [Goodfellow et al., 2014] uses the Jensen-Shannon divergence and WGAN uses the Wasserstein distance (which is *a fortiori* a divergence) but Dumoulin et al. [2017] empirically show the improvement attributed to the concatenation trick *vs.* without it in terms of image rendering. Interestingly, concatenation becomes a key element in our case to make the whole system work because of our clustering goal (that Dumoulin et al. [2017] do not have) to maintain data-code consistency through the encoder/decoder reciprocity. Intuitively, if the concatenated variables in the previous bulletted list have similar distributions with respect to the chosen GAN-specific divergence at hand, then the marginals are close too. Thus, our use of the concatenation trick gets handy because it ensures that our pair of encoder/decoder functions $(\mathcal{E}, \mathcal{D})$ still behaves in a reciprocal fashion in spite of the mixed GAN and mixture framework in a surprisingly stable manner in terms of optimization.

In fact, we first proposed a model without concatenation which meant the adversarial WGAN optimization did not involved the encoder $\mathcal{E}$ as only the distribution of $\mathbf{x}$ and $\mathcal{D}(\mathbf{z})$ were being put together: this was not principled especially because the clustering decision rule given in Eq. (6) involves $\mathcal{E}$ which forced us to retrain a final post-processing with an adhoc autoencoder loss with all parameters fixed except the ones of $\mathcal{E}$ for what seems to be an update with respect to the decoder $\mathcal{D}$. The contribution of Dumoulin et al. [2017] made all these considerations disappear in an easy yet principled fashion.

Now, we minimize the Wasserstein distance between these two augmented data associated distributions thanks to the Kantorovich-Rubinstein duality with a WGAN [Arjovsky et al., 2017, Salimans et al., 2016, Gulrajani et al., 2017, Miyato et al., 2018] :

$$\max_{\|\nabla \mathcal{C}\| \leq 1} \mathbb{E}_{\mathbf{x} \sim p}\left[\mathcal{C}\big(a(\mathbf{x})\big)\right] - \mathbb{E}_{\mathbf{z} \sim q}\left[\mathcal{C}\big(b(\mathbf{z})\big)\right] \tag{7}$$

with $\mathcal{C}$ called the critic and implemented in practice by a neural network of parameters $\theta_{\mathcal{C}}$ and constrained as 1-Lipschitzian by a chosen method among weight-clipping [Arjovsky et al., 2017], augmented Lagrangian [Gulrajani et al., 2017] or with more stability from online power iteration [Miyato et al., 2018].

Assuming that the codes $\mathbf{z}$ come from a mixture of Gaussians with full covariance matrices $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d \times d}$ ($d \ll D$), for each k among the K components the corresponding variable $\mathbf{z}_k$ follows:

$$\mathbf{y}_k = b(\mathbf{z}_k) = \Big[\mathcal{D}(\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top, (\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top\Big]^\top \tag{8}$$

where $\mathbf{e} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and $\mathbf{S}_k$ is a Cholesky decomposition-inspired representation (with non-zeros only in the lower-triangular part and strictly positive diagonal entries guaranteed by exponentials first and then affine-transformed sigmoids for better eigenvalues amplitude control) of the full covariance $\boldsymbol{\Sigma}_k = \mathbf{S}_k \times \mathbf{S}_k^\top$ such that the transformed random variable $\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k$ behaves as if it comes from $\mathcal{N}(\boldsymbol{\mu}_k, \mathbf{S}_k \times \mathbf{S}_k^\top) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^\top)$ in the spirit of the *Reparametrization Trick* [Kingma and Welling, 2013, Kingma et al., 2015] for unconstrained optimization which is much easier and well studied in stochastic gradient settings.

All the equations above meet in:

$$\begin{aligned}\mathcal{L}(\theta_{\mathcal{E}},\theta_{\mathcal{D}},\theta_{\mathcal{M}},\theta_{\mathcal{C}}) &= \mathbb{E}_{\mathbf{x}\sim p}\left[\mathcal{C}\left(\left[\mathbf{x}^\top,\mathcal{E}(\mathbf{x})^\top\right]^\top\right)\right] \\ &\quad -\sum_{k=1}^{K}\pi_k\times\mathbb{E}_{\mathbf{e}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)}\left[\mathcal{C}\left(\left[\mathcal{D}(\mathbf{S}_k\times\mathbf{e}\right.\right.\right. \\ &\quad \left.\left.\left.+\boldsymbol{\mu}_k)^\top,(\mathbf{S}_k\times\mathbf{e}+\boldsymbol{\mu}_k)^\top\right]^\top\right)\right]\end{aligned} \tag{9}$$

and thus we optimize:

$$\min_{\theta_{\mathcal{E}},\theta_{\mathcal{D}},\theta_{\mathcal{M}}}\max_{\theta_{\mathcal{C}}}\mathcal{L}(\theta_{\mathcal{E}},\theta_{\mathcal{D}},\theta_{\mathcal{M}},\theta_{\mathcal{C}}) \tag{10}$$

Indeed, $\max_{\theta_{\mathcal{C}}}\mathcal{L}(\theta_{\mathcal{E}},\theta_{\mathcal{D}},\theta_{\mathcal{M}},\theta_{\mathcal{C}})$ corresponds to the Wasserstein distance between real data $\left[\mathbf{x}^\top,\mathcal{E}(\mathbf{x})^\top\right]^\top$ and generated data $\left[\mathcal{D}(\mathbf{S}_k\times\mathbf{e}+\boldsymbol{\mu}_k)^\top,(\mathbf{S}_k\times\mathbf{e}+\boldsymbol{\mu}_k)^\top\right]^\top$ by the mixture $\mathcal{M}$ and decoder $\mathcal{D}$.

Originally Wasserstein GAN 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. Here, we use a tunable mixture distribution instead for clustering purposes. In a regular Wasserstein GAN, the fixed distribution has no parameter taking part in the data generation mechanism. For that reason among other vocabulary reasons, in our GeWaC algorithm, we call *generator* the union of the parametrized mixture distribution $\mathcal{M}$ associated with the decoder neural network $\mathcal{D}$ (that brings the noise generated from the mixture into fake data).

/////T»»> The mechanism that we dub concatenation trick proposed by Dumoulin et al. [2017] naturally enforces the encoder and the decoder being reciprocal which can be proven in a way that is very close to what Donahue et al. [2016] did for interested readers.

2.3 Algorithm

Our GeWaC algorithm can be decomposed in three successive steps:

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

   $$(\theta^0_{\mathcal{E}},\theta^0_{\mathcal{D}}) = \arg\min_{\theta_{\mathcal{E}},\theta_{\mathcal{D}}}\mathbb{E}_{\mathbf{x}\sim p}\left(\|\mathcal{D}\circ\mathcal{E}(\mathbf{x})-\mathbf{x}\|_2^2\right) \tag{11}$$

2. **EM-based Gaussian mixture initialization**
   We fit a Gaussian mixture model

   $$\mathcal{M}\left(\text{SoftMax}(\boldsymbol{\alpha}),\boldsymbol{\mu},(\mathbf{S}_k\times\mathbf{S}_k^\top)_{k=1,\dots,K}\right) \tag{12}$$

   parametrized by $\theta_{\mathcal{M}}=(\boldsymbol{\alpha}_k,\boldsymbol{\mu}_k,\mathbf{S}_k)_{k=1,\dots,K}$ with the Expectation-Maximization algorithm [Dempster et al., 1977] on the encoded data:

   $$\theta^0_{\mathcal{M}} = \arg\max_{\theta_{\mathcal{M}}}\mathbb{E}_{\mathbf{x}\sim p}\left[\log\left(\sum_{k=1}^{K}\pi_k\times\mathcal{N}(\boldsymbol{\mu}_k,\mathbf{S}_k\times\mathbf{S}_k^\top)(\mathcal{E}^0(\mathbf{x}))\right)\right] \tag{13}$$

   where the covariance matrices are parametrized by $\boldsymbol{\Sigma}_k=\mathbf{S}_k\times\mathbf{S}_k^\top$ to save tedious symmetry and eigenvalues signs constraints and the proportions are parametrized by $\boldsymbol{\pi}=\text{SoftMax}(\boldsymbol{\alpha})$ for easily imposing 1-sum positive constraints on the proportions.

3. **Critic Initialization**
   The critic function $\mathcal{C}$ role is to estimate the Wasserstein distance in order to get good gradient estimation for the rest of the parameters. The previous steps intialized a generator process that we should evaluate first before taking the gradient from it for the other parameters:

   $$\hat{\theta}_{\mathcal{C}} = \arg\max_{\theta_{\mathcal{C}}}\mathcal{L}(\theta^0_{\mathcal{E}},\theta^0_{\mathcal{D}},\theta^0_{\mathcal{M}},\theta_{\mathcal{C}}) \tag{14}$$

   from Eq. (10) which is optimized thanks to the algorithm 1 calling the algorithm 2 but without 3.

4. **Clustered Data Generation**

   The previous steps made this final step well initialized to optimize all the parameters thanks to the algorithms 1 calling both 2 and 3:

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

Our GeWaC technique is "end-to-end trainable". Steps 1, 2 and 3 are just reasonable initializations for step 4. Finally, we use the MAP rule in the code space (Bayes formula) in order to finally cluster the data points which makes our simple approach after training time particularly fit for clustering. For training this generative clustering, the optimization is orchestrated by algorithm 1 that calls Wasserstein distance estimation updates in iterations involving only critics neural networks parameters which is described by algorithm 2 and minimizes the Wasserstein distance estimates with respect to all parameters in algorithm 3 excluding the ones of the critics.

Algorithm 1 Optimization algorithm

while $\theta_{\mathcal{E}}$, $\theta_{\mathcal{D}}$ and $\theta_{\mathcal{M}}$ have not converged **do**
  Sample a mini-batch of size $B \times K$ from the dataset $\mathbf{x}_{i,k} i = 1, \ldots, B \quad k = 1, \ldots, K$
  Compute the critic evaluation on the mini-batch of points and codes concatenation

$$\mathbf{a}_{i,k} \leftarrow \mathcal{C}([\mathbf{x}_{i,k}^{\top}, \mathcal{E}(\mathbf{x}_{i,k})^{\top}]^{\top})$$

$$i = 1, \ldots, B \quad k = 1, \ldots, K$$

  for $j = 1, \ldots, N_{\text{critic}}$ **do**
    Wasserstein Estimation Step
  end for
  Wasserstein Minimization Step
end while

Algorithm 2 Wasserstein Estimation Step

Free critics gradients accumulators
Sample some $B \times K$ Gaussian noise codes from a pseudo-random generator

$$\mathbf{e}_{i,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$$

$$i = 1, \ldots, B \quad k = 1, \ldots, K$$

Compute the critic evaluation on the mini-batch of decoded noise and its original version concatenation

$$\mathbf{b}_{i,k} \leftarrow \mathcal{C}([\mathcal{D}(\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^{\top}, (\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^{\top}]^{\top})$$
$$i = 1, \ldots, B \quad k = 1, \ldots, K$$

Compute

$$w \leftarrow \frac{1}{B \times K} \sum_{i=1}^{B} \sum_{k=1}^{K} \mathbf{a}_{i,k} - \sum_{k=1}^{K} \boldsymbol{\pi}_k \frac{1}{B} \sum_{i=1}^{B} \mathbf{b}_{i,k}$$

Perform a gradient ascent step with w over $\theta_{\mathcal{C}}$

Algorithm 3 Wasserstein Minimization Step

1: Free encoder, decoder and mixture gradients accumulators
2: Sample some $B \times K$ Gaussian noise codes from a pseudo-random generator

$$\mathbf{e}_{i,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$$

$$i = 1, \ldots, B \quad k = 1, \ldots, K$$

3: Compute the critic evaluation on the mini-batch of decoded noise and its original version concatenation

$$\mathbf{b}_{i,k} \leftarrow \mathcal{C}([\mathcal{D}(\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top, (\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top]^\top)$$

$$i = 1, \ldots, B \quad k = 1, \ldots, K$$

4: Compute

$$w \leftarrow \frac{1}{B \times K} \sum_{i=1}^{B} \sum_{k=1}^{K} \mathbf{a}_{i,k} - \sum_{k=1}^{K} \boldsymbol{\pi}_k \frac{1}{B} \sum_{i=1}^{B} \mathbf{b}_{i,k}$$

5: Perform a gradient descent step with w over $(\theta_\mathcal{E}, \theta_\mathcal{D}, \theta_\mathcal{M})$

2.4 Model Selection for GeWaC

Once we get our trained data generator, we can measure the Wasserstein distance between generated data and some held-out validation data distributions (that we can sample from) to check under/over-fitting and ultimately choose the number of classes or the architecture of neurons and layers. There is one subtlety though: we must measure non-augmented data Wassertein distance between non-augmented generated data and decoded mixture noise with recomputed proportions from validation memberships probabilities means. Otherwise, the encoder-decoder part of our systems will not be fairly compared: by removing the dimensionality augmentation, we make possible the selection of the coding space dimensionality for example. Thus, algorithmically, transforming our training procedure into a validation one consists in keeping algorithms 1 and 2 and leaving 3 and the augmentation parts.

We could consider our generative attempt as a natural extension of what the Expectation-Maximization for the Gaussian mixture model algorithm does with the Kullback-Leibler divergence but with stochastic gradient descent for the Wasserstein Distance. One difficulty appears though from mixture distributions being problematic because of the partial discreteness of the parameter space [Graves, 2016]. Specifically, mixture weights are particularly difficult to optimize. Consequently, Dilokthanakul et al. [2016] assume that these weights are known beforehand, which is not always the case in real-world problems. Jiang et al. [2016] propose a method for optimizing these weights but does not provide empirical evidence on imbalanced data to support this scheme. On our side, we did manage to get reasonable results but without completely being fair about proportions: if we trust our careful initialization procedures and let proportions have very low learning rate, then in practice, it is as if we freeze these proportions to a constant vector value although we did not realize it at first while being deceived by apparently good results except when initial proportions are not valid.

2.5 Collapsing Effects

Since its introduction of GANs, Goodfellow et al. [2014] warned the reader about what they called the "Helvetica scenario" in which their generator is trained *too often* compared to the not-enough-updated *discriminator* (a word that is translated by *critic* since Wassertein GANs [Arjovsky et al., 2017]). Indeed, a generator can intuitively be good at generating data from a specific region of space without being able to generalize to other space zones of data as the discriminator is fooled when comparing good localized generated data compared to real data. This is a kind of a spatial unsupervised over-fitting that is commonly defined in supervised learning. We end up with a GAN that gets unable to generate the same variety of data as real data hence the *collapsing effect* phenomenon name.

In our GeWaC technique, we observed a similar problem on the mixture side. Indeed, because of the richness of potential functions expressed by neural networks, only one obviously non-clustering mode of a mixture is enough to go through the decoder and parse the whole data *manifold* space. We decided to give up this technique for that reason to prefer a discriminative approach that does not need to produce data. In fact, it would be interesting to adapt what Mukherjee et al. [2019] recently did but for the Wasserstein distance in order to see if their one-hot canonical latent augmentation clustering encoding circumevents our generative problems.

3 Discriminative Wasserstein Clustering

In this section, we do clustering with neural networks thanks to a discriminative (instead of generative) objective optimization: clusters distributions form mixture components like Fraley and Raftery [2002], Bouveyron et al. [2019] did in the past. The discriminative aspect comes from the fact that we are trying to optimally separate the data clusters in terms of Wasserstein distance (benefiting from the recent rising of deep learning scientific techniques since Arjovsky et al. [2017], Gulrajani et al. [2017], Miyato et al. [2018] but also Genevay [2019]) in a *one-versus-rest* fashion.

In a discriminative clustering, there is a fundamental limitation of Kullback-Leibler divergence techniques: distributions must share same (infinite) support. Indeed, if density supports are not the same, the Kullback-Leibler divergence is not defined (the logarithm of a zero probability being $-\infty$) so in these circumstances we can (must) use infinite support densities such as Gaussians or related to artificially separate clusters that have mathematically same support which is not natural: How come separated clusters share same model density supports? The unsatisfactory answer consists in having low density separation zones between them. From that perspective, Wasserstein distances are better because they are well-defined for non-equal density supports which constitutes its main advantage thanks to its geometric properties. Clusters can now mathematically have model densities without any overlap.

We propose an algorithm to perform unsupervised classification (a.k.a. clustering) within this framework that we call "DiWaC" for Discriminative Wasserstein Clustering. Using the idea of separating clusters as a discriminative objective function is not new (see for example Spectral Clustering [Von Luxburg, 2007] or DIFFRAC [Bach and Harchaoui, 2008]) but the fact that we handle distributions allows to enable both out-of-sample clustering and model selection at the same time which makes our technique appealing even in large scale settings.

This work, to the best of our knowledge, is the first that maximizes Wasserstein distances between clusters in a discriminative manner. The advantage of maximizing in our case is that the Kantorovich-Rubinstein formulation makes it an overall maximization which is good news in terms of programming and convergence ease compared to usual (and painful to monitor and debug) min-max-type of optimization in Generative Adversarial Networks.

DiWaC is built at the crossroads of the auto-encoders, generative adversarial networks, optimal transport and statistical mixture models literatures. We recall that an auto-encoder is a neural network made up of two parts: on the one hand, an encoder that transforms the initial data into a smaller code space followed by a decoder that sends the codes back to the initial data space by trying to reconstruct them approximately in the sense of a quadratic loss (for a conventional auto-encoder), a Kullback-Leibler divergence (for a variational auto-encoder [Kingma and Welling, 2013]) or the Wasserstein distance (for an adversarial auto-encoder [Tolstikhin et al., 2018]).

By presenting clustering in a discriminative fashion, we end up by defining a good data partitionning in clusters as one whose components are as far apart as possible from each other. Hence, mathematically, we maximize the weighted sum of Wasserstein's distances between each cluster components and all others. Thanks to Kantorovich's formulation of Wasserstein's distances, the optimization of this criterion is a maximization (without minimization) on probabilities and critics functions (also called potential in the optimal transport literature). Thus, we benefit from the algorithmic tools coming from adversarial neural networks, especially for the critics Lipschitzian functions [Miyato et al., 2018], without suffering from the hardship of an adversarial optimization (as there is only maximization and no minimization any more).

3.1 Wasserstein Distances between Clusters

Concretely, clustering is the task of gathering the data $\mathbf{x}$ in $K \geq 2, K \in \mathbb{N}$ well separated groups but here we relax the hard notion of group into soft memberships through:

$$\boldsymbol{\tau}(\mathbf{x}) = [\mathbb{P}(c = 1|\mathbf{x}), \dots \mathbb{P}(c = k|\mathbf{x}), \dots, \mathbb{P}(c = K|\mathbf{x})]^\top \tag{16}$$

The number K of groups can be found through model selection which is explained later (section 3.5)

Through clustering we infer a function to describe hidden structure from unlabeled data. We aim at clustering a dataset of N samples $\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N$ of the random variable $\mathbf{x}$ (of distribution p) living in a space $\mathcal{X}$ (say $\mathcal{X} = \mathbb{R}^D$) in K groups (or clusters). Usually, in the literature in Machine Learning (*a fortiori* including Deep Learning), scientists minimize probability divergences, namely: Kullback-Leibler divergence which is equivalent to maximizing the likelihood in many cases (with a wide range of tools from logistic regression i. e. cross-entropy loss for classification to Expectation-Maximization for clustering), the Jensen-Shannon divergence in Generative Adversarial Networks [Goodfellow et al., 2014] as remarked and extended to Wasserstein distance (which is thus also a divergence with nicer and geometric properties) by Arjovsky et al. [2017]. The originality of the current work in clustering settings where the grouping are unknown lies on the maximization of divergences between groups instead of the usual minimization of divergences. This makes our contribution close to Discriminative Latent Models [Bouveyron and Brunet-Saumard, 2014a], Fisher Expectation Maximization [Bouveyron and Brunet, 2012] and Spectral Clustering [Von Luxburg, 2007] and we took some ideas from this vein of research (especially about how we handle proportions to normalize our objectives as we will see). The Wasserstein distance is also a divergence and thus has better properties than the Kullback-Leibler which constitute the main motivation of this work: mainly symmetry and geometric interpretations such as the triangle inequality and being defined even when the compared distributions do not have the same support.

With this research background in mind, we choose to still benefit from the Wasserstein generative adversarial networks (WGAN) *without being adversarial*. Indeed, we just see WGAN as an algorithmic tool to manipulate Wasserstein distances for large scale datasets thanks to its neural networks stochastic optimization. Thus a probabilistic configuration is needed and we choose the one of mixture models which actually stood the test of time in several research milestones done by Machine Learning pioneers like Dempster et al. [1977], Lloyd [1982], Blei et al. [2003], Jain [2010]. Let us model our data as coming from a mixture distribution of K unknown bu separated components $p_1, \ldots, p_k, \ldots, p_K$ weighted by proportions $\boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_k, \cdots, \boldsymbol{\pi}_K$:

$$\mathbf{x} \sim p = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times p_k \tag{17}$$

which uses the classical data generation model of mixture models [Bouveyron et al., 2019]:

- Pick a cluster index k from the multinomial distribution of parameters $\boldsymbol{\pi}$
- Pick a data point $\mathbf{x}$ from the data component distribution p_k

Identifying the different components distributions p_ks (and the proportions $\boldsymbol{\pi}_k$s) is recasting clustering in a probabilistic manner. Based on the presented mixture model, we try to maximize the Wasserstein distances between the different p_ks themselves.

Meanwhile, thanks to the Bayes formula, we can get:

$$p_k(\mathbf{x}) = p(\mathbf{x}) \times \frac{\boldsymbol{\tau}_k(\mathbf{x})}{\boldsymbol{\pi}_k} \tag{18}$$

and similarly, we define:

$$\bar{p}_k(\mathbf{x}) = p(\mathbf{x}) \times \frac{1 - \boldsymbol{\tau}_k(\mathbf{x})}{1 - \boldsymbol{\pi}_k} \tag{19}$$

the distribution of the remaining points without the kth group. A reasonable objective appears to consist in simultaneously maximizing all inter-cluster Wasserstein distances $W(p_k, \bar{p}_k)$. Indeed, distributions p_ks with highly overlapping support correspond to low inter-cluster Wasserstein distances because each component corresponds to a cluster. In contrast, well separated distributions p_ks correspond to bigger inter-cluster Wasserstein distances $W(p_k, p_{k'})$ and $W(p_k, \bar{p}_k)$ (with $k \neq k'$).

3.2 Unormalized and Normalized sum of inter-Cluster Wasserstein Distances

We previously established that inter-cluster Wasserstein distances maximization for building a training objective could be interesting for clustering. In this part of the current work, we present two attempts: one simple sum and one weighted sum to combine the inter-cluster Wasserstein distances.

Definition 1. *Unormalized sum of inter-cluster Wasserstein distances*

$$\mathcal{L}_{\mathrm{u}}(p_1, \ldots, p_k, \ldots, p_K) = \frac{1}{K} \sum_{k=1}^{K} W(p_k, \bar{p}_k) \tag{20}$$

Definition 2. *Normalized sum of Wasserstein inter-cluster distances*

$$\mathcal{L}_{\mathrm{n}}(p_1, \ldots, p_k, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_k) \times W(p_k, \bar{p}_k) \tag{21}$$

In these two objectives definitions, we maximize over $p_1, \ldots, p_k, \ldots, p_K$ and $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K$ verifying:

- $\boldsymbol{\pi}_1 + \boldsymbol{\pi}_2 = 1$ with $\boldsymbol{\pi} \in \mathbb{R}_+^2$
- $p = \boldsymbol{\pi}_1 \times p_1 + \boldsymbol{\pi}_2 \times p_2$

3.2.1 Sanity Checks

Thanks to a theoretical example with an obvious clustering outcome one could wish, we now try to understand the interests of choosing one of the two defined objectives. In $\mathbb{R}^2$, let us take a data distribution made of Gaussian and Dirac components:

$$p = \frac{1}{\alpha + \beta + 1} \times (\alpha \mathcal{N}(\mathbf{0}_2, \sigma \times \mathbf{I}_2) + \beta \mathcal{N}(\mathbf{b}, \sigma \times \mathbf{I}_2) + \delta_{\mathbf{c}}) \tag{22}$$

with $\alpha = 10^4$, $\beta = 2 \times 10^4$, $\sigma = 10^{-3}$, $\mathbf{b} = [m, 0]^\top$ and $\mathbf{c} = [-M, 0]^\top$ ($m = 9$ and $M = 100$).

This *almost* Gaussian mixture case (corrupted by a down-weighted and far-located Dirac distribution), it is still interesting to analyze what would happen when clustering with $K = 2$ groups making Eq. (20) ending up with the maximization of $W(p_1, p_2)$ (p_1 and p_2 being unknown).

We consider two candidate clusterings to understand how compatible is the objective what we would expect as a good solution.

Example 1. Ideally, a satisfactory solution would be:

$$p_1^{\text{good}} = \frac{1}{\alpha + 1} \times (\alpha \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2) + \delta_{\mathbf{c}}) \text{ and } p_2^{\text{good}} = \mathcal{N}(\mathbf{b}, \mathbf{I}_2) \tag{23}$$

with proportions

$$\boldsymbol{\pi}_1^{\text{good}} = \frac{\alpha + 1}{\alpha + \beta + 1} \simeq 0.33 \text{ and } \boldsymbol{\pi}_2^{\text{good}} = \frac{\beta}{\alpha + \beta + 1} \simeq 0.66 \tag{24}$$

(switching the indices 1 and 2 does not break any generality). Indeed we believe that the oulier Dirac distribution $\delta_{\mathbf{c}}$ should be neglectable (thanks to the coefficients α and β being much bigger than one).

In other words, this would correspond to each cluster being associated with a single Gaussian beacuse the contribution of the Dirac in the mixture behaves like an outlier. Thus, we investigate here a rudimentary sanity check towards oultliers robustness.

Example 2. There is a bad clustering candidate that unfortunately gets a better score with respect to Eq. (20):

$$p_1^{\text{bad}} = \frac{1}{\alpha + \beta} \times (\alpha \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2) + \beta \mathcal{N}(\mathbf{b}, \mathbf{I}_2)) \text{ and } p_2^{\text{bad}} = \delta_{\mathbf{c}} \tag{25}$$

with proportions

$$\boldsymbol{\pi}_1^{\text{bad}} = \frac{\alpha + \beta}{\alpha + \beta + 1} \simeq 0.99 \text{ and } \boldsymbol{\pi}_2^{\text{bad}} = \frac{1}{\alpha + \beta + 1} \simeq 3.3 \times 10^{-5} \tag{26}$$

In the Normalized Spectral Clustering literature (well explained by Von Luxburg [2007]), a similar normalization is used for more robust clustering with respect to outliers.

Indeed for Eq. (20), the *good* and *bad* solutions give approximately (thanks to the chosen caricatural coefficients α, β, σ, m and M):

$$\mathcal{L}_{\mathrm{u}}(p_1^{\mathrm{good}}, p_2^{\mathrm{good}}) \simeq 9 \tag{27}$$

$$\mathcal{L}_{\mathrm{u}}(p_1^{\mathrm{bad}}, p_2^{\mathrm{bad}}) \simeq 100 \tag{28}$$

which selects the bad candidate. This prooves the sensitivity of the unormalized objective in $\mathcal{L}_{\mathrm{u}}$ with respect to the outlier Dirac distribution in Eq. (22).

In objectives represented in $\mathcal{L}_{\mathrm{u}}$ and $\mathcal{L}_{\mathrm{n}}$, the clusters are separated but $\mathcal{L}_{\mathrm{n}}$ avoids degenerate clustering candidates thanks to the $\boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_k)$ term. In our sanity check example, the previous bad clustering is smashed out by the normalization in $\mathcal{L}_{\mathrm{n}}$, the first cluster occupies more than $\boldsymbol{\pi}_1 \simeq 99\%$ of the data leaving the second singleton set with less than $\boldsymbol{\pi}_2 \simeq 0.003\%$) whereas the unormalized $\mathcal{L}_{\mathrm{u}}$ objective rewards a *bad* degenerate candidate solution. Indeed for the normalized objective $\mathcal{L}_{\mathrm{n}}$ gives approximately:

$$\mathcal{L}_{\mathrm{n}}(p_1^{\mathrm{good}}, p_2^{\mathrm{good}}, \boldsymbol{\pi}_1^{\mathrm{good}}, \boldsymbol{\pi}_2^{\mathrm{good}}) \simeq 3.92 \tag{29}$$

$$\mathcal{L}_{\mathrm{n}}(p_1^{\mathrm{bad}}, p_2^{\mathrm{bad}}, \boldsymbol{\pi}_1^{\mathrm{bad}}, \boldsymbol{\pi}_2^{\mathrm{bad}}) \simeq 5.9 \times 10^{-3} \tag{30}$$

selecting the expected good candidate clustering.

3.2.2 One-vs-One and One-vs-Rest Strategies

There is also a more principled way to look at our normalized objective Eq. (21) and the $\boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_k)$ term. In supervised classification (say logistic regression nicely explained by Hastie et al. [2005]), we have one-vs-one and one-vs-rest strategies that we use here. More concretely, in a one-vs-one strategy, it would be reasonable to consider the probability $\boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_{k'})$ of choosing one point from cluster k and the other point from cluster k' in an independent fashion.

Now we imagine two data generation models to add some theoretical justification to our objective Eq. (21). First, we can imagine a generation model:

1. Sample two clusters indices (k, k') independently from the multinomial distribution of parameters $\boldsymbol{\pi}$
2. Observe the Wasserstein distance $W(p_k, p_{k'})$ between the components/clusters associated with k and k' (if $k = k'$ the case is obvious since $W(p_k, p_k) = 0$)

The observed Wasserstein distance is a random variable whose mean is:

$$\mathcal{L}_{\mathrm{n}}^{\mathrm{OvO}}(p_1, \ldots, p_k, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K) = \sum_{k=1}^{K} \sum_{k'=1}^{K} \boldsymbol{\pi}_k \times \boldsymbol{\pi}_{k'} \times W(p_k, p_{k'}) \tag{31}$$

which is the mean one-vs-one inter-Wasserstein distance between clusters selected by a multinomial distribution of parameter $\boldsymbol{\pi}$.

The same reasoning can be done with a second and slightly different data generation model:

1. Sample two clusters indices (k, k') independently from the multinomial distribution of parameters $\boldsymbol{\pi}$
2. Observe the Wasserstein distance $W(p_k, \bar{p}_k)$ between the kth component and the rest

The observed Wasserstein distance is a random variable whose mean is now:

$$\mathcal{L}_{\mathrm{n}}^{\mathrm{OvR}}(p_1, \ldots, p_k, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_k) \times W(p_k, \bar{p}_k) \tag{32}$$

which is the mean one-vs-rest inter-Wasserstein distance between clusters selected by a multinomial distribution of parameter $\boldsymbol{\pi}$. This one-vs-rest approach is preferred rather than the one-vs-one for combinatorial reasons as K grows. One can notice that the previous normalized Wasserstin sum definition is equal to that one-vs-one point of view definition $\mathcal{L}_{\mathrm{n}}^{\mathrm{OvR}} = \mathcal{L}_{\mathrm{n}}$ but presented in a different fashion.

Decoupling the compared distributions in the Wasserstein distance computation is easier with the euclidean ℓ_2 distance. Indeed, the Kantorovich-Rubinstein [Dudley, 2018] as it is already used in WGAN [Arjovsky et al., 2017] tells us a decoupled re-definition:

$$W(\boldsymbol{\mu}, \boldsymbol{\nu}) \overset{\text{KR}}{=} \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim \boldsymbol{\mu}} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim \boldsymbol{\nu}} \mathcal{C}(\mathbf{y}) \tag{33}$$

where Lip_1 is the 1-Lipschitz functions set. Thus, we can write that for three distributions q_1, q_2, q_3 and proportions κ_1, κ_2 (with $\kappa_1 + \kappa_2 = 1$):

$$W(\kappa_1 \times q_1 + \kappa_2 \times q_2, q_3) = \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim \kappa_1 \times q_1 + \kappa_2 \times q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \tag{34}$$

but

$$\begin{aligned} \mathbb{E}_{\mathbf{x} \sim \kappa_1 \times q_1 + \kappa_2 \times q_2} \mathcal{C}(\mathbf{x}) &= \int_{\mathbb{R}^D} \mathcal{C}(\mathbf{x}) \left(\kappa_1 \times q_1(\mathbf{x}) + \kappa_2 \times q_2(\mathbf{x})\right) d\mathbf{x} & (35) \\ &= \kappa_1 \times \int_{\mathbb{R}^D} \mathcal{C}(\mathbf{x}) \times q_1(\mathbf{x}) d\mathbf{x} + \kappa_2 \times \int_{\mathbb{R}^D} \mathcal{C}(\mathbf{x}) \times q_2(\mathbf{x}) d\mathbf{x} \\ &= \kappa_1 \times \mathbb{E}_{\mathbf{x} \sim q_1} \mathcal{C}(\mathbf{x}) + \kappa_2 \times \mathbb{E}_{\mathbf{x} \sim q_2} \mathcal{C}(\mathbf{x}) & (36) \end{aligned}$$

and

$$\mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) = (\kappa_1 + \kappa_2) \times \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \tag{37}$$

thus

$$\begin{aligned} W(\kappa_1 \times q_1 + \kappa_2 \times q_2, q_3) &= \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim \kappa_1 \times q_1 + \kappa_2 \times q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) & (38) \\ &= \sup_{\mathcal{C} \in \text{Lip-1}} \kappa_1 \times \left(\mathbb{E}_{\mathbf{x} \sim q_1} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y})\right) \\ &\quad + \kappa_2 \times \left(\mathbb{E}_{\mathbf{x} \sim q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y})\right) \\ &\leq \kappa_1 \times \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim q_1} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \\ &\quad + \kappa_2 \times \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \end{aligned}$$

because the maximum of a sum is lower or equal than the sum of each term maxima (which is also true with supremum instead of maximum) which gives

$$W(\kappa_1 \times q_1 + \kappa_2 \times q_2, q_3) \leq \kappa_1 \times W(q_1, q_3) + \kappa_2 \times W(q_2, q_3) \tag{39}$$

and finally implies that

$$\mathcal{L}_{\text{n}}^{\text{OvR}}(p_1, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K) \leq \mathcal{L}_{\text{n}}^{\text{OvO}}(p_1, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K) \tag{40}$$

and thus maximizing the lower complexity one-vs-rest objective is maximizing a lower bound of the one-vs-one objective which is usual in Machine Learning. In this section, we tried to provide some theoretical justification for our training objective and it is time to tackle the deep learning angle of it.

3.3 Estimating Wasserstein Distances with Deep Learning

In terms of neural network optimizatin, if we sum up everything we described, membership probabilities $\boldsymbol{\tau}_k(\mathbf{x}) = \mathbb{P}(c = k|\mathbf{x})$ are the unknown function outputs of our clustering formulation which can be represented thanks to a positive function f chained with a sum-1 normalization layer:

$$\forall k \in [\![1, K]\!] \ \boldsymbol{\tau}_k(\mathbf{x}) = \frac{f_k(\mathbf{x})}{\sum_{\ell=1}^{K} f_\ell(\mathbf{x})} \tag{41}$$

On the Kantorovich side, the critics $\mathcal{C}_k$ are also neural networks (of parameters $\theta_{\mathcal{C}_k}$) but with a 1-Lipschitz property gracefuly provided by spectral normalization for each linear (or convolutional)

non-activation layers thanks to a simple yet efficient power iteration technique developped recently by Miyato et al. [2018] which gives enough stability to face large scale datasets in the work of Brock et al. [2019]. Injecting all these formulas in $\mathcal{L}_{\text{n}} = \mathcal{L}_{\text{n}}^{\text{OvR}}$, the maximization over all parameters θ (concatenating the θ_{f_k}s and the $\theta_{\mathcal{C}_k}$s) becomes (in several lines):

$$\mathcal{L}_{\text{n}}^{\text{OvR}}(p_1, \dots, p_K, \boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_K) = \mathbb{E}_{\mathbf{x}\sim p}\left[\sum_{k=1}^{K}\Big(\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k\Big) \times \mathcal{C}_k(\mathbf{x})\right] \tag{42}$$

while proportions $\boldsymbol{\pi}_k$ are equal to $\mathbb{E}_{\mathbf{x}\sim p}\Big(\mathbb{P}(c = k|\mathbf{x})\Big)$ which is kept and maintained *à la* online k-Means [Bottou and Bengio, 1995] in their means updates (which can be seen like an Optimal Control [Bertsekas et al., 1995] self-regulated closed loop mechanism).

Suppose that an oracle gave us the *optimal* $(\boldsymbol{\pi}^*, \theta_{\mathcal{E}}^*, \theta_{\mathcal{M}}^*)$ and we keep them fixed, does adding a constant of the $\mathcal{C}_k$s critics functions could diverge to $\pm\infty$? The answer is no because, by definition of the proportions in the mixture, we know that $\mathbb{E}_{\mathbf{x}\sim p}\left[\boldsymbol{\tau}_k(\mathbf{x})\right] = \boldsymbol{\pi}_k$ which means that the sign of $\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k$ cannot be constant (it can be always zero but that is an easy case for what we need to prove). Thus, adding a constant b_k to $\mathcal{C}_k$ will not change the objective. (and adding a constant multiplier a_k to a valid 1-Lipschitz $\mathcal{C}_k$ would violate that 1-Lipschitz constraint).

More generally, the fact that we maintain the null equalities

$$\forall k \in [\![1, K]\!]\ \mathbb{E}_{\mathbf{x}\sim p}\left[\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k\right] = 0 \tag{43}$$

(thanks to Bottou and Bengio [1995]) and the Lipschitz property together prevent the critics $\mathcal{C}_k$s f rom diverging;

The objective of Eq. (42) favors $\boldsymbol{\tau}_k(\mathbf{x})$ to be far from $\boldsymbol{\pi}_k$ thanks to the maximization: (i) if cluster memberships $\boldsymbol{\tau}_k(\mathbf{x})$ are too close to their means $\boldsymbol{\pi}_k$, then the objective would be close to zero (its lower bound because a positive linear combination of Wasserstein distances Eq. (42) is non-negative); (ii) when $\boldsymbol{\tau}_k(\mathbf{x})$ is high above its mean $\boldsymbol{\pi}_k$ (bounded by 1), $\mathcal{C}_k(\mathbf{x})$ will be high and when $\boldsymbol{\tau}_k(\mathbf{x})$ is low under its mean (bounded by 0), $\mathcal{C}_k(\mathbf{x})$ will be low too. Accordingly, it is interesting to see that $\mathcal{C}_k(\mathbf{x})$ can be seen as a relaxed decision function (i. e. high for points in k^{th} cluster and low for the other clusters, but bounded in terms of variation due to its Lipschitzian property);

Overlapping Gaussian components of mixture $\mathcal{M}$ are avoided. Intuitively, if we take a region of the data space where $\boldsymbol{\tau}_k(\mathbf{x})$ and $\boldsymbol{\tau}_\ell(\mathbf{x})$ (with $k \neq \ell$) are high (meaning $\mathcal{E}(\mathbf{x})$ is on an overlapping zone between two Gaussian components k and ℓ), then the Wasserstein distances $W(p_k, \bar{p}_k)$ and $W(p_\ell, \bar{p}_\ell)$ (parts of the objective sum) could have been even more maximized on this region because they are related to $W(p_k, p_\ell)$. Thus our algorithm favors partitions over covers (in spite of our soft relaxation of memberships).

3.4 Algorithm

Algorithm 4 optimizes Eq. (42) which has only a maximization steps which provides a certain engineering ease compared to the min-max optimization scheme required for the GANs. Nevertheless, initialization routines turned out to be crucial in practice. For this highly non-convex optimization, suggest three successive careful initialization steps before the real opitmization.

The optimization of Eq. (42) that we want to accomplish has no guarantee to converge in a global extrema with our neural networks approach, which is why careful initialization is crucial. That is why our DiWaC algorithm can be decomposed in four successive steps (three for intializations and one for the real optimization). In theory, if we had a universal (i. e. regardless of the convexity) ideal optimizer at our disposal, then the three first steps would be useless but to avoid useless and spurious local minima, we do a three steps initialization for a final one.

1. **Auto-encoder initialization** We train a classic auto-encoder $(\mathcal{E}, \mathcal{D})$ (for an encoder $\mathcal{E}$ and decoder $\mathcal{D}$ parametrized respectively by $\theta_{\mathcal{E}}$ and $\theta_{\mathcal{D}}$):

$$(\theta_{\mathcal{E}}^0, \theta_{\mathcal{D}}^0) = \arg\min_{\theta_{\mathcal{E}}, \theta_{\mathcal{D}}} \mathbb{E}_{\mathbf{x}\sim p}\|\mathcal{D} \circ \mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2^2 \tag{44}$$

   using an auto-encoder initialization is interestingly reminiscent to 1990s ans 2000s pretraining for neural networks at a time when training time was a much more a burden than today (although Goodfellow et al. [2016] also mention better activation functions, weight initialization, variants of gradient descent and regularization methods, we empirically observed that pretraining is still benefitial for many scenarios both in terms of performance and sometimes even overall training time).

2. **EM-based Gaussian mixture codes initialization** We fit a Gaussian mixture model $\mathcal{M}$ of proportions $\boldsymbol{\omega}$, means $\boldsymbol{\mu}$ and covariance matrices $\mathbf{S}$ with the Expectation-Maximization algorithm [Dempster et al., 1977] on the encoded data $\mathcal{E}(\mathbf{x})$:

$$\theta^0_{\mathcal{M}} = \arg\max_{\theta_{\mathcal{M}}} \mathbb{E}_{\mathbf{x}\sim p}\left[\log\left(\sum_{k=1}^{K} \boldsymbol{\omega}_k \times \mathcal{N}(\mathcal{E}^0(\mathbf{x}); \boldsymbol{\mu}_k, \mathbf{S}_k \times \mathbf{S}_k^\top)\right)\right] \tag{45}$$

where $\theta_{\mathcal{M}} = (\boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \mathbf{S}_k)_{k=1,\dots,K}$. We also precise that this EM-GMM optimization is itself initialized with k-Means++ [Arthur and Vassilvitskii, 2006][4] on the codes (or encoded inputs that lie at the bottleneck of the previous step's autoencoder).

3. **Critics initialization** We define K critics functions $(\mathcal{C}_k)_{k=1,\dots,K}$ implemented thanks to Miyato et al. [2018] and parametrized by $\theta_{\mathcal{C}}$ that would estimate the Wasserstein distances weighted sum with memberships probabilities provided by the previous step's EM. Indeed, we define clustering probability functions $\boldsymbol{\tau}_k(\mathbf{x}) = \frac{\boldsymbol{\omega}_k \times \mathcal{N}(\mathcal{E}(\mathbf{x}); \boldsymbol{\mu}_k, \mathbf{S}_k)}{\sum_{\ell=1}^{L} \boldsymbol{\omega}_\ell \times \mathcal{N}(\mathcal{E}(\mathbf{x}); \boldsymbol{\mu}_\ell, \mathbf{S}_\ell)}$ to maximize over $\theta_{\mathcal{C}}$:

$$\theta^0_{\mathcal{C}} = \arg\max_{\theta_{\mathcal{C}}} \mathbb{E}_{\mathbf{x}\sim p}\left[\sum_{k=1}^{K} \Big(\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k\Big) \times \mathcal{C}_k(\mathbf{x})\right] \tag{46}$$

Indeed, we recall that the critics functions $\mathcal{C}_k$s are just a convenient tool to estimate the Wasserstein distances. With bad critics, the gradient of the objective over the memberships probabilities parameters would be also wrong which is bad news especially when memberships functions are previously and nicely initialized. This suggests this warmup step 3 before the real clustering optimization.

Without that *warm-up* step 3 (right before the *real* step 4), we would deteriorate the previous clustering initializations quality provided by steps 1 and 2 (that is dubbed the "AE + GMM" initialization by Xie et al. [2015]) because the critics would not be trained enough to estimate that good clustering while still inflicting *ignorant* gradient steps on the clustering functions. To avoid such a bad scenario after steps 1 and 2, the last initialization step 3 corresponds to a *warm-up* before the real optimization step 4. This way, thrice the encoder, the mixture and the critics are reasonably well initialized and the real core step 4 can begin where all three are not free but just nicely initialized, relaxed and further optimized.

4. **Core clustering** With the same objective, we do the core, final and well-initialized optimization:

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{\mathbf{x}\sim p}\left[\sum_{k=1}^{K} \Big(\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k\Big) \times \mathcal{C}_k(\mathbf{x})\right] \tag{47}$$

as described in Algorithm 4.

3.5 Model Selection for DiWaC

The goal of model section is to check under/over-fitting and ultimately choose hyper-parameters such as the number of classes or the architecture of neurons and layers among several trained models with different hyper-parameters. In this work, our inter-cluster Wasserstein distance measured in Eq. (42) can be measured on some held-out validation data distributions. As stated earlier, critics are just a convenient tool to estimate the Wasserstein distances, so there are at least two ways to estimate the validation objective:

1. Directly (D) "as is": we simply apply Eq. (42) except for adapted proportions $\boldsymbol{\pi}$ measured on these held-out data:

$$\boldsymbol{\pi}_{\text{validation}} \leftarrow \frac{1}{N_{\text{validation}}} \sum_{i'=1}^{N_{\text{validation}}} \boldsymbol{\tau}(\mathbf{x}_{i'}) \tag{48}$$

2. Re-Fitted (R) "with re-optimized critics": we use the same previously adapted proportions Eq. (48) and we optimize again the objective Eq. (42) but only with respect to the critics which corresponds to step 4, Eq. (46). The goal is to refine the objective which corresponds to a separation power of the clustering in terms of interpretation.

[4] EM-GMM and k-Means++ provided by scikit-learn [Pedregosa et al., 2011]

Algorithm 4 Optimization algorithm (step 4)

1: **Input:** Data

$$(\mathbf{x}_i)_{i=1,\dots,N}$$

Number of clusters K

2: **Initialization:**

$$\theta_{\mathcal{E}} \text{ and } \theta_{\mathcal{M}} = (\boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \mathbf{S}_k)_{k=1,\dots,K}\text{, \# initialized from steps 1, 2 and 3}$$

3:

$$\boldsymbol{\pi} \leftarrow \boldsymbol{\omega} \text{ \# both } \boldsymbol{\pi} \text{ and } \boldsymbol{\omega}$$

are (re)-parametrized thanks to a softmax on free paramaters

4: $T \leftarrow 0$

5: **while** θ has not converged **do**

6: $T \leftarrow T + 1$

7: Free all gradients accumulators

8: Sample a mini-batch of size B from the dataset

$$(\mathbf{x}_{i_b})_{b=1,\dots,B} \text{ where } i_b \sim \mathcal{U}_{\mathbb{N}}(1, N)$$

9: Compute

$$g_{i_b k} \leftarrow \frac{\boldsymbol{\omega}_k \times \mathcal{N}(\mathcal{E}(\mathbf{x}_{i_b}); \boldsymbol{\mu}_k, \mathbf{S}_k)}{\sum_{\ell=1}^{K} \boldsymbol{\omega}_\ell \times \mathcal{N}(\mathcal{E}(\mathbf{x}_{i_b}); \boldsymbol{\mu}_\ell, \mathbf{S}_\ell)}$$

10: so that

$$\boldsymbol{\tau}_{i_b k} \leftarrow \frac{g_{i_b k}}{\sum_{\ell=1}^{K} g_{i_b \ell}}$$

11: Perform a gradient ascent step of $\frac{1}{B}\sum_{b=1}^{B}\sum_{k=1}^{K}(\boldsymbol{\tau}_{i_b k} - \boldsymbol{\pi}_k) \times \mathcal{C}_k(\mathbf{x}_{i_b})$ to update θ

12: Update proportions *à la* online k-means [Bottou and Bengio, 1995]

$$\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} + \frac{1}{T+1} \times \left(\left(\frac{1}{B} \sum_{b=1}^{B} \boldsymbol{\tau}_{i_b} \right) - \boldsymbol{\pi} \right)$$

13: **end while**

Traditionally, model selection is accomplished according to the likelihood (or completed likelihood) related to the Kullback-Leibler divergence (see the Bayesian Information Criterion BIC [Schwarz et al., 1978] or Information Completed Likelihood ICL [Biernacki et al., 2000] for examples of *extrapolated generalization power measurements*). 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 to maybe open new avenues of research for future investigations (e. g. adapting BIC and/or ICL beyond likelihood and Kullback-Leibler divergence).

In practice here, we train M times our model with different set of hyper-parameters (different number of clusters, different neural networks structure, different mixture components structure etc.) Models $(\hat{\theta}_m)_{m=1,\dots,M}$ are now evaluated on held-out validation data this time instead of training data (in which they are trained as usual). Finally, we select the best set of parameters: the model indexed by m for which the objective (defined in Eq. (42) with adapted proportions) is maximum.

3.6 Changing the Metric beyond the Euclidean Distance for DiWaC

In terms of ill-posed problem, the axioms of Kleinberg [2003] are *almost* satisfied: scale-invariance is valid, cluster-shapes-invariance is almost valid up to the expression power of the encoder and critic functions (a. k. a. the capacity of these neural networks in practice) that can be improved thanks to model selection and only metric-invariance remains but we have the intuition that it can be improved in future works thanks to the notion of *worst metric* among a large class of metrics as briefly evoked in the next chapter. Indeed, at the beginning of that clustering chapter, we mentioned that our clustering algorithm could not achieve *consistency* (i. e. metric invariance) because optimal transport required an initial and definitive commitment for the unique distance choice in the data space to build a Wasserstein distance for distributions over the data space. In fact, we could apply some ideas of the next contribution of this thesis to handle a *very large* class of distances at once which is possibly better than only one euclidean distance. Of course, according to the previously cited clustering no-free lunch theorem [Kleinberg, 2003][5], we will never be able to cover *all possible* distances. Nevertheless, we could contemplate a solution where our hereby euclidean distance clustering is an initialization for a newer and better clustering technique for a large class of distances simulataneously in the future towards more consistency with the work of Kleinberg [2003] in mind.

4 Experiments

4.1 Implementation details and experimental setup

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} with the Adam default optimization strategy [Kingma and Ba, 2014] everywhere. k-Means and GMM are not easily compatible with large scale datasets which is why we took only a reasonable subset of large datasets for these initializations (which is why this is not crucial). In fact, the online learning of k-Means [Bottou and Bengio, 1995] is also possible for EM thanks to Cappé and Moulines [2009] but we found our results already satisfying. In our preliminary experiments, the "AE + GMM" baseline (just the first 2 steps of our algorithm) performed poorly (and with bad reliability not reproducibility) without k-Means++ [Arthur and Vassilvitskii, 2006] and Xavier neural networks weights initialization [Glorot and Bengio, 2010]. Thus, all the experiments we report in this work use them.

For optimization reasons (unconstrained or implicitly constrained optimization is more stable than explicitly constrained optimization especially for stochastic gradient descent), we use two tricks:

- **the SoftMax trick** proportions are paremetrized by free logits that are converted into proportions through a SoftMax function[6];
- **the Cholesky trick** each covariance matrix is parametrized by its *square root* which always exists in the Cholesky decomposition sense for any covariance matrix (because it must be symmetric definite and positive, see [Press et al., 2007]) and is a lower-triangular matrix whose diagonal coefficients are strictly positive (which can be ensured thanks to the use of the exponential function) to guarantee that when multiplied by its transposed version we get the correct covariance properties throughout the optimization path.

  Unfortunately, we empirically realized that this kind of parametrization is not enough because of the eigenvalues behavior of that *square root* matrix: they numerically explode or implode resulting in ill-conditioned corresponding covariance matrices and create instability. More precisely, it appeared that when the means are far from the optimal means, the system chooses to modify its covariance matrices first instead of the means which should be prevented because of the spurious maxima. Limiting the eigenvalues range is better than an unbounded exponential function. We use a distorted Sigmoid[7] function bounded by two constants initially parametrized by the variable-wise standard deviations σ_j of all initial codes variables $\mathcal{E}^0(\mathbf{x}_i)_j$:

$$(\forall t \in \mathbb{R}) \;\; \lambda + (\Lambda - \lambda) \times \text{Sigmoid}(t) \in]\lambda, \Lambda[\tag{49}$$

[5]The "clustering no-free lunch theorem" is not an official nickname but we choose it because it helps understanding the work of Kleinberg [2003]. The original authors would rather use the clustering "impossibility theorem".

[6]$\text{SoftMax}(\mathbf{v})_k = \frac{\exp(\mathbf{v}_k)}{\sum_{k'=1}^{K} \exp(\mathbf{v}_{k'})}$

[7]$\text{Sigmoid}(t) = \frac{1}{1+\exp(-t)}$

where:

$$\lambda = 0.3 \times \min_{j \in \{1,\dots,d\}} \sigma_j \text{ and } \Lambda = 3 \times \max_{j \in \{1,\dots,d\}} \sigma_j \tag{50}$$

to be read with $m_j = \frac{1}{N}\sum_{i=1}^{N} \mathcal{E}^0(\mathbf{x}_i)^j$ and $\sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathcal{E}^0(\mathbf{x}_i)^j - m_j)^2}$.

In supervised classification, for a known labeled dataset (but hidden to the trained system), the confusion matrix (a. k. a. matching matrix, contingency matrix or table of confusion) is counting for every pair (actual class k, predicted class ℓ) the number of occurences of points falling into that pair configuration (being in group k and category ℓ). Thus, that table can be an interpretation and visualization tool after a supervised classification or an unsupervised clustering (with ground truth) on data.

From the confusion matrix $\mathbf{M}$ between a given clustering (indexed by the rows k) and a revealed ground truth (indexed by the columns ℓ), several accuracy measures can be built afterwards:

Hungarian Accuracy (ACC) Thanks to the Hungarian Method [Kuhn, 1955, Stephens, 2000] run on the confusion matrix (while taking care of the sign thanks to a minus sign and adding the maximum entry because the Hungarian Method minimizes a sum and we want to maximize that a sum of occurences), we can measure how good is a clustering

$$\text{ACC } = \max_{\sigma} \sum_{k=1}^{K} \mathbf{M}_{k,\sigma(k)} \iff \min_{\sigma} \sum_{k=1}^{K} \mathbf{C}_{k,\sigma(k)} \tag{51}$$

where the cost matrix $\mathbf{C}$ is adapted with these entries $\mathbf{C}_{k,\ell} = \left(\max_{k',\ell'} \mathbf{M}_{k',\ell'}\right) - \mathbf{M}_{k,\ell}$ and an optimal K-permutation σ tells which cluster $\sigma(k)$ should cluster k be assigned to.

Normalized Mutual Information (NMI) Measuring an estimator of dependence normalized by the entropy information contained in both signals is also a classic tool for clustering evaluation but this time evaluated on the divided confusion matrix $\tilde{\mathbf{M}} = \frac{\mathbf{M}}{\sum_{k,\ell} \mathbf{M}_{k,\ell}}$

$$\text{NMI } = \frac{\text{MI}}{\text{H}} \tag{52}$$

where

$$\text{MI} = \sum_{k,\ell} \tilde{\mathbf{M}}_{k,\ell} \log\left(\frac{\tilde{\mathbf{M}}_{k,\ell}}{\tilde{\mathbf{M}}_{k,\bullet} \times \tilde{\mathbf{M}}_{\bullet,\ell}}\right) \tag{53}$$

and

$$\text{H} = -\sum_{k,\ell} \tilde{\mathbf{M}}_{k,\ell} \log\left(\tilde{\mathbf{M}}_{k,\ell}\right) \tag{54}$$

and $\tilde{\mathbf{M}}_{k,\bullet} = \sum_{\ell} \tilde{\mathbf{M}}_{k,\ell}$ and $\tilde{\mathbf{M}}_{\bullet,\ell} = \sum_{k} \tilde{\mathbf{M}}_{k,\ell}$

Normalized Conditional Entropy (NCE) In the case where we need to over-cluster (a. k. a. over-segment the dataset) which is predicting more groups than ground truth semantic ones, it is interesting to see if the several small predicted clusters fit the fewer and bigger semantic *human* ones. In this case, normalized conditional entropy NCE seems appropriate:

$$\text{NCE } = \frac{\text{CE}}{\text{H}} \tag{55}$$

where the conditional entropy CE is:

$$\text{CE} = \sum_{k,\ell} \tilde{\mathbf{M}}_{k,\ell} \log\left(\frac{\tilde{\mathbf{M}}_{k,\ell}}{\tilde{\mathbf{M}}_{k,\bullet}}\right) \tag{56}$$

which makes NCE a kind of an asymetric version of NMI.

4.2 Introductory Examples for Generative Wasserstein Clustering

The probabilistic perspective tackled by Wasserstein-GAN in Arjovsky et al. [2017] approach can inspire one to think of the Kullback-Leibler alternative with the famous Expectation-Maximization algorithm for Gaussian mixture models (EM-GMM) [Dempster et al., 1977]. Let's try to solve the same problem as EM-GMM by using the Wasserstein distance with Wasserstein-GAN instead of the Kullback-Leibler divergence (or likelihood up to a sign and a constant term) with EM: take two Gaussian mixtures A and B both of $K = 3$ components each in a D-dimensional space ($D = 2$) defined by:

- K proportions $(\boldsymbol{\pi}_k^A)_{k=1,\dots,K}$ that are positive and sum to one for A and K proportions $(\boldsymbol{\pi}_k^B)_{k=1,\dots,K}$ with same properties;
- K means $(\boldsymbol{\mu}_k^A)_{k=1,\dots,K}$ for A and K means $(\boldsymbol{\mu}_k^B)_{k=1,\dots,K}$ for B where the means live in the same 2D space $\mathbb{R}^D$;
- K full covariance matrices $(\boldsymbol{\Sigma}_k^A)_{k=1,\dots,K}$ that are symmetric definite positive in $\mathbb{R}^{D\times D}$ for A and K full covariances $(\boldsymbol{\Sigma}_k^B)_{k=1,\dots,K}$ with same properties for B.

Now for good convergence at step 4, we decompose the successive minimizations and maximizations by just focusing on the maximization steps of Wasserstein-GANs in a *warm-up* phase. Indeed, in section 3.4, after steps 1, 2 and 3, we believe that the decoder and the mixture are well-initialized but the critic neural network is not. It means that our Wasserstein distance estimator (provided by the critic neural network) is not good and neither its gradient with respect to the decoder and the mixture. Concretely, this means that even though the generator (*i.e.* mixture and decoder) is well initialized, the first iterations will decrease the initial generation quality because of a poorly initialized critic that computes our Wasserstein distance value and gradients.

To cope with this bad critic initialization problem at step 4, we choose to optimize the critic alone which is estimating the Wasserstein distance between 2 fixed distributions (real and generated data do not change but the critic does) at the beginning of that step 4. This *warm-up* step concludes a nice initialization for every parameter as you can see in the toy example represented in Fig. 3 and the step 4 finally converges into Fig. 4. Of course, this warm-up step initially developed without generator neural network is also used in our case of WAMiC with the mixture and the decoder as the two components of the generator: first, we do not change the generator and we only change the critic neural network until convergence and then we trigger a second step after this warm-up and the actual min-max Wasserstein GAN optimization scheme takes place.

To illustrate our approach further, we now work on a toy dataset that we call "Three Moons" with 1000 points for each of the 3 groups in 2 dimensions as presented in Fig. 6. 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 in 3 groups 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) in step 3 of section 3.3 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 first 3 steps algorithm out of 4, 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. 5 actually selecting 3 clusters which is satisfactory.

4.3 Introductory Examples for Discriminative Wasserstein Clustering

To investigate model selection capabilities of our method and for explanatory reasons, we use here 2 synthetic datasets:

Three Moons 3M To illustrate our approach further, we now work on a toy dataset that we call "Three Moons" with 1000 points for each of the 3 groups (with a total of $N = 3000$ points) in dimensions $D = 2$ as presented in Fig. 6;

Various 2D Distributions VD Various distributions namely: 2 moons, 1 Gaussian, 1 non-isotropic Gaussian and 1 Student with different proportions (1000, 1500, 2000, 2500 and 3000 cardinalities respectively and a total of $N = 10000$ points) in dimension $D = 2$ in Fig. 7.

First, as described in section 3.4, we trained an auto-encoder on the 3M dataset with a code space dimension of $d = 1$: even though the clusters are not linearly separable in the original data space, our system is able to cluster them in 3 groups with 100% of accuracy (an EM-GMM or a simpler k-Means run in the codes data is enough!). 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 perfectly separated along one axis with respect to the cluster labels. Here, the point is not to check clustering capabilities but to see if model selection is plausible in this simple scenario.

Datasets	MNIST	Reuters	Reuters-10k	HHAR
DiWaC (ours)	**98.42**	**84.24**	**84.87**	**92.42**
GeWaC (ours with fixed proportions from AE+GMM)	**97.37**	**82.14**	**82.27**	**87.54**
ClusterGAN [Mukherjee et al., 2019]	90.97	–	–	–
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
k-Means	53.47	53.29	54.04	59.98

Table 1: Experimental accuracy results (%, the higher, the better) based on the Hungarian method. (the last rows correspond to methods without neural networks)

Now we study the case where that number of clusters K is not known in order to check if model selection is possible in these relatively easy settings. The intuition behind our validation procedure for model selection is that if a given hyper-parameter configuration (the number of clusters in this experiment) does not lead to under/over-fitting, our objective function will still be satisfactory on a validation set (i. e. that was not seen during training) compared to other configurations with one held-out validation set.

In the little more challenging (but still synthetic) VD dataset in Fig. 7, we observe that dealing with various kinds of distributions (even unknown by the system and beyond the Gaussian case) is still compatible with our mixture-type model-based clustering algorithm.

4.4 Real Data Experiments

For the real-world experiments, we were interested in 4 datasets with 3 different in nature (images, sparse and dense data) but all with several hundreds of dimensions per item:

- MNIST: 70 000 handwritten digits images dataset living in dimension 784 (for 28×28 pixels);
- 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;
- Reuters-10k: a random subset of Reuters with only 10 000 examples (selected with precisely the same random generator seed as DEC);
- HHAR: The Heterogeneity Human Activity Recognition (HHAR) dataset Stisen et al. [2015] contains 10,299 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.

On MNIST, in Fig. (8), we generated data from our GeWaC 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.

In these experiments, we used the same encoder (with symmetric decoder) MLP[8] structure from Xie et al. [2015] D-500-500-2000-d (D is the dimensionality of the input space e. g. $D = 784$ for MNIST and $d = 10$ is the dimensionality of the code space) (and ReLU activations) for fair comparisons with others. In fact, our conditions are tougher than the ones of IMSAT, GAR and DEPICT that use more sophisticated convolutions than matrix-vector products while DEC, VaDE and our DiWaC do not which is very encouraging.

The overall results of our DiWaC 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 ("AE + GMM" compared to GMM and even k-Means) when fed with the output of an AE which is

[8]MLP stands for Multi-Layer Perceptron

coherent with the results of Xie et al. [2015]. We must admit how surprised we are that this "AE + GMM" baseline works so well on various datasets. Undiscovered mathematical theories could tackle this phenomenon: *Why would a simple auto-encoder consistently map data in separated Gaussian groups?*. Maybe the auto-encoders layers loosely behave like successive *cluster-wise deterministic* random projections [Bingham and Mannila, 2001] and non-linear functions: indeed, the linear layers matrix entries have Gaussian frequencies but this would require further investigations to get thorough scientific interpretations. On a more practical level, we are particularly interested in real-world industrial cases where we highly recommend this "AE + GMM" baseline for its ease of implementation and good results in practice and also for fast prototyping.

There is a supplementary and significant improvement for DiWaC over its "AE + GMM" initialization. These good results are even comparable to those of supervised non-convolutional networks with just a few layers of the 1990s[9]. On MNIST, the important performance gap between VaDE and DiWaC can be explained by difficulties (already studied in [Jiang et al., 2016]) with data that lie extremely close to low-dimensional manifolds, like images. In that regard, our algorithm inherits the strenghts of WGAN [Arjovsky et al., 2017] and models data much more faithfully without our DiWaC algorithm being adversarial which means that Optimal Transport alone is a powerful tool as the non-adversarial work of Genevay et al. [2017] also tends to proove. Furthermore, as presented previously (section 3.5), our strategy has a natural criterion for model selection whereas VaDE has no principled model selection criterion; in theory, the validation likelihood could be used, however, computing the likelihood of for variational techniques remains an open question (see e.g. [Cremer et al., 2018]). In other works, DEC's authors propose to use the ratio between training and validation loss as a criterion, however this *ad hoc* solution has little statistical foundations. Model selection is definetely the main feature of our approach on top of providing good results at same hyper-parameter configurations. In fact, better neural networks structure can be found by cross-validation with our algorithm which could be done in future work.

[9] see `http://yann.lecun.com/exdb/mnist` for a complete supervised MNIST benchmark although they train with supervised labels whereas we do not have this supervised information.

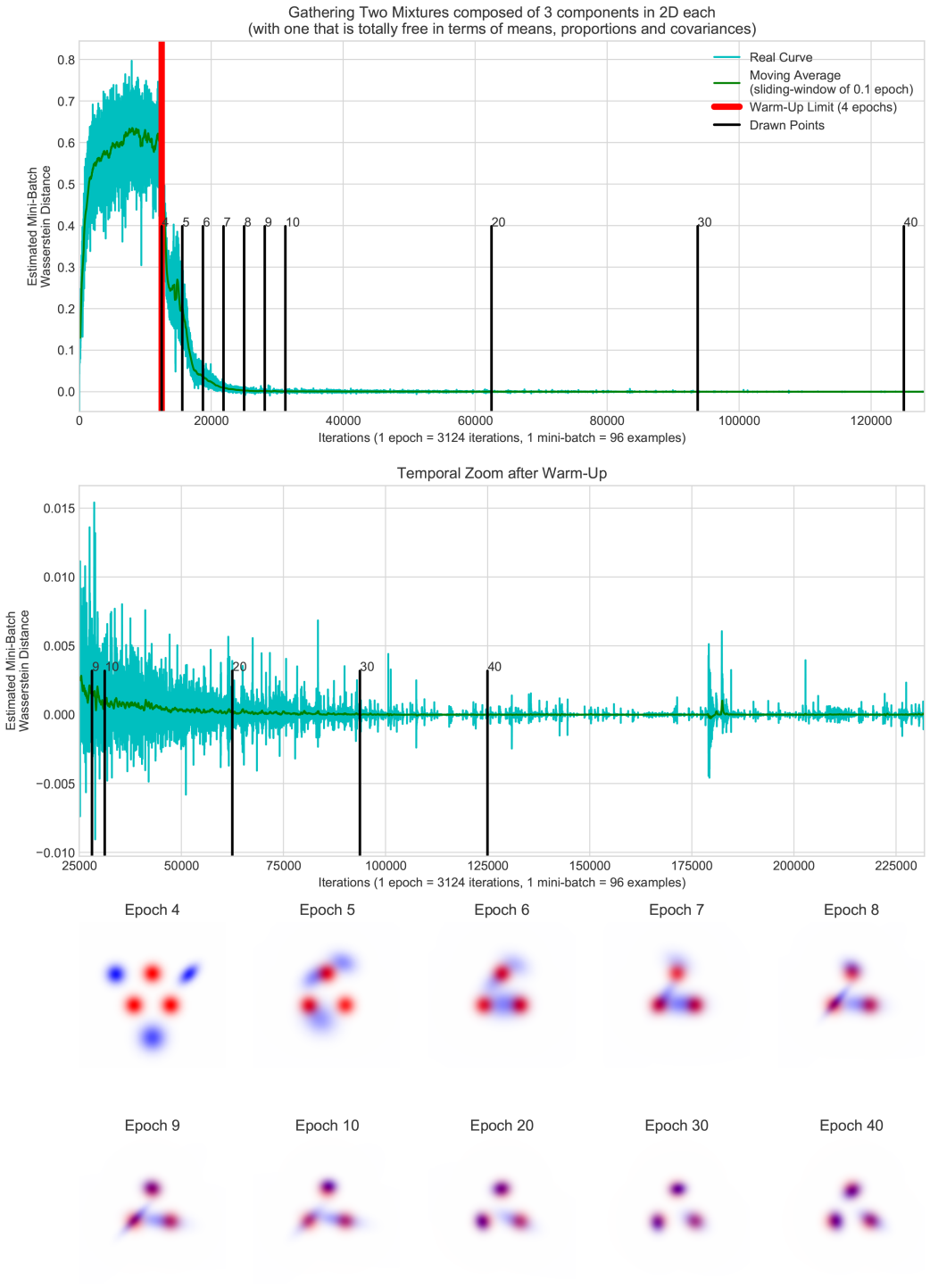

Figure 3: Same problem as EM-GMM but with a Wasserstein GAN without a generator neural network but a Gaussian mixture generator instead. A is the red distribution and B the blue one and they get purple when in local superposition — best seen in colors

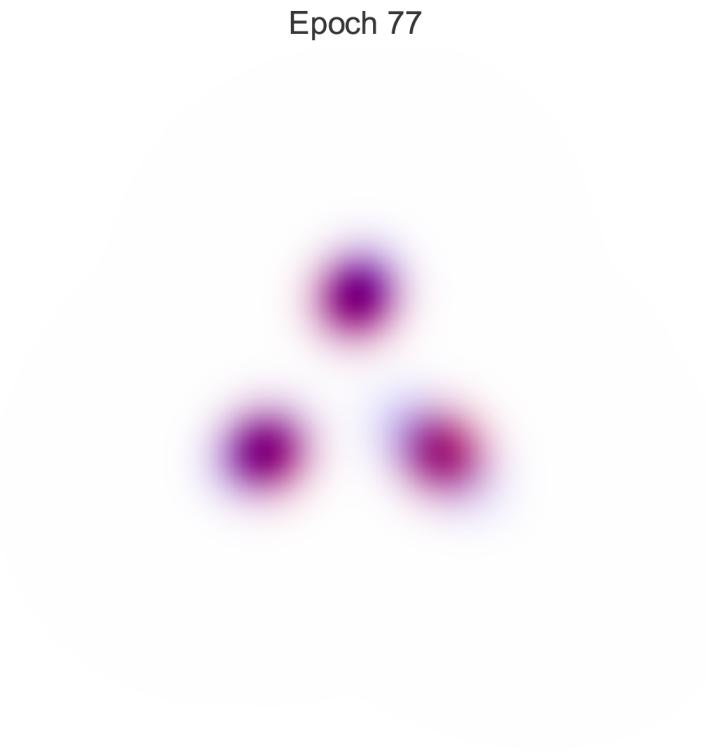

Figure 4: Converged mixtures (purple because of the superposition of the red and blue mixtures) — best seen in colors

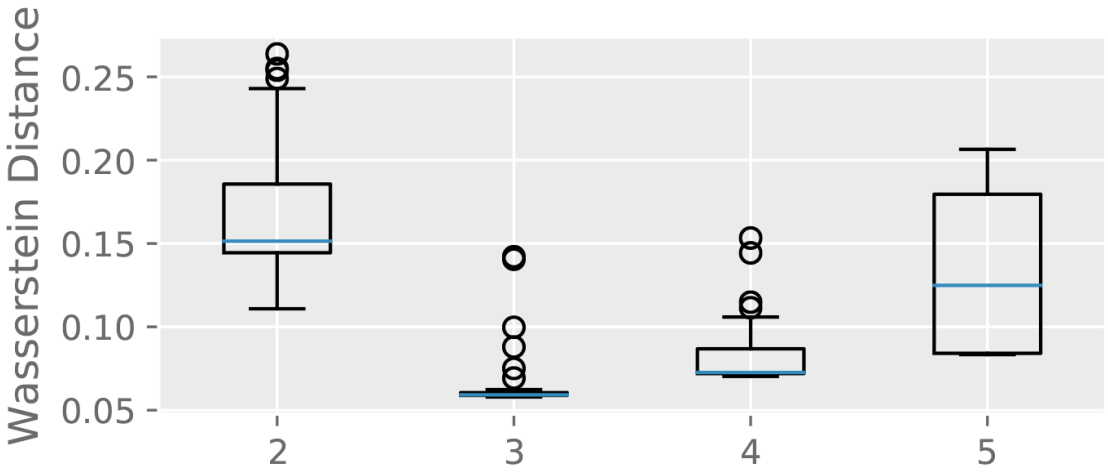

Figure 5: Wasserstein model selection on the three moons dataset for the number of clusters on a validation dataset on 30 runs for each number of clusters (the lower the better)

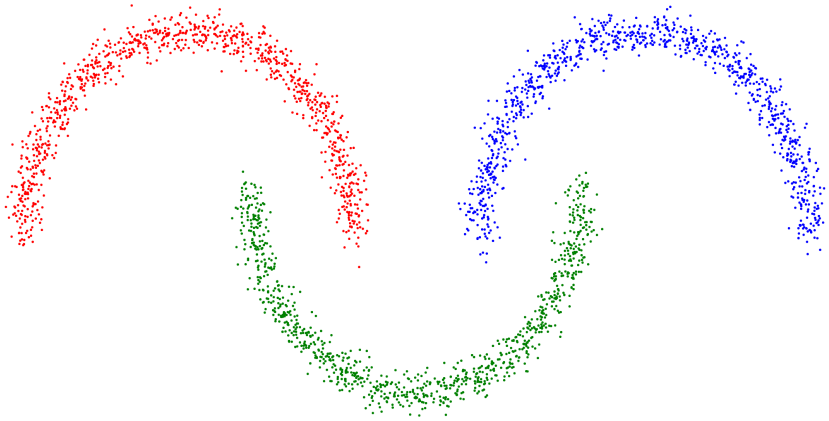

Figure 6: Three Moons 3M

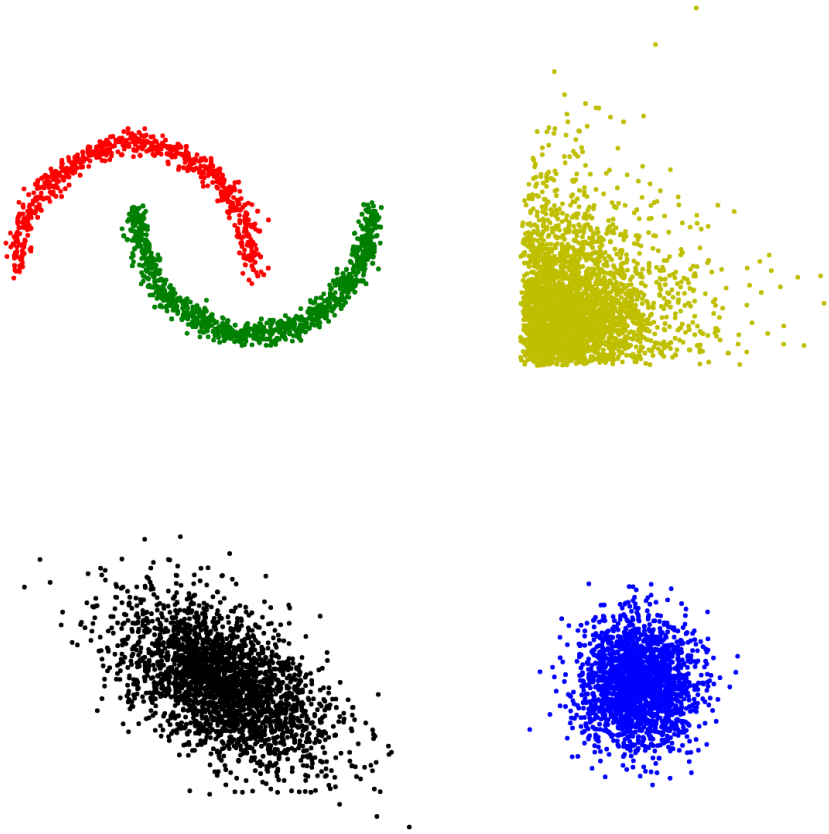

Figure 7: Various 2D Distributions VD

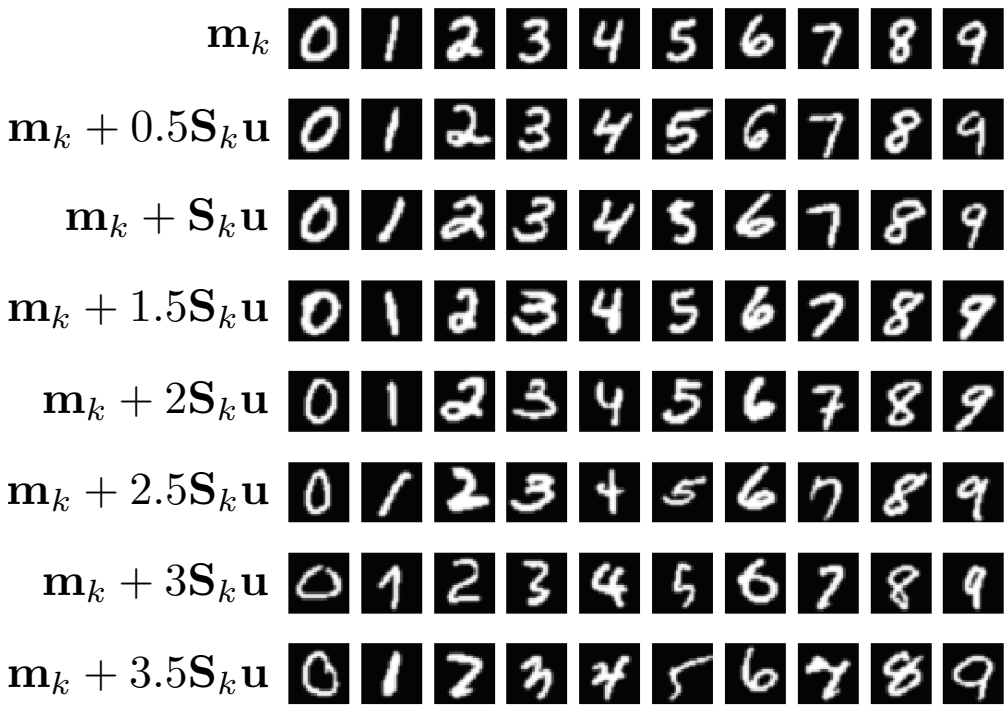

Figure 8: Generated digits images. From left to right, we have the ten classes found by GeWaC and ordered thanks to the Hungarian algorithm. From top to bottom, we go further and further in random directions from the centroids (the first row being the decoded centroids). More specifically, $\mathbf{u}$ is sampled from the uniform random density on the unit hypersphere in the code space.

5 Future Work and Conclusion

This work presents twos ways to use the Wasserstein distances literature with neural networks in order to achieve efficient clustering at the crossroad of Optimal Transport, Neural Networks and Model-based mixture techniques: first, we did generative clustering and second, we did discriminative clustering. In spite of three careful initialization steps, our technique is still end-to-end trainable as theoretically the final step could be done alone but in practice, with an associated higher risk of being stuck in spurious local extrema and much less reproducibility (i. e. higher dependence on the pseudo-random generators seeds which we do not have thanks to our robust and reasonable initialization steps).

In a mockingly accurate description of the generative part of our work, we could say that placing a mixture distribution at the input of new Generative Adversarial Networks seemed worthwhile for clustering purposes. Indeed, we inherit the recent wealth of literature (see adversarial autoencoders by Makhzani et al. [2015], Wasserstein autoencoders by Tolstikhin et al. [2018] and even adversarially learned inference Dumoulin et al. [2017]) with tunable input mixtures each mode corresponding to a cluster. In fact, it turns out that generating data in a clustered fashion with the desire to minimize the estimated Wasserstein distance with real data is difficult because of a mono-component collapsing effect due to a *LASSO*-kind of constraint on the mixture proportions which favors degenerated (sparse) proportions. Another way to explain this failure is the richness of the neural networks expression power that makes the Gaussian mixture clustering useless: one Gaussian prior is enough to generate whole datasets which looses any clustering capabilities hope if done carelessly. Eventually, we acknowledge the better and elegant solutions given in the recent work of Mukherjee et al. [2019] to cope with these problems and nothing can prevent us from re-using the same techniques with Wasserstein distance for us in lieu of the Jensen-Shannon divergence they use.

For the discriminative part, we observe that although our DiWaC system *sees* the data one by one because of its linear computation and memory complexity, it handles complex relationships between the points. In a way that is similar to spectral clustering (SC) with pairwise similarities in particular, but our approach has improved results with an objective function that does not full storage of pairwise similarities but only optimal transport that encodes it implicitly into the systems with still pairwises similarities but at a cluster distributions level. Our algorithm detects the space zones where the density mass is occupied thanks to the probabilistic model inherent to the optimal transport theory associated to our optimization. For that matter, we did manage to handle the trade-off between the flexibility of the neural networks functions and the rigidity of the mixture models choice of distributions. A counter-example 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.

Although our work is far from the well-established theory of the Reproducing Kernel Hilbert Spaces (RKHS), we realize that a similar phenomenon occurs when it comes to satisfactory results: from an input space of a given reasonable dimensionality D, it is better to first increase that dimensionality by changing space thanks to a mapping (the first layers of our encoder are increasing dimensionality compared to D) and only after that, shrink it to a small dimensionality $d \ll D$ (the last layer of our encoder). The main difference between our work and the RKHS theory is that our transformations are learned from the data and not given analytically *a priori* (e.g. by a Gaussian kernel) which echoes the work of Unser [2018].

Within the context of the algorithm laid out above, we empirically observe some symbiosis operating between generative or discriminative clustering and non-linear embedding. A great advantage of our approach is model selection especially for the number of clusters that seems to empirically work well. In other tasks different from clustering for itself, encouraged by future over-clustering investigations, one can be inspired by the work of Liao et al. [2016], that demonstrates the benefits of clustering-based regularization for supervised classification and generalization. For example, inside a supervised classification problem, identifying sub-categories might help to specialize classifiers on more homogeneous classes for improved generalization capabilities and ease of data interpretations. Finally, we now have the intriguing possibility of following other successes of the supervised classification and unsupervised clustering communities, such as the automatic selection of discriminative parts in the spirit of what Sun and Ponce [2016] and eventually Doersch et al. [2012] did and the automatic setting of the number of clusters. Beyond all these promising clustering results, our GeWaC algorithm has the ability to generate cluster-wise data. It is interesting to see that such conditional generation, already explored in the supervised setting [Mirza and Osindero, 2014] is not fundamentally harder without supervision.

References

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2017. URL http://proceedings.mlr.press/v70/arjovsky17a.html.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab, June 2006. URL http://ilpubs.stanford.edu:8090/778/.

F. R. Bach and Z. Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 49–56. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3269-diffrac-a-discriminative-and-flexible-framework-for-clustering.pdf.

D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995.

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Pattern Analysis and Machine Intelligence*, 22(7):719–725, 2000. doi: 10.1109/34.865189.

E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In D. Lee, M. Schkolnick, F. J. Provost, and R. Srikant, editors, *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge discovery and Data Mining, San Francisco, CA, USA, August 26-29, 2001*, pages 245–250. ACM, 2001. URL http://portal.acm.org/citation.cfm?id=502512.502546.

C. M. Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003. URL http://jmlr.org/papers/v3/blei03a.html.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

L. Bottou and Y. Bengio. Convergence Properties of the K-Means Algorithms. In *Advances in Neural Information Processing Systems*, pages 585–592, 1995.

C. Bouveyron and C. Brunet. On the estimation of the latent discriminative subspace in the Fisher-EM algorithm. *Journal de la Société Française de Statistique & revue de statistique appliquée*, 2011.

C. Bouveyron and C. Brunet. Simultaneous model-based clustering and visualization in the fisher discriminative subspace. *Statistics and Computing*, 22(1):301–324, 2012.

C. Bouveyron and C. Brunet-Saumard. Discriminative variable selection for clustering with the sparse fisher-em algorithm. *Computational Statistics*, 29(3-4):489–513, 2014a.

C. Bouveyron and C. Brunet-Saumard. Model-based clustering of high-dimensional data: A review. *Computational Statistics and Data Analysis*, 71:52–78, 2014b.

C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery. *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *7th International Conference on Learning Representations, ICLR 2019*, 2019.

O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*, 2018.

T. R. Davidson, L. Falorsi, N. D. Cao, T. Kipf, and J. M. Tomczak. Hyperspherical variational auto-encoders. pages 856–865, 2018. URL http://auai.org/uai2018/proceedings/papers/309.pdf.

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

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

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

K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. 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.

C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4):101:1–101:9, 2012.

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *Proceedings of the International Conference on Learning Representations*, 2016.

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

R. M. Dudley. *Real analysis and probability*. Chapman and Hall/CRC, 2018.

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

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188, 1936.

R. Flamary, M. Cuturi, N. Courty, and A. Rakotomamonjy. Wasserstein discriminant analysis. *Machine Learning*, 107(12):1923–1945, 2018. ISSN 15730565. doi: 10.1007/s10994-018-5717-1.

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

A. Genevay. *Entropy-regularized optimal transport for machine learning*. PhD thesis, 2019.

A. Genevay, G. Peyré, and M. Cuturi. Learning Generative Models with Sinkhorn Divergences. (2017-83), Oct. 2017. URL https://ideas.repec.org/p/crs/wpaper/2017-83.html.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, 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, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

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

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. 2016. URL http://arxiv.org/abs/1701.00160.

A. Graves. Stochastic backpropagation through mixture density distributions. *arXiv preprint arXiv:1607.05690*, 2016.

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

T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning discrete representations via information maximizing self-augmented training. In D. Precup and Y. W. 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.

A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 2010. ISSN 01678655.

Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: A generative approach to clustering. 2016. URL `http://arxiv.org/abs/1611.05148`.

O. Kilinc and I. 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`.

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

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

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

J. M. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems*, pages 463–470, 2003.

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

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

R. Liao, A. Schwing, R. Zemel, and R. Urtasun. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems*, pages 5076–5084, 2016.

S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–136, 1982. URL `https://doi.org/10.1109/TIT.1982.1056489`.

A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

S. Mukherjee, H. Asnani, E. Lin, and S. Kannan. ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4610–4617, 2019. doi: 10.1609/aaai.v33i01.33014610.

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

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.

G. Peyré, M. Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

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

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

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

J. Sun and J. Ponce. Learning Dictionary of Discriminative Part Detectors for Image Categorization and Cosegmentation. In *International Journal of Computer Vision*, volume 120, pages 111–133, 2016. doi: 10.1007/s11263-016-0899-0.

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkL7n1-0b`.

M. Unser. A representer theorem for deep neural networks. *arXiv preprint arXiv:1802.09210*, 2018.

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

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

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