Generalised Mutual Information for Discriminative Clustering

Louis Ohl
Université Côte d'Azur
Inria, CNRS
I3S, Maasai team
CHU de Québec Research Center
Laval University
louis.ohl@inria.fr

Pierre-Alexandre Mattei
Université Côte d'Azur
Inria, CNRS
LJAD, Maasai team
pierre-alexandre.mattei@inria.fr

Charles Bouveryon
Université Côte d'Azur
Inria, CNRS
LJAD, Maasai team

Warith Harchaoui
Jellysmack
AI Labs
Research and Development

Mickael Leclerq
CHU de Québec Research Center
Laval University

Arnaud Droit
CHU de Québec Research Center
Laval University

Frederic Precioso
Université Côte d'Azur
Inria, CNRS
I3S, Maasai team

Abstract

In the last decade, recent successes in deep clustering majorly involved the mutual information (MI) as an unsupervised objective for training neural networks with increasing regularisations. While the quality of the regularisations have been largely discussed for improvements, little attention has been dedicated to the relevance of MI as a clustering objective. In this paper, we first highlight how the maximisation of MI does not lead to satisfying clusters. We identified the Kullback-Leibler divergence as the main reason of this behaviour. Hence, we generalise the mutual information by changing its core distance, introducing the generalised mutual information (GEMINI): a set of metrics for unsupervised neural network training. Unlike MI, some GEMINIs do not require regularisations when training. Some of these metrics are geometry-aware thanks to distances or kernels in the data space. Finally, we highlight that GEMINIs can automatically select a relevant number of clusters, a property that has been little studied in deep clustering context where the number of clusters is a priori unknown.

1 Introduction

Clustering is a fundamental learning task which consists in separating data samples into several categories, each named cluster. This task hinges on two main questions concerning the assessment of correct clustering and the actual number of clusters that may be contained within the data distribution. However, this problem is ill-posed since a cluster lacks formal definitions which makes it a hard problem (Kleinberg, 2003).

Model-based algorithms make assumptions about the true distribution of the data as a result of some latent distribution of clusters (Bouveyron et al., 2019). These techniques are able to find the most

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

likely cluster assignment to data points. These models are usually generative, exhibiting an explicit assumption of the prior knowledge on the data.

Early deep models to perform clustering first relied on autoencoders, based on the belief that an encoding space holds satisfactory properties (Xie et al., 2016; Ghasedi Dizaji et al., 2017; Ji et al., 2019). However, the drawback of these architectures is that they do not guarantee that data samples which should meaningfully be far apart remain so in the feature space. Early models that dropped decoders notably used the mutual information (MI) (Krause et al., 2010; Hu et al., 2017) as an objective to maximise. The MI can be written in two ways, either as measure of dependency between two variables x and y, e.g. data distribution $p(x)$ and cluster assignment $p(y)$:

$$\mathcal{I}(x;y) = D_{\mathrm{KL}}(p(x,y)||p(x)p(y)), \tag{1}$$

or as an expected distance between implied distributions and the overall data:

$$\mathcal{I}(x;y) = \mathbb{E}_{p(y)}[D_{\mathrm{KL}}(p(x|y)||p(x))], \tag{2}$$

with D_{KL} being the Kullback-Leibler (KL) divergence. Related works often relied on the notion of MI as a measure of coherence between cluster assignments and data distribution (Hjelm et al., 2019). Regularisation techniques were employed to leverage the potential of MI, mostly by specifying model invariances, for example with data augmentation (Ji et al., 2019).

The maximisation of MI thus gave way to contrastive learning objectives which aim at learning stable representations of data through such invariance specifications (Chen et al., 2020; Caron et al., 2020). The contrastive loss maximises the similarity between the features of a sample and its augmentation, while decreasing the similarity with any other sample. Clustering methods also benefited from recent successful deep architectures (Li et al., 2021; Tao et al., 2021; Huang et al., 2020) by encompassing regularisations in the architecture. These methods correspond to discriminative clustering where we seek to directly infer cluster given the data distribution. Initial methods also focused on alternate schemes, for example with curriculum learning (Chang et al., 2017) to iteratively select relevant data samples for training: for example by alternating K-means cluster assignment with supervised learning using the inferred labels (Caron et al., 2018), or by proceeding to multiple distinct training steps (Van Gansbeke et al., 2020; Dang et al., 2021; Park et al., 2021).

However, most of the methods above rarely discuss their robustness when the number of clusters to find is different from the amount of preexisting known classes. While previous work was essentially motivated by considering MI as a dependence measure, we explore in this paper the alternative definition of the MI as the expected distance between data distribution implied by the clusters and the entire data. We extend it to incorporate cluster-wise comparisons of implied distributions, and question the choice of the KL divergence with other possible statistical distances.

Throughout the introduction of the generalised mutual information (GEMINI), the contributions of this paper are:

- A demonstration of how the maxima of MI are not sufficient criteria for clustering. This extends the contribution of (Tschannen et al., 2020) to the discrete case.
- The introduction of a set of metrics called GEMINIs involving different distances between distributions which can incorporate prior knowledge on the geometry of the data. Some of these metrics do not require regularisations.
- A highlight of the implicit selection of clusters from GEMINIs which allows to select a relevant number of cluster during training.

2 Is mutual information a good clustering objective?

We consider in this section a dataset consisting in N unlabelled samples $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$. We distinguish two major use cases of the mutual information: one where we measure the dependence between two continuous variables, as is the case in representation learning, and one where the random variable is discrete. In representation learning, the goal is to construct a continuous representation $\boldsymbol{z}$ extracted from the data $\boldsymbol{x}$ using a learnable distribution of parameters θ. In clustering, samples $\boldsymbol{x}$ are assigned to the discrete variable y through another learnable distribution.

2.1 Representation learning

Representation learning consists in finding high-level features $\boldsymbol{z}_i$ extracted from the data $\boldsymbol{x}_i$ in order to perform a *downstream task*, e.g. clustering or classification. MI between $\boldsymbol{x}$ and $\boldsymbol{z}$ is a common choice for learning features(Hjelm et al., 2019). However, estimating correctly MI between two random variables in continuous domains is often intractable when $p(\boldsymbol{x}|\boldsymbol{z})$ or $p(\boldsymbol{z}|\boldsymbol{x})$ is unknown, thus lower bounds are preferred, e.g. variational estimators such as MINE (Belghazi et al., 2018), $\mathcal{I}_{\text{NCE}}$(Van den Oord et al., 2018). Another common choice of loss function to train features are contrastive losses such as NT-XENT (Chen et al., 2020) where the similarity between the features $\boldsymbol{z}_i$ from data $\boldsymbol{x}_i$ is maximised with the features $\tilde{\boldsymbol{z}}$ from a data-augmented $\tilde{\boldsymbol{x}}_i$ against any other features $\boldsymbol{z}_j$. Recently, Do et al. (2021) achieved excellent performances in single-stage methods by highlighting the link between the $\mathcal{I}_{\text{NCE}}$ estimator (Van den Oord et al., 2018) and contrastive learning losses. Representation learning therefore comes at the cost of a complex lower bound estimator on MI, which often requires data augmentation. Moreover, it was noticed that the MI is hardly predictive of downstream tasks (Tschannen et al., 2020) when the variable y is continuous, i.e. a high value of MI does not clarify whether the discovered representations are insightful with regards to the target of the downstream task.

2.2 Discriminative clustering

The MI has been first used as an objective for learning discriminative clustering models (Bridle et al., 1992). Associated architectures went from simple logistic regression (Krause et al., 2010) to deeper architectures (Hu et al., 2017; Ji et al., 2019). Beyond architecture improvement, the MI maximisation was also carried with several regularisations. These regularisations include penalty terms such as weight decay (Krause et al., 2010) or Virtual Adversarial Training (VAT, Hu et al., 2017; Miyato et al., 2018b). Data augmentation was further used to provide invariances in clustering, as well as specific architecture designs like auxiliary clustering heads (Ji et al., 2019). Rewriting the MI in terms of entropies:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathcal{H}(y) - \mathcal{H}(y|\boldsymbol{x}) \tag{3}$$

highlights a requirement for balanced clusters, through the cluster entropy term $\mathcal{H}(y)$. Indeed, a uniform distribution maximises the entropy. This hints that an unregularised discrete mutual information for clustering can possibly produce uniformly distributed clusters among samples, regardless of how close they could be. We highlight this claim in section 2.3. As an example of regularisation impact: maximising the MI with ℓ_2 constraint can be equivalent to a soft and regularised K-Means in a feature space (Jabi et al., 2019). In clustering, the number of clusters to find is usually not known in advance. Therefore, an interesting clustering algorithm should be able to find a relevant number of clusters, i.e. perform model selection. However, model selection for parametric deep clustering models is expensive (Ronen et al., 2022). Cluster selection through MI maximisation has been little studied in related works, since experiments usually tasked models to find the (supervised) classes of datasets. Furthermore, the literature diverged towards deep learning methods focusing mainly on images, yet rarely on other type of data such as tabular data (Min et al., 2018).

2.3 Maximising the MI can lead to bad decision boundaries

Maximising the MI directly can be a poor objective: a high MI value is not necessarily predictive of the quality of the features regarding downstream tasks (Tschannen et al., 2020) when y is continuous. We support a similar argument for the case where the data x is a continuous random variable and the cluster assignment y a categorical variable. Indeed, the MI can be maximised by setting appropriately a sharp decision boundary which partitions evenly the data. This reasoning can be seen in the entropy-based formulation of the MI (Eq. 3): any sharp decision boundary minimises the negative conditional entropy, while ensuring balanced clusters maximises the entropy of cluster proportions. Consider for example Figure 1, where a mixture of Gaussian distributions with equal variances is separated by a sharp decision boundary. We highlight that both models will have the same mutual information on condition that the misplaced decision boundary of Figure 1b splits evenly the dataset (see Appendix A).

Globally, MI misses the idea in clustering that any two points close to one another may be in the same cluster according to some chosen metric. Hence regularisations are required to ensure this constraint. A sketch of these insights was mentionned by Bridle et al. (1992); Corduneanu and Jaakkola (2002).

(a) Good decision boundary (b) Misplaced decision boundary

Figure 1: Example of maximised MI for a Gaussian mixture $p(x) \sim \frac{1}{2}\mathcal{N}(\mu_0, \sigma^2) + \frac{1}{2}\mathcal{N}(\mu_1, \sigma^2)$. It is clear that Figure 1a presents the best decision boundary and posterior between the two Gaussian distributions. Yet, as $p(x|y)$ converges to a Dirac distribution, the MI difference converges to 0.

3 Extending the mutual information to the generalised mutual information

Given the identified limitations of MI, we now describe the discriminative clustering framework based on our perception of the mutual information. We then detail the different statistical distances we can use to extend MI to the generalised mutual information (GEMINI).

3.1 The discriminative clustering framework for GEMINIs

We change our view on the mutual information by seeing it as a discriminative clustering objective that aims at separating the data distribution given cluster assignments $p(\boldsymbol{x}|y)$ from the data distribution $p(\boldsymbol{x})$ according to the KL divergence:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p(y)}\left[D_{\mathrm{KL}}(p(\boldsymbol{x}|y)\|p(\boldsymbol{x}))\right]. \tag{4}$$

To highlight the discriminative clustering design, we explicitly remove our hypotheses on the data distribution by writing $p_{\text{data}}(\boldsymbol{x})$. The only part of the model that we design is a conditional distribution $p_\theta(y|\boldsymbol{x})$ that assigns a cluster y to a sample $\boldsymbol{x}$ using the parameters θ (Minka, 2005). This conditional distribution can typically be a neural network of adequate design regarding the data, e.g. a CNN, or a simple categorical distribution. Consequently, the cluster proportions are controlled by θ because $p_\theta(y) = \mathbb{E}[p_\theta(y|\boldsymbol{x})]$ and so is the conditional distribution $p_\theta(\boldsymbol{x}|y)$ even though intractable. This questions how Eq. (4) can be computed. Fortunately, well-known properties of MI can invert the distributions on which the KL divergence is computed (Bridle et al., 1992; Krause et al., 2010) via Bayes' theorem:

$$\mathcal{I}(\boldsymbol{x}; y) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[D_{\mathrm{KL}}(p_\theta(y|\boldsymbol{x})\|p_\theta(y))\right], \tag{5}$$

which is possible to estimate. Since we highlighted earlier that the KL divergence in the MI can lead to inappropriate decision boundaries, we are interested in replacing it by other distances or divergences. However, changing it in Eq. (5) would focus on the separation of cluster assignments from the cluster proportions which may be irrelevant to the data distribution. We rather alter Eq. (4) to clearly show that we separate data distributions given clusters from the entire data distribution because it allows us to take into account the data space geometry.

3.2 The GEMINI

The goal of the GEMINI is to separate data distributions according to an arbitrary distance D, i.e. changing the KL divergence for another divergence or distance in the MI. Moreover, we question the evaluation of the distance between the distribution of the data given a cluster assumption $p_\theta(\boldsymbol{x}|y)$ and the entire data distribution $p_{\text{data}}(\boldsymbol{x})$. We argue that it is intuitive in clustering to compare the distribution of one cluster against the distribution of *another cluster* rather than the data distribution. This raises the definition of two GEMINIs, one named *one-vs-all* (OvA):

$$\mathcal{I}_D^{\text{OvA}}(\boldsymbol{x}; y) = \mathbb{E}_{y\sim p_\theta(y)}\left[D(p_\theta(\boldsymbol{x}|y)\|p_{\text{data}}(\boldsymbol{x}))\right], \tag{6}$$

which compares the cluster distributions to the data distribution, and the *one-vs-one* (OvO) in which we independently draw cluster assignments y_a and y_b (see App. B for an OvO justification):

$$\mathcal{I}_D^{\text{OvO}}(\boldsymbol{x}; y) = \mathbb{E}_{y_a, y_b\sim p_\theta(y)}\left[D(p_\theta(\boldsymbol{x}|y_a)\|p_\theta(\boldsymbol{x}|y_b))\right], \tag{7}$$

Table 1: Definition of the GEMINI for f-divergences, MMD and the Wasserstein distance. We directly write here the equation that can be optimised to train a discriminative model $p_\theta(y|\boldsymbol{x})$ via stochastic gradient descent since they are expectations over the data.

Name	Equation
KL OvA/MI	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[D_{\text{KL}}(p_\theta(y\|\boldsymbol{x})\|p_\theta(y))\right]$
KL OvO	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}[D_{\text{KL}}(p_\theta(y\|\boldsymbol{x})\|p_\theta(y))) + D_{\text{KL}}(p_\theta(y)\|p_\theta(y\|\boldsymbol{x})))]$
Squared Hellinger OvA	$1 - \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}[\mathbb{E}_{p_\theta(y)}[\sqrt{\frac{p_\theta(y\|\boldsymbol{x})}{p_\theta(y)}}]]$
Squared Hellinger OvO	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}[\mathbb{V}_{p_\theta(y)}[\sqrt{\frac{p_\theta(y\|\boldsymbol{x})}{p_\theta(y)}}]]$
TV OvA	$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}[D_{\text{TV}}(p_\theta(y\|\boldsymbol{x})\|p_\theta(y))]$
TV OvO	$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}[\mathbb{E}_{y_a,y_b\sim p_\theta(y)}[\lvert\frac{p_\theta(y_a\|\boldsymbol{x})}{p_\theta(y_a)} - \frac{p_\theta(y_b\|\boldsymbol{x})}{p_\theta(y_b)}\rvert]]$
MMD OvA	$\mathbb{E}_{p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[k(\boldsymbol{x}_a,\boldsymbol{x}_b)\left(\frac{p_\theta(y\|\boldsymbol{x}_a)p_\theta(y\|\boldsymbol{x}_b)}{p_\theta(y)^2} + 1 - 2\frac{p_\theta(y\|\boldsymbol{x}_a)}{p_\theta(y)}\right)\right]^{\frac{1}{2}}\right]$
MMD OvO	$\mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[k(\boldsymbol{x}_a,\boldsymbol{x}_b)\left(\frac{p_\theta(y_a\|\boldsymbol{x}_a)p_\theta(y_a\|\boldsymbol{x}_b)}{p_\theta(y_a)^2} + \frac{p_\theta(y_b\|\boldsymbol{x}_a)p_\theta(y_b\|\boldsymbol{x}_b)}{p_\theta(y_b)^2} - 2\frac{p_\theta(y_a\|\boldsymbol{x}_a)p_\theta(y_b\|\boldsymbol{x}_b)}{p_\theta(y_a)p_\theta(y_b)}\right)\right]^{\frac{1}{2}}\right]$
Wasserstein OvA	$\mathbb{E}_{p_\theta(y)}\left[\mathcal{W}_c\left(\sum_{i=1}^N m_i^y\delta_{\boldsymbol{x}_i}, \sum_{i=1}^N \frac{1}{N}\delta_{\boldsymbol{x}_i}\right)\right]$
Wasserstein OvO	$\mathbb{E}_{y_a,y_b\sim p_\theta(y)}\left[\mathcal{W}_c\left(\sum_{i=1}^N m_i^{y_a}\delta_{\boldsymbol{x}_i}, \sum_{i=1}^N m_i^{y_b}\delta_{\boldsymbol{x}_i}\right)\right]$

There exists other distances than the KL to measure how far two distributions p and q are one from the other. We can make a clear distinction between two types of distances, Csiszar's f-divergences (Csiszár, 1967) and integral probability metrics (IPM) (Sriperumbudur et al., 2009). However, unlike f-divergences, IPM-derived distances like the Wasserstein distance or the maximum mean discrepancy (MMD)(Gneiting and Raftery, 2007; Gretton et al., 2012) bring knowledge about the data throughout either a distance c or a kernel κ: these distances are geometry-aware.

f-divergence GEMINIs: These divergences involve a convex function $f : \mathbb{R}^+ \to \mathbb{R}$ such that $f(1) = 0$. This function is applied to evaluate the ratio between two distributions p and q, as in Eq. (8):

$$D_{\text{f-div}}(p, q) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[f\left(\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right)\right]. \tag{8}$$

We will focus on three f-divergences: the KL divergence, the total variation (TV) distance and the squared Hellinger distance. While the KL divergence is the usual divergence for the MI, the TV and the squared Hellinger distance present different advantages among f-divergences. First of all, both of them are bounded between 0 and 1. It is consequently easy to check when any GEMINI using those is maximised contrarily to the MI that is bounded by the minimum of the entropies of $\boldsymbol{x}$ and y (Gray and Shields, 1977). When used as distance between data conditional distribution $p_\theta(\boldsymbol{x}|y)$ and data distribution $p_{\text{data}}(\boldsymbol{x})$, we can apply Bayes' theorem in order to get an estimable equation to maximise, which only involves cluster assignment $p_\theta(y|\boldsymbol{x})$ and marginals $p_\theta(y)$ (see Table 1).

MMD GEMINIs: The MMD corresponds to the distance between the respective expected embedding of samples from distribution p and distribution q in a reproducing kernel hilbert space (RKHS) $\mathcal{H}$:

$$\text{MMD}(p\|q) = \|\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\varphi(\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\varphi(\boldsymbol{z})]\|_{\mathcal{H}}, \tag{9}$$

where φ is the RKHS embedding. To compute this distance we can use the kernel trick (Gretton et al., 2012) by involving the kernel function $\kappa(\boldsymbol{a}, \boldsymbol{b}) = \langle\varphi(\boldsymbol{a}), \varphi(\boldsymbol{b})\rangle$. We use Bayes' theorem to uncover a version of the MMD that can be estimated through Monte Carlo using only the predictions $p_\theta(y|\boldsymbol{x})$ (see Table 1).

Wasserstein GEMINIs: This distance is an optimal transport distance, defined as:

$$\mathcal{W}_c(p,q) = \left(\inf_{\gamma \in \Gamma(p,q)} \mathbb{E}_{\boldsymbol{x},\boldsymbol{z} \sim \gamma(\boldsymbol{x},\boldsymbol{z})} \left[c(\boldsymbol{x},\boldsymbol{z}) \right] \right), \tag{10}$$

where $\Gamma(p,q)$ is the set of all couplings between p and q and c a distance function in $\mathcal{X}$. Computing the Wasserstein distance between two distributions $p_\theta(\boldsymbol{x}|y=k_1)$ and $p_\theta(\boldsymbol{x}|y=k_2)$ is difficult in our discriminative context because we only have access to a finite set of samples N. To achieve the Wasserstein-GEMINI, we instead use approximations of the distributions with weighted sums of Diracs:

$$p_\theta(\boldsymbol{x}|y=k) \approx \sum_{i=1}^{N} m_i^k \delta_{\boldsymbol{x}_i} = p_N^k, \quad \text{with} \quad m_i^k = \frac{p_\theta(y=k|\boldsymbol{x}_i)}{\sum_{j=1}^{N} p_\theta(y=k|\boldsymbol{x}_j)}, \tag{11}$$

where $\delta_{\boldsymbol{x}_i}$ is a Dirac located on sample location $\boldsymbol{x}_i \in \mathcal{X}$. The Wasserstein-OvA and -OvO applied to Dirac sums are compatible with the `emd2` function of the Python optimal transport package (Flamary et al., 2021) which gracefully supports automatic differentation (see Appendix D.3 for convergence to the expectation). All GEMINIs are summarised in Table 1, (see Appendix D for derivations).

4 Experiments

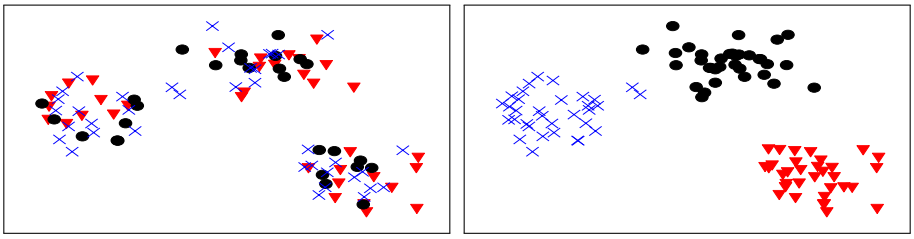
For all experiments below, we report the adjusted rand index (ARI) (Hubert and Arabie, 1985), a common metric in clustering. This metric is external as it requires labels for evaluation. It ranges from 0, when labels are independent from cluster assignments, to 1, when labels are equivalent to cluster assignments up to permutations. An ARI close to 0 is equivalent to the best accuracy when voting constantly for the majority class, e.g. 10% on a balanced 10-class dataset. Regarding the MMD- and Wasserstein-GEMINIs, we used by default a linear kernel and the Euclidean distance unless specified otherwise. All discriminative models are trained using the Adam optimiser (Kingma and Ba, 2014). We estimate a total of 450 hours of GPU consumption. (See Appendix I for the details of Python packages for experiments and Appendix F for further experiments regarding model selection). The code is available at `https://github.com/oshillou/GEMINI`

4.1 When the MI fails because of the modelling

We first took the most simple discriminative clustering model, where each cluster assignment according to the input datum follows a categorical distribution:

$$y|\boldsymbol{x} = \boldsymbol{x}_i \sim \text{Cat}(\theta_i^1, \theta_i^2, \cdots, \theta_i^K).$$

We generated $N = 100$ samples from a simple mixture of $K = 3$ Gaussian distributions. Each model thus only consists in NK parameters to optimise. This is a simplistic way of describing the most flexible deep neural network. We then maximised on the one hand the KL-OvA (MI) and on the other hand the MMD-OvA. Both clustering results can be seen in Figure 2. We concluded that

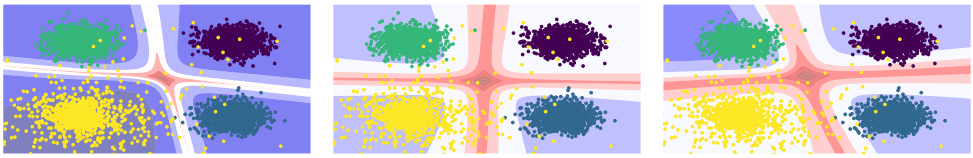


(a) KL-OvA (MI) (b) MMD OvA with linear kernel

Figure 2: Clustering of a mixture of 3 Gaussian distributions with MI (left) and a GEMINI (right) using categorical distributions. The MI does not have insights on the data shape because of the model, and clusters points uniformly between the 3 clusters (black dots, red triangles and blue crosses) whereas the MMD is aware of the data shape through its kernel.

without any function, e.g. a neural network, to link the parameters of the conditional distribution with $\boldsymbol{x}$, the MI struggles to find the correct decision boundaries. Indeed, the position of $\boldsymbol{x}$ in the 2D space plays no role and the decision boundary becomes only relevant with regards to cluster entropy maximisation: a uniform distribution between 3 clusters. However, it plays a major role in the kernel of the MMD-GEMINI thus solving correctly the problem.

4.2 Resistance to outliers



(a) KL-OvA (MI) Entropy Map (b) MMD-OvO Entropy Map (c) Wasserstein-OvO Entropy Map

Figure 3: Entropy maps of the predictions of each MLP trained using a GEMINI or the MI. The bottom-left distribution (yellow) is a Student-t distribution with 1 degree of freedom that produces samples far from the origin. The Rényi entropy of prediction is highlighted from lowest (red background) to highest (blue background). MI on the left has the most confident predictions overall and the smallest uncertainty around the decision boundary, i.e. high entropy variations.

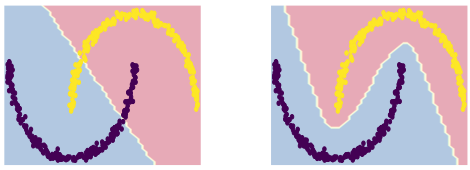
To prove the strength of using neural networks for clustering trained with GEMINI, we introduced extreme samples in Gaussian mixtures by replacing a Gaussian distribution with a Student-t distribution for which the degree of freedom ρ is small. We fixed $K = 4$ clusters, 3 being drawn from multivariate Gaussian distributions and the last one from a multivariate Student-t distribution in 2 dimensions for visualisation purposes with 1 degree of freedom (see AppendixE for other parameters and results). Thus, the Student-t distribution has an undefined expectation and produces samples that can be perceived as outliers regarding a Gaussian mixture owing to its heavy tail. We report the ARIs of multi-layered perceptron (MLP) trained 20 times with GEMINIs in Table 2. The presence of "outliers" leads K-Means and Gaussian Mixture models to fail at grasping the 4 distributions when tring to find 4 clusters. Meanwhile, GEMINIs perform better. Note that all MMD and Wasserstein-OvO-GEMINI present lower standard deviation for high scores compared to f-divergence GEMINIs. We attribute these performances to both the MLP that tries to find separating hyperplanes in the data space and the absence of hypotheses regarding the data. Moreover, as mentioned in section 2.3, the usual MI is best maximised when its decision boundary presents little entropy $\mathcal{H}(p_\theta(y|\boldsymbol{x}))$. As neural networks can be overconfident (Guo et al., 2017), MI is likely to yield overconfident clustering by minimizing the conditional entropy. We highlight such behaviour in Figure 3 where the Rényi entropy (Rényi, 1961) associated to each sample in the MI (Figure 3a) is much lower, if not 0, compared to MMD-OvO and Wasserstein-OvO (figures 3b and 3c). We conclude that Wasserstein- and MMD-GEMINIs train neural networks not to be overconfident, hence yielding more moderate distributions $p_\theta(y|\boldsymbol{x})$.

Table 2: Mean ARI (std) of a MLP fitting a mixture of 3 Gaussian and 1 Student-t multivariate distributions compared with Gaussian Mixture Models and K-Means. The models try to find 4 at best and the Student-t distribution has ρ=1 degree of freedom. We write the ARI for the maximum a posteriori of an oracle aware of all parameters of the data.

K-Means	GMM		MMD		Wasserstein	
	full cov	diagonal cov	$\mathcal{I}^{\text{ova}}_{\text{MMD}}$	$\mathcal{I}^{\text{ovo}}_{\text{MMD}}$	$\mathcal{I}^{\text{ova}}_{\mathcal{W}}$	$\mathcal{I}^{\text{ovo}}_{\mathcal{W}}$
0 (0)	0 (0)	0.024 (0.107)	0.922 (0.004)	0.921 (0.007)	0.915 (0.131)	0.922 (0.006)
Oracle	f-divergences					
	$\mathcal{I}^{\text{ova}}_{\text{KL}}$	$\mathcal{I}^{\text{ovo}}_{\text{KL}}$	$\mathcal{I}^{\text{ova}}_{\text{H}^2}$	$\mathcal{I}^{\text{ovo}}_{\text{H}^2}$	$\mathcal{I}^{\text{ova}}_{\text{TV}}$	$\mathcal{I}^{\text{ovo}}_{\text{TV}}$
0.989	**0.939 (0.006)**	0.723 (0.114)	0.906 (0.103)	0.858 (0.143)	0.904 (0.104)	**0.938 (0.005)**

4.3 Leveraging a manifold geometry

We highlighted that MI can be maximised without requiring to find the suitable decision boundary. Here, we show how the provided distance to the Wasserstein-OvO GEMINI can leverage appropriate clustering when we have a good a priori on the data.

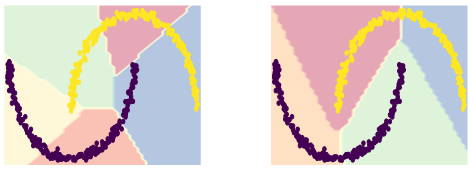


(a) $\mathcal{I}_{\text{kl}}^{\text{ova}}$ for 2 clusters (MI) (b) $\mathcal{I}_{\mathcal{W}}^{\text{ovo}}$ for 2 clusters

Figure 4: Fitting hand-generated moons using the GEMINI on top of an MLP with 2 clusters to find.

The importance of the distance c: We generated a dataset consisting of two facing moons on which we trained a MLP using either the MI or the Wasserstein-OvO GEMINI. To construct a distance c for the Wasserstein distance, we derived a distance from the Floyd-Warshall algorithm (Warshall, 1962; Roy, 1959) on a sparse graph describing neighborhoods of samples. This distance describes how many neighbors are in between two samples, further details are provided in appendix H. We report the different decision boundaries in Figure 4. We observe that the insight on the neighborhood provided by our distance c helped the MLP to converge to the correct solution with an appropriate decision boundary unlike the MI. Note that the usual Euclidean distance in the Wasserstein metric would have converged to a solution similar to the MI. Indeed for 2 clusters, the optimal transport plan has a larger value, using a distribution similar to Figure 4a, than in Figure 4b. This toy example shows how an insightful metric provided to the Wasserstein distance in GEMINIs can lead to correct decision boundaries while only designing a discriminative distribution $p_\theta(y|\boldsymbol{x})$ and a distance c.

Not using all clusters In addition, we highlight an interesting behaviour of all GEMINIs. During optimisation, it is possible that the model converges to using fewer clusters than the number to find. For example in Figure 5, for 5 clusters, the model can converge to 4 balanced clusters and 1 empty cluster (Figure 5b) unlike MI that produced 5 misplaced clusters (Figure 5a). Indeed, the entropy on the cluster proportion in the MI forces to use the maximum number of clusters.



(a) $\mathcal{I}_{\text{kl}}^{\text{ova}}$ for 5 clusters (MI) (b) $\mathcal{I}_{\mathcal{W}}^{\text{ovo}}$ for 5 clusters

Figure 5: The Wasserstein-ovo model with 5 clusters eventually found 4 unlike the MI that maintained 5 clusters.

Table 3: ARI for deep neural network trained with GEMINIs on MNIST for 500 epochs. Models were trained either with either 10 clusters to find or 15. We indicate in parentheses the number of used clusters by the model after training.

GEMINI		10 clusters		15 clusters	
		MLP	LeNet-5	MLP	LeNet-5
KL	OvA	0.320 (10)	0.138 (8)	0.271 (15)	0.136 (12)
	OvO	0.348 (7)	0.123 (4)	0.333 (8)	0.104 (4)
Squared Hellinger	OvA	0.301 (10)	0.207 (6)	0.224 (13)	0.162 (7)
	OvO	0.287 (10)	0.161 (6)	0.305 (13)	0.157 (7)
TV	OvA	0.299 (10)	0.171 (6)	0.277 (15)	0.140 (6)
	OvO	0.422 (10)	0.161 (9)	0.330 (15)	0.182 (14)
MMD	OvA	0.373 (10)	0.382 (10)	0.345 (15)	0.381 (15)
	OvO	0.361 (10)	0.373 (10)	0.364 (15)	0.379 (15)
Wasserstein	OvA	**0.471** (10)	**0.463** (10)	0.390 (15)	**0.446** (11)
	OvO	0.450 (10)	0.383 (10)	**0.415** (15)	0.414 (15)
K-Means		0.367		0.385	

4.4 Fitting MNIST

We trained a neural network using either MI or GEMINIs. Following Hu et al. (2017), we first tried with a MLP with one single hidden layer of dimension 1200. To further illustrate the robustness of the method and its adaptability to other architectures, we also experimented using a LeNet-5 architecture (LeCun et al., 1998) since it is adequate to the MNIST dataset. We report our results in Table 3. Since we are dealing with a clustering method, we may not know the number of clusters a priori in a dataset. The only thing that can be said about MNIST is that there are *at least* 10 clusters, one per digit. Indeed, writings of digits could differ leading to more clusters than the number of classes. That is why we further tested the same method with 15 clusters to find in Table 3. We first see that the scores of the MMD and Wasserstein GEMINIs are greater than the MI, with the highest performances for the Wasserstein-OvAWe also observe that no f-divergence-GEMINI always yield best ARIs. Nonetheless, we observe better performances in the case of the TV GEMINIs owing to its bounded gradient. This results in controlled stepsize when doing gradient descent contrarily to KL- and squared Hellinger-GEMINIs. Notice that the change of architecture from a MLP to a LeNet-5 unexpectedly halves the scores for the f-divergences. We believe this drop is due to the change of notion of neighborhood implied by the network architecture.

4.5 Cifar10 clustering using a SIMCLR-derived kernel

To further illustrate the benefits of the kernel or distance provided to GEMINIs, we continue the same experiment as in section 4.4. However, we focus this time on the CIFAR10 dataset. As improved distance, we chose a linear kernel and ℓ_2 norm between features extracted from a pretrained SIMCLR model (Chen et al., 2020). We provide results for two different architectures: LeNet-5 and ResNet-18 both trained from scratch on raw images, the latter being a common choice of models in deep clustering literature (Van Gansbeke et al., 2020; Tao et al., 2021). We report the results in Table 4 and provide the baseline of MI. We also write the baselines from related works when not using data augmentations to make a fair comparison. Indeed, models trained with GEMINIs do not use data augmentation: only the architecture and the kernel or distance function in the data space plays a role. We observe here that the choice of kernel or distance can be critical in GEMINIs. Indeed, while the Euclidean norm between images does not provide insights on how images of cats and dogs are far as shown by K-Means, features derived from SIMCLR carry much more insight on the proximity of images. This shows that the performances of GEMINIs depend on the quality of distance functions. Interestingly, we observe that for the Resnet-18 using SIMCLR features to guide GEMINIs was not as successful as it has been on the LeNet-5. We believe that the ability of this network to draw any decision boundary makes it equivalent to a categorical distribution model as in Sec. 4.1. Finally, to the best of our knowledge, we are the first to train from scratch a standard discriminative neural network on CIFAR raw images without using labels or direct data augmentations, while getting sensible clustering results. However, other recent methods achieve best scores using data augmentations which we do not (Park et al., 2021).

Table 4: ARI score of models trained for 200 epochs on CIFAR10 with different architectures using GEMINIs. The kernel for the MMD is either a linear kernel or the dot product between features extracted from a pretrained SIMCLR model. Both the Euclidean norm between images and SIMCLR features are considered for the Wasserstein metric. We report the ARI of related works when not using data augmentation for comparison.*: scores reported from Li et al., (2021)

Architecture	No kernel	Linear kernel / ℓ_2 norm				SIMCLR (Chen et al., 2020)			
	$\mathcal{I}_{\mathrm{KL}}^{\mathrm{ova}}$	$\mathcal{I}_{\mathrm{MMD}}^{\mathrm{ova}}$	$\mathcal{I}_{\mathrm{MMD}}^{\mathrm{ovo}}$	$\mathcal{I}_{\mathcal{W}}^{\mathrm{ova}}$	$\mathcal{I}_{\mathcal{W}}^{\mathrm{ovo}}$	$\mathcal{I}_{\mathrm{MMD}}^{\mathrm{ova}}$	$\mathcal{I}_{\mathrm{MMD}}^{\mathrm{ovo}}$	$\mathcal{I}_{\mathcal{W}}^{\mathrm{ova}}$	$\mathcal{I}_{\mathcal{W}}^{\mathrm{ovo}}$
LeNet-5	0.026	0.049	0.048	0.043	0.041	**0.157**	0.145	0.079	0.138
Resnet-18	0.008	0.047	0.044	0.037	0.036	0.122	**0.145**	0.052	0.080
KMeans (images / SIMCLR)		0.041	0.147	CC (Li et al., 2021)				0.030	
IDFD (Tao et al., 2021)		0.060		JULE (Yang et al., 2016)*				0.138	

5 Conclusion

We highlighted that the choice of distance at the core of MI can alter the performances of deep learning models when used as an objective for a deep discriminative clustering. We first showed that MI maximisation does not necessarily reflect the best decision boundary in clustering. We introduced the GEMINI, a method which only needs the specification of a neural network and a kernel or distance in the data space. Moreover, we showed how the notion of neighborhood built by the neural network can affect the clustering, especially for MI. To the best of our knowledge, this is the first method that trains single-stage neural networks from scratch using neither data augmentations nor regularisations, yet achieving good clustering performances. We emphasised that GEMINIs are only searching for a maximum number of clusters: after convergence some may be empty. However, we do not have insights to explain this convergence which is part of future work. Finally, we introduced several versions of GEMINIs and would encourage the MMD-OvA or Wasserstein-OvA as a default choice, since it proves to both incorporate knowledge from the data using a kernel or distance while remaining the less complex than MMD-OvO and Wasserstein-OvO in time and memory. OvO versions could be privileged for fine-tuning steps. Future works could include an optimisation of the time performances of the Wasserstein-OvO to make it more competitive.

Acknowledgements

This work has been supported by the French government, through the 3IA Côte d'Azur, Investment in the Future, project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. We would also like to thank the France Canada Research Fund (FFCR) for their contribution to the project. This work was partly supported by EU Horizon 2020 project AI4Media, under contract no. 951911. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein Generative Adversarial Networks. 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 214–223. PMLR, Aug. 2017.

S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and Equilibrium in Generative Adversarial Nets (GANs). 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 224–232. PMLR, 06–11 Aug 2017.

M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm. Mutual Information Neural Estimation. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540. PMLR, 10–15 Jul 2018.

P. Billingsley. *Convergence of Probability Measures*. John Wiley & Sons, 2013.

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

J. Bridle, A. Heading, and D. MacKay. Unsupervised Classifiers, Mutual Information and 'Phantom Targets. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1992.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep Clustering for Unsupervised Learning of Visual Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan. Deep Adaptive Image Clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888, 2017. doi: 10.1109/ICCV.2017.626.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

A. Corduneanu and T. Jaakkola. On Information Regularization. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 151–158, 2002.

I. Csiszár. Information-Type Measures of Difference of Probability Distributions and Indirect Observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318, 1967.

Z. Dang, C. Deng, X. Yang, K. Wei, and H. Huang. Nearest Neighbor Matching for Deep Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13693–13702, 2021.

K. Do, T. Tran, and S. Venkatesh. Clustering by Maximizing Mutual Information Across Views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9928–9938, 2021.

W. Falcon et al. Pytorch Lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. Pot: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5736–5745, 2017.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree. Regularisation of Neural Networks by Enforcing Lipschitz Continuity. *Machine Learning*, 110(2):393–416, Feb. 2021. ISSN 1573-0565. doi: 10.1007/s10994-020-05929-w.

R. M. Gray and P. C. Shields. The Maximum Mutual Information between Two Random Processes. *Information and Control*, 33(4):273–280, 1977.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A Kernel Two-Sample Test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. Publisher: JMLR. org.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

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, Aug. 2017.

J. Huang, S. Gong, and X. Zhu. Deep Semantic Clustering by Partition Confidence Maximisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8849–8858, 2020.

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

M. Jabi, M. Pedersoli, A. Mitiche, and I. B. Ayed. Deep clustering: On the link between discriminative models and K-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019. doi: 10.1109/TPAMI.2019.2962683.

X. Ji, J. F. Henriques, and A. Vedaldi. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. Kleinberg. An Impossibility Theorem for Clustering. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

A. Krause, P. Perona, and R. Gomes. Discriminative Clustering by Regularized Information Maximization. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng. Contrastive Clustering. In *2021 AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. *IEEE Access*, PP:1–1, 2018. doi: 10.1109/ACCESS.2018.2855437.

T. Minka. Discriminative models, not discriminative training. Technical report, Technical Report MSR-TR-2005-144, Microsoft Research, 2005.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018a.

T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018b. Publisher: IEEE.

A. B. Owen. *Monte Carlo theory, methods and examples*. Stanford, 2013.

S. Park, S. Han, S. Kim, D. Kim, S. Park, S. Hong, and M. Cha. Improving Unsupervised Image Clustering With Robust Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12278–12287, 2021.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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.

A. Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press, 1961.

M. Ronen, S. E. Finder, and O. Freifeld. DeepDPM: Deep Clustering With an Unknown Number of Clusters. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9851–9860, 2022. doi: 10.1109/CVPR52688.2022.00963.

B. Roy. Transitivité et connexité. *Comptes Rendus Hebdomadaires Des Séances De L'Académie Des Sciences*, 249(2):216–218, 1959.

B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet. On integral probability metrics,ϕ-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

Y. Tao, K. Takagi, and K. Nakata. Clustering-friendly representation learning via instance discrimination and feature decorrelation. In *International Conference on Learning Representations*, 2021.

M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. On Mutual Information Maximization for Representation Learning. In *International Conference on Learning Representations*, 2020.

A. Van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, pages arXiv–1807, 2018.

W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool. Scan: Learning to Classify Images without Labels. In *European Conference on Computer Vision*, pages 268–285. Springer, 2020.

C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

S. Warshall. A Theorem on Boolean Matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.

J. Xie, R. Girshick, and A. Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA, June 2016. PMLR.

J. Yang, D. Parikh, and D. Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We clearly state that we focus on mutual information for clustering and how we improve it.
   (b) Did you describe the limitations of your work? [Yes] We describe shortcomings in Sec. 4.5,5 and provide more in App. G.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Yes, we confirm having read the ethics review guidelines and conform to it.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We clearly highlighted that we compute the GEMINI between a continuous variable, namely the data, and a discrete variable, namely the cluster assignment. The framework was detailed precisely in Sec. 3.1 and all distances have been introduced.
   (b) Did you include complete proofs of all theoretical results? [Yes] Please refer to appendices A, D.3 or D for proofs of our claims.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Link to github will be available upon release of paper and a sample is zipped aside in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Data generation processes are reported in supplementary materials and main parameters are described at the head of Sec. 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] When applicable due to multiple runs in table form, e.g. Table 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We mention it at the beginning of Sec. 4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] All packages for the experiments are mentionned in Appendix I
   (b) Did you mention the license of the assets? [Yes] We are under licenses of MIT (NumPy, POT,), BSD (torch,pandas, scikit-learn) and the python software foundation (matplotlib, python).
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The code once released will contain the specific file `losses.py` that will contain only the definitions of GEMINIs, alongside all configurations to reproduce the experiments.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Demonstration of the convergence to 0 of the MI for a Gaussian Mixture

A.1 Models definition

Let us consider a mixture of two Gaussian distributions, both with different means μ_0 and μ_1, s.t. $\mu_0 < \mu_1$ and of same standard deviation σ:

$$p(x|y=0) = \mathcal{N}(x|\mu_0, \sigma^2), p(x|y=1) = \mathcal{N}(x|\mu_1, \sigma^2),$$

where y is the cluster assignment. We take balanced clusters proportions, i.e. $p(y=0) = p(y=1) = \frac{1}{2}$. This first model is the basis that generated the complete dataset $p(x)$. When performing clustering with our discriminative model, we are not aware of the distribution. Consequently: we create other models. We want to compute the difference of mutual information between two decision boundaries that discriminative models $p_\theta(y|x)$ may yield.

We define two decision boundaries: one which splits evenly the data space called p_A and another which splits it on a closed set p_B:

$$p_A(y=1|x) = \begin{cases} 1-\epsilon & x > \frac{\mu_1-\mu_0}{2} \\ \epsilon & \text{otherwise} \end{cases},$$

$$p_B(y=1|x) = \begin{cases} 1-\epsilon & x \in [\mu_0, \mu_1] \\ \epsilon & \text{otherwise} \end{cases}. \tag{12}$$

Our goal is to show that both models p_A and p_B will converge to the same value of mutual information as ϵ converges to 0.

A.2 Computing cluster proportions

A.2.1 Cluster proportion of the correct decision boundary

To compute the cluster proportions, we estimate with samples x from the distribution $p_{\text{data}}(x)$. Since we are aware for this demonstration of the true nature of the data distribution, we can use $p(x)$ for sampling. Consequently, we can compute the two marginals:

$$\begin{aligned} p_A(y=1) &= \int_{\mathcal{X}} p(x) p_A(y=1|x) dx, \\ &= \int_{-\infty}^{\frac{\mu_1-\mu_0}{2}} p(x)\epsilon dx + \int_{\frac{\mu_1-\mu_0}{2}}^{+\infty} p(x)(1-\epsilon) dx, \\ &= \epsilon \left(\int_{-\infty}^{\frac{\mu_1-\mu_0}{2}} p(x) dx \right) + (1-\epsilon) \left(\int_{\frac{\mu_1-\mu_0}{2}}^{+\infty} p(x) dx \right), \\ &= \frac{1}{2}. \end{aligned}$$

A.2.2 Cluster proportion of the misplaced decision boundary

For the misplaced decision boundary, the marginal is different:

$$\begin{aligned} p_B(y=1) &= \int_{\mathcal{X}} p(x) p_B(y=1|x) dx, \\ &= \epsilon \left(\int_{-\infty}^{\mu_0} p(x) dx + \int_{\mu_1}^{+\infty} p(x) dx \right) + (1-\epsilon) \int_{\mu_0}^{\mu_1} p(x) dx, \\ &= \epsilon \left(1 - \int_{\mu_0}^{\mu_1} p(x) dx \right) + (1-\epsilon) \int_{\mu_0}^{\mu_1} p(x) dx. \end{aligned} \tag{13}$$

Here, we simply introduce a new variable named β that will be a shortcut for noting the proportion of data between μ_0 and μ_1:

$$\beta = \int_{\mu_0}^{\mu_1} p(x)dx.$$

And so can we simply write the cluster proportion of decision boundary model B as:

$$p_B(y=1) = \epsilon(1-\beta) + (1-\epsilon)\beta,$$

Leading to the summary of proportions in Table 5. For convenience, we will write the proportions of model B using the shortcuts:

$$\pi_B = p_B(y=1) = \epsilon + \beta(1-2\epsilon),$$

$$\bar{\pi}_B = p_B(y=0) = 1 - \epsilon - \beta(1-2\epsilon).$$

Table 5: Proportions of clusters for models A and B

$\mathcal{M}$	A	B
$p_{\mathcal{M}}(y=1)$	$\frac{1}{2}$	$\epsilon + \beta(1-2\epsilon)$
$p_{\mathcal{M}}(y=0)$	$\frac{1}{2}$	$1 - \epsilon - \beta(1-2\epsilon)$

A.3 Computing the KL divergences

A.3.1 Correct decision boundary

We first start by computing the Kullback-Leibler divergence for some arbitrary value of $x \in \mathbb{R}$:

$$D_{\text{KL}}(p_A(y|x)||p_A(y)) = \sum_{i=0}^{1} p_A(y=i|x) \log \frac{p_A(y=i|x)}{p_A(y=i)}.$$

We now need to detail the specific cases, for the value of $p(y=i|x)$ is dependent on x. We start $\forall x < \frac{\mu_1 - \mu_0}{2}$:

$$\begin{aligned} D_{\text{KL}}(p_A(y|x)||p_A(y)) &= p_A(y=0|x) \log \frac{p_A(y=0|x)}{\frac{1}{2}} + p_A(y=1|x) \log \frac{p_A(y=1|x)}{\frac{1}{2}}, \\ &= (1-\epsilon)\log 2(1-\epsilon) + \epsilon \log 2\epsilon. \end{aligned}$$

The opposite case, $\forall x \geq \frac{\mu_1 - \mu_0}{2}$ yields:

$$\begin{aligned} D_{\text{KL}}(p_A(y|x)||p_A(y)) &= p_A(y=0|x) \log \frac{p_A(y=0|x)}{\frac{1}{2}} + p_A(y=1|x) \log \frac{p_A(y=1|x)}{\frac{1}{2}}, \\ &= \epsilon \log 2\epsilon + (1-\epsilon)\log 2(1-\epsilon). \end{aligned}$$

Since both cases are equal, we can write down:

$$D_{\text{KL}}(p_A(y|x)||p_A(y)) = \epsilon \log 2\epsilon + (1-\epsilon)\log 2(1-\epsilon), \forall x \in \mathbb{R}. \tag{14}$$

A.3.2 Misplaced boundary

We proceed to the same detailing of the Kullback-Leibler divergence computation for the misplaced decision boundary. We start with the set $x \in [\mu_0, \mu_1]$:

$$\begin{aligned} D_{\text{KL}}(p_B(y|x)||p_B(y)) &= p_B(y=0|x) \log \frac{p_B(y=0|x)}{p_B(y=0)} + p_B(y=1|x) \log \frac{p_B(y=1|x)}{p_B(y=1)}, \\ &= \epsilon \log \frac{\epsilon}{\bar{\pi}_B} + (1-\epsilon) \log \frac{1-\epsilon}{\pi_B}. \end{aligned}$$

When x is out of this set, the divergence becomes:

$$\begin{aligned} D_{\text{KL}}(p_B(y|x)||p_B(y)) &= p_B(y=0|x) \log \frac{p_B(y=0|x)}{p_B(y=0)} + p_B(y=1|x) \log \frac{p_B(y=1|x)}{p_B(y=1)}, \\ &= (1-\epsilon) \log \frac{1-\epsilon}{\bar{\pi}_B} + \epsilon \log \frac{\epsilon}{\pi_B}. \end{aligned}$$

To fuse the two results, we will write the KL divergence as such:

$$D_{\text{KL}}(p_B(y|x)||p_B(y)) = \epsilon \log \epsilon + (1-\epsilon) \log(1-\epsilon) - C(x), \forall x \in \mathbb{R}, \tag{15}$$

where $C(x)$ is a constant term depending on x defined by:

$$C(x) = \begin{cases} \epsilon \log \bar{\pi}_B + (1-\epsilon) \log \pi_B & x \in [\mu_0, \mu_1] \\ \epsilon \log \pi_B + (1-\epsilon) \log \bar{\pi}_B & x \in \mathbb{R} \setminus [\mu_0, \mu_1] \end{cases}. \tag{16}$$

For simplicity of later writings, we will shorten the notations by:

$$C(x) = \begin{cases} \alpha_1 & x \in [\mu_0, \mu_1] \\ \alpha_0 & x \in \mathbb{R} \setminus [\mu_0, \mu_1] \end{cases}.$$

A.4 Evaluating the mutual information

A.4.1 Correct decision boundary

We inject the value of the Kullback-Leibler divergence from Eq. (14) inside an expectation performed over the data distribution $p_{\text{data}}(x)$:

$$\begin{aligned} \mathcal{I}_A(X;Y) &= \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[D_{\text{KL}}(p_A(y|x)||p_A(y)) \right], & (17) \\ &= \int_{\mathcal{X}} p(x) \left(\epsilon \log(2\epsilon) + (1-\epsilon) \log(2(1-\epsilon)) \right) dx, & (18) \\ &= \epsilon \log(2\epsilon) + (1-\epsilon) \log(2(1-\epsilon)). & (19) \end{aligned}$$

Since the KL divergence was independent of x, we could leave the constant outside of the integral which is equal to 1.

We can assess the coherence of Eq. (19) since its limit as ϵ approaches 0 is $\log 2$. In terms of bits, this is the same as saying that the information on X directly gives us information on the Y of the cluster.

A.4.2 Odd decision boundary

We inject the value of the KL divergence from Eq. (15) inside the expectation of the mutual information:

$$
\begin{aligned}
\mathcal{I}_B(X;Y) &= \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[D_{\text{KL}}(p_B(y|x)||p_B(y))\right], \\
&= \int_{\mathcal{X}} p(x)\left(\epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - C(x)\right), dx \\
&= \epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - \int_{\mathcal{X}} p(x)C(x)dx.
\end{aligned}
$$

The first terms are constant with respect to x and the integral of $p(x)$ over $\mathcal{X}$ adds up to 1. We finally need to detail the expectation of the constant $C(x)$ from Eq. (16):

$$
\begin{aligned}
\mathbb{E}_x[C(x)] &= \int_{-\infty}^{\mu_0} C(x)p(x)dx + \int_{\mu_0}^{\mu_1} C(x)p(x)dx + \int_{\mu_1}^{+\infty} C(x)p(x)dx, \\
&= \alpha_0\left(\int_{-\infty}^{\mu_0} p(x)dx + \int_{\mu_1}^{+\infty} p(x)dx\right) + \alpha_1\int_{\mu_0}^{\mu_1} p(x)dx, \\
&= \alpha_0(1-\beta) + \alpha_1\beta.
\end{aligned}
$$

This can be further improved by unfolding the description of α_0 and α_1 from Eq. (16):

$$
\begin{aligned}
\alpha_0(1-\beta) + \beta\alpha_1 &= \alpha_0 + \beta(\alpha_1 - \alpha_0), \\
&= \epsilon\log\pi_B + (1-\epsilon)\log\bar{\pi}_B + \beta\left[\epsilon\log\bar{\pi}_B + (1-\epsilon)\log\pi_B\right. \\
&\quad \left.-\epsilon\log\pi_B - (1-\epsilon)\log\bar{\pi}_B\right], \\
&= \left[1-\epsilon+\beta\epsilon-\beta+\beta\epsilon\right]\log\bar{\pi}_B + \left[\epsilon+\beta-\beta\epsilon-\beta\epsilon\right]\log\pi_B, \\
&= \log\bar{\pi}_B + \left[2\beta\epsilon-\beta-\epsilon\right]\log\frac{\bar{\pi}_B}{\pi_B}.
\end{aligned}
$$

We can finally write down the mutual information for the model B:

$$
\mathcal{I}_B(X;Y) = \epsilon\log\epsilon + (1-\epsilon)\log(1-\epsilon) - \log\bar{\pi}_B - \left[2\beta\epsilon-\beta-\epsilon\right]\log\frac{\bar{\pi}_B}{\pi_B}. \tag{20}
$$

A.5 Differences of mutual information

Now that we have the exact value of both mutual informations, we can compute their differences:

$$
\begin{aligned}
\Delta_{\mathcal{I}} &= \mathcal{I}_A(X;Y) - \mathcal{I}_B(X;Y), \\
&= \epsilon\log(2\epsilon) + (1-\epsilon)\log(2(1-\epsilon)) - \epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon) \\
&\quad + \log\bar{\pi}_B + \left[2\beta\epsilon-\beta-\epsilon\right]\log\frac{\bar{\pi}_B}{\pi_B}, \\
&= \epsilon\log 2 + (1-\epsilon)\log 2 + \log\bar{\pi}_B + \left[2\beta\epsilon-\beta-\epsilon\right]\log\frac{\bar{\pi}_B}{\pi_B}.
\end{aligned}
$$

We then deduce how the difference of mutual information evolves as the decision boundary becomes sharper, i.e. when ϵ approaches 0:

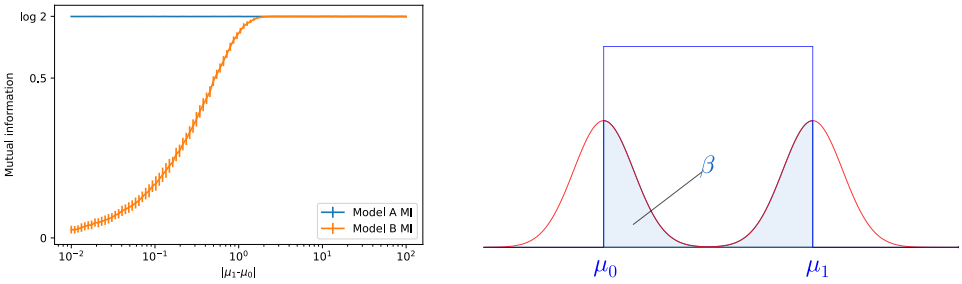
$$
\lim_{\epsilon\to 0}\Delta_{\mathcal{I}} = \log 2 + \log\bar{\pi}_B - \beta\log\frac{\bar{\pi}_B}{\pi_B}.
$$

However, the cluster proportions by B π_B also take a different value as ϵ approaches 0. Recalling Eq. (12):

$$
\lim_{\epsilon\to 0} p_B(y=1) = \beta, \lim_{\epsilon\to 0} p_B(y=0) = 1-\beta.
$$

And finally can we write that:

$$\begin{aligned}\lim_{\epsilon \to 0} \Delta_{\mathcal{I}} &= \log 2 + \log(1-\beta) - \beta \log \frac{1-\beta}{\beta}, \\ &= \log 2 + (1-\beta)\log(1-\beta) + \beta \log \beta, \\ &= \log 2 - \mathcal{H}(\beta).\end{aligned} \tag{21}$$



(a) Differences of MI between models A and B

(b) Gaussian mixture distribution $p(x)$ with proportion β in between the two means μ_0 and μ_1

Figure 6: Value of the mutual information for the two models splitting a Gaussian mixture dependening on the distance between the two means μ_0 and μ_1 of the two generating Gaussian distributions. We estimate the MI by computing it 50 times on 1000 samples drawn from the Gaussian mixture.

To conclude, as the decision boundaries turn sharper, i.e. when ϵ approaches 0, the difference of mutual information between the two models is controlled by the entropy of proportion of data β between the two means μ_0 and μ_1. We know that for binary entropies, the optimum is reached for $\beta = 0.5$. In other words having μ_0 and μ_1 distant enough to ensure balance of proportions between the two clusters of model B leads to a difference of mutual information equal to 0. We experimentally highlight this convergence in Figure 6 where we compute the mutual information of models A and B as the distance between the two means μ_0 and μ_1 increases in the Gaussian distribution mixture.

B A geometrical perspective on OvA and OvO GEMINIs

Considering the topology of the data space through a kernel in the case of the MMD or a distance in the case of the Wasserstein metric implies that we can effectively measure how two distributions are close to another. In the formal design of the mutual information, the distribution of each cluster

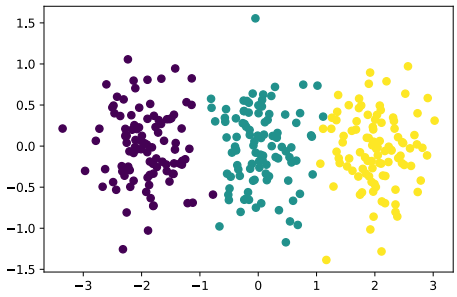


Figure 7: Here, 3 clusters of equal proportions from isotropic Gaussian distributions are located in -2, 0 and 2 on the x-axis, with small covariance. The complete data distribution hence has its expectation in 0 on the x-axis. Consequently, maximising the MMD OvA GEMINI can lead to 2 clusters at best while the MMD OvO is able to see all 3 clusters.

$p(\boldsymbol{x}|y)$ is compared to the complete data distribution $p(\boldsymbol{x})$. Therefore, if one distribution of a specific cluster $p(\boldsymbol{x}|y)$ were to look alike the data distribution $p(\boldsymbol{x})$, for example up to a constant in some areas of the space, then its distance to the data distribution could be 0, making it unnoticed when maximising the OvA GEMINI.

Take the example of 3 distributions $\{p(\boldsymbol{x}|y=i)\}_{i=1}^3$ with respective different expectations $\{\mu_i\}_{i=1}^3$. We want to separate them using the MMD OvA GEMINI with linear kernel. The mixture of the 3 distributions creates a data distribution with expectation $\mu = \sum_{i=1}^3 p(y=i)\mu_i$. However if the distributions satisfy that this data expectation μ is equal to one of the sub-expectations μ_i, then the associated distribution i will be non-evaluated since its MMD to the data distribution is equal to 0. We illustrate this example in figure 7. We tackled the problem by introducing the OvO setup where each pair of different cluster distribution is compared.

C Another approach to Wasserstein maximisation

The Wasserstein-1 metric can be considered as an IPM defined over a set of 1-Lipschitz functions. Indeed, such writing is the dual representation of the Wasserstein-1 metric:

$$W_c(p\|q) = \sup_{f, \|f\|_L \leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{z\sim q}[f(z)].$$

Yet, evaluating a supremum as an objective to maximise is hardly compatible with the usual back-propagration in neural networks. This definition was used in attempts to stabilise GAN training (Arjovsky et al., 2017) by using 1-Lipschitz neural networks (Gouk et al., 2021). However, the Lipschitz continuity was achieved at the time by weight clipping, whereas other methods such as spectral normalisation (Miyato et al., 2018a) now allow arbitrarily large weights. The restriction of 1-Lipscthiz functions to 1-Lipschitz neural networks does not equal the true Wasserstein distance, and the term "neural net distance" is sometimes preferred (Arora et al., 2017). Still, estimating the Wasserstein distance using a set of Lipschitz functions derived from neural networks architectures brings more difficulties to actually leverage the true distance according to the energy cost c of Eq. 10.

Globally, we hardly experimented the generic IPM for GEMINIs. Our efforts for defining a set of 1-Lipschitz critics, one per cluster for OvA or one per pair of clusters for OvO, to perform the neural net distance (Arora et al., 2017) were not fruitful. This is mainly because such objective requires the definition of one neural network for the posterior distribution $p_\theta(y|\boldsymbol{x})$ and K (resp. $K(K-1)/2$) other 1-Lipschitz neural networks for the OvA (resp. OvO) critics, i.e. a large amount of parameters. Moreover, this brings the problem of designing not only one, but many neural networks while the design of one accurate architecture is already a sufficient problem.

D Deriving GEMINIs

We show in this appendix how to derive all estimable forms of the GEMINI.

D.1 f-divergence GEMINI

We detail here the derivation for 3 f-divergences that we previously chose: the KL divergence, the TV distance and the squared Hellinger distance, as well as the generic scenario for any function f.

D.1.1 Generic scenario

First, we recall that the definition of an f-divergence involves a convex function:

$$\begin{aligned} f : \mathbb{R}^+ &\to \mathbb{R} \\ x &\to f(x) \\ \text{s.t.} \quad f(1) &= 0, \end{aligned}$$

between two distributions p and q as described:

$$D_{\text{f-div}}(p, q) = \mathbb{E}_{\boldsymbol{x}\sim q}\left[f\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right)\right].$$

We simply inject this definition in the OvA-GEMINI and directly obtain both an expectation on the cluster assignment y and on the data variable $\boldsymbol{x}$. We then merge the writing of the two expectations for the sake of clarity.

$$\begin{aligned}
\mathcal{I}^{\text{ova}}_{\text{f-div}} &= \mathbb{E}_{p_\theta(y)}\left[D_{\text{f-div}}(p_\theta(\boldsymbol{x}|y)||p_{\text{data}}(\boldsymbol{x}))\right], \\
&= \mathbb{E}_{p_\theta(y)}\left[\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[f\left(\frac{p_\theta(\boldsymbol{x}|y)}{p_{\text{data}}(\boldsymbol{x})}\right)\right]\right], \\
&= \mathbb{E}_{p_\theta(y), p_{\text{data}}(\boldsymbol{x})}\left[f\left(\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right)\right].
\end{aligned}$$

Injecting the f-divergence in the OvO-GEMINI first yields:

$$\begin{aligned}
\mathcal{I}^{\text{ovo}}_{\text{f-div}} &= \mathbb{E}_{p_\theta(y_a), p_\theta(y_b)}\left[D_{\text{f-div}}(p_\theta(\boldsymbol{x}|y_a)||p_\theta(\boldsymbol{x}|y_b))\right], \\
&= \mathbb{E}_{p_\theta(y_a), p_\theta(y_b)}\left[\mathbb{E}_{p_\theta(\boldsymbol{x}|y_b)}\left[f\left(\frac{p_\theta(\boldsymbol{x}|y_a)}{p_\theta(\boldsymbol{x}|y_b)}\right)\right]\right].
\end{aligned}$$

Now, by using Bayes theorem, we can perform the inner expectation over the data distribution. We then merge the expectations for the sake of clarity.

$$\begin{aligned}
\mathcal{I}^{\text{ovo}}_{\text{f-div}} &= \mathbb{E}_{p_\theta(y_a), p_\theta(y_b)}\left[\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} f\left(\frac{p_\theta(\boldsymbol{x}|y_a)}{p_\theta(\boldsymbol{x}|y_b)}\right)\right]\right], \\
&= \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} f\left(\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}\right)\right].
\end{aligned}$$

Notice that we also changed the ratio of conditional distributions inside the function by a ratio of posteriors through Bayes' theorem, weighted by the relative cluster proportions.

Now, we can derive into details these equations for the 3 f-divergences we focused on: the KL divergence, the TV distance and the squared Hellinger distance.

D.1.2 Kullback-Leibler divergence

The function for Kullback-Leibler is $f(t) = t \log t$. We do not need to write the OvA equation: it is straightforwardly the usual MI. For the OvO, we inject the function definition by replacing:

$$t = \frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)},$$

in order to get:

$$\mathcal{I}^{\text{ovo}}_{\text{KL}} = \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \times \frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}\right].$$

We can first simplify the factors outside of the logs:

$$\mathcal{I}_{\text{KL}}^{\text{ovo}} = \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)} \right].$$

If we use the properties of the log, we can separate the inner term in two sub-expressions:

$$\mathcal{I}_{\text{KL}}^{\text{ovo}} = \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} + \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})} \right].$$

Hence, we can use the linearity of the expectation to separate the two terms above. The first term is constant w.r.t. y_b, so we can remove this variable from the expectation among the subscripts:

$$\mathcal{I}_{\text{KL}}^{\text{ovo}} = \mathbb{E}_{p_\theta(y_a), p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \right] + \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \log \frac{p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})} \right].$$

Since the variables y_a and y_b are independent, we can use the fact that:

$$\mathbb{E}_{p_\theta(y_a)} \left[\frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} \right] = \int p_\theta(y_a) \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} dy_a = 1,$$

inside the second term to reveal the final form of the equation:

$$\mathcal{I}_{\text{KL}}^{\text{ovo}} = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}), p_\theta(y)} \left[\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \log \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \right] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}), p_\theta(y)} \left[\log \frac{p_\theta(y)}{p_\theta(y|\boldsymbol{x})} \right].$$

Notice that since both terms did not compare one cluster assignment y_a against another y_b, we can switch to the same common variable y. Both terms are in fact KL divergences depending on the cluster assignment y. The first is the reverse of the second. This sum of KL divergences is sometimes called the *symmetric* KL, and so can we write in two ways the KL-OvO GEMINI:

$$\begin{aligned} \mathcal{I}_{\text{KL}}^{\text{ovo}} &= \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[D_{\text{KL}}(p_\theta(y|\boldsymbol{x}) || p_\theta(y)) \right] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[D_{\text{KL}}(p_\theta(y) || p_\theta(y|\boldsymbol{x})) \right], \\ &= \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[D_{\text{KL-sym}}(p_\theta(y|\boldsymbol{x}) || p_\theta(y)) \right]. \end{aligned}$$

We can also think of this equation as the usual MI with an additional term based on the reversed KL divergence.

D.1.3 Total Variation distance

For the total variation, the function is $f(t) = \frac{1}{2}|t-1|$. Thus, the OvA GEMINI is:

$$\mathcal{I}_{\text{TV}}^{\text{ova}} = \frac{1}{2} \mathbb{E}_{p_\theta(y), p_{\text{data}}(\boldsymbol{x})} \left[\left| \frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} - 1 \right| \right].$$

And the OvO is:

$$\begin{aligned} \mathcal{I}_{\text{TV}}^{\text{ovo}} &= \frac{1}{2} \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \left| \frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_a)} - 1 \right| \right], \\ &= \frac{1}{2} \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})} \left[\left| \frac{p_\theta(y_a|\boldsymbol{x})}{p_\theta(y_a)} - \frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \right| \right]. \end{aligned}$$

We did not find any further simplification of these equations.

D.1.4 Squared Hellinger distance

Finally, the squared Hellinger distance is based on $f(t) = 2(1 - \sqrt{t})$. Hence the OvA unfolds as:

$$\begin{aligned}\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ova}} &= \mathbb{E}_{p_\theta(y), p_{\text{data}}(\boldsymbol{x})}\left[2\left(1 - \sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right)\right], \\ &= 2 - 2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}), p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right].\end{aligned}$$

The idea of the squared Hellinger-OvA GEMINI is therefore to minimise the expected square root of the relative certainty between the posterior and cluster proportion.

For the OvO setting, the definition yields:

$$\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ovo}} = \mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} \times 2 \times \left(1 - \sqrt{\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_b|\boldsymbol{x})p_\theta(y_a)}}\right)\right],$$

which we can already simplify by putting the constant 2 outside of the expectation, and by inserting all factors inside the square root before simplifying and separating the expectation:

$$\begin{aligned}\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ovo}} &= 2\mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)} - \frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)}\sqrt{\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b)}{p_\theta(y_a)p_\theta(y_b|\boldsymbol{x})}}\right], \\ &= 2\mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_b)}\right] - 2\mathbb{E}_{p_\theta(y_a), p_\theta(y_b), p_{\text{data}}(\boldsymbol{x})}\left[\sqrt{\frac{p_\theta(y_a|\boldsymbol{x})p_\theta(y_b|\boldsymbol{x})}{p_\theta(y_a)p_\theta(y_b)}}\right].\end{aligned}$$

We can replace the first term by the constant 1, as shown for the KL-OvO derivation. Since we can split the square root into the product of two square roots, we can apply twice the expectation over y_a and y_b because these variables are independent:

$$\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ovo}} = 2 - 2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right]^2\right].$$

To avoid computing this squared expectation, we use the equation of the variance $\mathbb{V}$ to replace it. Thus:

$$\begin{aligned}\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ovo}} &= 2 - 2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{p_\theta(y)}\left[\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right] - \mathbb{V}_{p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right]\right], \\ &= 2 - 2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{p_\theta(y)}\left[\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}\right]\right] + 2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{V}_{p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right]\right].\end{aligned}$$

Then, for the same reason as before, the second term is worth 1, which cancels the first constant. We therefore end up with:

$$\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ovo}} = 2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{V}_{p_\theta(y)}\left[\sqrt{\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)}}\right]\right].$$

Similar to the KL-OvO case, the squared Hellinger OvO converges to an OvA setting, i.e. we only need information about the cluster distribution itself without comparing it to another. Furthermore, the idea of maximising the variance of the cluster assignments is straightforward for clustering.

D.2 Maximum Mean Discrepancy

When using an IPM with a family of functions that project an input of $\mathcal{X}$ to the unit ball of an RKHS $\mathcal{H}$, the IPM becomes the MMD distance.

$$\begin{aligned}\text{MMD}(p,q) &= \sup_{f:||f||_{\mathcal{H}}\leq 1} \mathbb{E}_{\boldsymbol{x}_a\sim p}[f(\boldsymbol{x}_a)] - \mathbb{E}_{\boldsymbol{x}_b\sim q}[f(\boldsymbol{x}_b)], \\ &= \|\mathbb{E}_{\boldsymbol{x}_a\sim p}[\varphi(\boldsymbol{x}_a)] - \mathbb{E}_{\boldsymbol{x}_b\sim q}[\varphi(\boldsymbol{x}_b)]\|_{\mathcal{H}},\end{aligned}$$

where φ is a embedding function of the RKHS.

By using a kernel function $\kappa(\boldsymbol{x}_a, \boldsymbol{x}_b) =< \varphi(\boldsymbol{x}_a), \varphi(\boldsymbol{x}_b) >$, we can express the square of this distance thanks to inner product space properties (Gretton et al., 2012):

$$\text{MMD}^2(p,q) = \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}'_a\sim p}[\kappa(\boldsymbol{x}_a,\boldsymbol{x}'_a)] + \mathbb{E}_{\boldsymbol{x}_b,\boldsymbol{x}'_b\sim q}[\kappa(\boldsymbol{x}_b,\boldsymbol{x}'_b)] - 2\mathbb{E}_{\boldsymbol{x}_a\sim p,\boldsymbol{x}_b\sim q}[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)].$$

Now, we can derive each term of this equation using our distributions $p \equiv p_\theta(\boldsymbol{x}|y)$ and $q \equiv p_{\text{data}}(\boldsymbol{x})$ for the OvA case, and $p \equiv p_\theta(\boldsymbol{x}|y_a), q \equiv p_\theta(\boldsymbol{x}|y_b)$ for the OvO case. In both scenarios, we aim at finding an expectation over the data variable x using only the respectively known and estimable terms $p_\theta(y|\boldsymbol{x})$ and $p_\theta(y)$.

OvA scenario For the first term, we use Bayes' theorem twice to get an expectation over two variables $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ drawn from the data distribution.

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}'_a\sim p} &= \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}'_a\sim p_\theta(\boldsymbol{x}|y)}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}'_a)\right], \\ &= \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}'_a\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)p_\theta(y|\boldsymbol{x}'_a)}{p_\theta(y)^2}\kappa(\boldsymbol{x}_a,\boldsymbol{x}'_a)\right].\end{aligned}$$

For the second term, we do not need to perform anything particular as we directy get an expectation over the data variabes $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$.

$$\mathbb{E}_{\boldsymbol{x}_b,\boldsymbol{x}'_b\sim q} = \mathbb{E}_{\boldsymbol{x}_b,\boldsymbol{x}'_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}_b,\boldsymbol{x}'_b)\right].$$

The last term only needs Bayes theorem once, for the distribution q is directly replaced by the data distribution $p_{\text{data}}(\boldsymbol{x})$:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}_a\sim p,\boldsymbol{x}_b\sim q} &= \mathbb{E}_{\boldsymbol{x}_a\sim p_\theta(\boldsymbol{x}|y),\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right], \\ &= \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim px}\left[\frac{p_\theta(y|\boldsymbol{x}_a)}{p_\theta(y)}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right].\end{aligned}$$

Note that for the last term, we could replace $p_\theta(y|\boldsymbol{x}_a)$ by $p_\theta(y|\boldsymbol{x}_b)$; that would not affect the result since $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are independently drawn from $p_{\text{data}}(\boldsymbol{x})$. We thus replace all terms, and do not forget to put a square root on the entire sum since we have computed so far the squared MMD:

$$\begin{aligned}\mathcal{I}^{\text{ova}}_{\text{MMD}} &= \mathbb{E}_{p_\theta(y)}\left[\text{MMD}(p_\theta(\boldsymbol{x}|y), p_{\text{data}}(\boldsymbol{x}))\right], \\ &= \mathbb{E}_{p_\theta(y)}\left[\left(\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}'_a\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)p_\theta(y|\boldsymbol{x}'_a)}{p_\theta(y)^2}\kappa(\boldsymbol{x}_a,\boldsymbol{x}'_a)\right]\right.\right. \\ &\qquad \left.\left. +\mathbb{E}_{\boldsymbol{x}_b,\boldsymbol{x}'_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\kappa(\boldsymbol{x}_b,\boldsymbol{x}'_b)\right] - 2\mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_b\sim p_{\text{data}}(\boldsymbol{x})}\left[\frac{p_\theta(y|\boldsymbol{x}_a)}{p_\theta(y)}\kappa(\boldsymbol{x}_a,\boldsymbol{x}_b)\right]\right)^{\frac{1}{2}}\right].\end{aligned}$$

Since all variables $\boldsymbol{x}_a$, $\boldsymbol{x}'_a$, $\boldsymbol{x}_b$ and $\boldsymbol{x}'_b$ are independently drawn from the same distribution $p_{\text{data}}(\boldsymbol{x})$, we can replace all of them by the variables $\boldsymbol{x}$ and $\boldsymbol{x}'$. We then use the linearity of the expectation and factorise by the kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}')$:

$$\mathcal{I}^{\text{ova}}_{\text{MMD}} = \mathbb{E}_{p_\theta(y)} \left[\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}' \sim p_{\text{data}}(\boldsymbol{x})} \left[\kappa(\boldsymbol{x}, \boldsymbol{x}') \left(\frac{p_\theta(y|\boldsymbol{x}) p_\theta(y|\boldsymbol{x}')}{p_\theta(y)^2} + 1 - 2\frac{p_\theta(y|\boldsymbol{x})}{p_\theta(y)} \right) \right]^{\frac{1}{2}} \right].$$

OvO scenario The two first terms of the OvO MMD are the same as the first term of the OvA setting, with a simple subscript a or b at the appropriate place. Only the negative term changes. We once again use Bayes' theorem twice:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}_a \sim p, \boldsymbol{x}_b \sim q}[\kappa(\boldsymbol{x}_a, \boldsymbol{x}_b)] &= \mathbb{E}_{\boldsymbol{x}_a \sim p_\theta(\boldsymbol{x}|y_a), \boldsymbol{x}_b p_\theta(\boldsymbol{x}|y_b)} \left[\kappa(\boldsymbol{x}_a, \boldsymbol{x}_b) \right], \\
&= \mathbb{E}_{\boldsymbol{x}_a, \boldsymbol{x}_b \sim p_{\text{data}}(\boldsymbol{x})} \left[\frac{p_\theta(y_a|\boldsymbol{x}_a)}{p_\theta(y_a)} \frac{p_\theta(y_b|\boldsymbol{x}_b)}{p_\theta(y_b)} \kappa(\boldsymbol{x}_a, \boldsymbol{x}_b) \right].
\end{aligned}$$

The final sum is hence similar to the OvA:

$$\begin{aligned}
\mathcal{I}^{\text{ovo}}_{\text{MMD}} &= \mathbb{E}_{p_\theta(y_a), p_\theta(y_b)} \left[\text{MMD}(p_\theta(\boldsymbol{x}|y_a), p_\theta(\boldsymbol{x}|y_b)) \right], \\
&= \mathbb{E}_{p_\theta(y_a), p_\theta(y_b)} \left[\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}' \sim p_{\text{data}}(\boldsymbol{x})} \left[\kappa(\boldsymbol{x}, \boldsymbol{x}') \left(\frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_a|\boldsymbol{x}')}{p_\theta(y_a)^2} + \frac{p_\theta(y_b|\boldsymbol{x}) p_\theta(y_b|\boldsymbol{x}')}{p_\theta(y_b)^2} \right.\right.\right. \\
&\quad \left.\left.\left. -2\frac{p_\theta(y_a|\boldsymbol{x}) p_\theta(y_b|\boldsymbol{x}')}{p_\theta(y_a) p_\theta(y_b)} \right) \right]^{\frac{1}{2}} \right].
\end{aligned}$$

D.3 Wasserstein distance

To compute the Wasserstein distance between the distributions $p_\theta(\boldsymbol{x}|y = k)$, we estimate it using approximate distributions. We replace $p_\theta(\boldsymbol{x}|y = k)$ by a weighted sum of Dirac measures on specific samples $\boldsymbol{x}_i$: p^k_N:

$$p_\theta(\boldsymbol{x}|y = k) \approx \sum_{i=1}^{N} m^k_i \delta_{\boldsymbol{x}_i} = p^k_N,$$

where $\{m^k_i\}_{i=1}^N$ is the set of weights. We now show that computing the Wasserstein distance between these approximates converges to the correct distance. We first need to show that p^k_N weakly converges to p. To that end, let f be any bounded and continuous function. Computing the expectation of such through p_θ is:

$$\mathbb{E}_{\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y=k)}[f(\boldsymbol{x})] = \int_{\mathcal{X}} f(\boldsymbol{x}) p_\theta(\boldsymbol{x}|y = k) d\boldsymbol{x},$$

which can be estimated using self-normalised importance sampling (Owen, 2013, Chapter 9). The proposal distribution we take for sampling is $p_{\text{data}}(\boldsymbol{x})$. Although we cannot evaluate both $p_\theta(\boldsymbol{x}|y)$ and $p_{\text{data}}(\boldsymbol{x})$ up to a constant, we can evaluate their ratio up to a constant which is sufficient:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y=k)}[f(\boldsymbol{x})] &= \int_{\mathcal{X}} f(\boldsymbol{x}) \frac{p_\theta(\boldsymbol{x}|y = k)}{p_{\text{data}}(\boldsymbol{x})} p_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}, \\
&= \int_{\mathcal{X}} f(\boldsymbol{x}) \frac{p_\theta(y = k|\boldsymbol{x})}{p_\theta(y = k)} p_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}, \\
&\approx \sum_{i=1}^{N} f(\boldsymbol{x}_i) \frac{p_\theta(y = k|\boldsymbol{x} = \boldsymbol{x}_i)}{\sum_{j=1}^{N} p_\theta(y = k|\boldsymbol{x} = \boldsymbol{x}_j)}.
\end{aligned}$$

Now, by noticing in the last line that the importance weights are self normalised and add up to 1, we can identify them as the point masses of our previous Dirac approximations:

$$m_i^k = \frac{p_\theta(y = k|\boldsymbol{x} = \boldsymbol{x}_i)}{\sum_{j=1}^N p_\theta(y = k|\boldsymbol{x} = \boldsymbol{x}_j)}.$$

This allows to write that the Monte Carlo estimation through importance sampling of the expectation w.r.t $p_\theta(\boldsymbol{x}|y = k)$ is directly the expectation taken on the discrete approximation p_N^k. We can conclude that there is a convergence between the two expectations owing to the law of large numbers:

$$\lim_{N \to +\infty} \mathbb{E}_{\boldsymbol{x} \sim p_N^k}[f(\boldsymbol{x})] = \mathbb{E}_{\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y=k)}[f(\boldsymbol{x})].$$

Since f is bounded and continuous, the portmanteau theorem (Billingsley, 2013) states that p_N^k weakly converges to $p_\theta(\boldsymbol{x}|y = k)$ when defining the importance weights as the normalised predictions cluster-wise.

To conclude, when two series of measures p_N and q_N weakly converge respectively to p and q, so does their Wasserstein distance (Villani, 2009, Corollary 6.9), hence:

$$\lim_{N \to +\infty} \mathcal{W}_c(p_N^{k_1}, p_N^{k_2}) = \mathcal{W}_c\left(p_\theta(\boldsymbol{x}|y = k_1) \| p_\theta(\boldsymbol{x}|y = k_2)\right). \tag{22}$$

E More information and experiment on the Gaussian and Student-t distributions mixture

E.1 Generative process of Gaussian and Student Mixture

We describe here the generative protocol for the Gaussian and Student mixture dataset. Each cluster distribution is centered around a mean μ_i which proximity is controlled by a scalar α. For simplicity, all covariance matrices are the identity scaled by a scalar σ. We define:

$$\mu_1 = [\alpha, \alpha], \quad \mu_2 = [\alpha, -\alpha], \quad \mu_3 = [-\alpha, \alpha], \quad \mu_4 = [-\alpha, -\alpha].$$

To sample from a multivariate Student-t distribution, we first draw samples x from a centered multivariate Gaussian distribution. We then sample another variable u from a χ^2-distribution using the degrees of freedom ρ as parameter. Finally, x is multiplied by $\sqrt{\frac{\rho}{u}}$, yielding samples from the Student-t distribution.

E.2 Extended experiment

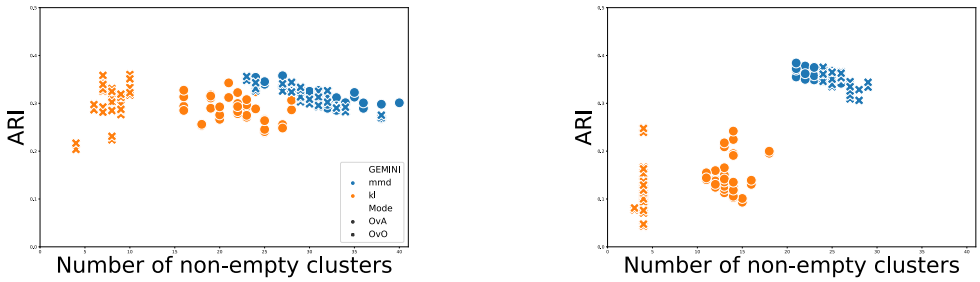
For the main experiment, we fixed $\sigma = 1$, $\alpha = 5$ and the degree of freedom $\rho = 1$. We further tested our method when training a MLP for 4 or 8 clusters. For both cases, we also considered $\rho = 1$ and $\rho = 2$. We include the complete results in Table 6.

F Model selection on MNIST

We repeat our previous experiment on the MNIST dataset from Sec. 4.4. We choose this time to get 50 clusters at best for both the MI and the MMD GEMINI and train the models for 100 epochs. We repeat the experiment 20 times per model and plot the resulting scores in figures 8a and 8b. We did not choose to test with the Wasserstein GEMINI because its complexity implies a long training time for 50 clusters, as explained in App. G. We first observe in Fig. 8 that the MMD-GEMINI with linear kernel has a tendency to exploit more clusters than the MI. The model converges to approximately 30 clusters in the case of the MLP and 25 for the LeNet-5 model with less variance. We can further observe that for all metrics the choice of architecture impacted the number of non-empty clusters after training. Indeed, by playing a key role in the decision boundary shape, the architecture may limit the number of clusters to be found: the MLP can draw more complex boundaries compared to

Table 6: Mean ARI (std) of a MLP fitting a mixture of 3 Gaussian and 1 Student-t multivariate distributions compared with Gaussian Mixture Models and K-Means. The model can be tasked to find either 4 or 8 clusters at best and the Student-t distribution has ρ=1 or 2 degrees of freedom. Bottom line presents the ARI for the maximum a posteriori of an oracle aware of all parameters of the data.

Model	$\rho = 2$		$\rho = 1$	
	4 clusters	8 clusters	4 clusters	8 clusters
K-Means	0.965 (0)	0.897 (0.040)	0 (0)	0.657 (0.008)
GMM (full covariance)	0.972 (0)	0.868 (0.042)	0 (0)	0.610 (0.117)
GMM (diagonal covariance)	**0.973 (0)**	0.862 (0.048)	0.024 (0.107)	0.660 (0.097)
$\mathcal{I}_{\mathrm{KL}}^{\mathrm{ova}}$	0.883 (0.182)	0.761 (0.101)	**0.939 (0.006)**	0.742 (0.092)
$\mathcal{I}_{\mathrm{KL}}^{\mathrm{ovo}}$	0.731 (0.140)	0.891 (0.129)	0.723 (0.114)	0.755 (0.163)
$\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ova}}$	0.923 (0.125)	**0.959 (0.043)**	0.906 (0.103)	0.86 (0.087)
$\mathcal{I}_{\mathrm{H}^2}^{\mathrm{ovo}}$	0.926 (0.112)	0.951 (0.059)	0.858 (0.143)	0.887 (0.074)
$\mathcal{I}_{\mathrm{TV}}^{\mathrm{ova}}$	0.940 (0.097)	0.973 (0.004)	0.904 (0.104)	**0.925 (0.103)**
$\mathcal{I}_{\mathrm{TV}}^{\mathrm{ovo}}$	0.971 (0.005)	0.620 (0.053)	**0.938 (0.005)**	0.595 (0.055)
$\mathcal{I}_{\mathrm{MMD}}^{\mathrm{ova}}$	0.953 (0.060)	0.940 (0.033)	0.922 (0.004)	**0.908 (0.016)**
$\mathcal{I}_{\mathrm{MMD}}^{\mathrm{ovo}}$	0.968 (0.001)	0.771 (0.071)	0.921 (0.007)	0.849 (0.048)
$\mathcal{I}_{\mathcal{W}}^{\mathrm{ova}}$	0.897 (0.096)	0.896 (0.021)	0.915 (0.131)	0.889 (0.051)
$\mathcal{I}_{\mathcal{W}}^{\mathrm{ovo}}$	0.970 (0.002)	0.803 (0.067)	0.922 (0.006)	0.817 (0.042)
Oracle	0.991		0.989	



(a) MLP (b) LeNet5

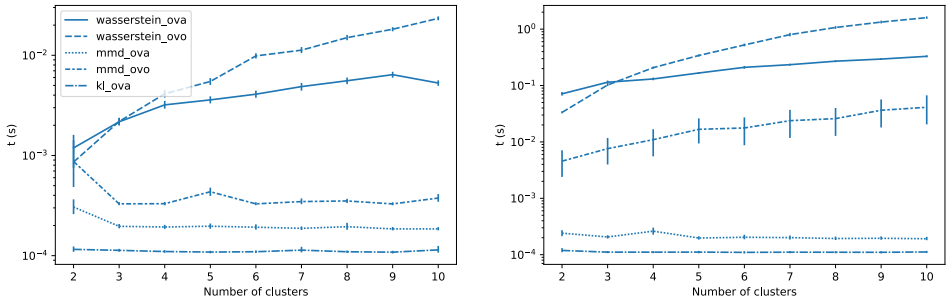
Figure 8: Distributions of the ARI scores given a number of non-empty clusters after 100 epochs of training on MNIST on two different architectures.

the LeNet5 model. Moreover, we suppose that the cluster selection behaviour of GEMINI may be due to optimisation processes. Indeed, we optimise estimators of the GEMINI rather than the exact GEMINI. Finally, Fig. 8 also confirms from Table. 3 the stability of the MMD-GEMINI regarding the ARI despite the change of architecture whereas the MI is affected and shows poor performance with the LeNet-5 architecture.

G Choosing a GEMINI

The complexity of GEMINI increases with the distances previously mentionned depending on the number of clusters K and the number of samples per batch N. It ranges from $\mathcal{O}(NK)$ for the usual MI to $\mathcal{O}(K^2N^3 \log N)$ for the Wasserstein-GEMINI-OvO. As an example, we show in Figure 9 the average time of GEMINI as the number of tasked clusters increases for both 10 samples per batch (Figure 9a) and 500 samples (Figure 9b). The batches consists in randomly generated prediction and distances or kernel between randomly generated data.

The Wasserstein-OvO is the most complex, and so its usage should remain for 10 clusters or less overall. The second most time-consuming loss is the Wasserstein-OvA, however its tendency in



(a) 10 samples per batch

(b) 500 samples per batch

Figure 9: Average performance time (in seconds) of GEMINIs as the number of tasked clusters grows for batches of size 10 and 500 samples.

optimisation to only find 2 clusters makes it a inappropriate. The main difference also to notice between the following MMD is regarding their memory complexity. The MMD-OvA requires only $\mathcal{O}(KN^2)$ while the MMD-OvO requires $\mathcal{O}(K^2N^2)$. This memory complexity should be the major guide to choosing one MMD-GEMINI or the other. Thus, the minimal time-consuming and resource-demanding GEMINI is the MMD-OvA if we consider GEMINIs that incorporates knowledge of data through kernels and distances. Other versions involving f-divergences have in fact the same complexity as MI in our implementations, apart from the TV-OvO which reaches $\mathcal{O}(K^2N)$ in our implementation.

H All pair shortest paths distance

Sometimes, using distances such as the ℓ_2 may not capture well the shape of manifolds. To do so, we derive a metric using the all pair shortest paths. Simply put, this metric consists in considering the number of closest neighbors that separates two data samples. To compute it, we first use a sub-metric that we note d, say the ℓ_2 norm. This allows us to compute all distances d_{ij} between every sample i and j. From this matrix of sub-distances, we can build a graph adjacency matrix W following the rules:

$$W_{ij} = \begin{cases} 1 & d_{ij} \leq \epsilon \\ 0 & d_{ij} > \epsilon \end{cases}, \tag{23}$$

where ϵ is a chosen threshold such that the graph has sparse edges. Our typical choice for ϵ is the 5% quantile of all d_{ij}.

We chose the graph adjacency matrix to be undirected, owing to the symmetry of d_{ij} and unweighted. Indeed, solving the all-pairs shortest paths involves the Floyd-Warshall algorithm (Warshall, 1962; Roy, 1959) which complexity $\mathcal{O}(n^3)$ is not affordable when the number of samples n becomes large. An undirected and unweighted graph leverages performing n times the breadth-first-search algorithm, yielding a total complexity of $\mathcal{O}(n^2 + ne)$ where e is the number of edges. Consequently, setting a good threshold ϵ controls the complexity of the shortest paths to finds. Our final distance between two nodes i and j is eventually:

$$c_{ij} = \begin{cases} \text{Shortest-path}^W(i,j) & \text{if it exists.} \\ n & \text{otherwise} \end{cases}. \tag{24}$$

This metric c can then be incorporated inside the Wasserstein-GEMINI.

I Packages for experiments

For the implementation details, we use several packages with a python 3.8 version.

- We use PyTorch (Paszke et al., 2019) for all deep learning models and automatic differentiation, as well as NumPy (Harris et al., 2020) for arrays handling.
- We use Python Optimal Transport's function `emd2`(Flamary et al., 2021) to compute the Wasserstein distances between weighted sums of Diracs.
- We used the implementation of SIMCLR from PyTorch Lightning (Falcon et al., 2019).
- Small datasets such as isotropic Gaussian Mixture of score computations are performed using scikit-learn(Pedregosa et al., 2011).
- All figures were generated using Pyplot from matplotlib (Hunter, 2007).