Learning Representations using Neural Networks and Optimal Transport

Warith Harchaoui

LEARNING REPRESENTATIONS USING NEURAL NETWORKS AND OPTIMAL TRANSPORT

RÉSEAUX DE NEURONES ET TRANSPORT OPTIMAL POUR L'APPRENTISSAGE DE REPRÉSENTATIONS

PAR WARITH HARCHAOUI

DIRIGÉ PAR PR. CHARLES BOUVEYRON

OSCARO.COM – RECHERCHE ET DÉVELOPPEMENT

UNIVERSITÉ DE PARIS
LABORATOIRE MAP5 – UMR CNRS 8145
ÉCOLE DOCTORALE 386
SCIENCES MATHÉMATIQUES DE PARIS CENTRE

THÈSE DE DOCTORAT EN MATHÉMATIQUES APPLIQUÉES
PRESENTÉE ET SOUTENUE À PARIS, LE 8 OCTOBRE 2020

DEVANT UN JURY COMPOSÉ DE

ERWANN LE PENNEC	RAPPORTEUR	PROFESSEUR	ÉCOLE POLYTECHNIQUE
NICOLAS COURTY	RAPPORTEUR	PROFESSEUR	UNIVERSITÉ BRETAGNE SUD
JULIE DELON	EXAMINATRICE	PROFESSEURE	UNIVERSITÉ DE PARIS
LAURE SOULIER	EXAMINATRICE	MAÎTRE DE CONFÉRENCES	SORBONNE UNIVERSITÉ
PIERRE-ALEXANDRE MATTEI	EXAMINATEUR	CHARGÉ DE RECHERCHE	INRIA SOPHIA-ANTIPOLIS
ANDRÉS ALMANSA	MEMBRE INVITÉ	DIRECTEUR DE RECHERCHE	UNIVERSITÉ DE PARIS
CHARLES BOUVEYRON	DIRECTEUR DE THÈSE	PROFESSEUR	UNIVERSITÉ CÔTE D'AZUR
STÉPHANE RAUX	CO-ENCADRANT	RESPONSABLE DÉVELOPPEMENT R&D	OSCARO.COM

LEARNING REPRESENTATIONS USING NEURAL NETWORKS AND OPTIMAL TRANSPORT

BY WARITH HARCHAOUI

SUPERVISED BY PR. CHARLES BOUVEYRON

OSCARO.COM – R&D

UNIVERSITÉ DE PARIS

LABORATOIRE MAP5 – UMR CNRS 8145

La pire chose qu'on puisse te dire
c'est que t'es intelligent.

The worst thing one can tell you is that you're intelligent.

A. Harchaoui to his son, 1991

ABSTRACTS IN FRENCH AND ENGLISH

RÉSUMÉ EN FRANÇAIS

La dernière décennie a vu les réseaux de neurones devenir un outil de référence dans l'apprentissage statistique. En effet, cette technologie s'est imposée comme un outil essentiel pour des types de données aussi variés que les images, le texte, le son, etc. dans des contextes à grande échelle. Les succès des réseaux neuronaux s'étendent également à l'apprentissage par renforcement (jeux et robotique) et à l'apprentissage non-supervisé (analyse et génération de données), avec notamment une qualité inégalée pour l'imitation d'images avec les réseaux génératifs adversaires. Néanmoins, les réseaux neuronaux restent difficiles à interpréter en tant qu'estimateurs statistiques. L'objectif de cette thèse est d'atténuer cet inconvénient et d'accroître encore la portée des réseaux de neurones grâce à trois applications dites d'intelligence artificielle : (i) le partitionnement des données en groupes (*clustering*) grâce à un algorithme qu'on propose, (ii) la recherche des coordonnées pertinentes d'un signal avec une notion qu'on a baptisé « la pire distance de Wasserstein » et (iii) la prédiction d'un résultat munie de l'estimation d'une incertitude associée en revisitant et en étendant les méthodes connues.

A travers ces trois contributions, nous nous proposons de répondre à trois questions sur les représentations : (i) Comment représentons-nous les objets qui appartiennent aux groupes que nous essayons de former ? (ii) Comment représenter ce qui fait qu'un objet ressemble au reste des objets de son type ? (iii) Comment représenter une incertitude associée à une prédiction automatique ? Ainsi, ce travail présente des moyens de s'éloigner de l'utilisation supervisée classique du *deep learning* (le domaine de l'apprentissage statistique des réseaux de neurones) avec la volonté d'interpréter ces mystérieuses « boîtes noires » grâce à de nouveaux outils algorithmiques et statistiques. Nous avons veillé à ce que notre utilisation des réseaux de neurones soit la plus interprétable possible pour une meilleure compréhension des données en jeu, au-delà des décisions automatiques.

Mots-clefs

Réseaux de neurones, apprentissage de représentations, partionnement de données (clustering), réseaux génératifs adversaires de Wasserstein, dualité de Kantorovich, transport optimal de Monge, modèle de mélange gaussien, modèles génératifs profonds, auto-encodeur, estimation de l'incertitude

ABSTRACT IN ENGLISH

The last decade has seen neural networks become a reference tool in statistical learning. Indeed, this technology has established itself as an essential tool for data types as varied as images, text, sound, etc. in large scales settings. The success of neural networks also extends to reinforcement learning (games and robotics) and unsupervised learning (data analysis and generation) including unparalleled quality for image imitation with generative adversary networks. Nevertheless, neural networks remain difficult to interpret as statistical estimators. The aim of this thesis is to mitigate this drawback and further enhance the scope of neural networks through three so-called artificial intelligence applications: (i) partitioning data into groups (*clustering*) thanks to a proposed algorithm, (ii) finding the relevant coordinates of a signal with a notion that we dub "the worst Wasserstein distance" and (iii) predicting a result while estimating an associated uncertainty by revisiting and extending known methods.

Through these three contributions, we propose to answer three questions about representations: (i) How do we represent objects that belong to the groups we are trying to form? (ii) How do we represent what makes an object resemble its kind? (iii) How do we represent an uncertainty associated with an automatic prediction? Thus, this work presents ways to get away from the classic supervised use of *deep learning* (the statisical learning field related to neural networks) with the desire to interpret these mysterious so-called *black boxes* thanks to new algorithmic and statistical tools. We tried hard to make sure that our use of neural networks is as interpretable as possible for a better understanding of the data at stake, beyond automatic decisions.

Keywords

Neural networks, representation learning, clustering, Wasserstein generative adversarial networks, Kantorovich duality, Monge optimal transport, Gaussian mixture model, deep generative models, auto-encoder, uncertainty estimation

REMERCIEMENTS

À mes amis,

ACKNOWLEDGEMENTS

To my friends,

CONTENTS

CONTENTS

1

STATE OF THE ART

1.1 INTRODUCTION

Through perception and experience, each human being gathers data all the time and eventually process that data into representations that become our grasp onto the universe for interacting. We represent ideas and concepts for thinking and language and even get *pre*-representations of things we do not know. Although *artificial intelligence* has become a very popular expression, one can remark that representation is a key prerequisite for any of *processing* that we consider whether we talk about our intelligence as human beings our artificial intelligence for automatic machines. On one side of the spectrum of possibilities, excellent representations do not even require further processing and simple decision mechanisms on raw data are enough for good results. On the other side of the spectrum of possibilities, poor representations actually do require sophisticated expert-based or advanced statistical processing to output good results by taking into account domain-specific knowledge. During the 2010s, some major technological obstacles were crossed allowing the development of the so-called deep learning marked by the coming of a new era where the frontier between representation and processing becomes very blurry.

In many machine learning fields, defining a clear and sound objective is always key to produce good research but unfortunately, several so-called more intelligent tasks are impossible to define that well which leads us in this dissertation in statistics to insist on the statistical representation side rather than on the statistical processing side. Therefore, our main subject is representation: (i) of data among its groups in the first chapter, (ii) of data among its characteristics in the second chapter and (iii) of predictions with uncertainty in the third chapter. At the crossroads of three different fields namely statistics, deep learning and optimal transport that recently gained much scientific attention, our effort leverages that wealth of existing research to tackle representations in three contributions with different contexts: Wasserstein Clustering (DiWaC and GeWaC) to identify groups among data, Infinitesimal Wasserstein Maximal Distortion (InWaMaDi) to highlight relevant characteristics of data and Hypothesis of an Uncertainty Model (HUM) to estimate both supervised predictions and some uncertainty information.

1.2 MACHINE LEARNING LANDSCAPE

This dissertation focuses on neural networks (a. k. a. *deep* techniques) that enjoy recent tremendous sucess as a technology but much of the current described work is applicable to other tools like decision trees [Breiman, 2017] or kernel-based methods [Andrew, 2001]. That being said, *deep learning* changed scientists traditions about data: usually researchers separated feature-extraction and automatic-decision making tasks into two jobs. Nowadays, this frontier becomes merely a blurred line as best systems are built by doing feature extraction and automatic decision simultaneously along layers (hence the "deep" adjective as more layers give more sophisticated systems). Mixing feature extraction and automatic decision in layers amplified by much more computation power than ever let neural networks become fancy again. The whole machine learning scientific community beyond those who uses neural networks refer to these low-financial-support periods as "artificial intelligence winters". In the 1990s and 2000s, even in a major conference named *Neural Information Processing Systems*, popular methods like kernel-based ones were more popular than neural networks at a time when the keyphrase "deep learning" was merely confidential. As announced in the seminal keynote at the International Joint Conference on neural networks in 2011, neural networks were back again in the performance leaderboards top methods thanks to recent hardware considerable computational improvements. Then, between 2011 and 2014, neural networks algorithms beat state-of-the-art records in several fields such as image processing, computer vision, speech recognition, machine translation with almost only artificial neural networks scientists and without researchers from specific domain expertise as explained by Ng [2013] which gave the surprising hope for an increasing ease in a growing list of applications.

During the last decade, a pleasing *end-to-end* paradigm emerged and says that systems should not be trained sequentially (or even independently) but rather simultaneously as a whole [Ng, 2018]. This intuitive belief that consists in training several layers of a neural network at once is widespread for better empirical results. Unfortunately, doing an end-to-end training also means dealing with black boxes as intermediate neural networks layers that are difficult to interpret (and are even not identifiable whereas other less efficient methods still provide interpretation ease). End-to-end training seems to be encouraged for better than other known styles of training empirically and especially in Deep Learning [Bojarski et al., 2016].

Nevertheless, this statement must be handled with care for pragmatic reasons beyond industrial scalability and loss of interpretable modularity as Glasmachers [2017] points out there are also some other effects: feeding a deep neural network with the concatenation of raw data and some non-deep-learning algorithms outputs is often hard to beat. For example, in video recognition [Schmid, 2013, Crasto et al., 2019], it is recommended to augment the raw video voxels input with optical flow (which is a processed version of the same raw video pixels

but for motion estimation). Indeed, because of the *data starvation* phenomenon (a. k. a. over-fitting), using off-the-shelves pretrained algorithms is a simplistic form of transfer learning combining knowledge (and sometimes data) from the current and from the previous tasks. Thus, pragmatically, it is sometimes useful not to follow the end-to-end-training approach just for the sake of it: sequentially trained and/or optimized modules can work very well and still provide the easiest interpretation for what each module does. One too naive end-to-end-training approach would end up with a black box trained from scratch.

In this dissertation, there is a will to emphasize our scientific need to crack in deep learning black boxes (end-to-end or not) because that's how better data understanding gets in, beyond automatic decisions.

1.2.1 *Dimensions*

In this dissertation, we will consider large scale settings so we must be specific about what is considered *large*. Throughout the machine-learning-related fields, we can consider:

CARDINALITY N, the number of data points for train;

DIMENSIONALITY D, the dimensionality of one data point;

OUTPUT DIMENSION K, the dimensionality of the automatic output decision (number of classes in classification, the output space dimension for regression and beyond, the approximate number of nodes in a grammar tree, or number of atoms of a chemical molecule graph...).

as emphasized by Harchaoui [2013] in the concept of *machine learning cuboid*. For each edge of this cuboid, we have direct optimization implications for feasible computations and best results so far:

- $N \gg 1$ stochastic-gradient-based optimization algorithms are more suitable than in-memory alternatives because we only need a few data points (mini-batches) at the same time per iteration;
- $D \gg 1$ some further analysis should be conducted: dimensionality reduction and domain-specific knowledge must be used at once for fighting against the *curse of dimensionality* phenomenon;
- $K \gg 1$ one-class-vs-rest strategies are preferred rather than one-class-vs-one strategies for computational reasons. Indeed in a one-class-vs-rest strategy, we only need to combine K decisions separating each class with all the rest whereas in a one-class-vs-one strategy, we would have $\frac{K \times (K-1)}{2}$ decisions separating all combinations of classes pair.

Throughout this work, we are mainly interested in large N and large D data configurations without considering large K issues. For example, in clustering settings, K should be small because otherwise it defeats the data analysis purpose: having too many clusters does not help human beings to understand data. Although the large N and large D case has already been successfully investigated in the supervised classification context, at the beginning of this work (in 2016) little research had been conducted but we observe that this key preoccupation is finally entitled to scientific attention today.

Along these three dimensions of data analysis in machine learning, this Ph.D. dissertation proposes new (or revisited) representations: (i) simplifying the *cardinality axis of N* thanks to clustering in our first contribution, (ii) an attempt to better understand data at a coordinate level for a local *dimensionality axis of D* relevance assessment in our second contribution, and (iii) re-interpreting the *output axis of K* through uncertainty estimation in our third contribution.

1.2.2 *Epistemology*

Epistemology is the theory of knowledge [Newman, 2018, Ahmad, 2003]. In particular, in machine learning, one epistemology has consequences on our beliefs, opinions, justifications and finally our scientific methodology in our studies. Several epistemological ways to describe machine learning exist and for this dissertation we choose one with probabilistic perspectives [Murphy, 2012] because of its ease of sophisticated interpretation for insights. We believe data is coming from a phenomenon that we call *Nature* that we represent by an idealized probabilistic distribution associated with a random and often multivariate variable (or a pair or a tuple of this random multivariate variables). In practice, we consider datasets as extracts of Nature. An (annotated) dataset is some collection of independent realizations that are identically (sampled) distributed from what we call Nature. Of course, this statement is falsifiable [Bernard, 1898] and maybe counter-intuitive but computer science, statistical learning, machine learning, data science and all these young sciences mixing mathematics and programming are just a few decades old, compared to more established several centuries old (or even millenia old) fields such as mathematics, biology, physics, chemistry, medecine etc.

Mathematically, it is convenient to choose that epistemology (rather than an other one) in order to introduce the notion of generalization capabilities of machine learning systems. Indeed, with other epistemology, datasets have a higher status and generalization becomes ill-defined (with the *ad hoc* notion of training error and testing generalization error). In this work, we accept that datasets empirical distributions (sum of Dirac distributions) are approximated and noisy versions of an idealized (probably smoother) distribution.

1.2.3 *Different kinds of machine learning*

Generally, for a given project, the job of a machine learning scientist (or data scientist depending on the name given by economic trends) is decomposed in two phases often in a loop: (i) training time (previous to or interleaved with a validation time) to match/imitate/reproduce the phenomenon in the presence of groundtruth information (labels, reward) for learning a model, (ii) test time (or execution time of the system we have just built) without access to groundtruth information because we are using our trained model. Following pioneers in machine learning [LeCun, 2015], we can roughly separate the machine learning landscape in three depending on what is accessible during training, validating and testing times: first supervised learning, second unsupervised learning and third reinforcement learning as this dissertation can find applications in all of these three machine learning fields.

For the purposes of notation, univariate functions are here generalized to the multivariate case by applying the associated univariate function to each entry and then concatenating everything such that the function output has the same shape as the input. For example, $\mathbf{z} \in \mathbb{R}^K$ is a vector whose $K \in \mathbb{N}^*$ coordinates $\mathbf{z}^{(k)}$ are indexed by k, then $\log(\mathbf{z}) = \left[\log(\mathbf{z}^{(1)}), \ldots, \log(\mathbf{z}^{(k)}), \ldots, \log(\mathbf{z}^{(K)})\right]^\top$.

Supervised Learning

Nature provides a pair of (input, output) random variables $(\mathbf{x}, \mathbf{y})$.

$$(\mathbf{x}, \mathbf{y}) \sim \text{Nature} \tag{1}$$

collected in a training labelled dataset (or *Nature extract* as stated above):

$$\begin{gathered} \text{dataset} = (\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_i, \mathbf{y}_i), \ldots, (\mathbf{x}_N, \mathbf{y}_N) \\ \text{where each } (\mathbf{x}_i, \mathbf{y}_i) \text{ is a realization of } (\mathbf{x}, \mathbf{y}) \sim \text{Nature} \end{gathered} \tag{2}$$

On the one hand, input $\mathbf{x}$ often represents a question in various forms such as: a vector of numbers (categories, integers or floating decimal numbers), an image or a video (made of pixels, voxels in channels and beyond [Ponce and Forsyth, 2011]), a sound (its waveform or its time-frequency representation [Li et al., 2016, Mallat, 2008]), a gene (its ATGC or RNA-seq representation [Barillot et al., 2012]), a chemical molecule (its 3D graphical representation [Zaslavskiy, 2010]) etc. On the other hand, output $\mathbf{y}$ represents the answer of the question $\mathbf{x}$ in two main classes of problems: regression in which we deal real numbers and classification dealing with categories and integers. Of course, these kinds of separations are limited but somewhat useful to describe the main problems.

Many supervised learning problems share the same kind of optimization objective:

$$\min_{\mathcal{F}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{Nature}} \left(\ell\left(\mathbf{y}, \mathcal{F}(\mathbf{x})\right)\right) \tag{3}$$

where $\ell(\mathbf{y}, \mathcal{F}(\mathbf{x}))$ measures the discrepancy of predicting $\mathcal{F}(\mathbf{x})$ from an input $\mathbf{x}$ instead of the ground truth label $\mathbf{y}$. This equation Eq. (3) behaves like an aggregation of errors (when $\mathbf{y} \neq \mathcal{F}(\mathbf{x})$) summed up into one value (the lower, the better) over data. In layman's terms, $\ell(\mathbf{y}, \mathcal{F}(\mathbf{x}))$ is *how much the system is punished for a mistake during training* and is preferably zero for no error: a perfect $\mathbf{y} = \mathcal{F}(\mathbf{x})$ scenario. More precisely, statistical learning becomes the task of finding parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathcal{F}}$ of function $\mathcal{F}$ that has the right structure (tree, random forest, linear or kernel-based support vector machines, neural networks etc.) that minimizes $\mathcal{L}$:

$$\begin{gathered} \min_{\boldsymbol{\theta}_{\mathcal{F}}} \mathcal{L}(\boldsymbol{\theta}_{\mathcal{F}}) \\ \mathcal{L}(\boldsymbol{\theta}_{\mathcal{F}}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim \text{Nature}} \left(\ell(\mathbf{y}, \mathcal{F}(\mathbf{x})) \right) \end{gathered} \tag{4}$$

Indeed, the formulation of Eq. (4) has the merit of generalizing almost all supervised learning problems.

Beyond the scope of this dissertation, there is a considerable amount of scientific works about regularization notably inherited from Lagrangian optimization [Boyd and Vandenberghe, 2014] and similar techniques. In a nutshell, our Eq. (3) is still valid to fit in this kind of research by simply replacing the current function ℓ by function $\tilde{\ell}$:

$$\tilde{\ell}(\mathbf{y}, \mathcal{F}(\mathbf{x})) \triangleq \ell(\mathbf{y}, \mathcal{F}(\mathbf{x})) + \lambda \Omega(\mathcal{F}) \tag{5}$$

where Ω gives the predictor function $\mathcal{F}$ some desirable properties (see the books of Bonnans et al. [2003] and of Boyd and Vandenberghe [2014] for further details) with a relative importance given by $\lambda \in \mathbb{R}_+$ in order to provide generalization capabilities coping with the fact that we only have access to a limited training dataset instead of Nature itself in practice.

In summary, supervised learning is finding a function $\mathcal{F}$ that maps $\mathbf{x}$ to $\mathbf{y}$ based on a training dataset assuming that such an idealized function $\mathcal{F}^*$ exists (sometimes we only need $\mathcal{F}^*$ to only be a relation and not necessarily a function):

$$\mathbf{y} \simeq \mathcal{F}^*(\mathbf{x}) \tag{6}$$

In order to build an estimator $\hat{\mathcal{F}}$ of the desired decision function $\mathcal{F}^*$, we usually solve an optimization problem:

$$\hat{\mathcal{F}} = \min_{\mathcal{F}} \mathcal{L}(\mathcal{F}) \tag{7}$$

where the loss function $\mathcal{L}$ is a proxy of all errors that we want to ideally minimize in one value $\mathcal{L}(\mathcal{F})$ measuring all aggregated discrepancies between the ground truth $\mathbf{y}$ and prediction $\mathcal{F}(\mathbf{x})$.

In practice, we only have access to a limited amount of annotated $(\mathbf{x}, \mathbf{y})$ data that we call training data to *fit* our prediction function $\hat{\mathcal{F}}$, so the $\mathcal{L}(\mathcal{F})$s are

estimated by approximations $\mathcal{L}(\hat{\mathcal{F}})$ as if the available data was all the data in the universe: almost as if the Nature smooth distribution was replaced by the dataset empirical distribution. The hope consists in saying that at test time, for a new and unseen input $\mathbf{x}$ coming from the same $(\mathbf{x}, \mathbf{y})$ distribution as in training time but where the true output $\mathbf{y}$ is unknown, we can predict an estimated output $\hat{\mathbf{y}} = \hat{\mathcal{F}}(\mathbf{x})$ that is close to the true output $\mathbf{y}$.

Learning is possible because at training time, we have access to both the input $\mathbf{x}$ and output $\mathbf{y}$ in a dataset. Back to our philosophical considerations about epistemology, we may consider that the supervised learning task consists in *compressing* the relationship between the training pairs $(\mathbf{x}, \mathbf{y})$ in a fitted predictor $\hat{\mathcal{F}}$ that once trained, one only needs $\mathbf{x}$ to recover a lossy version of $\mathbf{y} \simeq \hat{\mathcal{F}}(\mathbf{x})$. During the "compression process of learning" we accept some loss in information that we sacrifice to get better compression rate in general non-lab conditions. This way, we can study many machine learning tools such linear dot products, tree, random forest of trees, non-linear kernel evaluations, vanilla neural networks, convolutional or recurrent neural networks etc. and their respective algorithms, computations and structures with the same compression-flavored point of view.

In supervised learning we have:

$\mathbf{x}$ and $\mathbf{y}$ at training time
$\mathbf{x}$ without $\mathbf{y}$ at testing time that we estimate

Looking at supervised learning as a compression problem is interesting for understanding recurring trade-off through out this scientific literature: models should be sophisticated enough to recover outputs information from inputs without too much loss of information (complexity and number shoud go up) while still being sufficiently sober (i. e. not too sophisticated,(complexity and number goes down) otherwise generalization capabilities dramatically drop down while compression is given up.

Unsupervised Learning

In unsupervised learning, Nature does not provide any more information than the data $\mathbf{x}$ itself.

$$\mathbf{x} \sim \text{Nature} \tag{8}$$

This machine learning field is useful for data analysis that has more broader scientific purposes than the industrial applications of supervised learning.

$\mathbf{x}$ without any labeled $\mathbf{y}$ information that we still estimate to get structure from data *for the sake of interpretable knowledge discovery*

Imitating data (Generative Adversarial Networks GANs for example) [Goodfellow et al., 2014] and clustering [Jain, 2010] are two unsupervised learning tools to exhibit data analysis as a fundamentally intelligent tool for scientists, *intelligent*

etymologically meaning from latin understanding the underlying structure of data. Unsupervised learning is often ill-posed which remains a mystery because the same automatic tasks can be evaluated subjectively many times by different human beings with still some consistency (e. g. for clustering) which gives hopes for improvements. More mathematically, indeed, we use the Hadamard definition [Maz'ya and Shaposhnikova, 1999] for a well-posed problem and we understand that all these three points cannot apply in clustering objectives optimization:

1. a solution exists,
2. the solution is unique,
3. the solution's behaviour changes continuously with the initial conditions.

In this dissertation, clustering is tackled in a chapter as the computation of the maximal optimal transports between groups (e. g. clusters) while leveraging the GANs scientific literature in terms of numerical tools. Furthermore, another chapter is trying to make use of enhanced Wasserstein distance among distributions for relevant features weighting of data without explicit supervision from any task but rather the *likeness* of each point with the remaining dataset points that we express with the notion of *worst* optimal transport once again thanks to its powerful mathematics and algorithmics machineries.

Reinforcement Learning

Reinforcement Learning [Sutton and Barto, 2018] is an area of machine learning about how agents (say robots) sequentially observing environment inputs $\mathbf{x}$ from Nature could take the best sequence of actions $\mathbf{y}$ in order to maximize some untimely reward $\mathbf{r}$ also given by Nature and not necessarily after each action while maintaining a state representation $\mathbf{s}$ (or $\mathbf{z}$) of both history and environment.

$$(\mathbf{x}, \mathbf{r}) \sim \text{Nature, but } \mathbf{r} \text{ is not always given (i. e. we often have } \mathbf{r} = 0\,) \tag{9}$$

For example, playing automatically chess, Go, cards are famous applications associated with artificial intelligence victories in controlled settings face to expert human beings.

In unsupervised learning, we have:

> $\mathbf{x}$ but no supervised action $\mathbf{y}$ is given, only some reward $\mathbf{r}$ once in a while during training time
>
> $\mathbf{x}$ with some reward $\mathbf{r}$ once in a while at testing time and we estimate the best sequence of actions $\mathbf{y}$

In reinforcement learning, taking into account uncertainty estimation is certainly helpful for the distentangling the recurring exploration / exploitation dilemma [Sutton and Barto, 2018]. This opens up a potentially large range of possible applications for our contribution dealing with uncertainty.

1.3 NEURAL NETWORKS

Since their introduction in computer science [Rosenbaltt, 1957], artificial neural networks loosely inspired by the biological neurons did not stop fascinating researchers until today. In a nutshell, a neural network implements a function $\mathcal{F}$ from $\mathbb{R}^D$ to $\mathbb{R}^K$ by the composition of $L \in \mathbb{N}^*$ layers (or sub-functions) $(\mathcal{F}_\ell)_{\ell=1,\dots,L}$) of the form:

$$\begin{array}{rcl} (\forall \ell \in [\![1, L-1]\!]) \;\; \mathcal{F}_\ell & = & a \circ \text{Linear}_\ell \\ \mathcal{F}_L & = & \text{Linear}_L \end{array} \tag{10}$$

where:

- The non-linear element-wise function a called activation is often chosen among the hyperbolic tangent function, the positive part function (or rectified linear unit ReLU), the sigmoid function (or logit function).
- The Linear_ℓs functions are matrix-vector product linear operators in the form of:

$$(\forall \mathbf{x} \in \mathbb{R}^{i_\ell}) \;\; \text{Linear}_\ell(\mathbf{x}) = \mathbf{M}_\ell \mathbf{x} + \mathbf{b}_\ell \tag{11}$$

  for $\mathbf{M}_\ell \in \mathbb{R}^{o_\ell \times i_\ell}$ and $\mathbf{b}_\ell \in \mathbb{R}^{o_\ell}$ with $(i_\ell, o_\ell) \in \mathbb{N}^* \times \mathbb{N}^*$ and $(\forall \ell \in [\![1, L-1]\!]) \;\; o_\ell = i_{\ell+1}$ ($i_1 = D$ and $o_L = K$)

Thanks to the universal approximation theorem of Hornik [1991], we only need mild conditions on a (unbounded and non-constant) for the set of functions expressed in this form to be dense over the set of Lebesgue-integrable functions.

This means that each time, we have an objective function to minimize over a set a function, there exists a neural network $\mathcal{F}$ that is able to approximate the optimal solution arbitrarily well. In terms of computer science, this is good news because instead of minimizing over a set of functions, we can reasonably minimize over a set of parameters approximating that function we are looking for. Thus, we transformed a functional optimization problem into a numerical optimization problem over the matrices $\mathbf{M}_\ell$s, the biases $\mathbf{b}_\ell$s but also the number of layers L and the input/output parameters (i_ℓ, o_ℓ). The initial enthusiasm provoked by this statement was dampened through decades because this is only an existence theorem that does not provide a way in itself to find these parameters. Moreover it turns out that the statistical estimation of these parameters is difficult due to the over-fitting phenomenon [Scholkopf and Smola, 2001] (a. k. a. *data starvation* which is a complementary metaphor). Optimization is also slow (especially in the 1960s, 1970s, 1980s and even 1990s compared to nowadays).

Recently, neural networks became suddenly more plausibly useful since storage and computations were getting much faster and cheaper. Indeed, on top of a dramatic computational speed improvement, fast random data access is

also crucial for realistic real-world applications. During the 1990s, even with industrial-level quality results, scientists in neural networks for machine learning did not catch the whole Research community world wide attention at first. As data storage and collection problems have been nicely solved thanks to the decrease of hardware's costs (and thus the increase of available computational power especially with the rise of CPUs and GPUs parallelism and distribution), increase of data access speed and software solutions (like HDFS [Shvachko et al., 2010] and Spark [Zaharia et al., 2010]) that all appeared in an era around the 2000s called *Big data*. In the 2010s, these tools for manipulating data were key for large-scale training and execution of machine learning engines [Castelluccio, 2017]. Another factor of scientific success is the extensive use of world-wide open source repository which pioneered in terms of reproducibility and best practice sharing. On the theoretical part, the mild conditions of the universal approximation theorems [Cybenko, 1989, Hornik, 1991, Gao and Jojic, 2016] only gives approximation ability up to our statistical estimation ability (hence the need of big cardinality datasets compared to dimensionality). Otherwise, neural networks notoriously suffer from data starvation (a. k. a. overfitting) and the neural networks need appropriate structures, computations and optimization procedures to inject enough knowledge into the systems in order to get the tremendous sucess we benefit today. Still, in spite of these recent and great improvements, there is a lack of solid theoretical grounds (especially compared to its Reproducing Kernel Hilbert Space RKHS counterpart) and many open questions remain unanswered as of today although we can mention the works of Vidal et al. [2017] and Arora et al. [2017].

1.3.1 *Input Data*

Thanks to the universal approximation theorem presented previouly, it is reasonable to parse several kinds of input data to see how neural networks can be fed. Indeed, neural networks universal approximation theorems do only provide an existence result of a desired function but no explicit way to get or estimate it. Even with this important theorem, scientists sill have to work to build an adequate deep learning structure to cope with the estimation problem of an almost unreachably ideal neural networks parameters. Even if we had a procedure that could reach that ideal neural networks, this would be fitted on training data only which is not a guarantee for good results on unseen yet data.

Historically [Rosenbaltt, 1957], input data $\mathbf{x} \in \mathbb{R}^D$ is a vector fed to a function implemented by a one-layer perceptron: a composition of a matrix-vector product and an element-wise sigmoid function. At the same time, stochastic optimization tools were revisited, implemented for large scale settings with the associated theoretical background provided by the Robbins-Monro theorem [Robbins and Monro, 1951] (we will come back on it later). The idea of composition for more sophisticated neural networks came fast with the introduction of multilayer

perceptron [Rosenblatt, 1961] along with the back-propagation algorithm which gracefully adapts the differenciation chain rule for efficient algorithms (the *forward* step is the evaluation of the neural network function and the *backward* step is the computation of its gradient with respect its parameter). Nowadays, thanks to much engineering progress, more and more artificial intelligence promises are kept.

Since the dawn of the 1990s, new kinds of information media became available on computers storage systems with increasing sophistication and some dramatic research-to-product type of improvements emerged:

TEXT Word processor softwares rapidly replaced typewriters and made natural language processing possible and one can cite the old Reuters text dataset [Hayes and Weinstein, 1990] for example. In terms of artificial intelligence, it seems that this medium along with genetics data are the most difficult one;

IMAGE Likewise, digital recording of photographs made the *singleton* dataset of Lena [Roberts, 1962] in image processing grow from only one to several thousands images size datasets with MNIST [LeCun et al., 1989b] and to millions images size datasets such as ImageNet [Fei-Fei, 2010] and beyond. The associated tremendous research progress made possible real-world applications in everyday applications;

SOUND Early speech recognition systems also benefit from datasets collection since one of the oldest: TIMIT [Zue et al., 1990], even the music processing got its own MusicNet dataset [Thickstun et al., 2018];

VIDEO the recent 2018 YouTube 8 millions videos dataset [Abu-El-Haija et al., 2016] seem promising for same kind of quantitative-qualitative upward gap and improvements as image in a near future.

There is an interesting hypothesis to maybe understand why the text medium is so hard to manipulate within statistical frameworks compared to the other ones although it has been the first to be digitized with enormous ever-growing quantity data: close features in the other media are more clearly dependent (two neighboring pixels in images and even voxels in videos are highly dependent like consecutive sound samples are) and this dependence fades away with longer horizons but unfortunately text features (words or even letters) have stronger and longer range interactions that suggests to see them through grammar in spite of the scientifically obsolete and harmful but widespread saying from Frederick Jelinek in 1985:

> "Every time we fire a phonetician/linguist, the performance of our system goes up."

Combining strong statistical tools and linguistics is probably the best alternative for future natural language representations.

For all these different media (or information supports), a major research pattern can be analyzed: most data contain redundant information so discarding stuttered information is useful in order to manipulate the relevant infortmation only. For example, spatial data like images, have highly dependent close pixels and thus intuitively small-sized convolution kernels seem appropriate to decorrelate the redundant information. Another example is sequence data or smooth time series data where neighboring data (with respect to the sequence or temporal axes) should also be decorrelated thus 1D-convolutions [Zhang et al., 2015] or recurrence seem appropriate as much as recurrent neural networks can [Murakami and Taguchi, 1991, Hochreiter and Schmidhuber, 1997] as described in Fig. 1.

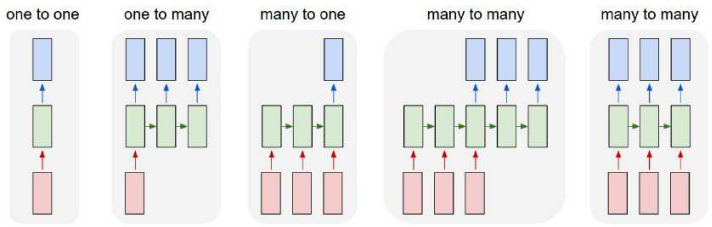

Figure 1: Several input/processing/output scenarios from standard neural networks to recursive ones. Red: Input, Green: Processing, Blue: Output. From left to right: one (input) to one (output) vanilla neural network, one to many like in image captioning (one image to many words in a sentence), many to one like in sentiment analysis (many words in a sentence to a category of mood), and the (delayed or not) many to many case like in langauage translation (many words of a sentence in one language to many words of a new sentence in a different language). Diagrams taken from the pedagogical blog of Andrej Karpathy: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

This huge research effort allowed for large scale applications (both in terms of training size or cardinality N and data size or dimensionality D and in the end the output size K). Meanwhile, image and sound processing were increasingly developping methods using convolutions when LeCun et al. [1989b,a] applied those convolutions for image classification with both iterated compositions and backpropagation which turned out to be also useful for sound and even text processing [LeCun and Bengio, 1995]. For time series, there is still a scientific debate among the deep learning community about whether recurrent or convolutional neural networks (which abreviate to RNN and CNN respectively). Meanwhile the sequential nature of words enumeration in text is tackled as time series but there is probably more hidden structure yet to get from grammar (as pioneering work from Socher et al. [2011] pointed out since 2011).

These combined breakthroughs allowed ambitious real-world applications throughout the 1990s and the 2000s. Today, new constibutions with improved engineering, structure, optimization and regularization tools are still referring to early works with even homage (e. g. GoogLeNet by Szegedy et al. [2015] referring to LeNet by LeCun et al. [1989b]). Image and Speech recognition is embedded in many everyday products, even video recognition begins to have reliable industrial applications. Natural language processing also gets impressive translation results[1] but there is still room for improvements. For all these different media, it seems that the same phenomenon occurs: unleashing clean dataset with thorough engineering effort astonishingly helps the scientific community to bring back high quality prototypes and ultimately products. Indeed, if we measure the time delay between a dataset release and available products and the best example is the AdaBoost implementation for face detection with Haar features (MIT-CMU frontal faces dataset [Sung et al., 1998] and prize-wining paper [Viola and Jones, 2001] less than 3 years after). In computer vision, the same phenomenon occured with more and more available large cardinality datasets and thus ready-to-use products as Fig. 2 shows.

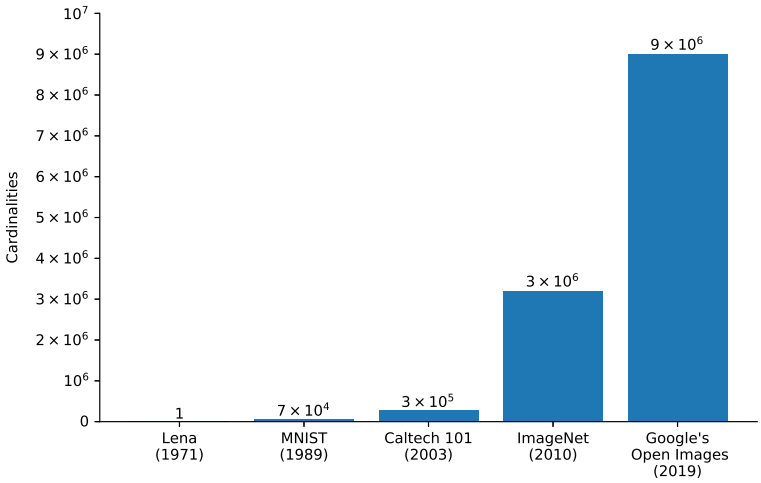

Figure 2: Images Datasets Explosion

1.3.2 *Output Data and Functions Properties*

Now we present a table of tips to adapt unconstrained vanilla neural networks functions to constrained custom ones without the need of tediously maintaining the constraints (which is not recommended for stochastic gradient descent optimization, the weapon of choice to face large scale datasets according to the neural networks literature [LeCun et al., 1998]).

1 `https://www.deepl.com/press.html`

Mathematically, the universal approximation theorem allows to parse a very large class of function (more precisely we only need the Lebesgue-integrability [Hornik, 1991]):

$$\mathbf{x} \in \mathbb{R}^D \text{ and } \mathcal{F}(\mathbf{x}) \in \mathbb{R}^K \tag{12}$$

and Table 1 enumerates many kinds of functions and their corresponding implementations thanks to the unconstrained original neural network function $\mathcal{F}$.

Property	Implementation
Positivity	$\exp(\mathcal{F}(\mathbf{x}))$ or $\mathcal{F}(\mathbf{x})^2$ or even $\max(0, \mathcal{F}(\mathbf{x}))$
Boundness between m and M ($m < M$)	$m + (M-m)\sigma(\mathcal{F}(\mathbf{x}))$ where $\sigma(\mathbf{z}) = \frac{1}{1+\exp(-\mathbf{z})}$ which is related to the SoftMax function when $K = 2$
Probability Vector (i. e. in K-dimensional simplex)	$\text{SoftMax}(\mathcal{F}(\mathbf{x}))$ where $\text{SoftMax}(\mathbf{z})_k = \frac{\exp(\mathbf{z}_k)}{\sum_{\ell=1}^{K}\exp(\mathbf{z}_\ell)}$ which is related to the multilogit model [Hastie et al., 2005] and the logsumexp trick
Positivity and (semi)-definiteness matrix	$\mathcal{C}(\mathbf{x}) \times \mathcal{C}(\mathbf{x})^\top$ where $\mathcal{C}(\mathbf{x})$ is a lower triangular free matrix with positive diagonal entries (even bounded for more stability in practice) which is related to the Cholesky decomposition [Golub and Van Loan, 2012]
1-Lipschitz function	Online power iteration on each matrix-vector product inside the neural network that implements the function $\mathcal{F}$ which is described in the spectral normalization work by Miyato et al. [2018]
Bijection (one-to-one function)	Composition of layers such as $\left[\mathbf{x}_{[:d]}^\top, \left(\boldsymbol{s}(\mathbf{x}_{[:d]}) \times \mathbf{x}_{[d:]} + \boldsymbol{t}(\mathbf{x}_{[:d]})\right)^\top\right]^\top$ (with *pythonic* indexation of coordinates) where $\boldsymbol{s}$ and $\boldsymbol{t}$ are regular neural network functions. This technique also gives to the log-determinant of the Jacobi matrix without too much computation burden thanks to the original paper of Dinh et al. [2017]
Recursive Function	If t is time and $\mathbf{y}_0$ an initialization: $\mathbf{y}_{t+1} = \mathcal{G}(\mathbf{y}_t, \mathcal{F}(\mathbf{x}_t))$ which is pedagogically well presented by Karpathy [2015] and Chakraborty et al. [2014] with improved LSTMs variants by Hochreiter and Schmidhuber [1997] and GRUs [Cho et al., 2014]

Table 1: Implementations for several types of Functions with respect to their inputs nature and functional properties

1.4 OPTIMAL TRANSPORT

The optimal transport research field has proven to be crucial in redefining our modern world as we know it, across a stunningly wide range of applications, since its French birth in the XVIIIth century [Monge, 1781], with scientists from very different backgrounds and application areas such as Rabin et al. [2012] in image processing, Courty et al. [2017] in near-general data domain adaptation, Abouchar [1970] for airports management, decisive World War II military battle victories [Smolinski, 1962], breakthrough innovation in modern economy [Galbraith, 2019] and flabbergasting futuristic industrial revolutions such as semi-automatic objects generative design [Shu et al., 2019]. After this long one-sentence celebration, we now briefly review the basics of such a prolific mathematical offspring.

1.4.1 *Formulations*

In a data space $\mathcal{X}$ equipped with a metric c, we want to measure a distance between two *piles* of data that is related to that metric. We are willingly using the vague word *pile* because thanks to the notion of Dirac distibutions we have access to both smooth densities, empirical distributions and a wide variety of distributions in general. Mathematically, we thus manipulate two distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ with associated two variables $\mathbf{x} \sim \boldsymbol{\mu}$ and $\mathbf{y} \sim \boldsymbol{\nu}$ both living in $\mathcal{X}$. On those distributions, we want to compute the quantity $W_c(\boldsymbol{\mu}, \boldsymbol{\nu})$ measuring how much different the *piles* $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are. To that end, three equivalent formulations exist for that same quantity:

MONGE FORMULATION

$$W_c(\boldsymbol{\mu}, \boldsymbol{\nu}) = \inf_{T_*(\boldsymbol{\mu})=\boldsymbol{\nu}} \mathbb{E}_{\mathbf{x}\sim\boldsymbol{\mu}}\big[c\left(\mathbf{x}, T(\mathbf{x})\right)\big] \tag{13}$$

where $T_*(\boldsymbol{\mu})$ denotes the push forward of $\boldsymbol{\mu}$ by transport map T. For real understanding, we refer the reader looking for details to academic textbooks on Probability such as *Random Measures, Theory and Applications* [Kallenberg, 2017].

WASSERSTEIN FORMULATION

$$W_c(\boldsymbol{\mu}, \boldsymbol{\nu}) = \inf_{\gamma\in\Gamma(\boldsymbol{\mu},\boldsymbol{\nu})} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}\big[c\left(\mathbf{x}, \mathbf{y}\right)\big] \tag{14}$$

where $\Gamma(\boldsymbol{\mu}, \boldsymbol{\nu})$ denotes the set of couplings γ based on the two distributions $\boldsymbol{\mu}, \boldsymbol{\nu}$: the collection of all probability measures on $\mathcal{X} \times \mathcal{X}$ with marginals $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ in order to maintain two coupling properties:

$$\forall \mathbf{y} \in \mathcal{X},\ \mathbb{E}_{\mathbf{x}\sim\boldsymbol{\mu}}\left[\gamma(\mathbf{x}, \mathbf{y})\right] = \boldsymbol{\nu}(\mathbf{y}) \tag{15}$$

and

$$\forall \mathbf{x} \in \mathcal{X},\ \mathbb{E}_{\mathbf{y}\sim\nu}\left[\gamma(\mathbf{x},\mathbf{y})\right] = \mu(\mathbf{x}) \tag{16}$$

KANTOROVICH-RUBINSTEIN FORMULATION (for the L_2 euclidean distance cost: $c(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2$)

$$W(\mu,\nu) = \sup_{\mathcal{C}\in\text{Lip-1}} \mathbb{E}_{\mathbf{x}\sim\mu}\left[\mathcal{C}(\mathbf{x})\right] - \mathbb{E}_{\mathbf{y}\sim\nu}\left[\mathcal{C}(\mathbf{y})\right] \tag{17}$$

where Lip-1 is the 1-Lipschitz functions set (from $\mathcal{X} \subset \mathbb{R}^D$ to $\mathbb{R}$). More general formulas exist in *Real analysis and Probability* by Dudley [2018], but changing the euclidean distance cost is possible up to how difficult redefining the Lipschitz property of $\mathcal{C}$ is.

As a scientist apprentice, one can notice that for producing research work, it seems that the Monge formulation is more amenable to intuition, the Wasserstein formulation is more suitable for probabilistic and geometric perspectives and the Kantorovich-Rubinstein formulation is more convenient for devising algorithms thanks the decoupling of μ and ν in the formulas which makes computations easier (simple expectations that a usual Monte Carlo estimation can handle).

1.4.2 *Algorithms*

Since the middle of the XXth century, three major algorithmical tools emerged:

1. Hungarian Discrete Method [Kuhn, 1955]
2. Entropy-Regularized Sinkhorn Fixed-Point Method [Cuturi, 2013, Genevay, 2019]
3. Wasserstein Generative Adversarial Networks [Goodfellow, 2016]

but we are well aware that with a certain amount of pragmatism it is beyond the scope of this humble state of the art to present the great mathematical achievements in optimal transport. Daring to write a summary of such a huge mathematical and on-going research field is unsettling and we refer the reader to three great references for best and rather exhaustive optimal transport overview: (i) *Optimal Transport for Applied Mathematicians* by Santambrogio [2015] on the general scientific culture side, (ii) *Computational Optimal Transport* by Peyré et al. [2019] on the statistical and programming side, (iii) *Optimal Transport: Old and New* by Villani [2008] on the probabilistic and theoretical side.

Historically, Kuhn [1955] found a cubic complexity algorithm to solve an optimal transport in the discrete case which was internationally widespread as the so-called *Hungarian Method*. Linear cost optimization of one-to-one assignments between two sets of elements is reccuring in many real-world applications as

briefly enumerated above. Indeed, once a pairwise assignment cost matrix is given, minimizing the associated cost sum with respect to the best possible assignment map boils down to the ticking of one matrix entry per row and per column (with some additional dummy entries for coping with the rectangular matrix case i. e. different cardinalities for the two sets at hand). Beyond this seminal successful attempt to cast optimal transport as a linear problem, some other works pushed the analysis further with network flows [Ahuja et al., 1989], with graph theory angle [Goldberg and Tarjan, 1989], then Dynamic Programming mixed with fluid mechanics reasoning came in with Benamou and Brenier [2000] for improved computational speed. Special discrete-continuous distributions cases were also efficiently tackled by Mérigot [2011] with some exceedingly fast convergence thanks to Lévy [2015]. First in industrial logistics, these approaches were used in a surprisingly wide range of applications from e. g. worker to work assignments, airplane to airport assignments, to communication protocol load balancing systems etc.

Recently, Cuturi [2013] revisited entropy regularized transport to efficiently solve almost the same optimal transport problem with the Sinkhorn iterative fixed-point-type algorithm, which is especially handy when polynomial complexity is not realistic in large scale settings. The idea is that they are willing to trade some approximated *optimal* transport due to transport entropic regularization for realistic speed and doable computations. Surprisingly, even exploiting the entropy-regularized properties of the Sinkhorn (and thus *non-optimal*) transport itself has value in many applications such as robust finance [De March, 2018], ranking [Vert], photo album summarization [Liu et al., 2020]... This is explainable because traditionally, regularizations schemes are meant to make numerical and stability problems vanish. That fixed-point Sinkhorn theorem gives extremely fast convergence rate of transport entropy regularized over Wasserstein distances computations: less than a dozen of iterations are enough in practice. This approach on top of dramatic computational accelerations makes it indispensable both for theoretical analysis and for many real-world applications. In spite of diligent progress for Wasserstein Generative Adversarial Networks (as we will describe later), the work accomplished by Genevay [2019] still presents Sinkhorn-based techniques as a great mathematical and numerical alternative for efficient optimal-transport-related solutions in machine learning.

In order to imitate high dimensional data, the principle of the milestone work of Goodfellow et al. [2014] about Generative Adversarial Networks (GAN) is prototypically new for a fascinating worldwide series of research papers. As illustrated in Fig. 3, from a pseudo-random generator, we can sample some low dimensional noise that is transformed thanks to a variable generator function to get generated data within the original data space. The role of the critic function is to estimate a divergence between the real data and the generated data distributions and the generator's role is to minimize it. The most common metaphor for this mainstream press acclaimed technique is considering the generator as a

forger trying to fool the detective embodied by the critic within an adversarial objective. The detective wants to distinguish generated and real data and the forger wants to produce generated data that are indistinguishable when compared to real data. More mathematically, it turns out that there is indeed a link between that min-max optimization and a Nash equilibrium as emphasized by Fedus et al. [2017].

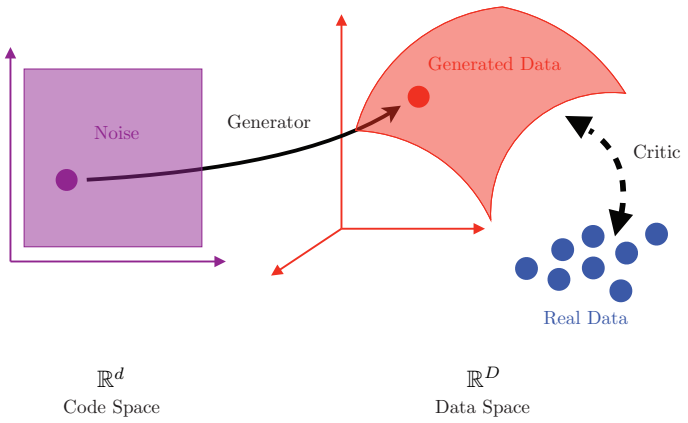

Figure 3: GAN Principle: Imitating Data thanks to some Artificial Low Dimensional Noise in Purple from a Code Space ($\mathbb{R}^d$) transformed thanks to a Generator function (or forger) into Generated Data in Red living in the Data Space (($\mathbb{R}^D$)) so that they are supposed to be close to the Real Data in Blue thanks to the Critic function (or detective) – Adapted from `https://optimaltransport.github.io`

Following the seminal Generative Adversarial Networks (GANs) work of Goodfellow et al. [2014], Arjovsky et al. [2017] used the Kantorovich-Rubinstein formulation in order to imitate data extending GAN from Jensen-Shannon divergence minimization to Wasserstein distance minimization between generated and real data distributions which paves the way for revisiting optimal transport in the context of unsupervised learning. Answering the high research expectations for GANs, the recent contribution of Miyato et al. [2018] called spectral normalization had a tremendous impact. Indeed, revisiting the power iteration numerical recipe [Press et al., 2007] at each linear or convolutional steps to elegantly enforces the Lipschitz property required by the Kantorovich-Rubinstein duality allows variation constraints withtout unstable stochastic gradient projection techniques beyond Wasserstein distances and optimal transport. Until the spectral normalization technique, neural networks had a tendency to implement function that are not regular and enforcing the Lipschitz property (i. e. constraining the variations of the functions implemented this way) gives a beneficial regularization effect beyond Wasserstein distance estimation.

This spectral normalization technique gave so much optimization stability that facing unheard-of large scale settings is made possible and provides the extraordinary images imitation results of the BigGAN approach [Brock et al., 2019]. This engineering achievement is also convincing thanks the already-mastered residual convolutional neural networks [He et al., 2016] ResNet tool coming from the supervised image classification research. As a matter of fact, it is fair to say that ResNet [He et al., 2016] gave the fascinating ability to neural networks of handling recursively raw, moderately pre-processed and highly pre-processed data at each layer by *short-cutting* the traditional successive-layers structure thanks to additional cross-layer connections. All combined, it made BigGAN [Brock et al., 2019] impressive rendering results possible, neatly fighting against the curse of dimensionality effect on the learned parameters side confronted with exceedlingly large scale unsupervised conditions for both data cardinality N (number of images in the dataset) and dimensionality D (number of pixels for the image high resolution) under unsupervised conditions.

1.5 REPRESENTATIONS

Knowing how to represent data is understanding data and the underlying structure beneath it (and *vice versa*). As is, data has in general too much dimensionality to be plotted ($D > 3$) and finding a useful projections seems to interleave regular dimensionality reduction tehniques and clustering (i. e. the task of building groups as we will see later).

Since 2012, several researchers identified what large scale settings for computations and memory meant not only for applied mathematics but also for the worldwide economy describing it at the *Big Data* era: Among them Zikoupoulos and Eaton [2016], Peters [2012], Jordan [2013]. Consider N, the number of elements of a database (the cardinality), D the size of each element (the dimensionality), then we observe two training conditions or regimes for machine learning algorithms:

BIG DATA REGIME $\frac{N}{D} \gg 1$ Statistical theorems behave nicely but software programming was difficult before data storage and computations speed dramatically improved (even on the software side for parallelization)

SMALL DATA REGIME $\frac{N}{D} \ll 1$ or $N \simeq D$ As we will see, the *curse of dimensionality* make things very difficult in terms of statistics but computations are easy.

In practice, software and hardware issues of the big data regime have been solved in the 2000s and the beginning 2010s at an industry-quality level. Domain specific and statistical expertise are still required for small data regime. Indeed with images for example, hierarchical convolutions organized layers in neural networks has a decorrelating effect which reduces the impact of the huge dimensionality of data: intermediate results (called feature maps) are still very big on

the first layers but the number of parameters has been substantially decreased thanks to the small-sized convolution kernels. For other domains, applying Convolutional Neural Networks (CNN) did work but mastery of each domain must not be ignored. In other words, injecting enough knowledge into an automatic system is basically diminishing the dimensionality (i. e. removing redundancy with respect to the task at hand) which is good news towards coping with the curse of dimensionality: this transforms a difficult small data regime into an easier big data regime.

1.5.1 *Big data and neural networks*

We previouly established that a convincing decrease of dimensionality D thanks to the appropriate representation tools is key for good empirical results in machine learning. Now it is time to explain how to optimize an objective function under large *cardinality* conditions. Since the 1950s, some early work mixing statistics and optimization, Robbins and Monro [1951] allowed large cardinality training dataset thanks to stochastic optimization. One interesting aspect of the Robbins-Monro theorem is that the objective (nor the full-gradient) does not need to be evaluated directly anymore but only a biased-free estimate of its gradient with respect to the learned parameters. In the end, the large scale constraint is relieved because only randomly picked mini-batches of data are needed instead of the whole dataset during the optimization. This early mathematical finding paved the way for contemporary large scale learning fulfilled ambitions [Bonnans et al., 2003] beyond deep learning. Several scientific avenues have been taken with sometimes sophisticated algorithmical tools [Bertsekas, 1997] with new programming context (e. g. distributed systems [Hendrikx et al., 2019] or limited memory systems [Defazio et al., 2014]).

In a nutshell, the Robbins-Monro theorem allows to manipulate an idealized loss function (over all the data of the universe) without the need to compute its values nor its gradients with respect to its parameters as long as one can provide a biased-free estimator of the loss function gradient needed for the stochastic gradient descent optimization scheme. Surprisingly, it turns out that artificial neural networks actually look like biological neural networks for learning (but much less for its structure and running behavior inspite of 1950s predictions). Indeed, stochastic gradient descent follows a Hebbian rule[2]:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \mathbf{f}_t \tag{18}$$

where t is an iteration index, $\boldsymbol{\theta}$ the parameters that are learned, α is a learning rate (often constant) and finally $\mathbf{f}_t$ is an incremental progress computed thanks to a measured error. Numerically, $\mathbf{f}_t$ is the gradient of an objective $\mathcal{L}$ to minimize:

$$\mathbf{f}_t = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \tag{19}$$

2 Donald O. Hebb was an influential neuro-psychologist from the 1950s

which can be approximated by a biased-free estimate of $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$ according to the Robbins-Monro theorem. For a general objective to minimize such as $\mathcal{L}(\boldsymbol{\theta}_{\mathcal{F}}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{Nature}}\left(\ell\left(\mathbf{y},\mathcal{F}(\mathbf{x})\right)\right)$ the hebbian rule has a replaced stochastic gradient $\hat{\mathbf{f}}_t$ instead of the "true" one $\mathbf{f}_t$:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t\hat{\mathbf{g}}_t \tag{20}$$

where

$$\hat{\mathbf{f}}_t = \frac{1}{B}\sum_{b=1}^{B}\nabla_{\boldsymbol{\theta}}\left(\ell\left(\mathbf{y}_{i_b},\mathcal{F}(\mathbf{x}_{i_b})\right)\right) \tag{21}$$

and $i_b \sim \mathcal{U}_{\mathbb{N}}(1,N)$ is an uniform index parsing the N-cardinality training dataset in mini-batches of size B.

It is noteworthy to recall that there is a simple case where the best learning rate is given in closed form: the online estimation of a mean random multivariable which is related to the least squares problem:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{1}{t+1}(\mathbf{x}_{t+1} - \boldsymbol{\mu}_t) \tag{22}$$

From a practitioner point of view, large cardinality problems have been solved both on the software/hardware side and on the optimization side. The dimensionality issues still remain and is in essence more specific to each application we tackle.

1.5.2 *The Curse of Dimensionality*

Enhancing data structure in the data representation is an efficient way to cope with the curse of dimensionality issue that we focus on in this section. As said above, in images and speech, convolutions provided enormous improvements in terms of accuracy for the given task. Word ordering structure is today considered as standard since the rise of embeddings techniques with *word2vec* [Mikolov et al., 2013] and variants even with the recent *BERT* method. The fundamental idea is to transform observed occurrences of words ordering on large corpora into a regression problem providing relevant words representations into real-valued vectors. The situation is much more difficult in genomics despite worldwide sincere attention [Barillot et al., 2012]. Indeed 1 human is DNA-represented by $D \simeq 10^{12}$ nucleotides and the worldwide population is $N \simeq 10^9$ which corresponds to a small data regime once again. Even in diagnostics problem *healthy v.s. ill* detection or classification problem, in theory a balanced and worldwide-sized annotated DNA-dataset is not enough for classification nor interpretation which is quiet an embarassing (theoretical) scenario. We did not yet leverage enough desirable and yet unknown DNA structural information among those nucleotides to get an easier big data regime.

Now we know that high cardinality is not a problem anymore if dimensionality is not too high keeping the problem under the big data regime. In small data regime when dimensionlity is too high, devising algorithms is difficult because of the well-known *Curse of Dimensionality*[3] that we mentioned earlier. To explain that phenomenon dubbed the *Curse of Dimensionality*, a classic example considers the volume of a sphere of unit radius in dimension D which follows

$$V(D) = \frac{\pi^k}{(k)!} \text{ if } D = 2k \text{ is even, or } V(D) = \frac{2(k!)(4\pi)^k}{(2k+1)!} \text{if } D = 2k+1 \text{ is odd} \quad (23)$$

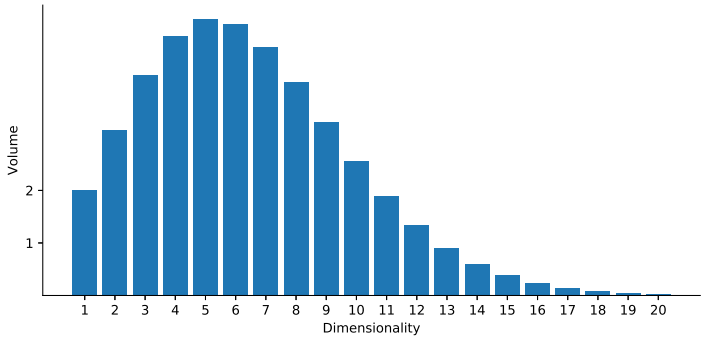

Figure 4: Volume of an euclidean sphere of unit radius in dimension D

In Fig 4, first we can see that volume $V(D)$ reaches its maximum for $D = 5$ and right after, that same volume $V(D)$ dramatically decreases towards 0 as D grows ($\lim_{D\to\infty} V(D) = 0$) which is not intuitive and rather deceiving because of how we experience spatial neighborhood notions in our 2D and 3D living environment as human beings (we would expect that volume to grow indefinitely as it does for $D = 1, 2, 3, 4$). For example, in low dimensionality regimes ($D = 1, 2$ or 3), if a point lies close to the origin inside an unit ball neighborhood, then the volume to exhaustively parse for finding it is reasonably big ($V(D) \simeq 2, 3.14$ or 4.18 respectively). But for high dimensionality regimes (e. g. $D = 30$), the corresponding volume to parse gets extremely small ($V(D) =\simeq 2 \times 10^{-5}$) which is unsettling: we would think that looking for a point inside a higher dimensional sphere would be larger but this is not true. In the end, this means that rudimentary notions like distances or even similarities behave unexpectedly in such high dimensionality data regimes. Another pedagical example considers the ratio $R(D)$ between a 0.9-radius and 1-radius balls' volumes ($R(D) = 0.9^D$). In Fig. 5, that $R(D)$ ratio also exhibits an embarassing phenomenon with respect to our intuition: as the dimensionality D increases, the volume ratio $R(D)$ goes to zero

3 The keyphrase *Curse of Dimensionality* was first mentioned by Bellman [1957] to describe the need of efficient algorithmic tools like Dynamic Programming [Dasgupta et al., 2008] to efficiently explore huge discrete solutions spaces, historically later it also concerned many kinds of large data spaces

($lim_{D\to\infty} V(D) = 0$) which means that in high dimensionality regimes the *orange 0.1-peel* occupies almost all the *entire orange* volume in layman's terms.

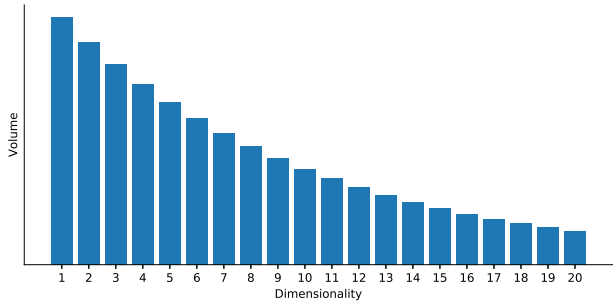

Figure 5: 0.1-peel ratio $R(D)$f or an euclidean sphere of unit radius in dimension D

For pedagogical reasons, we may present a metaphor that we dubbed the "crinkled paper in a room". Indeed, in general, real data distribution seem to behave like a crinkled sheet of paper with low intrinsic manifold dimensionality (say $d = 2$) inside of a large space of dimensionality D (say a $d = 2$ crinkled sheet of paper living in a room of $D = 3$ dimensions to help our $3D$-living creatures intuition). In this metaphor, revealing the data structure and focusing on the independent variables ruling the data boils down to how we *iron* or *un-crinkle* that sheet of paper to explicitly have access to the real degrees of freedom parsing the data.

Interestingly, following this metaphor helps us understand than the GAN approach is doing the opposite: starting from a low dimensional uniform law (the ironed sheet of paper) transformed by the generator into a higher dimensional and much more sophisticated law. GAN optimization is basically learning how to *un-iron* or *crinkle* a clean sheet of paper into something close to the real data manifold.

Unfortunately, in terms of measures theories, neither the Riemann nor Lebesgue measures allocate weight to that sheet of paper (crinkled or not, it is respectively not defined and zero) which may lead to explore new theoretical research avenues and maybe studying other probability foundations like maybe not yet sufficiently explored in machine learning based on the Hausdorff measure [Abbott and Rogers, 1999] or using probability freed from measure theory like the acclaimed attempt of Breiman [1992]. Another way to cope with this difficulties is to abandon the notion of probability distributions to embrace energy-based models as initiated by LeCun et al. [2006] which require less assumptions to model data and allows to generalize a distribution into an energy (through a neg-exponentiation from an energy analogy from Physics but without the constraint of having a unit *measure* over space).

On the kernel-based techniques [Shawe-Taylor et al., 2004] side during the 1990s and 2000s, the dimensionality problems disappear because once a kernel

similarity matrix is built, one does almost not need the data anymore to operate analysis (with supervised kernel-based support vector machines or unsupervised spectral clustering). With kernels, dimensionality problems are avoided thanks to the kernel similarity matrix. All dimensionality-wise considerations are relegated to the crucial kernel definition. Unfortunately, such techniques are limited due to the inherent quadratic memory complexity of such pairwise structures implied by the similarity matrix.

Meanwhile, on the Model-based side, naive approaches do not succeed to achieve good results because of that high dimensionality: they ultimately loose their specific model selection capabilities because of over-parametrization to match that high data dimensionality. Indeed, in reasonable dimensionality conditions, the main advantage of such techniques is their ease for probabilistic interpretation and model selection. Over-parametrization (or over-fitting) can be seen as a *data starvation* phenomenon: a large number of parameters to fit would require a huge amount of data to get reliable estimations which in practice leads to poor performance. We see that once again, the interesting factor is the ratio $\frac{N}{D}$ of cardinality N over D and not one of them without considering the other one.

In the context clustering of large scale dimensionality, parcimonious and cluster-wise representations [Bouveyron and Brunet-Saumard, 2014b] circumvent these high dimensionality problems and still keep the appealing probabilistic properties of model-based clustering without sacrificing accuracy. One can remark that in the supervised classification literature, sparsity (and even structured sparsity [Jenatton, 2011]) also did cope with dimensionality problems that are similar in essence.

Model-based clustering algorithms are popular because they are renowned for their probabilistic foundations and their flexibility [Duda et al., 2012]. Indeed, even for non-statisticians, the possibility to output meaningful probabilities is intuitive and principled. The main drawback of mixture-based and model-based methods for clustering is the lack of richness (in Kleinberg's sense see [Kleinberg, 2015] but we will come back on it later) due to necessary distribution assumptions that may not be necessarily true for real data which justifies our attempt to alleviate this limitation thanks to the functional expressivity of neural networks.

One fundamental machine learning hypothesis is recurring in the literature [Murphy, 2012, Duda et al., 2012, Bishop, 2006]: real data live in a low-dimensional (of dimension D) manifold in a much higher dimensional space (of dimension D and $d \ll D$). A classic pedagogical example consists of the independent and uniform sampling of each pixel of an image: there is no realistic chance to produce a convincing photograph! This means that even sophisticated mathematical object such as photographs lie on a manifold of lower intrinsic dimensionality than the number of pixels multiplied by the number of channels. In a reverse fasion, this has been confirmed by the DCGAN work of Radford et al. [2015] that is able to generate $D = 3 \times 256 \times 256 \simeq 2 \times 10^5$ convincing DCGAN images from a random uniform variable made of $d = 100$ independent coordinates). Thanks to

this low-manifold-dimensionality hypothesis for data in mind, it is reasonable to investigate some dimensionality reduction techniques.

1.5.3 *Dimensionality Reduction*

At the beginning of the XXth century, Principal Component Analysis (PCA) was invented [Pearson, 1901]. This technique finds an optimal linear (or affine) projection with respect to compression/decompression quadratic reconstruction error. This algorithm gave birth to two more recent ones: (i) its kernelized extension [Schölkopf et al., 1998] (euclidean distances can be expressed with dot products that are in turn replaced by kernel evaluations in a Reproducing Kernel Hilbert Space RKHS following the well-known *kernel trick*) and (ii) auto-encoders [Kramer, 1991, Bourlard and Kamp, 1988, Vincent et al., 2010] which replaces compression and decompression by one neural network each.

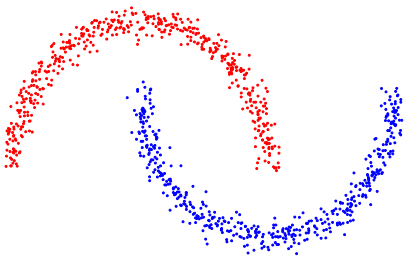

Figure 6: "Two Moons" Toy Dataset

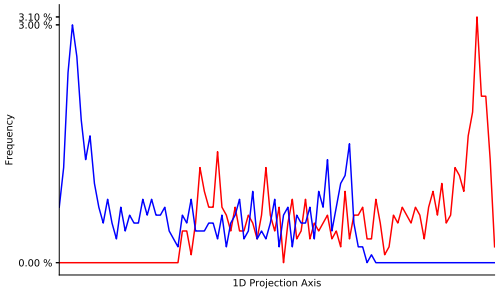

Figure 7: Two Moons Projection through PCA into 1D

Fig. 7, 8 and 9 show PCA, (Gaussian) kernel-PCA and AE dealing with non-linearly-structured yet simple 2D distributions shown in Fig. 6. It turns out that (up to the Gaussian parameter of our kernel-PCA), the non-linearity improvements of PCA in two different variants namely kernel-PCA and AE does help *ironing the crinkled distribution of interest* (to follow our metaphor in section 1.5.2).

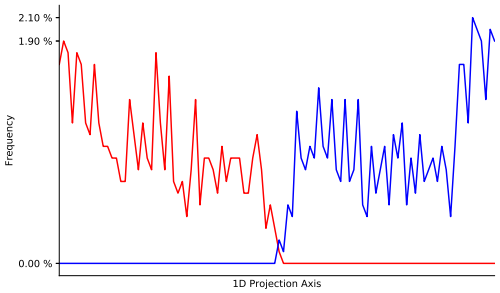

Figure 8: Two Moons Projection through kernel-PCA into 1D (with Gaussian Kernel)

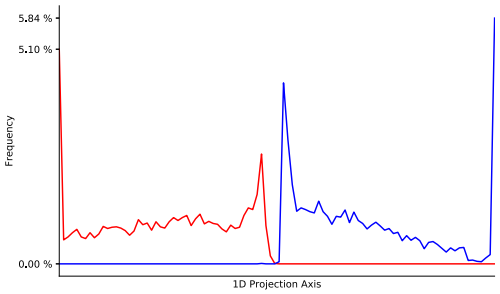

Figure 9: Two Moons Projection through an Auto-Encoder into 1D

Dimension reduction approaches such as principal components analysis (PCA) or even autoe-encoders (AE) [Vincent et al., 2010] may help for clustering but, as is, are not designed with a clustering mindset which causes poor results in practice. These global dimension reduction techniques are pragmatic but ignore information which is discriminant for separating clusters. Indeed, clusters are usually living in different sub-spaces between clusters if there exist. Back in the original data space, there is no reason to find an easy-to-find common linear sub-space that is discriminant enough to separate all the classes at the same time. For example, if clustering is taken into account while reducing the dimensionality, then one solution could be to divide the reduced space into as many zones as clusters such that they do not overlap while still reducing the dimensionality. Thus, we avoid generic approaches because they cannot afford by themselves to capture these subtleties in the data. One must combine dimensionality reduction and clustering. Clustering could be looked at as an extremely simplified version of the data by just keeping the index of the cluster the data belong to.

The high dimensionality clustering literature [Bouveyron et al., 2007] tends to show that clustering and dimensionality should be done a the same time (*i.e.* in an end-to-end fashion) and not sequentially. Indeed, on the one hand, doing clustering first for huge dimensionality data is computationally difficult for obvious reasons and also statistically difficult because of the *curse of dimensionality*.

On the other hand, doing dimensionality reduction first looses hidden cluster-wise information about data. One major issue of this work is precisely trying to tackle this "chicken or egg" problem.

Vanilla Auto-encoders alone do not allow to specify a precise probabilistic structure for a low-dimensional representation. This limits their combination with model-based clustering techniques. Furthermore, optimizing an auto-encoder and a Gaussian mixture generally implies the use of a trade-off hyper-parameter to combine these two objectives. This hyper-parameter is possibly hard to tune as cross-validation is not an option in our unsupervised settings as no validation score can be used by definition.

The problem of learning representations from data in an unsupervised manner is a long-standing problem in machine learning [Bengio et al., 2013, LeCun et al., 2015]. Principal Components analysis (PCA) and auto-encoders (AE) which can be seen as non-linear extension of PCA [Baldi and Hornik, 1989] have been used for representing faces [Turk and Pentland, 1991] or to produce a hierarchy of features [Chan et al., 2015]. Other techniques have been used such as sparse coding [Mairal et al., 2008] where the representation of one image is a linear combination of a few elements in a dictionary of features. More recently Bojanowski and Joulin [2017] learned features unsupervisedly by a procedure that consists in mapping a large collection of images to noise vectors through a deep convolutional neural networks.

Clustering and dimensionality reduction are interleaved. The importance of finding a suitable representation for unsupervised tasks was first highlighted by Chang [1983], who showed that embeddings based on principal component analysis were often unfit for clustering purposes so we suggest the idea of learning both clustering and dimensionality at the same time in an end-to-end deep learning fahsion. In a more model-based literature [Bouveyron et al., 2019], combining clustering and dimensionality reduction simultaneously also proved more successful than separating dimensionality reduction and clustering sequentially, which in turn, was already more successful than doing only one of them for both results. This means that both deep learning and Bayesian literatures tend to show a certain symbiosis between clustering and dimensionality reduction towards data analysis and understanding.

In the context of linear embeddings (that offers dimensionality reduction), the main approach was to combine linear discriminant analysis with the k-Means (k-Means) algorithm (DisKMeans) [De la Torre and Kanade, 2006] or more generally a Gaussian Mixture Model (Fisher-EM) [Bouveyron and Brunet, 2012]. Much less research is available in relation to non-linear embeddings. Archambeau and Verleysen [2005] however proposed to use manifold learning in combination with GMM. Combining clustering with representation learning has been done with deep learning techniques in the past. An early attempt was explored by Trigeorgis et al. [2014] who used a deep semi-non-negative-matrix-factorization (NMF)

model to specifically factorize the input into multiple stacking factors which are initialized and updated layer by layer with k-Means on the last layer.

Neural networks have proven successful in the context of supervised classification and even regression [Goodfellow et al., 2016]. Indeed, their ability to transform data such that the frontiers between classes are hyperplanes in the classification setting have made them very popular. In spite of the non-convexity of their optimization scheme, today, they are superior to convex machineries such as Support Vector Machines even for kernelized ones in almost every domain in Computer Vision and sound processing for example. The idea of having learned features has already been tackled by Chen [2015] used Deep Belief Networks together with maximum-margin clustering. Wang et al. [2016] jointly optimized a sparse coding objective and a clustering loss. Eventually, all these recent approaches have been empirically outperformed by auto-encoders' style machineries.

When it comes to compressing data while limiting loss of reconstruction information, auto-encoders have proved efficient [Vincent et al., 2010]. Briefly, an auto-encoder is a neural network made of two parts: (i) the *encoder* maps the data in a low-dimension space, (ii) the *decoder* maps them back to the original space. An auto-encoder is trained to reconstruct the data in the original space (usually in a least squares fashion but it could be any differentiable metric). At the end, if the reconstruction error is low, then codes resulting from the encoder (also called "bottleneck") have compressed data without loosing too much information (because by construction, it is possible to rebuild data from codes thanks to the decoder). The main assumption behind this technique is that the input data space of high dimensionality contains structure that could be successfully embedded in a lower-dimensionality manifold [Alain and Bengio, 2014, Sonoda and Murata, 2016] and the code space plays that embedding role.

Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] establish a min-max game between a generator neural network on one side and a discriminator or critic neural network on the other side in order to generate data (from random noise) that the critic cannot distinguish from the real data. From that influential work emerged Adversarial Auto-Encoders (AAE) by Makhzani et al. [2015], Wasserstein Auto-Encoders (WAE) by Tolstikhin et al. [2018] and Adversarially Learned Inference (ALI) by Dumoulin et al. [2017]. In a few words, thrice are turning an auto-encoder into a generative model. They are trained in different ways that put an arbitrary fixed prior distribution in the code space. For the clustering chapter, we were initially inspired by these approaches with learned mixture distribution instead of a fixed prior one.

1.6 DISSERTATION OUTLINE

This thesis is illustrating the process of learning representations using neural networks and optimal transport through three applications:

CLUSTERING *joint work with Pierre-Alexandre Mattei, Andrés Almansa and Charles Bouveyron* It is about unsupervisedly representing large-scale datasets in groups. This offers data tools to get a better intimate knowledge of the data with whereas the usual deep learning supervised classification algorithms do not so easily unless tedious manual annotation is already *paid for* at least on training data;

UNSUPERVISED FEATURE IMPORTANCE It consists in analyzing data at a coordinates level with a wide of applications from pure data understanding to background/foreground image segmentation in an unsupervised manner (the only remaining supervision being a pile of images containing the same semantic class of content). In this work, we only propose an attempt to accomplish that desirable goal and we provide a sound theoretical framework to do it;

PREDICTION WITH UNCERTAINTY We insist on a better interpretation of supervisedly trained neural networks output in terms of uncertainty (especially for classification probabilities) towards a simple yet efficient way to improve uncertainty estimation in such supervised learning scenarios. In this contribution, sensitive applications can be a little more reliably envisioned when simple industrial constraints or more complex health, security and even justice issues are involved.

At the end of this mauscript, we provide an appendix *Generative Adversarial Networks Initialization with Auto-Encoders*. This is an heuristic for practical initialization of GAN training with tips revisiting pre-training traditions from the 1990s but for contemporary machine learning tools.

In this *state of the art* chapter, we introduced the challenges related to thrice learning representations, optimal transport and neural networks and the next chapters will be devoted to applications in clustering, unsupervised feature importance extraction and supervised uncertainty estimation thanks to the aforementioned tools. Some ongoing and future work are envisioned as a conclusion and an appendix presents practical yet effective techniques for training GANs that we gathered from experience.

1.6.1 *Clustering*

Clustering is one of the oldest unsupervised learning task [Jain, 2010]. Clustering [Duda et al., 2012] is the task of making groups without the need of any manual annotations. Along with dimensionality reduction, clustering is a desirable goal in data analysis, visualization and is often a preliminary step in many algorithms for example in computer vision [Ponce and Forsyth, 2011] and natural language processing [Goldberg, 2017]. Clustering and more generally data analysis does not only consist in pre-processing steps, it is about helping us (as human beings) understanding the underlying structure of data at hand.

Meanwhile, the computer vision field has recently witnessed major progress thanks to end-to-end deep-learning systems since the seminal work of LeCun et al. [1990] and more recently of Krizhevsky et al. [2012]. Most of the work however has been carried out in a supervised context. Our effort leverages that wealth of existing research but in an unsupervised framework.

While optimal transport [Villani, 2008] have gained recent attention especially for generating data (*i.e.* imitating data) in large scale settings (large both in terms of dataset cardinality N and dimensionality D) with Generative Adversarial Networks (GAN) originated by Arjovsky et al. [2017] and Sinkhorn divergences by Genevay [2019], we chose to ignore imitating capabilities and just use this literature to algorithmically manipulate Wasserstein distances. For example, this considerable amount of anterior work gives us a significant ease for optimization with helpful tools such as stochastic gradient descent.

The purpose of this research is to build a linear-complexity algorithms that use non-linear embeddings into code spaces. Indeed, in the clustering literature, one can distinguish two kinds of clustering algorithms with respect to their computation and memory complexity as function of the cardinality N. On one side, we have linear algorithms such as k-Means (k-Means) and Gaussian Mixture Models (GMM), which usually work directly on the data (*i.e.* without any medium such as embeddings and transformed version of the raw data). On the other side, we also have quadratic and cubic algorithms such as hierarchical clustering [Duda et al., 2012] and spectral clustering [Ng et al., 2001, Zelnik-Manor and Perona, 2004, Von Luxburg, 2007] that use pairwise similarities to emphasize the latent clustering structure lying on the data. Now, we describe some statistical problems related to the clustering task, and we will enumerate some famous clustering algorithms.

Clustering is an ill-posed problem

In general, there exists no clear, objective means of defining a "good clustering". For a fixed number of groups, Kleinberg [2015] presents three desirable properties for a given clustering algorithm, namely:

SCALE INVARIANCE Clustering output should not change if we multiply data by a constant

RICHNESS OR CLUSTER SHAPES INVARIANCE all separable cluster shapes should be possible (e. g. beyond linear separation or ball-shaped clusters)

CONSISTENCY OR METRIC INVARIANCE Clustering output should not change with respect to the choice of distance

and he prooved the impossible existence of such an algorithm featuring all these three properties simultaneously. In other words, his clustering impossibility theorem tells us that clustering is an ill-posed problem. To add insult to injury,

when data representation (or embedding) is involved, that clustering task becomes all the more unclear because the underlying metric is allowed to change arbitrarily. Indeed, the algorithm could expand the distance between points in the embedding space that initially were located near to each another, which would break initial pairwise "distance" constraints between the initial points and would inevitably violate the internal structure relating data. Well aware about these difficulties, we decided to try anyway following our scientific predecessors as clustering is useful in practice "as is".

Taking advantage of the abundant optimal transport literature with probabilities (relaxing hard clusters memberships definition to prefer probabilities) and also the neural networks literature (which successfully handles arbitrary classes shapes in supervised contexts) make our efforts reasonable towards a useful clustering algorithm for practioners. Let's review the clustering axioms of Kleinberg [2015]. First, once a metric is chosen, *scale-invariance* can be given for free thanks to the geometric optimal transport interpretations i. e. all 1-Wasserstein distance would be multiplied like the data accordingly without changing the optimization results. Second, Wasserstein distances operate on *all* pairs of distributions to the contrary of the Kullback-Leibler divergence (which requires common support which explains the use of infinite support distributions for the models like the Gaussian), and thus no cluster shapes assumption is required, thus *richness* would be fulfilled by the functional expressivity power provided by neural networks. Third, unfortunately, we would not be able to achieve *consistency* because our models and algorithms strongly depend on the euclidean distance. In fact, this third *consistency* property could be partially reached thanks to a generalized notion of Wasserstein distance defined by the maximum Wasserstein distance when the metric parses a family of distances (which makes the maximum of them still a distance) but this would require further scientific work that we just skimmed in our unsupervised feature importance contribution.

Richness (i. e. free clusters shapes robustness) is obtained thanks to an intermediate space that we call *embedding space* or *code space* which has lower dimensionality that the data space. If theoretical or practical tools are given to navigate between these spaces, back and forth without loosing too much information between data and codes thanks to encoder and decoder functions, then, model-based distribution assumptions can be made on the code side which gives us the richness property in return on the data side like in the work of Jiang et al. [2016]. Indeed, at the beginning of this thesis (mid-2016), our first intuition was to put a mixture distribution (which is a typical model-based idea) at the bottleneck of an auto-encoder (which is a typical deep learning unsupervised tool) with the hope of gathering the best of these two universes: probabilistic ease for model selection coming from model-based legacy on one side and rich representations with neural networks coming from deep learning legacy on the other side.

Fig. 10 represents our strategy to alleviate dimensionality issues while simulataneously performing clustering in a symbiotic fashion: this strategy proved

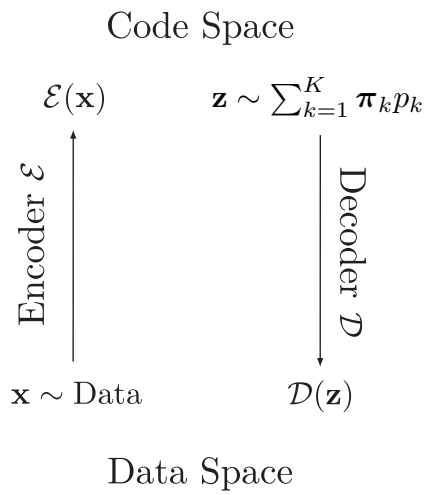

Figure 10: Data and codes spaces

successful in several papers that we briefly present here. The first work we saw doing clustering in the code space of an auto-encoder is the one of Song et al. [2014] and Yang et al. [2016a]. The idea is to cope with large data space dimensionality $D \gg 1$ thanks to an intermediate code space of lower dimensionality $d \ll D$. More precisely, Song et al. [2014] considered a k-Means-regularized auto-encoder loss to get a code space that is more easily clustered with k-Means namely their loss is the sum of the reconstruction and the k-Means residual with a chosen hyper-parameter combining both. This philosophy is the one adopted for our own approach but with a mixture of distributions $\sum_{k=1}^{K} \boldsymbol{\pi}_k \times p_k$ of distributions p_k weighted by proportions $\boldsymbol{\pi}$. We use a code space where clusters memberships (not data themselves but encoded versions of them) are easily computed. In our preliminary experiments, we found that optimizing the k-Means objective (online) when doing joint clustering and feature learning did not work well. We believe this is because it creates high magnitude gradients for points that are far away from cluster centers. Moreover there are sharp discontinuities at cluster boundaries whereas GMM diminishes that effect thanks to low density/probability values for far points. This empirical conclusion seems to confirm what Xie et al. [2015] also observed.

In a similar spirit, [Huang et al., 2014] have developped a locality-preserving and group-sparsity constraints method to handle the clustering. Yang et al. [2016b] alternate between supervised classification and feature learning through Convolutional neural networks (CNN) for images clustering. They significantly improved the state-of-the-art but their method is limited by its intrinsically quadratic complexity. In a similar spirit, Xie et al. [2015] embrace the t-SNE framework [Maaten and Hinton, 2008] in a clustering context through an auto-encoder in a non-model-based fashion. But doing t-SNE first and then clustering is not a good idea because of the same loss of useful cluster-wise information that occurred with PCA or auto-encoders.

All these works tend to show that simultaneous representation learning (by dimensionality reduction for example) and clustering actually do help each other. The reader will find an excellent review in the work of Aljalbout et al. [2018].

Clustering in Large-Scale Cardinality Regimes

k-Means and Mixture Models have been studied in large scale cardinality settings [Bottou and Bengio, 1995, Cappé and Moulines, 2009] but these algorithms work thanks to strong distribution assumptions (like cluster-wise Gaussian clusters shapes for GMM) directly in the original data space (*i.e.* without embeddings). Agglomerative clustering methods greedily use a square similarity matrix to fusion data into clusters but the building of that $N \times N$ matrix is undoable for large cardinality N. Spectral clustering [Zelnik-Manor and Perona, 2004] works with very mild (or no) cluster shapes assumptions thanks to the kernel trick which gives access to high (and even infinite) dimensional representations space at the cost of a square similarity matrix once again which inevitably blocks the way leading to large cardinality datasets. Nevertheless, Choromanska et al. [2013] found a way to gracefuly alleviate this problem through to the the Nyström method that only demands the computations of only few entries of that non-storable square similarity matrix thanks to a low-rank approximation (which is justified by the low intrinsic manifold dimensionality hypothesis related to our *2D crinkled sheet of paper in a 3D room* metaphor). The present work is an attempt to provide a scalable method with the mildest possible cluster shapes assumptions thanks to neural networks that already have these two desirable properties in the supervised context: scalability and mild data assumptions.

The universal approximation theorems [Hanin and Sellke, 2017] allow neural networks to achieve richness (in the sense of Kleinberg [2015]) in theory. But in practice, that richness requires the estimation of a lot of parameters which is not reliable unless we have a large cardinality dataset during training compared to the dimensionality as explained above in section 1.5. In this case, large cardinality datasets are handled thanks to stochastic gradient optimization [Bach, 2016]. Indeed, considerable research in supervised classification has been conducted based on these foundations during the last decades but this work's challenge is about extending this success to unsupervised classification (a. k. a. clustering).

Generative approaches produce a model in the form of a synthetic data distribution that is supposed to be close to the original data distribution with respect to a criterion such as the Kullback-Leibler divergence (which is equivalent to maximizing the likelihood as explained by the *Pattern Recognition and Machine Learning* textbook of Bishop [2006]) typically optimized with Expectation-Maximization [Dempster et al., 1977]. Parameters and hyper-parameters are two different things: parameters are optimized whereas hyper-parameters are imposed before optimization and can be selected after optimization among a set of optimized models (i. e. model selection). One considerable advantage of generative techniques over others is that hyper-parameter selection is made

easy through model selection thanks to principled mathematical (often Bayesian) foundations. Inddeed, building such a model for clustering gives strong tools to evaluate generalization capabilities (with famous criteria such as Akaike Information Criterion AIC, Bayesian Information Criterion BIC summarized by Duda et al. [2012] or even Integrated Completed Likelihood ICL [Biernacki et al., 2000] etc.).

Discriminative methods for clustering were initially inherited from supervised classification these last two or three decades. They are also extended to unsupervised classification (a. k. a. clustering). In clustering these discriminative approaches would not build a model that would fit the data but would rather separate the ouput classes or groups from each other (*e.g.* in a one-vs-one or one-vs-rest manner) focusing on the boundaries of the groups rather than on the groups themselves. Spectral Clustering [Von Luxburg, 2007] or DIFFRAC [Bach and Harchaoui, 2008] are two examples of such techniques.

k-Means solves an (Optimal) Transport Problem

We take a close look at the k-Means loss for data $(\mathbf{x}_i)_{i=1\dots N}$ into K groups:

$$\min_{\sigma,\boldsymbol{\mu}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_{\sigma(i)}\|_2^2 \tag{24}$$

that we optimize over the centroids $\boldsymbol{\mu} = (\boldsymbol{\mu}_k)_{k=1\dots K}$ and the assignments function σ from $[\![1, N]\!] \subset \mathbb{N}$ to $[\![1, K]\!] \subset \mathbb{N}$.

We studied k-Means which is probably both the oldest and most famous clustering algorithm [Jain, 2010] and we realized that it tries to efficiently solve an optimal transport problem. Put differently, the global minimum of the k-Means loss satisfies the optimal transport problem of choosing a limited K number centroids $(\boldsymbol{\mu}_k)_{k=1,\dots,K}$ such that the data empirical distribution $p = \frac{1}{N}\sum_{i=1}^{N} \delta_P \mathbf{x}_i$ on the one hand and the centroids weighted distribution $q = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times \delta_{\boldsymbol{\mu}_k}$ on the other hand would be the closest possible in the 2-Wasserstein sense associated to the squared euclidean distance, although usually, we use the 1-Wasserstein distance associated with the plain and simple euclidean distance:

$$W_{c_2}(p, q) = \min_{\gamma \in \Gamma(p,q)} \mathbb{E}_{(\mathbf{x},\mathbf{m})\sim\gamma} \left[\|\mathbf{x} - \mathbf{m}\|_2^2 \right] \tag{25}$$

and for discrete distributions:

$$W_{c_2}(p, q) = \frac{1}{N} \sum_{k=1}^{K} \boldsymbol{\pi}_k \times \mathbb{E}_{\mathbf{x}\sim p} \left[\|\mathbf{x} - \boldsymbol{\mu}_k\|_2^2 \right] \tag{26}$$

which is the k-Means loss and of course we have a link between the proportions $\boldsymbol{\pi}$ and the assignments σ: $\boldsymbol{\pi}_k = \frac{\#\{\sigma(i)=k |\, i \in [\![1,N]\!]\}}{N}$ as for discrete distributions, optimal transport plans are degenerated [Peyré et al., 2019].

This interesting link between clustering and optimal transport encouraged us to investigate further between these two scientific literatures: clustering and optimal transport. Generalizing this observation to more sophisticated distributions thanks to Generative Adversarial Networks could lead to having a *fatter* support distribution than just Diracs:

$$\min_{\sigma,(\boldsymbol{\theta}_{p_k})_{k=1\ldots K}} W(\sum_{k=1}^{K} \boldsymbol{\pi}_k \times p_k, \frac{1}{N}\sum_{i=1}^{N} \times \delta_{\mathbf{x}_i}) \tag{27}$$

where $\boldsymbol{\theta}_{\mathbf{p}_k}$ parametrized the kth cluster generator distribution $\mathbf{p}_k$ (that was previously reduced to a Dirac distribution located on the kth centroid $\boldsymbol{\mu}_k$). In practice, we can have latent variables $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and we define in $\mathbf{y}_k \sim p_k$ by $\mathbf{y} = \mathcal{G}_k(\mathbf{z})$ as we will develop later. That initial idea we had is encouraging in a sense that it suggests an interesting mixing between optimal transport, neural networks and model-based clustering.

We think that there is an opportunity here to mention that k-Means is *not* a regular special case of Gaussian Mixture Models through Expectation Maximization (EM-GMM) although this false idea is widespread. It is true that when neg-exponentiated, the k-Means looks like the negative likelihood of a Gaussian mixture fitted on the data with identity (or equally proportional to) covariance matrices and equal proportions but there is major difference: memberships probabilities in EM-GMM are not degenerated but they are in k-Means. k-Means can still be seen as an example of the Expectation-Maximization technique but these distinctions have been clearly made by Celeux and Govaert [1992]. The link between k-Means and optimal transport is stronger than the one between k-Means and EM-GMM and also more fruitful in terms of open research.

1.6.2 *Unsupervised Feature Importance*

As suggested earlier when we analyzed the consequences of the impossibility theorem by Kleinberg [2015], metric invariance is an interesting subject. To handle this difficulty related to the choice of the metric, one can work with a set of metrics because we know that the metric defined by the upper bound evaluation over a set of ditances is also a distance. To the best of our knowledge, this angle has not been tackled by the research community to study unsupervised feature importance extraction.

In supervised learning, Breiman [2001] proposed routines based on permutation and mean decrease in impurity but much less work has been done in the unsupervised context. This is probably due to the fact two interleaved problems remain: metric learning, feature selection which makes our task ill-posed. We actually suggest that it is worth trying to improve regular and generic euclidean approaches.

1.6.3 *Uncertain Predictions*

In this chapter devoted to uncertainty in supervised learning, we borrow scientific items from Bayesian and frequentist scientific communities. Indeed, we believe that mixing both scientific cultures is beneficial in general and for uncertainty estimation in particular: probabilistic interpretations provided by Bayesian formulations and function expressivity fitted to large scale data provided by the large frequentist fauna of algorithms which is of course not limited to deep neural networks.

Bayesian and Frequentist scientists in Statistical Learning

There is a so-called rivalry in automatic statistical learning between Bayesians on one side and frequentists on the other side that is quiet disturbing when seen from a young scientific point of view. Indeed, we can laughably notice that Bayesian scientists are allowed to use histograms of frequencies and frequentist scientists are allowed to use the Bayes rule. This kind of debate is often sterile but it is probably a characteristic of still young non-unified sciences with varying names accross trends. In this section, we briefly describe the specific issues taken into account by these two commonly separated communities.

The Bayesian framework [Barber, 2011] is characterized by its use of parametrized probability laws which allows to benefit from interpretation ease (including uncertainty) when predicting information of producing description about data. The preferred statistical tool is usually the so-called bayes rule, hence the name of this scientific community. The choice of the modelled distributions carries interpretation and knowledge that is elegantly injected into the trained systems.

In contrast, within the frequentist approach [Bishop, 2006], we assume that uncertainty is inherenty present due to the randomness coming from repeatable experiments producing empirical observations. Hence, many machine learning problems tackled with a frequentist point of view is a statistical estimation problem based on observed data. If one could accept caricatures, then we would say that Bayesian statistical learning is a principled probabilistic and thus interpretable framework (hence their appealing reputation but at the cost of often wrong model assumptions) which offers extraordinary research avenues such as model selection without extra data and meaningful probability interpretations whereas frequentist statistical learning produce good-results-oriented blackboxes [Neal, 1995] with impressive recognition rates resculpting our modern world.

Recently, in a Ph.D. thesis Gal [2016], dared the idea of taking advantage of both worlds in a Bayesian deep learning approach (although the merit was certainly to revisit such a counter-intuitive approach that in fact dates back to at least in the 1990s by Neal [1995] or even by Bishop [1994]). Today in 2020, there is a still a controversial debate on this subject: Bayesian machine learning injects some knowledge through a distribution prior (not always a Gaussian prior even if

we can recognize this is the best studied distribution and the most frequently used) for inputs and outputs of statistical predictors. This guiding of the machine learning at both optimization and prediction steps assumes some knowledge about input and output data but most of the time that knowledge is not existing hence the debate. Frequentist neural networks with all their parameters and Bayesian statistics with all their parametrized distributions might seem uneasy to coexist in the same unique machine learning method.

Sources of Uncertainty

Beyond frequentist and Bayesian considerations, according to recent research work [Gal, 2016] (we recommend the reader to read this Ph.D. thesis for details and comprehensive bibliography about prediction with uncertainty), uncertainty can be broken down into several facets:

EXTRAPOLATION UNCERTAINTY The prediction could be wrong because test data do not come from the same distribution as training data. Out from the training distribution, test data go against one of the most fundamental machine learning hypothesis to make any system work. For example, online self-adapting systems [Bertsekas et al., 1995] look like reinforcement learning systems and deal with out of distribution data uncertainty purposefully. During training, prediction systems never see anything but data coming from training data. Neural networks have the deserved reputation of quickly over-fitting on training data empirical distribution (if used carelessly) which is both a warning against both interpolation and extrapolation data prediction in worst case scenarios as explained by **?** partially avoided by regularization [Srivastava, 2013]. Overfitting is bad extrapolation in essence. Thus, neural networks are particularly prone to extrapolation issues like extrapolation uncertainty.

ALEATORIC UNCERTAINTY Some noise could have been introduced in training data (some wrong coordinates, some wrong labels etc.). Well-studied statistical machineries like linear and kernel-based support vector machines [Andrew, 2001] proposed the notion of margin to cope with some part of that uncertainty but it seems that linking that margin to probability estimation is still an open research problem even with a logistic regression loss instead of a hyperplane in practice;

EPISTEMIC UNCERTAINTY The initial machine learning problem might be ill-defined: many solutions can solve the problem, so we do not know which one to choose objectively. For example, the *butterfly effect* (in layman's terms) is a source of epistemic uncertainty against meteorological forecasts meaning that uncertainty is inherently related to the studied laws of physics as a scientific standing point (see for example Epstein [1969]). Thus, the problem modeling could be unsufficient which introduces randomness.

There exists more precise analysis for describing these interleaved sources of uncertainty and readers may find more exhaustive research in the work of Kennedy and O'Hagan [2001]. But all these phenomena boil down to how systems should and could handle the possible lack of confidence tainting automatic predictions.

2

WASSERSTEIN CLUSTERING

Man Gave Names to All the Animals

Bob Dylan *in* Slow Train Coming, 1979

Joint work with Charles Bouveyron, Pierre-Alexandre Mattei and Andrés Almansa

ABSTRACT

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

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

KEYWORDS

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

2.1 INTRODUCTION

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

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

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

2.1.1 *Related Work*

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

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

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

At the very start of this thesis in 2016, in our preliminary experiments, a Gaussian mixture model trained with Expectation-Maximization [Dempster et al., 1977] on codes coming from a vanilla MLP[1] auto-encoder without convolutions is able to reach above 80% of unsupervised clustering accuracy on the famous digits MNIST images dataset[2] directly on raw pixels. Encouraged by this surprisingly good result, we pursue our efforts towards an attempt to take advantage of that empirical fact in a sound framework.

While supervised classification has been a long-standing problem for many decades until recently thanks to computational hardware dramatic improvements and a considerable research effort towards statistical tools and algorithms, unsupervised classification (a. k. a. clustering) is still a difficult area but taking advantage of the supervised findings proved efficient in terms of research. In fact, although we can only confirm that clustering is an ill-posed problem as explained earlier due to the Kleinberg's impossibility theorem [Kleinberg, 2015], we still find

1 Multi-Layered Perceptron

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

it desirable with the same issues that the supervised classification research had to solve in easier settings because supervised classification has a clearer objective. Beyond ill-posed problems, the way we tackle model-based clustering in this work has at least two issues, namely model selection and mixture parametrization that we brielfy describe here before presenting two techniques, one generative which we improved thanks into a discriminative one.

2.1.2 *Model Selection*

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

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

WITHTOUT ANY OTHER DATASET THAN THE INITIAL TRAINING SET a vast literature exists with Kullback-Leibler divergences and Likelihood Maximization for selecting a good model among many [Schwarz et al., 1978, Biernacki et al., 2000] using the number of model parameters following an Occam's razor for Bayesian Machine Learning to compensate for extra data validation but to the best of our knowledge, we do not know non-likelihood-based techniques (here we use Wasserstein distances instead).

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

2.1.3 *Reparametrization Trick for a Mixture*

In order to handle distributions and more precisely being able to differentiate our objective functions with respect to the parameters of these distributions for learning purposes, we adapt the *Reparametrization Trick* from Kingma et al. [2015] first used in a variational context[3]. Indeed, we have chosen a probabilistic framework from which probability distribution parameters are estimated. Usually, in deep learning, first, the optimization process finds functional parameters for output prediction thanks to the minimization (or maximization) of an objective funcion and the notion of gradient gets easy under mild (sub)-differentiability conditions towards training optimization. In our case, this is different: we do not optimize functional parameters for an output prediction but we do find density parameters for a distribution which plays the role of the output prediction in a probabilistic fashion. In less than a decade, the variational (see the surveys accomplished by Kingma et al. [2019]) and optimal transport (see the book of Peyré et al. [2019]) literatures in machine learning got much attention for that estimation scenario with probability distribution parameters. Instead of using some sophisticated mathematical tools for differentiating some objective over some distribution parameters, many researchers used the *Reparametrization Trick* thanks to Kingma et al. [2015]: transforming a known pseudo-random noise as an estimator of the theorereticall random variable for optimization reasons which makes differentiation easier. Graves [2016] gives a more comprehensive overview of the problem.

Indeed, Kingma et al. [2015] allow to directly specify a prior distribution over the code space of a variational auto-encoder (VAE). Inference is done using a stochastic gradient variational bayesian (SGVB) method, based on a reparametrization of the variational lower bound. In this work, we will revisit and adapt this technique called the *Reparametrization Trick* for our distributions settings. Deep generative models for clustering may be built using a mixture model as prior distribution. This approach was recently explored by Dilokthanakul et al. [2016] and Jiang et al. [2016] who used a Gaussian mixture prior.

In practice, the Reparametrization Trick consists in manipulating a random variable $\mathbf{z}$ coming from a parametrized distribution (say Gaussian of mean $\mathbf{a}$ and covariance matrix $\mathbf{B} = \mathbf{C}\mathbf{C}^\top$ such that $\mathbf{z} \sim \mathcal{N}(\mathbf{a}, \mathbf{B})$) thanks to a default random generator (here $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$) that is transformed through ($\mathbf{z} = \mathbf{a} + \mathbf{C}\boldsymbol{\epsilon}$ which simulates $\mathbf{z} \sim \mathcal{N}(\mathbf{a}, \mathbf{B})$) in order to get a differentiable version of the random

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

variable we needed. This technique can used in many different ways according to the literature [Graves, 2016, Doersch, 2016, Blei et al., 2017].

2.2 GENERATIVE WASSERSTEIN CLUSTERING

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

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

2.2.1 *Deep Generative Models for Clustering*

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

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

Second, inverting the problem of accessing to the parametrized model data distribution is to sample generated data from it which what Goodfellow et al. [2014] astutely proposed with GANs: they minimize a divergence between real and generated data over the generator function parameters implemented by a

neural network thanks to a critic function that evaluates that divergence. Since then, the Wasserstein distance [Gulrajani et al., 2017] through the Wasserstein GAN (WGAN) was also proposed among other divergences summarized by a survey done by Goodfellow [2016] gives a glimpse of what is offered by that thriving on-going scientific literature.

We aim at clustering a dataset of N points $\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N$ samples of the random variable $\mathbf{x}$ living in a space $\mathcal{X}$ (say $\mathbb{R}^D$) into K homogeneous groups. We suppose there exists a latent code space $\mathcal{Z}$ of a low dimension d (say $d = 10$ which is low compared to the original data dimensionality D: $d \ll D$) such that there is a mapping $\mathcal{D}$ between $\mathcal{Z}$ to $\mathcal{X}$ connecting the random variable $\mathbf{x}$ in $\mathcal{X}$ and its latent counterpart $\mathbf{z}$ in $\mathcal{Z}$. We also assume that $\mathbf{z}$ follows a mixture $\mathcal{M}$ of rather common distributions as components (say Gaussian).

At the heart of our Generative Wasserstein Clustering (GeWaC) model, there is an auto-encoder made of: (i) an encoder network $\mathcal{E}$ (parametrized by $\theta_{\mathcal{E}}$) and (ii) a decoder network $\mathcal{D}$ (parametrized by $\theta_{\mathcal{D}}$). That auto-encoder plays the role of a two-ways bridge between the data space and a code (or latent) space which more akin to clustering alleviating the curse of dimensionality thanks to its lower dimensionality (d instead of D).

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

$$c \sim \text{Cat}(\boldsymbol{\pi}) \tag{28}$$

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

$$\mathbf{z}|c = k \sim g_k(.;\xi_k) \qquad\qquad \mathbf{z} \sim \sum_{k=1}^{K} \boldsymbol{\pi}_k g_k(.;\xi_k) \tag{29}$$

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

$$\mathbf{x}|\mathbf{z} \sim \delta_{\mathcal{D}(\mathbf{z})} \tag{30}$$

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

$$\mathbb{P}(c = k|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x})}[P(c = k|\mathbf{z})] \tag{31}$$

Although any parametric density functions can be used for each mixture component g_k, we restrict ourselves in this work to densities allowing the use of the

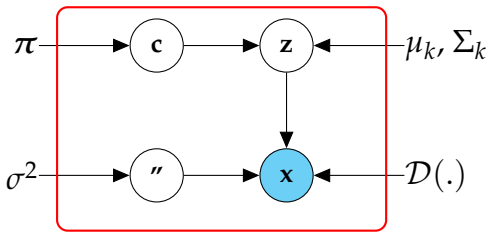

Figure 11: Graphical Model for Data Generation

reparameterization trick [Kingma and Welling, 2013, Kingma et al., 2015] which has been described above and has a central role in our adversarial optimization as we will see later.

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

$$g_k(.;\xi_k) = \mathcal{N}(.;\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{32}$$

(a Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$) where $(\boldsymbol{\pi}_k)_{k=1,\dots,K}$, $(\boldsymbol{\mu}_k)_{k=1,\dots,K}$, and $(\boldsymbol{\Sigma}_k)_{k=1,\dots,K}$ are the mixture parameters stored in $\theta_{\mathcal{M}}$. Note that, beyond the Gaussian mixture prior that we consider here, our approach could be extended to any mixture of reparametrizable distributions: one might for example consider a mixture of von Mises as the prior distribution, in order to obtain interesting visualizations on a hyper-sphere, such as the ones of Davidson et al. [2018]. Our graphical model displayed in Fig. 11 summarizes the chosen data generation modelling.

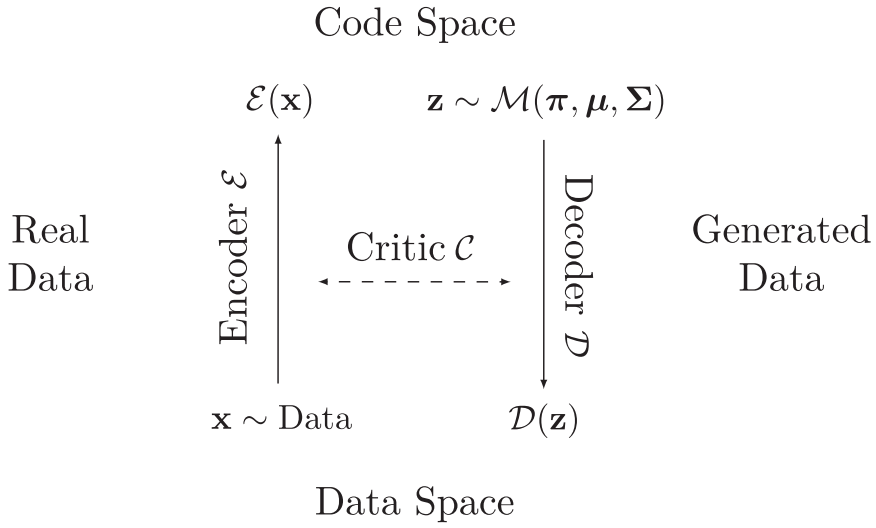

Figure 12: GeWaC Optimization Scheme

In order to fit our generative model, we generate a random variable $\mathcal{D}(\mathbf{z})$ to match $\mathbf{x}$ in terms of Wasserstein distance in an adversarial fashion. We only have access to samples of $\mathbf{x}$ (through the dataset) and $\mathcal{D}(\mathbf{z})$ (through a pseudo-random

generator for $\mathbf{z}$ and the decoder or generator $\mathcal{D}$), which is a scenario where the WGANs proved successful as Fig. 12 summarizes the modeling proposed here.

The posterior probability $p(\mathbf{z}|\mathbf{x})$ in Eq. (31) is hard to compute because of the nonlinearity of the decoder. But, as in the work of Kingma and Welling [2013], we can approximate it using an inference network $q(\mathbf{z}|\mathbf{x})$ built according to the encoder $\mathcal{E}$. As emphasized by Kingma et al. [2015], minimizing the Kullback-Leibler divergence between the true posterior and $q(\mathbf{z}|\mathbf{x})$ leads to minimizing a penalized quadratic auto-encoder loss. Since $\sigma \to 0$, the dominating term in this loss will precisely be the loss of a vanilla auto-encoder which is what we do in practice for the sake of simplicity. Eventually, we can compute an approximation of $P(c = k|\mathbf{x})$ by simply replacing the true posterior by the approximation, which leads to the maximum-a-posteriori (MAP) rule in code space:

$$P(c = k|\mathbf{x}) \simeq \frac{\boldsymbol{\pi}_k g_k(\mathcal{E}(\mathbf{x}); \xi_k)}{\sum_{k'=1}^{K} \boldsymbol{\pi}_{k'} g_{k'}(\mathcal{E}(\mathbf{x}); \xi_k)} \tag{33}$$

following the Bayes formula.

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

2.2.2 *Concatenation Trick*

Our concatenation approach consists in operating a clustering that is consistent both in the embedding code and input data spaces as described in Fig. 12:

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

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

everywhere on the data and code manifolds at hand (see Donahue et al. [2016] for proof).

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

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

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

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

In fact, we first proposed a model without concatenation which meant the adversarial WGAN optimization did not involved the encoder $\mathcal{E}$ as only the distribution of $\mathbf{x}$ and $\mathcal{D}(\mathbf{z})$ were being put together: this was not principled especially because the clustering decision rule given in Eq. (33) involves $\mathcal{E}$ which forced us to retrain a final post-processing with an adhoc autoencoder loss with all parameters fixed except the ones of $\mathcal{E}$ for what seems to be an update with respect to the decoder $\mathcal{D}$. The contribution of Dumoulin et al. [2017] made all these considerations disappear in an easy yet principled fashion.

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

$$\max_{\|\nabla\mathcal{C}\|\leq 1} \mathbb{E}_{\mathbf{x}\sim p}\left[\mathcal{C}\big(a(\mathbf{x})\big)\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\mathcal{C}\big(b(\mathbf{z})\big)\right] \tag{34}$$

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

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

$$\mathbf{y}_k = b(\mathbf{z}_k) = \left[\mathcal{D}(\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top, (\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top\right]^\top \tag{35}$$

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

All the equations above meet in:

$$\begin{aligned}\mathcal{L}(\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{M}}, \theta_{\mathcal{C}}) \;=\; & \mathbb{E}_{\mathbf{x}\sim p}\left[\mathcal{C}\left(\left[\mathbf{x}^\top, \mathcal{E}(\mathbf{x})^\top\right]^\top\right)\right] \\ & - \sum_{k=1}^{K} \boldsymbol{\pi}_k \times \mathbb{E}_{\mathbf{e}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)}\left[\mathcal{C}\left(\left[\mathcal{D}(\mathbf{S}_k \times \mathbf{e} \right.\right.\right. \\ & \left.\left.\left. + \boldsymbol{\mu}_k)^\top, (\mathbf{S}_k \times \mathbf{e} + \boldsymbol{\mu}_k)^\top\right]^\top\right)\right]\end{aligned} \tag{36}$$

and thus we optimize:

$$\min_{\theta_{\mathcal{E}},\theta_{\mathcal{D}},\theta_{\mathcal{M}}} \max_{\theta_{\mathcal{C}}} \mathcal{L}(\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{M}}, \theta_{\mathcal{C}}) \tag{37}$$

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

Originally Wasserstein GAN uses a simple fixed distribution (Gaussian or uniform) for the random noise generator that is transformed by a neural network

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

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

2.2.3 *Algorithm*

Our GeWaC algorithm can be decomposed in three successive steps:

1. **Auto-Encoder Initialization**

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

   $$(\theta^0_{\mathcal{E}}, \theta^0_{\mathcal{D}}) = \arg\min_{\theta_{\mathcal{E}}, \theta_{\mathcal{D}}} \mathbb{E}_{\mathbf{x}\sim p}\left(\|\mathcal{D}\circ\mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2^2\right) \tag{38}$$

2. **EM-based Gaussian mixture initialization**

   We fit a Gaussian mixture model

   $$\mathcal{M}\left(\text{SoftMax}(\boldsymbol{\alpha}), \boldsymbol{\mu}, (\boldsymbol{S}_k \times \boldsymbol{S}_k^\top)_{k=1,\dots,K}\right) \tag{39}$$

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

   $$\theta^0_{\mathcal{M}} = \arg\max_{\theta_{\mathcal{M}}} \mathbb{E}_{\mathbf{x}\sim p}\left[\log\left(\sum_{k=1}^{K} \boldsymbol{\pi}_k \times \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{S}_k \times \boldsymbol{S}_k^\top)(\mathcal{E}^0(\mathbf{x}))\right)\right] \tag{40}$$

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

3. **Critic Initialization**

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

   $$\hat{\theta}_{\mathcal{C}} = \arg\max_{\theta_{\mathcal{C}}} \mathcal{L}(\theta^0_{\mathcal{E}}, \theta^0_{\mathcal{D}}, \theta^0_{\mathcal{M}}, \theta_{\mathcal{C}}) \tag{41}$$

from Eq. (37) which is optimized thanks to the algorithm 1 calling the algorithm 2 but without 3.

4. **Clustered Data Generation**

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

$$(\hat{\theta}_{\mathcal{E}}, \hat{\theta}_{\mathcal{D}}, \hat{\theta}_{\mathcal{M}}) = \arg \min_{\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{M}}} \max_{\theta_{\mathcal{C}}} \mathcal{L}(\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{M}}, \theta_{\mathcal{C}}) \tag{42}$$

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

Algorithm 1 Optimization algorithm

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

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

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

4: **for** $j = 1, \ldots, N_{\text{critic}}$ **do**
5: Wasserstein Estimation Step
6: **end for**
7: Wasserstein Minimization Step
8: **end while**

Algorithm 2 Wasserstein Estimation Step

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

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

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

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

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

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

4: Compute

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

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

2.2.4 *Model Selection for GeWaC*

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

We could consider our generative attempt as a natural extension of what the Expectation-Maximization for the Gaussian mixture model algorithm does with the Kullback-Leibler divergence but with stochastic gradient descent for the Wasserstein Distance. One difficulty appears though from mixture distributions being problematic because of the partial discreteness of the parameter

Algorithm 3 Wasserstein Minimization Step

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

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

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

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

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

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

4: Compute

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

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

space [Graves, 2016]. Specifically, mixture weights are particularly difficult to optimize. Consequently, Dilokthanakul et al. [2016] assume that these weights are known beforehand, which is not always the case in real-world problems. Jiang et al. [2016] propose a method for optimizing these weights but does not provide empirical evidence on imbalanced data to support this scheme. On our side, we did manage to get reasonable results but without completely being fair about proportions: if we trust our careful initialization procedures and let proportions have very low learning rate, then in practice, it is as if we freeze these proportions to a constant vector value although we did not realize it at first while being deceived by apparently good results except when initial proportions are not valid.

2.2.5 *Collapsing Effects*

Since its introduction of GANs, Goodfellow et al. [2014] warned the reader about what they called the "Helvetica scenario" in which their generator is trained *too often* compared to the not-enough-updated *discriminator* (a word that is translated by *critic* since Wassertein GANs [Arjovsky et al., 2017]). Indeed, a generator can intuitively be good at generating data from a specific region of space without

being able to generalize to other space zones of data as the discriminator is fooled when comparing good localized generated data compared to real data. This is a kind of a spatial unsupervised over-fitting that is commonly defined in supervised learning. We end up with a GAN that gets unable to generate the same variety of data as real data hence the *collapsing effect* phenomenon name.

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

2.3 DISCRIMINATIVE WASSERSTEIN CLUSTERING

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

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

We propose an algorithm to perform unsupervised classification (a.k.a. clustering) within this framework that we call "DiWaC" for Discriminative Wasserstein Clustering. Using the idea of separating clusters as a discriminative objective function is not new (see for example Spectral Clustering [Von Luxburg, 2007] or DIFFRAC [Bach and Harchaoui, 2008]) but the fact that we handle distributions

allows to enable both out-of-sample clustering and model selection at the same time which makes our technique appealing even in large scale settings.

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

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

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

2.3.1 *Wasserstein Distances between Clusters*

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

$$\boldsymbol{\tau}(\mathbf{x}) = [\mathbb{P}(c = 1|\mathbf{x}), \dots \mathbb{P}(c = k|\mathbf{x}), \dots, \mathbb{P}(c = K|\mathbf{x})]^\top \tag{43}$$

The number K of groups can be found through model selection which is explained later (section 2.3.5)

Through clustering we infer a function to describe hidden structure from unlabeled data. We aim at clustering a dataset of N samples $\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N$ of the random variable $\mathbf{x}$ (of distribution p) living in a space $\mathcal{X}$ (say $\mathcal{X} = \mathbb{R}^D$) in K groups (or clusters). Usually, in the literature in Machine Learning (*a fortiori* including

Deep Learning), scientists minimize probability divergences, namely: Kullback-Leibler divergence which is equivalent to maximizing the likelihood in many cases (with a wide range of tools from logistic regression i. e. cross-entropy loss for classification to Expectation-Maximization for clustering), the Jensen-Shannon divergence in Generative Adversarial Networks [Goodfellow et al., 2014] as remarked and extended to Wasserstein distance (which is thus also a divergence with nicer and geometric properties) by Arjovsky et al. [2017]. The originality of the current work in clustering settings where the grouping are unknown lies on the maximization of divergences between groups instead of the usual minimization of divergences. This makes our contribution close to Discriminative Latent Models [Bouveyron and Brunet-Saumard, 2014a], Fisher Expectation Maximization [Bouveyron and Brunet, 2012] and Spectral Clustering [Von Luxburg, 2007] and we took some ideas from this vein of research (especially about how we handle proportions to normalize our objectives as we will see). The Wasserstein distance is also a divergence and thus has better properties than the Kullback-Leibler which constitute the main motivation of this work: mainly symmetry and geometric interpretations such as the triangle inequality and being defined even when the compared distributions do not have the same support.

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

$$\mathbf{x} \sim p = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times p_k \tag{44}$$

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

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

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

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

$$p_k(\mathbf{x}) = p(\mathbf{x}) \times \frac{\boldsymbol{\tau}_k(\mathbf{x})}{\boldsymbol{\pi}_k} \tag{45}$$

and similarly, we define:

$$\bar{p}_k(\mathbf{x}) = p(\mathbf{x}) \times \frac{1 - \boldsymbol{\tau}_k(\mathbf{x})}{1 - \boldsymbol{\pi}_k} \tag{46}$$

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

2.3.2 *Unormalized and Normalized sum of inter-Cluster Wasserstein Distances*

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

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

$$\mathcal{L}_{\mathrm{u}}(p_1, \ldots, p_k, \ldots, p_K) = \frac{1}{K} \sum_{k=1}^{K} W(p_k, \bar{p}_k) \tag{47}$$

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

$$\mathcal{L}_{\mathrm{n}}(p_1, \ldots, p_k, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_k) \times W(p_k, \bar{p}_k) \tag{48}$$

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

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

Sanity Checks

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

$$p = \frac{1}{\alpha + \beta + 1} \times (\alpha \mathcal{N}(\mathbf{0}_2, \sigma \times \mathbf{I}_2) + \beta \mathcal{N}(\mathbf{b}, \sigma \times \mathbf{I}_2) + \delta_{\mathbf{c}}) \tag{49}$$

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

This *almost* Gaussian mixture case (corrupted by a down-weighted and far-located Dirac distribution), it is still interesting to analyze what would happen when clustering with $K = 2$ groups making Eq. (47) ending up with the maximization of $W(p_1, p_2)$ (p_1 and p_2 being unknown).

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

Example 1. Ideally, a satisfactory solution would be:

$$p_1^{\text{good}} = \frac{1}{\alpha + 1} \times (\alpha \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2) + \delta_{\mathbf{c}}) \text{ and } p_2^{\text{good}} = \mathcal{N}(\mathbf{b}, \mathbf{I}_2) \tag{50}$$

with proportions

$$\boldsymbol{\pi}_1^{\text{good}} = \frac{\alpha + 1}{\alpha + \beta + 1} \simeq 0.33 \text{ and } \boldsymbol{\pi}_2^{\text{good}} = \frac{\beta}{\alpha + \beta + 1} \simeq 0.66 \tag{51}$$

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

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

Example 2. There is a bad clustering candidate that unfortunately gets a better score with respect to Eq. (47):

$$p_1^{\text{bad}} = \frac{1}{\alpha + \beta} \times (\alpha \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2) + \beta \mathcal{N}(\mathbf{b}, \mathbf{I}_2)) \text{ and } p_2^{\text{bad}} = \delta_{\mathbf{c}} \tag{52}$$

with proportions

$$\boldsymbol{\pi}_1^{\text{bad}} = \frac{\alpha + \beta}{\alpha + \beta + 1} \simeq 0.99 \text{ and } \boldsymbol{\pi}_2^{\text{bad}} = \frac{1}{\alpha + \beta + 1} \simeq 3.3 \times 10^{-5} \tag{53}$$

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

Indeed for Eq. (47), the *good* and *bad* solutions give approximately (thanks to the chosen caricatural coefficients α, β, σ, m and M):

$$\mathcal{L}_{\text{u}}(p_1^{\text{good}}, p_2^{\text{good}}) \simeq 9 \tag{54}$$

$$\mathcal{L}_{\mathrm{u}}(p_1^{\mathrm{bad}}, p_2^{\mathrm{bad}}) \simeq 100 \tag{55}$$

which selects the bad candidate. This prooves the sensitivity of the unormalized objective in $\mathcal{L}_{\mathrm{u}}$ with respect to the outlier Dirac distribution in Eq. (49).

In objectives represented in $\mathcal{L}_{\mathrm{u}}$ and $\mathcal{L}_{\mathrm{n}}$, the clusters are separated but $\mathcal{L}_{\mathrm{n}}$ avoids degenerate clustering candidates thanks to the $\pi_k \times (1 - \pi_k)$ term. In our sanity check example, the previous bad clustering is smashed out by the normalization in $\mathcal{L}_{\mathrm{n}}$, the first cluster occupies more than $\pi_1 \simeq 99\%$ of the data leaving the second singleton set with less than $\pi_2 \simeq 0.003\%$) whereas the unormalized $\mathcal{L}_{\mathrm{u}}$ objective rewards a *bad* degenerate candidate solution. Indeed for the normalized objective $\mathcal{L}_{\mathrm{n}}$ gives approximately:

$$\mathcal{L}_{\mathrm{n}}(p_1^{\mathrm{good}}, p_2^{\mathrm{good}}, \pi_1^{\mathrm{good}}, \pi_2^{\mathrm{good}}) \simeq 3.92 \tag{56}$$

$$\mathcal{L}_{\mathrm{n}}(p_1^{\mathrm{bad}}, p_2^{\mathrm{bad}}, \pi_1^{\mathrm{bad}}, \pi_2^{\mathrm{bad}}) \simeq 5.9 \times 10^{-3} \tag{57}$$

selecting the expected good candidate clustering.

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

There is also a more principled way to look at our normalized objective Eq. (48) and the $\pi_k \times (1 - \pi_k)$ term. In supervised classification (say logistic regression nicely explained by Hastie et al. [2005]), we have one-vs-one and one-vs-rest strategies that we use here. More concretely, in a one-vs-one strategy, it would be reasonable to consider the probability $\pi_k \times (1 - \pi_{k'})$ of choosing one point from cluster k and the other point from cluster k' in an independent fashion.

Now we imagine two data generation models to add some theoretical justification to our objective Eq. (48). First, we can imagine a generation model:

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

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

$$\mathcal{L}_{\mathrm{n}}^{\mathrm{OvO}}(p_1, \ldots, p_k, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K) = \sum_{k=1}^{K} \sum_{k'=1}^{K} \boldsymbol{\pi}_k \times \boldsymbol{\pi}_{k'} \times W(p_k, p_{k'}) \tag{58}$$

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

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

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

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

$$\mathcal{L}_{\mathrm{n}}^{\mathrm{OvR}}(p_1, \ldots, p_k, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k, \ldots, \boldsymbol{\pi}_K) = \sum_{k=1}^{K} \boldsymbol{\pi}_k \times (1 - \boldsymbol{\pi}_k) \times W(p_k, \bar{p}_k) \tag{59}$$

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

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

$$W(\boldsymbol{\mu}, \boldsymbol{\nu}) \overset{\mathrm{KR}}{=} \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim \boldsymbol{\mu}} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim \boldsymbol{\nu}} \mathcal{C}(\mathbf{y}) \tag{60}$$

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

$$W(\kappa_1 \times q_1 + \kappa_2 \times q_2, q_3) = \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim \kappa_1 \times q_1 + \kappa_2 \times q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \tag{61}$$

but

$$\begin{aligned} \mathbb{E}_{\mathbf{x} \sim \kappa_1 \times q_1 + \kappa_2 \times q_2} \mathcal{C}(\mathbf{x}) &= \int_{\mathbb{R}^D} \mathcal{C}(\mathbf{x}) \left(\kappa_1 \times q_1(\mathbf{x}) + \kappa_2 \times q_2(\mathbf{x})\right) d\mathbf{x} && (62) \\ &= \kappa_1 \times \int_{\mathbb{R}^D} \mathcal{C}(\mathbf{x}) \times q_1(\mathbf{x}) d\mathbf{x} + \kappa_2 \times \int_{\mathbb{R}^D} \mathcal{C}(\mathbf{x}) \times q_2(\mathbf{x}) d\mathbf{x} \\ &= \kappa_1 \times \mathbb{E}_{\mathbf{x} \sim q_1} \mathcal{C}(\mathbf{x}) + \kappa_2 \times \mathbb{E}_{\mathbf{x} \sim q_2} \mathcal{C}(\mathbf{x}) && (63) \end{aligned}$$

and

$$\mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) = (\kappa_1 + \kappa_2) \times \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \tag{64}$$

thus

$$\begin{aligned} W(\kappa_1 \times q_1 + \kappa_2 \times q_2, q_3) &= \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim \kappa_1 \times q_1 + \kappa_2 \times q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \\ &= \sup_{\mathcal{C} \in \text{Lip-1}} \kappa_1 \times \left(\mathbb{E}_{\mathbf{x} \sim q_1} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \right) \\ &\quad + \kappa_2 \times \left(\mathbb{E}_{\mathbf{x} \sim q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \right) \\ &\leq \kappa_1 \times \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim q_1} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \\ &\quad + \kappa_2 \times \sup_{\mathcal{C} \in \text{Lip-1}} \mathbb{E}_{\mathbf{x} \sim q_2} \mathcal{C}(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim q_3} \mathcal{C}(\mathbf{y}) \end{aligned} \tag{65}$$

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

$$W(\kappa_1 \times q_1 + \kappa_2 \times q_2, q_3) \leq \kappa_1 \times W(q_1, q_3) + \kappa_2 \times W(q_2, q_3) \tag{66}$$

and finally implies that

$$\mathcal{L}_{\text{n}}^{\text{OvR}}(p_1, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K) \leq \mathcal{L}_{\text{n}}^{\text{OvO}}(p_1, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K) \tag{67}$$

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

2.3.3 *Estimating Wasserstein Distances with Deep Learning*

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

$$\forall k \in [\![1, K]\!] \ \boldsymbol{\tau}_k(\mathbf{x}) = \frac{f_k(\mathbf{x})}{\sum_{\ell=1}^{K} f_\ell(\mathbf{x})} \tag{68}$$

On the Kantorovich side, the critics $\mathcal{C}_k$ are also neural networks (of parameters $\theta_{\mathcal{C}_k}$) but with a 1-Lipschitz property gracefuly provided by spectral normalization for each linear (or convolutional) non-activation layers thanks to a simple yet efficient power iteration technique developped recently by Miyato et al. [2018] which gives enough stability to face large scale datasets in the work of Brock et al. [2019]. Injecting all these formulas in $\mathcal{L}_{\text{n}} = \mathcal{L}_{\text{n}}^{\text{OvR}}$, the maximization over all parameters θ (concatenating the θ_{f_k}s and the $\theta_{\mathcal{C}_k}$s) becomes (in several lines):

$$\mathcal{L}_{\text{n}}^{\text{OvR}}(p_1, \ldots, p_K, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K) = \mathbb{E}_{\mathbf{x} \sim p} \left[\sum_{k=1}^{K} \Big(\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k \Big) \times \mathcal{C}_k(\mathbf{x}) \right] \tag{69}$$

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

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

More generally, the fact that we maintain the null equalities

$$\forall k \in [\![1, K]\!]\ \mathbb{E}_{\mathbf{x}\sim p}\left[\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k\right] = 0 \tag{70}$$

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

The objective of Eq. (69) favors $\boldsymbol{\tau}_k(\mathbf{x})$ to be far from $\boldsymbol{\pi}_k$ thanks to the maximization: (i) if cluster memberships $\boldsymbol{\tau}_k(\mathbf{x})$ are too close to their means $\boldsymbol{\pi}_k$, then the objective would be close to zero (its lower bound because a positive linear combination of Wasserstein distances Eq. (69) is non-negative); (ii) when $\boldsymbol{\tau}_k(\mathbf{x})$ is high above its mean $\boldsymbol{\pi}_k$ (bounded by 1), $\mathcal{C}_k(\mathbf{x})$ will be high and when $\boldsymbol{\tau}_k(\mathbf{x})$ is low under its mean (bounded by 0), $\mathcal{C}_k(\mathbf{x})$ will be low too. Accordingly, it is interesting to see that $\mathcal{C}_k(\mathbf{x})$ can be seen as a relaxed decision function (i. e. high for points in k^{th} cluster and low for the other clusters, but bounded in terms of variation due to its Lipschitzian property);

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

2.3.4 *Algorithm*

Algorithm 4 optimizes Eq. (69) which has only a maximization steps which provides a certain engineering ease compared to the min-max optimization scheme required for the GANs. Nevertheless, initialization routines turned out to be crucial in practice. For this highly non-convex optimization, suggest three successive careful initialization steps before the real opitmization.

The optimization of Eq. (69) that we want to accomplish has no guarantee to converge in a global extrema with our neural networks approach, which is

why careful initialization is crucial. That is why our DiWaC algorithm can be decomposed in four successive steps (three for intializations and one for the real optimization). In theory, if we had a universal (i. e. regardless of the convexity) ideal optimizer at our disposal, then the three first steps would be useless but to avoid useless and spurious local minima, we do a three steps initialization for a final one.

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

$$(\theta^0_{\mathcal{E}}, \theta^0_{\mathcal{D}}) = \arg\min_{\theta_{\mathcal{E}}, \theta_{\mathcal{D}}} \mathbb{E}_{\mathbf{x}\sim p} \|\mathcal{D} \circ \mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2^2 \tag{71}$$

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

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

$$\theta^0_{\mathcal{M}} = \arg\max_{\theta_{\mathcal{M}}} \mathbb{E}_{\mathbf{x}\sim p} \left[\log \left(\sum_{k=1}^{K} \boldsymbol{\omega}_k \times \mathcal{N}(\mathcal{E}^0(\mathbf{x}); \boldsymbol{\mu}_k, \mathbf{S}_k \times \mathbf{S}_k^\top) \right) \right] \tag{72}$$

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

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

$$\theta^0_{\mathcal{C}} = \arg\max_{\theta_{\mathcal{C}}} \mathbb{E}_{\mathbf{x}\sim p} \left[\sum_{k=1}^{K} \Big(\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k \Big) \times \mathcal{C}_k(\mathbf{x}) \right] \tag{73}$$

   Indeed, we recall that the critics functions $\mathcal{C}_k$s are just a convenient tool to estimate the Wasserstein distances. With bad critics, the gradient of

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

the objective over the memberships probabilities parameters would be also wrong which is bad news especially when memberships functions are previously and nicely initialized. This suggests this warmup step 3 before the real clustering optimization.

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

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

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{\mathbf{x}\sim p}\left[\sum_{k=1}^{K}\Big(\boldsymbol{\tau}_k(\mathbf{x}) - \boldsymbol{\pi}_k\Big) \times \mathcal{C}_k(\mathbf{x})\right] \tag{74}$$

as described in Algorithm 4.

2.3.5 *Model Selection for DiWaC*

The goal of model section is to check under/over-fitting and ultimately choose hyper-parameters such as the number of classes or the architecture of neurons and layers among several trained models with different hyper-parameters. In this work, our inter-cluster Wasserstein distance measured in Eq. (69) can be measured on some held-out validation data distributions. As stated earlier, critics are just a convenient tool to estimate the Wasserstein distances, so there are at least two ways to estimate the validation objective:

1. Directly (D) "as is": we simply apply Eq. (69) except for adapted proportions $\boldsymbol{\pi}$ measured on these held-out data:

$$\boldsymbol{\pi}_{\text{validation}} \leftarrow \frac{1}{N_{\text{validation}}} \sum_{i'=1}^{N_{\text{validation}}} \boldsymbol{\tau}(\mathbf{x}_{i'}) \tag{75}$$

2. Re-Fitted (R) "with re-optimized critics": we use the same previously adapted proportions Eq. (75) and we optimize again the objective Eq. (69) but only with respect to the critics which corresponds to step 4, Eq. (73). The goal is to refine the objective which corresponds to a separation power of the clustering in terms of interpretation.

Algorithm 4 Optimization algorithm (step 4)

1: **Input:** Data

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

Number of clusters K

2: **Initialization:**

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

3:

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

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

4: $T \leftarrow 0$

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

6: $T \leftarrow T + 1$

7: Free all gradients accumulators

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

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

9: Compute

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

10: so that

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

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

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

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

13: **end while**

Traditionally, model selection is accomplished according to the likelihood (or completed likelihood) related to the Kullback-Leibler divergence (see the

Bayesian Information Criterion BIC [Schwarz et al., 1978] or Information Completed Likelihood ICL [Biernacki et al., 2000] for examples of *extrapolated generalization power measurements*). One of the originality of our approach is that our model selection technique is done according to an other divergence which is the Wasserstein distance. This parallel allows us to use the historical likelihood-based literature by replacing the well-known Kullback-Leibler divergence by the Wasserstein distance to maybe open new avenues of research for future investigations (e. g. adapting BIC and/or ICL beyond likelihood and Kullback-Leibler divergence).

In practice here, we train M times our model with different set of hyper-parameters (different number of clusters, different neural networks structure, different mixture components structure etc.) Models $(\hat{\theta}_m)_{m=1,\dots,M}$ are now evaluated on held-out validation data this time instead of training data (in which they are trained as usual). Finally, we select the best set of parameters: the model indexed by m for which the objective (defined in Eq. (69) with adapted proportions) is maximum.

2.3.6 *Changing the Metric beyond the Euclidean Distance for DiWaC*

In terms of ill-posed problem, the axioms of Kleinberg [2015] are *almost* satisfied: scale-invariance is valid, cluster-shapes-invariance is almost valid up to the expression power of the encoder and critic functions (a. k. a. the capacity of these neural networks in practice) that can be improved thanks to model selection and only metric-invariance remains but we have the intuition that it can be improved in future works thanks to the notion of *worst metric* among a large class of metrics as briefly evoked in the next chapter. Indeed, at the beginning of that clustering chapter, we mentioned that our clustering algorithm could not achieve *consistency* (i. e. metric invariance) because optimal transport required an initial and definitive commitment for the unique distance choice in the data space to build a Wasserstein distance for distributions over the data space. In fact, we could apply some ideas of the next contribution of this thesis to handle a *very large* class of distances at once which is possibly better than only one euclidean distance. Of course, according to the previously cited clustering no-free lunch theorem [Kleinberg, 2015][5], we will never be able to cover *all possible* distances. Nevertheless, we could contemplate a solution where our hereby euclidean distance clustering is an initialization for a newer and better clustering technique for a large class of distances simulataneously in the future towards more consistency with the work of Kleinberg [2015] in mind.

5 The "clustering no-free lunch theorem" is not an official nickname but we choose it because it helps understanding the work of Kleinberg [2015]. The original authors would rather use the clustering "impossibility theorem".

2.4 EXPERIMENTS

2.4.1 *Implementation details and experimental setup*

We did our experiments in Python by using the `pyTorch` [Paszke et al., 2017] and `scikit-learn` [Pedregosa et al., 2011] libraries. with the same learning rate of 10^{-5} with the Adam default optimization strategy [Kingma and Ba, 2014] everywhere. k-Means and GMM are not easily compatible with large scale datasets which is why we took only a reasonable subset of large datasets for these initializations (which is why this is not crucial). In fact, the online learning of k-Means [Bottou and Bengio, 1995] is also possible for EM thanks to Cappé and Moulines [2009] but we found our results already satisfying. In our preliminary experiments, the "AE + GMM" baseline (just the first 2 steps of our algorithm) performed poorly (and with bad reliability not reproducibility) without k-Means++ [Arthur and Vassilvitskii, 2006] and Xavier neural networks weights initialization [Glorot and Bengio, 2010]. Thus, all the experiments we report in this work use them.

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

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

  Unfortunately, we empirically realized that this kind of parametrization is not enough because of the eigenvalues behavior of that *square root* matrix: they numerically explode or implode resulting in ill-conditioned corresponding covariance matrices and create instability. More precisely, it appeared that when the means are far from the optimal means, the system chooses to modify its covariance matrices first instead of the means which should be prevented because of the spurious maxima. Limiting the eigenvalues range is better than an unbounded exponential function. We use a distorted Sigmoid[7] function bounded by two constants initially

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

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

parametrized by the variable-wise standard deviations σ_j of all initial codes variables $\mathcal{E}^0(\mathbf{x}_i)_j$:

$$(\forall t \in \mathbb{R}) \;\; \lambda + (\Lambda - \lambda) \times \text{Sigmoid}(t) \in]\lambda, \Lambda[\tag{76}$$

where:

$$\lambda = 0.3 \times \min_{j \in \{1,\dots,d\}} \sigma_j \text{ and } \Lambda = 3 \times \max_{j \in \{1,\dots,d\}} \sigma_j \tag{77}$$

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

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

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

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

$$\text{ACC} = \max_{\sigma} \sum_{k=1}^{K} \mathbf{M}_{k,\sigma(k)} \iff \min_{\sigma} \sum_{k=1}^{K} \mathbf{C}_{k,\sigma(k)} \tag{78}$$

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

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

$$\text{NMI} = \frac{\text{MI}}{\text{H}} \tag{79}$$

where

$$\text{MI} = \sum_{k,\ell} \tilde{\mathbf{M}}_{k,\ell} \log \left(\frac{\tilde{\mathbf{M}}_{k,\ell}}{\tilde{\mathbf{M}}_{k,\bullet} \times \tilde{\mathbf{M}}_{\bullet,\ell}} \right) \tag{80}$$

and

$$\mathrm{H} = -\sum_{k,\ell} \tilde{\mathbf{M}}_{k,\ell} \log\left(\tilde{\mathbf{M}}_{k,\ell}\right) \tag{81}$$

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

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

$$\mathrm{NCE} = \frac{\mathrm{CE}}{\mathrm{H}} \tag{82}$$

where the conditional entropy CE is:

$$\mathrm{CE} = \sum_{k,\ell} \tilde{\mathbf{M}}_{k,\ell} \log\left(\frac{\tilde{\mathbf{M}}_{k,\ell}}{\tilde{\mathbf{M}}_{k,\bullet}}\right) \tag{83}$$

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

2.4.2 *Introductory Examples for Generative Wasserstein Clustering*

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

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

Now for good convergence at step 4, we decompose the successive minimizations and maximizations by just focusing on the maximization steps of Wasserstein-GANs in a *warm-up* phase. Indeed, in section 2.3.4, after steps 1, 2

and 3, we believe that the decoder and the mixture are well-initialized but the critic neural network is not. It means that our Wasserstein distance estimator (provided by the critic neural network) is not good and neither its gradient with respect to the decoder and the mixture. Concretely, this means that even though the generator (*i.e.* mixture and decoder) is well initialized, the first iterations will decrease the initial generation quality because of a poorly initialized critic that computes our Wasserstein distance value and gradients.

To cope with this bad critic initialization problem at step 4, we choose to optimize the critic alone which is estimating the Wasserstein distance between 2 fixed distributions (real and generated data do not change but the critic does) at the beginning of that step 4. This *warm-up* step concludes a nice initialization for every parameter as you can see in the toy example represented in Fig. 13 and the step 4 finally converges into Fig. 14. Of course, this warm-up step initially developed without generator neural network is also used in our case of WAMiC with the mixture and the decoder as the two components of the generator: first, we do not change the generator and we only change the critic neural network until convergence and then we trigger a second step after this warm-up and the actual min-max Wasserstein GAN optimization scheme takes place.

To illustrate our approach further, we now work on a toy dataset that we call "Three Moons" with 1000 points for each of the 3 groups in 2 dimensions as presented in Fig. 16. With a code space of dimension 1, even though the clusters are not linearly separable in the original data space, our system is able to cluster them in 3 groups with 100% of accuracy.

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

For the number of clusters, without labels in our unsupervised context, while a classification-score-based cross-validation is not an option, one can still measure an estimate of the Wasserstein distance (our loss) in step 3 of section 3.3 but on a validation set. The intuition behind is that if we did not overfit, our loss function will still be satisfactory on that validation set (that was not seen during training). After running our first 3 steps algorithm out of 4, we can train a new neural network f to measure the Wasserstein distance between a held-out validation dataset and some generated data empirical distributions. More precisely, we find the results presented in Fig. 15 actually selecting 3 clusters which is satisfactory.

2.4.3 *Introductory Examples for Discriminative Wasserstein Clustering*

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

THREE MOONS 3M To illustrate our approach further, we now work on a toy dataset that we call "Three Moons" with 1000 points for each of the 3

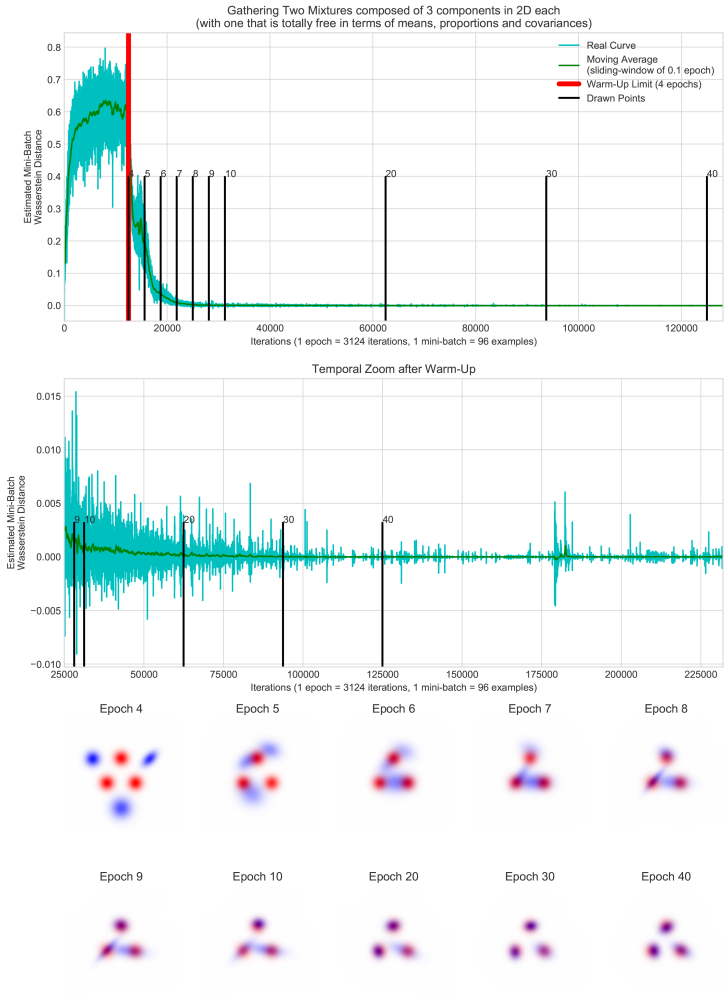

Figure 13: Same problem as EM-GMM but with a Wasserstein GAN without a generator neural network but a Gaussian mixture generator instead. A is the red distribution and B the blue one and they get purple when in local superposition — best seen in colors

groups (with a total of $N = 3000$ points) in dimensions $D = 2$ as presented in Fig. 16;

VARIOUS 2D DISTRIBUTIONS VD Various distributions namely: 2 moons, 1 Gaussian, 1 non-isotropic Gaussian and 1 Student with different proportions

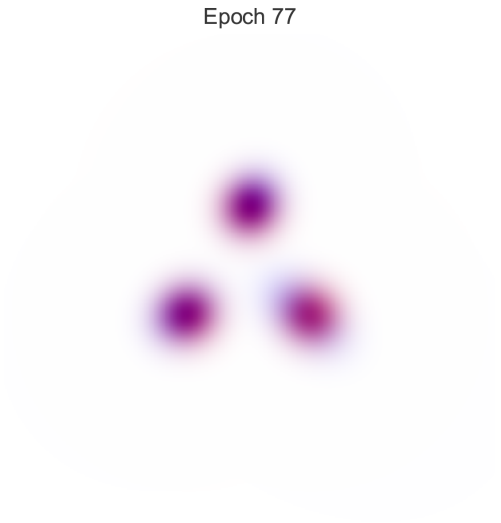

Figure 14: Converged mixtures (purple because of the superposition of the red and blue mixtures) — best seen in colors

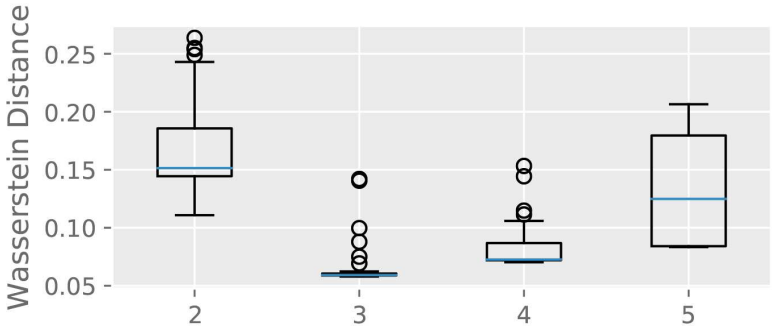

Figure 15: Wasserstein model selection on the three moons dataset for the number of clusters on a validation dataset on 30 runs for each number of clusters (the lower the better)

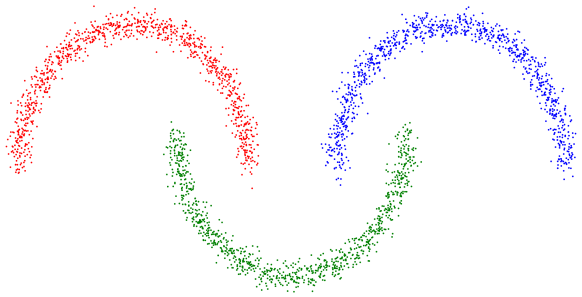

Figure 16: Three Moons 3M

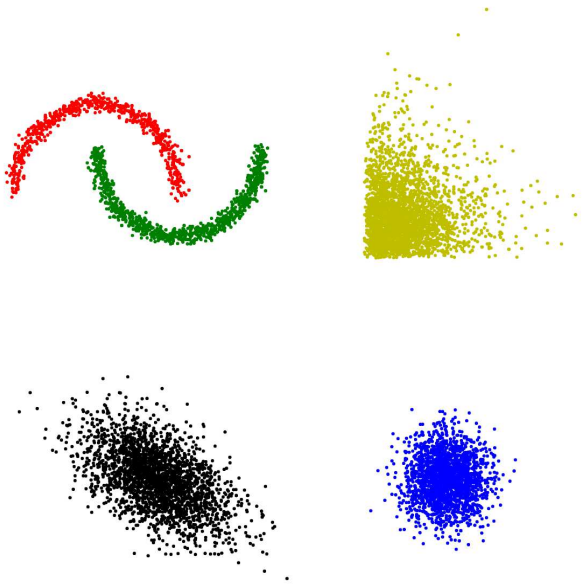

Figure 17: Various 2D Distributions VD

(1000, 1500, 2000, 2500 and 3000 cardinalities respectively and a total of $N = 10000$ points) in dimension $D = 2$ in Fig. 17.

First, as described in section 2.3.4, we trained an auto-encoder on the 3M dataset with a code space dimension of $d = 1$: even though the clusters are not linearly separable in the original data space, our system is able to cluster them in 3 groups with 100% of accuracy (an EM-GMM or a simpler k-Means run in the codes data is enough!). For that simple toy dataset, most of the clustering work is done by the vanilla auto-encoder. Indeed, once the auto-encoder is trained, we observed that the 1D codes are already perfectly separated along one axis with respect to the cluster labels. Here, the point is not to check clustering capabilities but to see if model selection is plausible in this simple scenario.

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

In the little more challenging (but still synthetic) VD dataset in Fig. 17, we observe that dealing with various kinds of distributions (even unknown by the system and beyond the Gaussian case) is still compatible with our mixture-type model-based clustering algorithm.

Datasets	MNIST	Reuters	Reuters-10k	HHAR
DiWaC (ours)	**98.42**	**84.24**	**84.87**	**92.42**
GeWaC (ours with fixed proportions from AE+GMM)	**97.37**	**82.14**	**82.27**	**87.54**
ClusterGAN [Mukherjee et al., 2019]	90.97	–	–	–
VaDE [Jiang et al., 2016]	94.06	79.38	79.83	84.46
DEC [Xie et al., 2015]	84.30	75.63	72.17	79.82
AE + GMM (full covariance)	82.56	70.98	70.12	78.48
IMSAT [Hu et al., 2017]	98.40	–	71.00	–
GAR [Kilinc and Uysal, 2018]	98.32	–	–	–
DEPICT [Dizaji et al., 2017]	96.50	–	–	–
GMM (diagonal covariance)	53.73	55.81	54.72	60.34
k-Means	53.47	53.29	54.04	59.98

Table 2: Experimental accuracy results (%, the higher, the better) based on the Hungarian method. (the last rows correspond to methods without neural networks)

2.4.4 *Real Data Experiments*

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

- MNIST: 70 000 handwritten digits images dataset living in dimension 784 (for 28×28 pixels);
- Reuters: English news stories labeled with a category tree Lewis et al. [2004]. Following DEC Xie et al. [2015], we used 4 root categories: corporate/industrial, government/social, markets, and economics as labels and discarded all documents with multiple labels. We computed tf-idf features on the 2000 most frequent words to represent all articles;
- Reuters-10k: a random subset of Reuters with only 10 000 examples (selected with precisely the same random generator seed as DEC);
- HHAR: The Heterogeneity Human Activity Recognition (HHAR) dataset Stisen et al. [2015] contains 10,299 sensor records from smart phones and smart watches. All samples are partitioned into 6 categories of human activities and each sample is of 561 dimensions.

On MNIST, in Fig. (18), we generated data from our GeWaC further and further in random directions from the centroids: we see that the digits get *fancier* away from the centroids. The good quality of the generation is comparable to those of regular GANs but in a cluster-wise fashion.

In these experiments, we used the same encoder (with symmetric decoder) MLP[8] structure from Xie et al. [2015] D-500-500-2000-d (D is the dimensionality of the input space e. g. $D = 784$ for MNIST and $d = 10$ is the dimensionality of

8 MLP stands for Multi-Layer Perceptron

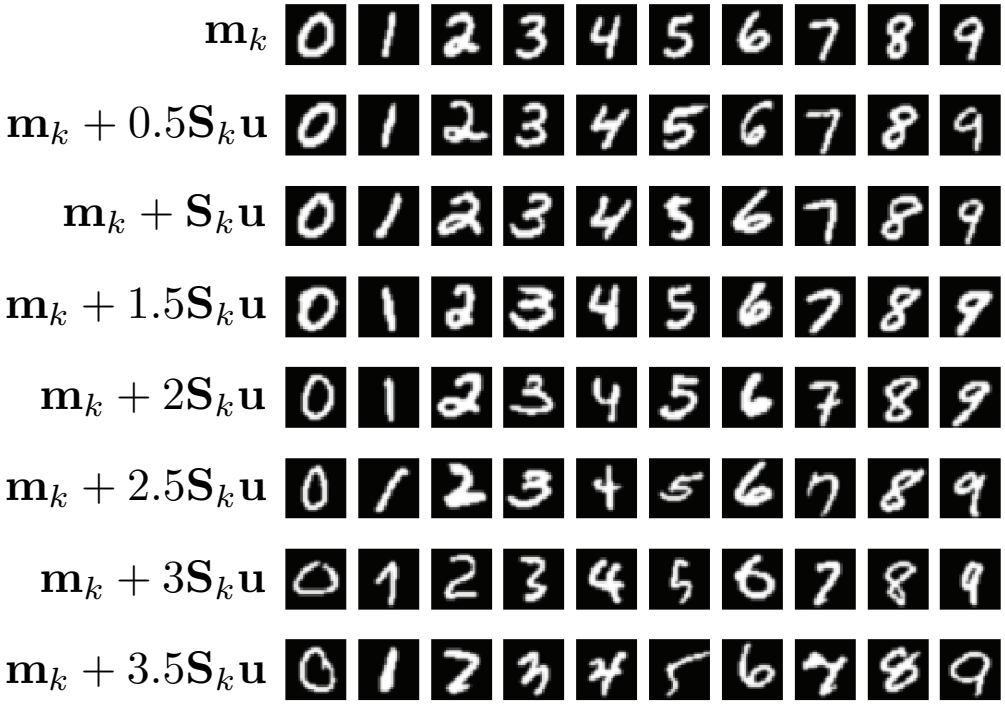

Figure 18: Generated digits images. From left to right, we have the ten classes found by GeWaC and ordered thanks to the Hungarian algorithm. From top to bottom, we go further and further in random directions from the centroids (the first row being the decoded centroids). More specifically, $\mathbf{u}$ is sampled from the uniform random density on the unit hypersphere in the code space.

the code space) (and ReLU activations) for fair comparisons with others. In fact, our conditions are tougher than the ones of IMSAT, GAR and DEPICT that use more sophisticated convolutions than matrix-vector products while DEC, VaDE and our DiWaC do not which is very encouraging.

The overall results of our DiWaC approach compare favorably to the deep clustering state-of-the-art in Table. 2. First, we observe that there is an improvement over standard baseline algorithms ("AE + GMM" compared to GMM and even *k*-Means) when fed with the output of an AE which is coherent with the results of Xie et al. [2015]. We must admit how surprised we are that this "AE + GMM" baseline works so well on various datasets. Undiscovered mathematical theories could tackle this phenomenon: *Why would a simple auto-encoder consistently map data in separated Gaussian groups?*. Maybe the auto-encoders layers loosely behave like successive *cluster-wise deterministic* random projections [Bingham and Mannila, 2001] and non-linear functions: indeed, the linear layers matrix

entries have Gaussian frequencies but this would require further investigations to get thorough scientific interpretations. On a more practical level, we are particularly interested in real-world industrial cases where we highly recommend this "AE + GMM" baseline for its ease of implementation and good results in practice and also for fast prototyping.

There is a supplementary and significant improvement for DiWaC over its "AE + GMM" initialization. These good results are even comparable to those of supervised non-convolutional networks with just a few layers of the 1990s[9]. On MNIST, the important performance gap between VaDE and DiWaC can be explained by difficulties (already studied in [Jiang et al., 2016]) with data that lie extremely close to low-dimensional manifolds, like images. In that regard, our algorithm inherits the strenghts of WGAN [Arjovsky et al., 2017] and models data much more faithfully without our DiWaC algorithm being adversarial which means that Optimal Transport alone is a powerful tool as the non-adversarial work of Genevay et al. [2017] also tends to proove. Furthermore, as presented previously (section 2.3.5), our strategy has a natural criterion for model selection whereas VaDE has no principled model selection criterion; in theory, the validation likelihood could be used, however, computing the likelihood of for variational techniques remains an open question (see e.g. [Cremer et al., 2018]). In other works, DEC's authors propose to use the ratio between training and validation loss as a criterion, however this *ad hoc* solution has little statistical foundations. Model selection is definetely the main feature of our approach on top of providing good results at same hyper-parameter configurations. In fact, better neural networks structure can be found by cross-validation with our algorithm which could be done in future work.

2.5 FUTURE WORK AND CONCLUSION

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

In a mockingly accurate description of the generative part of our work, we could say that placing a mixture distribution at the input of new Generative Adversarial Networks seemed worthwhile for clustering purposes. Indeed, we

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

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

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

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

Within the context of the algorithm laid out above, we empirically observe some symbiosis operating between generative or discriminative clustering and non-linear embedding. A great advantage of our approach is model selection especially for the number of clusters that seems to empirically work well. In other

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

3

UNSUPERVISED FEATURE IMPORTANCE

What Makes Paris Look like Paris?

probably Alexei A. Efros *in* SIGGRAPH, 2012

ABSTRACT

Machine learning and pattern recognition requires data analysis: a process of inspecting, cleansing, transforming and modeling data with the goal of discovering useful information, informing conclusion and supporting decision-making. Unsupervised learning as a scientific field provide tools for dimensionality reduction, visuzalization procedures, features extraction etc. in order to empower human beings with enough computational power to grasp the environment we live in.

This chapter is more of plea for further investigations towards unsupervised feature importance rather than a scientific contribution. Indeed, we revisit notions such as differenciation for distributions, distribution Wasserstein-based metrics and manifold normal in order to call for both further theoretical and practical research work.

KEYWORDS

Feature importance, relevant features, manifold, foreground/background segmentation, cosegmentation, hypersurface normal

3.1 INTRODUCTION

This chapter aims at exploring possible statistical and algorithmical tools for unsupervised feature importance extraction. Indeed, the problem of extracting the absolute or relative importance of data coordinates is of high interest for understanding data. In supervised learning, revisiting the principle of Occam's razor gave birth to a considerable literature and we suggest at least one thesis to the reader: *Structured Sparsity-Inducing Norms: Statistical and Algorithmic Properties with Applications to Neuroimaging*, the Ph.D. manuscript of Jenatton [2011] which combines coordinate selection, weighting and structure in high-dimensional problems such as neuro-imaging. At the same time, the decision trees literature also carefuly studied the problem in several works by Breiman [2001, 2017] and more recently, there is a will to break the *black box taboo* of neural networks being supposedly non-interpretable with interesting attempts by Knight [2007] and de Sá [2019]. The emerging popularity of add-on toolboxes such as Captum [Kokhlikyan et al., 2019] is a solid proof showing a research trend towards input space interpretability in supervised settings at least.

Back in unsupervised learning, the problem of selecting or weighting coordinates by relevance is probably an ill-posed problem because there is no supervision. As usual in pattern recognition, we assume that data live in an instrisically low-dimensional manifold compared to the whole data space dimensionality. If we are able to find a machine learning procedure able to compute an hypersurface normal of that manifold at each point of it, then we can interpret that normal direction coordinates as relevant or not for describing the manifold. More precisely, we can ask ourselves *Is it possible to maximally change a manifold of data with an infinitesimally small distortion?* and the distortion would be a function of space giving high amplitudes to coordinates that one should not change in order to preserve the manifold consistency. This question is reminiscent to the notion of gradient and we make the hypothesis that perturbating data in an infinitesimally small fashion can be done with gradient of a Wasserstein distance between the real data distribution and a pertubated version of it.

One possible application of this work could be unsupervised foreground / background segmentation. Indeed, from a dataset of images containing the same high-level semantic category of content (e. g. "wolves") in several outdoor / indoor conditions, the revealed coordinates would select foreground pixels from background non-content-manifold-specific pixels (that can intuitively be changed without breaking the semantic meaning of the image category). Generalizing such an automatic tool would be of great interest in many scientific fields beyond computer vision.

3.2 INWAMADI: INFINITESIMAL WASSERSTEIN MAXIMAL DISTORTION

In the previous chapter 2, we reviewed some consequences of the impossibility theorem by Kleinberg [2015]. In particular, the metric invariance is an interesting and difficult subject. Indeed, it seems that choosing a particular metric is a heavy commitment. This is especially true in unsupervised learning probably because determining a metric is choosing the algorithms *lenses* for seeing the data without supervision which is redefining a notion of neighborhood tainted by the curse of dimensionality we mentioned earlier in introduction section 1.5.2.

For a random variable $\mathbf{x}$ coming from distribution p living in a space $\mathcal{X}$ (say $\mathbb{R}^D$), we can consider the distortion function $\mathcal{D} = t \times \mathcal{F}$ mapping $\mathcal{X}$ to $\mathcal{X}$ (and $t \in \mathbb{R}^+$) which creates a second random variable $\mathbf{y}$ defined by:

$$\mathbf{y} = \mathbf{x} + \mathcal{D}(\mathbf{x}) = \mathbf{x} + t \times \mathcal{F}(\mathbf{x}) \tag{84}$$

which defines a new distribution $q_{t,\mathcal{F}}$. For the sake of simplicity, we impose:

$$(\forall \mathbf{x} \in \mathbb{R}^D) \quad \|\mathcal{F}(\mathbf{x})\|_2 = 1 \tag{85}$$

so that the length of the distortion is simply t. Our goal is to measure how different a perturbated distribution $q_{t,\mathcal{F}}$ can be from the original distribution p with an infinitesimal length t and constrained energy $\|\mathcal{F}(\mathbf{x})\|_2 = 1$. Thanks to an already successful probabilistic approach in Machine Learning [Murphy, 2012], we rephrase our question set out in our introduction: *What is the infinitesimal steepest distortion of data?* Indeed, this kind of approach would give a function $\mathcal{F}$ such that when applied to each data point $\mathbf{x}_i \in \mathbb{R}^D$, the computed vector $\mathcal{F}(\mathbf{x}_i) \in \mathbb{R}^D$ would tell which coordinate $(\mathbf{x}_i^{(j)})_{j=1,\dots,D}$ is relevant i. e. characteristic in an interpretable way for deeper data analysis especially when the dimensionality D is high.

There is a natural mathematical and geometric tool to measure a distortion for distributions: the Wasserstein distance when the associated data space metric is d. Thus, inspired by the optimization idea of *steepest gradient direction*, we can first define such a function measuring a quantity corresponding to the discrepancy $\mathcal{L}(t, \mathcal{F}, d)$ induced by the distortion $\mathcal{D} = t \times \mathcal{F}$, namely the Wasserstein distance (based on metric d) between the distributions p and $q_{t,\mathcal{F}}$:

$$\mathcal{L}(t, \mathcal{F}, d) = \min_{\gamma \in \Gamma(p, q_{t,\mathcal{F}})} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \left[d(\mathbf{x}, \mathbf{y}) \right] \tag{86}$$

and because

$$\mathbf{y} \sim q_{t,\mathcal{F}} \iff \mathbf{y} = \mathbf{x}' + t \times \mathcal{F}(\mathbf{x}') \text{ and } \mathbf{x}' \sim p \tag{87}$$

we get (without applying a change of variable formula involving a Jacobian term):

$$\mathcal{L}(t, \mathcal{F}, d) = \min_{\gamma' \in \Gamma(p, p)} \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim\gamma'} \left[d(\mathbf{x}, \mathbf{x}' + t \times \mathcal{F}(\mathbf{x}')) \right] \tag{88}$$

with $\Gamma(p,q)$ being the set of coupled distributions with marginals p and q. Please note that without the min operator, this change of variable would not have been valid: between Eq. (86) and Eq. (88), the optimal transport changed from γ to γ' because they are different since Eq. (87).

Now we revisit the notion of *steepest function Wasserstein direction* gives an optimization problem:

$$\mathcal{F}_d^* = \arg\max_{\mathcal{F}} |\nabla_t \mathcal{L}(0, \mathcal{F}, d)| \tag{89}$$

where for a given direction $\mathcal{F}$ and metric d, the quantity $\nabla_t \mathcal{L}(0, \mathcal{F}, d)$ is the derivative of function $t \mapsto \mathcal{L}(t, \mathcal{F}, d)$ on $t = 0^+$ which is:

$$\nabla_t \mathcal{L}(0, \mathcal{F}, d) = \lim_{t \to 0^+} \frac{\mathcal{L}(t, \mathcal{F}, d) - \mathcal{L}(0, \mathcal{F}, d)}{t - 0} \tag{90}$$

which simplifies in:

$$\nabla_t \mathcal{L}(0, \mathcal{F}, d) = \lim_{t \to 0} \frac{\mathcal{L}(t, \mathcal{F}, d)}{t} \tag{91}$$

because $p = q_{0,\mathcal{F}}$ for all direction $\mathcal{F}$ and so, we get:

$$\mathcal{F}_d^* = \arg\max_{\mathcal{F}} \lim_{t \to 0^+} \frac{\mathcal{L}(t, \mathcal{F}, d)}{t} \tag{92}$$

At this point, we ignored the distance d operating in the data space $\mathcal{X} = \mathbb{R}^D$ but we can build such a distance in a form that parses a large variety of metrics:

$$\begin{aligned} d_\phi : \quad \mathbb{R}^D \times \mathbb{R}^D &\to \mathbb{R}_+ \\ (\mathbf{x}, \mathbf{y}) &\mapsto \|\phi(\mathbf{y}) - \phi(\mathbf{x})\|_2 \end{aligned} \tag{93}$$

and we note that $d_\phi(\mathbf{x}, \mathbf{y}) = (L_2 \circ \phi)(\mathbf{x}, \mathbf{y}) = L_2(\phi(\mathbf{x}), \phi(\mathbf{y}))$ (L_2 being the euclidean distance).

We make sure that ϕ is a smooth bijection so that we inherit injectivity (and differentiability for optimization reasons we will see later). Indeed, such a bijective ϕ allows the associated function d_ϕ to verify the distinguishability property of a distance, namely:

$$(\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^D \times \mathbb{R}^D) \quad \mathbf{x} = \mathbf{y} \iff \phi(\mathbf{x}) = \phi(\mathbf{y}) \tag{94}$$

$$\iff 0 = L_2(\phi(\mathbf{x}), \phi(\mathbf{y})) = d_\phi(\mathbf{x}, \mathbf{y}) \tag{95}$$

and the other required properties for being a distance: positivity, symmetry and triangular inequality are given for free thanks to the euclidean distance.

Moreover, we also want to avoid some *equivalence class explosion* effect due to the fact that there is no practical difference of interpretation between choosing a given metric d and a proportional one $\alpha \times d$ (with $\alpha > 0$) especially for explosively large coefficient α. Avoiding such annoying properties can be obtained

by constraining the variations of the function ϕs indexing the distances space made of d_ϕs. Mathematically, the notion of variation for a multivariate bijective function ϕ is studied thanks to the derivative matrix $\nabla\phi(\mathbf{x}) \in \mathbb{R}^{D\times D}$ called the Jacobi matrix at each point $\mathbf{x} \in \mathbb{R}^D$ and the main variation directions are given by its eigen values $(\lambda(\mathbf{x})^{(j)})_{j=1,\dots,D}$. Thus we can propose two ways to constrain these variations:

BOUNDING AMPLITUDE (BA) We limit the Jacobi eigenvalues amplitude:

$$(\forall \mathbf{x} \in \mathbb{R}^D)(\forall j \in [\![1, D]\!]) \;\; \lambda(\mathbf{x})^{(j)} \in [J_{\min}, J_{\max}] \subset \mathbb{R}_+^* \tag{96}$$

ZEROING GLOBAL LOG-AMPLITUDE (ZGLA) We enforce a null logarithm of Jacobi determinant mean:

$$0 = \kappa = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{U}(p,q_{t,\mathcal{F}})} \log|\det \nabla\phi(\tilde{\mathbf{x}})| = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{U}(p,q_{t,\mathcal{F}})} \left[\sum_{j=1}^{D} \log \lambda(\tilde{\mathbf{x}})^{(j)}\right] \tag{97}$$

where:

$$\begin{aligned} \tilde{\mathbf{x}} \sim \mathcal{U}(p, q_{t,\mathcal{F}}) \quad \text{means} \quad & \tilde{\mathbf{x}} = u \times \mathbf{x} + (1-u) \times \mathbf{y} \\ & \text{with} \;\; u \sim \mathcal{U}_{\mathbb{R}}(0,1) \\ & \text{and} \;\; \mathbf{x} \sim p \\ & \text{and} \;\; \mathbf{y} = \mathbf{x}' + t \times \mathcal{F}(\mathbf{x}') \;\; \text{with} \;\; \mathbf{x}' \sim p \end{aligned} \tag{98}$$

Indeed, we need that "zero overall Jacobian property" being true but only over the convex enveloppe of the original and distorted points and not necessarily on the whole space $\mathbb{R}^D$ (manily because we will not evaluate the functions at hand anywhere else outside that enveloppe). This *convex enveloppe* sampling technique has been used for maintaining a Lipschitzian constraint by Gulrajani et al. [2017].

These two combined constraints over the set of smooth bijections defines the functions set Φ. For more mathematical details, we highly recommend the reader the thorough academic book by Ambrosio et al. [2008] to conduct further and more principled studies.

3.2.1 *A Simplified Case: Empirical Distributions*

Before doing some mathematical proposals, we study in this section a simplified case where only empirical distributions are at stake. For that simplified scenario with euclidean distance, we handle $\tilde{p} = \frac{1}{B}\sum_{b=1}^{B} \delta_{\mathbf{x}_{i_b}}$ an empirical distribution from p, we also have $\tilde{q}_{t,\mathcal{F}} = \frac{1}{B}\sum_{b=1}^{B} \delta_{\mathbf{x}_{i_b}+t\times\mathcal{F}(\mathbf{x}_{i_b})}$ from $q_{t,\mathcal{F}}$. For all

$t \in \left[0, \frac{1}{2}\min_{i,i'} \|\mathbf{x}_{i'} - \mathbf{x}_i\|_2\right]$ the optimal transport between $\tilde{p}$ and $\tilde{q}_{t,\mathcal{F}}$ is the natural one as the Fig. 19 shows (we will proove the general case later):

$$\begin{aligned} W_{L_2}(\tilde{p}, \tilde{q}_{t,\mathcal{F}}) &= \min_{\gamma \in \Gamma(\tilde{p}, \tilde{q}_{t,\mathcal{F}})} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \|\mathbf{y} - \mathbf{x}\|_2 \\ &= \min_{\gamma' \in \Gamma(\tilde{p}, \tilde{p})} \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim\gamma'} \|\mathbf{x}' + t \times \mathcal{F}(\mathbf{x}') - \mathbf{x}\|_2 \end{aligned} \tag{99}$$

but with empirical distributions, the transport γ' is in fact an assignment π of integers i_b parsing the points in the $\tilde{p} = \frac{1}{B}\sum_{b=1}^{B} \delta_{\mathbf{x}_{i_b}}$ sum to the integers $i_{b'}$ parsing the points in the $\tilde{q}_{t,\mathcal{F}} = \frac{1}{B}\sum_{b'=1}^{B} \delta_{\mathbf{x}_{i_{b'}} + t \times \mathcal{F}(\mathbf{x}_{i_{b'}})}$ with $\pi(i) = i'$:

$$\begin{aligned} W_{L_2}(\tilde{p}, \tilde{q}_{t,\mathcal{F}}) &= \min_{\pi} \frac{1}{B} \sum_{b=1}^{B} \|\mathbf{x}_{\pi(i_b)} + t \times \mathcal{F}(\mathbf{x}_{\pi(i_b)}) - \mathbf{x}_{i_b}\|_2 \\ &= \frac{1}{B} \sum_{b=1}^{B} \|\mathbf{x}_{i_b} + t \times \mathcal{F}(\mathbf{x}_{i_b}) - \mathbf{x}_{i_b}\|_2 = t \end{aligned} \tag{100}$$

so that $\lim_{t\to 0^+} \frac{W_{L_2}(\tilde{p}, \tilde{q}_{t,\mathcal{F}})}{t} = 1$ for all directions $\mathcal{F}$.

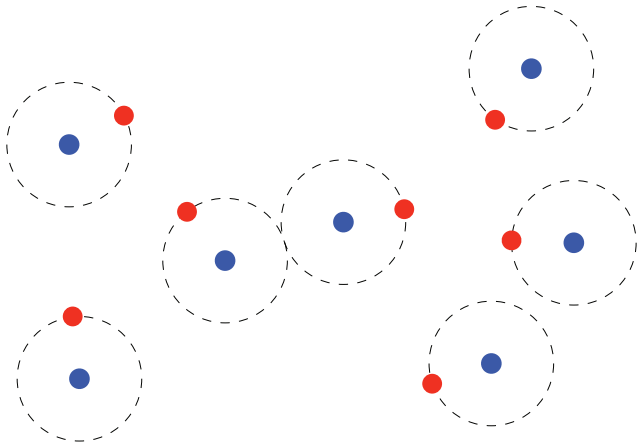

Figure 19: Optimal and Natural Transports are the same in Euclidean Distance Case for close Disttributions

Unfortunately, this is not useful because it means that all $\mathcal{F}$ directions are equally distorting the original distribution with respect to the euclidean Wasserstein distance in a quantity that does not even depend on the distribution $\tilde{p}$.

Based on that failed study, we decide to also optimize the distance d_ϕ to get non trivial $\mathcal{F}$ directions of distortion because we just saw that the euclidean distance is independent from the manifold at hand and thus *too much* isotropic in a data-independent fashion. Optimizing the distance, gives a new optimization problem:

$$\max_{\mathcal{F},\phi} \left(\lim_{t\to 0^+} \frac{W_{d_\phi}(p, q_{t,\mathcal{F}})}{t} \right) \tag{101}$$

instead of $\max_{\mathcal{F}} \left(\lim_{t \to 0} \frac{W_{L_2}(p, q_{t,\mathcal{F}})}{t} \right) = 1$ that previously left us with no optimization hope. This Eq. (101) is much more powerful because we do not only get the steepest distortion but also the pair of steepest distortion direction $\mathcal{F}$ and optimized associated metric d_ϕ.

Now that we have a better grasp on what is mathematically going on, we tackle the general (smooth) distributions case.

3.2.2 *General Distribution Case*

The ideas we just briefly tackled are appealing and now we give a rather theoretical result in order to pursue the optimization side in a general case beyond the euclidean distance.

Proposition. For an infinitesimally small distortion $\mathcal{D} = t \times \mathcal{F}$, the optimal transport between the original distribution p and distorted distribution $q_{t,\mathcal{F}}$ is the natural transport for all smooth metric d_ϕ indexed by bijection ϕ:

$$\begin{aligned}(\exists M \in \mathbb{R}_+^*)(\forall t \in \mathbb{R}_+^*) \;\; t < M \implies W_{d_\phi}(p, q_{t,\mathcal{F}}) &= \min_{\gamma \in \Gamma(p, q_{t,\mathcal{F}})} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \left[d_\phi(\mathbf{x}, \mathbf{y}) \right] \\ &= \mathbb{E}_{\mathbf{x}\sim p} \left[d_\phi(\mathbf{x}, \mathbf{x} + t \times \mathcal{F}(\mathbf{x})) \right]\end{aligned}$$

Proof. For all pair $(\mathbf{x}, \mathbf{y})$ of points drawn from any transport $\gamma \in \Gamma(p, q_{t,\mathcal{F}})$, there exists a point $\mathbf{x}'$ such that $\mathbf{y} = \mathbf{x}' + t \times \mathcal{F}(\mathbf{x}')$ because this is how $q_{t,\mathcal{F}}$ is built. Applying a Taylor expansion on the function $f_{\mathbf{x},\mathbf{x}'}$ defined by:

$$f_{\mathbf{x},\mathbf{x}'}(t) = d_\phi(\mathbf{x}, \mathbf{x}' + t \times \mathcal{F}(\mathbf{x}')) \tag{102}$$

gives:

$$\begin{aligned} d_\phi(\mathbf{x}, \mathbf{y}) &= d_\phi(\mathbf{x}, \mathbf{x}' + t \times \mathcal{F}(\mathbf{x}')) && (103) \\ &= f_{\mathbf{x},\mathbf{x}'}(t) \\ &= f_{\mathbf{x},\mathbf{x}'}(0) + t \times \nabla_t f_{\mathbf{x},\mathbf{x}'}(0) + o(t) \\ &= d_\phi(\mathbf{x}, \mathbf{x}') + t \times \nabla_t f_{\mathbf{x},\mathbf{x}'}(0) + o(t) && (104) \end{aligned}$$

Let's focus on the second term $\nabla_t f_{\mathbf{x},\mathbf{x}'}(0)$, we know thanks to a Taylor expansion on ϕ around $\mathbf{x}'$ that:

$$\begin{aligned} f_{\mathbf{x},\mathbf{x}'}(t) &= \|\phi(\mathbf{x}' + t \times \mathcal{F}(\mathbf{x}')) - \phi(\mathbf{x})\|_2 && (105) \\ &= \|\phi(\mathbf{x}') - \phi(\mathbf{x}) + t \times \nabla\phi(\mathbf{x}') \times \mathcal{F}(\mathbf{x}') + o(t)\|_2 \end{aligned}$$

and thus the only remaining non-negligible term depending on t gives:

$$\nabla_t f_{\mathbf{x},\mathbf{x}'}(0) = \|\nabla\phi(\mathbf{x}') \times \mathcal{F}(\mathbf{x}')\|_2 \tag{106}$$

and in the end, for Eq. (103), we obtain:

$$d_\phi(\mathbf{x},\mathbf{y}) = d_\phi(\mathbf{x},\mathbf{x}') + t \times \|\nabla\phi(\mathbf{x}') \times \mathcal{F}(\mathbf{x}')\|_2 + o(t) \tag{107}$$

and from that Eq. (107) as for all transport $\gamma \in \Gamma(p, q_{t,\mathcal{F}})$ there is an associated coupling $\gamma' \in \Gamma(p,p)$ such that each pair $(\mathbf{x},\mathbf{y}) \sim \gamma$ corresponds to the pair $(\mathbf{x},\mathbf{x}') \sim \gamma'$ (as seen earlier) we get:

$$\begin{aligned} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}\left[d_\phi(\mathbf{x},\mathbf{y})\right] &= \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim\gamma'}\left[d_\phi(\mathbf{x},\mathbf{x}')\right] \\ &\quad + t \times \mathbb{E}_{\mathbf{x}'\sim p}\left[\|\nabla\phi(\mathbf{x}') \times \mathcal{F}(\mathbf{x}')\|_2\right] \\ &\quad + t \times \epsilon_{\mathcal{F},\gamma}(t) \end{aligned} \tag{108}$$

where $\lim_{t\to 0} \epsilon_{\mathcal{F},\gamma}(t) = 0$ and one can note that the second term decouples the $(\mathbf{x},\mathbf{x}')$ pairing. If we consider the natural transport γ^* (i. e. $(\mathbf{x},\mathbf{y}) \sim \gamma^* \iff (\mathbf{x} \sim p \text{ and } \mathbf{y} = \mathbf{x} + t \times \mathcal{F}(\mathbf{x}))$, then we can measure the difference $D(\gamma,\gamma^*)$ for any other non-natural transport γ:

$$\begin{aligned} D(\gamma,\gamma^*) &= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}\left[d_\phi(\mathbf{x},\mathbf{y})\right] - \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma^*}\left[d_\phi(\mathbf{x},\mathbf{y})\right] \\ &= \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim\gamma'}\left[d_\phi(\mathbf{x},\mathbf{x}')\right] + t \times \left(\epsilon_{\mathcal{F},\gamma}(t) - \epsilon_{\mathcal{F},\gamma^*}(t)\right) \quad \text{thanks to Eq. (108)} \end{aligned} \tag{109}$$

which is positive for a $t > 0$ sufficiently small because $\lim_{t\to 0} |\epsilon_{\mathcal{F},\gamma}(t) - \epsilon_{\mathcal{F},\gamma^*}(t)| = 0$ which proves that the natural transport γ^* has the minimal transport cost and thus is the optimal transport. □

When we briefly saw the empirical distribution and euclidean case, it appeared that when the distortion is sufficiently low in amplitude (i. e. with a small t), the optimal transport is the natural one (assigning each original point to its distorted version). With that proposition above in mind and in order to build an optimization objective and an algorithm, we propose a procedure to get a value of such a $t > 0$ satisfying:

$$\max_b \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_b} + t \times \mathcal{F}(\mathbf{x}_{i_b}))\|_2 \leq \frac{1}{2} \min_{b,b'} \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_{b'}})\|_2 \tag{110}$$

The right term does not depend on t and is easily computed from mini-batches of data. The left term can be approximated to guess the right order of magnitude for t that we can divide later until the inequality Eq. (110) is satisfied in a dichotomic fashion. Indeed:

$$\begin{aligned} \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_b} + t \times \mathcal{F}(\mathbf{x}_{i_b}))\|_2 &\simeq t \times \|\nabla\phi(\mathbf{x}_{i_b}) \times \mathcal{F}(\mathbf{x}_{i_b})\|_2 \\ &\leq t \times J_{\max} \text{ because } \|\mathcal{F}(\mathbf{x}_{i_b})\|_2 = 1 \end{aligned} \tag{111}$$

and thus

$$t_k = \frac{\min_{b,b'} \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_{b'}})\|_2}{2^{k+1} \times J_{\max}} \tag{112}$$

with an increasing $k \in \mathbb{N}^*$ is a good strategy until satisfying Eq. (110).

3.3 OPTIMIZATION

In this section, we present a draft of an optimization strategy to summarize the ideas we just presented with neural networks implementations for functions. This is about leveraging the research effort for deep learning in general and GANs in particular for our representation learning and data analysis purposes.

Indeed, we have the function $\mathcal{F}$ implemented by a neural network of pararameters $\theta_{\mathcal{F}}$ mapping $\mathbb{R}^D$ to $\mathbb{R}^D$. We add some layers: a SoftMax layer followed by an element-wise square-root layer with kept sign layer such that we maintain the norm 1 constraint on $\mathcal{F}$.

Thanks to some work accomplished in a different context by Dinh et al. [2017], the implementation of bijection ϕ in a special neural network of parameters θ_ϕ is already done and the constraints needed in section 3.2 are easily applicable. More specifically, we keep on zeroing the overall jacobian mean in a way that is similar to online k-Means [Bottou and Bengio, 1995] via an intermediate bijection ψ that we divide by scalar κ for control the variation of $\phi = \exp(\frac{-\kappa}{D}) \times \psi$. All these parameters are concatenated in θ that the algorithm 20 optimizes.

Maintaining the constrains that we required Eq. (96) and Eq. (97) in section 3.2 is facilitated by the structure of our bijective neural network that we took from Dinh et al. [2017]. Indeed, for bijection ψ implemented by L bijective layers $\psi = \text{Layer}_L \circ \cdots \circ \text{Layer}_\ell \circ \cdots \circ \text{Layer}_1$, we have at each layer ℓ among L:

$$\begin{aligned} \mathbf{x}_{[:d]} &\mapsto \mathbf{y}_{[:d_\ell]} = \text{Layer}_\ell(\mathbf{x})_{[:d_\ell]} = \mathbf{x}_{[:d_\ell]} \\ \mathbf{x}_{[d_\ell:]} &\mapsto \mathbf{y}_{[d_\ell:]} = \text{Layer}_\ell(\mathbf{x})_{[d_\ell:]} = \mathbf{x}_{[d_\ell:]} \times \exp(s_\ell(\mathbf{x}_{[:d_\ell]})) + t_\ell(\mathbf{x}_{[:d_\ell]}) \end{aligned} \tag{113}$$

where the functions s_ℓs and t_ℓs are *free* neural networks (with *pythonic* notations for dimensions and without loss of generality in the coordinates order but with an arbitrary pivot d_ℓ) and then we can bound the associated Jacobi matrices:

$$\begin{aligned} \nabla \text{Layer}_\ell(\mathbf{x}) &= \begin{pmatrix} \mathbf{I}_{d_\ell} & \mathbf{0}_{d_\ell \times (D-d_\ell)} \\ - & \text{diag}(\exp(s_\ell(\mathbf{x}_{[:d_\ell]}))) \end{pmatrix} \\ \text{and } \log|\det \nabla \text{Layer}_\ell(\mathbf{x})| &= \sum_{j=1}^{D-d_\ell} s_\ell^{(j)}(\mathbf{x}_{[:d_\ell]}) \end{aligned} \tag{114}$$

Thanks to the chain rule applied to such a bijection ψ as a composition of layers from $\mathbf{x}^0 = \mathbf{x}$ to $\mathbf{x}^\ell = \text{Layer}_\ell(\mathbf{x}^{\ell-1})$ for $\ell \in [\![1, L]\!]$, we can collect through the forward computations of the function ψ output and sum the outputs of s_ℓ in order to get κ:

$$\kappa = \mathbb{E}_{\mathbf{x}\sim\mathcal{U}(p,q_{t,\mathcal{F}})}\left[\log|\det \nabla\psi(\mathbf{x})|\right] = \mathbb{E}_{\mathbf{x}\sim\mathcal{U}(p,q_{t,\mathcal{F}})}\left[\sum_{\ell=1}^{L}\sum_{j=1}^{D-d_\ell} s_\ell^{(j)}(\mathbf{x}_{[:d_\ell]}^{\ell-1})\right] \tag{115}$$

Having an updated estimate of κ allows us to use $\phi = \exp(\frac{-\kappa}{D}) \times \psi$ instead of ψ directly such that $\log|\det \nabla\phi(\mathbf{x})| = \log|\det \nabla\psi(\mathbf{x})| - \kappa$ which as zero mean.

For numerical stability reasons, we can impose on the neural networks s_ℓs to have a final element-wise $\frac{3\times\tanh}{\sum_{\ell=1}^{L}(D-d_\ell)}$ layer so that $\kappa \in [-3,3]$ is bounded and thus for the local stretching amplitude it gives: $J^{\ell}_{\min/\max} = \exp(\frac{\pm 3}{D-d_\ell})$.

Recently, some Generative Adversarial Networks articles by Pan et al. [2019] and Detlefsen et al. [2019] report some evidence that the Kantorovich-Rubinstein formulation implemented by neural networks in Lipschitz functions [Miyato et al., 2018] has positive regularization effects on the computation and optimization of the Wasserstein-based losses. To benefit from these advances, let's recall what we wanted:

$$\max_{\mathcal{F},\phi}\left(\lim_{t\to 0^+}\frac{W_{d_\phi}(p,q_{t,\mathcal{F}})}{t}\right) \tag{116}$$

Thanks to that Kantorovich-Rubinstein duality (used for example for Wasserstein Generative Adversarial Networks first by Arjovsky et al. [2017]), we know that for sufficiently low value of t (with some approximation):

$$W_{d_\phi}(p,q_{t,\mathcal{F}}) = W_{L_2}(p_\phi,q_{\phi,t,\mathcal{F}}) = \max_{\mathcal{C}\in \mathrm{Lip}_1}\mathbb{E}_{\mathbf{x}\sim p}\left[\mathcal{C}\big(\phi(\mathbf{x})\big) - \mathcal{C}\big(\phi(\mathbf{x}+t\times\mathcal{F}(\mathbf{x}))\big)\right] \tag{117}$$

In fine, we can forget about the limit operator in Eq. (116) for a low value of t because the optimal transport is the natural transport even in the Kantorovich-Rubinstein formulation and thus we have:

$$\max_{\theta_\mathcal{F},\theta_\phi,\theta_\mathcal{C}}\mathcal{L}(\theta_\mathcal{F},\theta_\phi,\theta_\mathcal{C}) \tag{118}$$

with

$$\mathcal{L}(\theta_\mathcal{F},\theta_\phi,\theta_\mathcal{C}) = \frac{\mathbb{E}_{\mathbf{x}\sim p}\left[\mathcal{C}\big(\phi(\mathbf{x})\big) - \mathcal{C}\big(\phi(\mathbf{x}+t\times\mathcal{F}(\mathbf{x}))\big)\right]}{t} \tag{119}$$

and in a stochastic gradient descent strategy, the only important quantity is a bias-free estimate of the gradient [Robbins and Monro, 1951]:

$$\hat{\nabla}\mathcal{L}(\theta_\mathcal{F},\theta_\phi,\theta_\mathcal{C}) = \frac{\nabla\left[\sum_{b=1}^{B}\mathcal{C}\big(\phi(\mathbf{x}_{i_b})\big) - \mathcal{C}\big(\phi(\mathbf{x}_{i_b}+t\times\mathcal{F}(\mathbf{x}_{i_b}))\big)\right]}{t\times B} \tag{120}$$

for a small t and some B random indices $i_b \sim \mathcal{U}_{\mathbb{N}}(1,N)$.

In conclusion of this section, the optimization problem finally gets:

$$\max_{\theta_\mathcal{F},\theta_\phi,\theta_\mathcal{C}}\left(\frac{\mathbb{E}_{\mathbf{x}\sim p}\left[\mathcal{C}\big(\phi(\mathbf{x})\big) - \mathcal{C}\big(\phi(\mathbf{x}+t\times\mathcal{F}(\mathbf{x}))\big)\right]}{t}\right) \tag{121}$$

such that ϕ is bijective, $\mathcal{C}$ is 1-Lipschitz and $\mathcal{F}$ Jacobian the BA and ZGLA properties in Eq. (96) and Eq. (97) for small t given by Eq. (112).

3.4 ALGORITHM

Input: Data $(\mathbf{x}_i)_{i=1,\dots,N}$ where $\mathbf{x}_i \in \mathbb{R}^D$, a mini-batch size B

Initialization:

$\theta_{\mathcal{F}}$ initialized such that $\mathcal{F}$ is uniform throughout coordinates

θ_{ϕ} initialized such that ϕ corresponds to the identity matrix

$\theta_{\mathcal{C}}$ for the 1-Lipschitz critic function

$$\theta = \{\theta_{\mathcal{F}}, \theta_{\phi}, \theta_{\mathcal{C}}\}$$

$$\kappa \leftarrow 0$$

$$T \leftarrow 0$$

while θ has not converged **do**

Free all gradients accumulators

Sample a mini-batch of size B from the dataset

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

Sample B values from a pseudo-random generator

$$u_b \sim \mathcal{U}_{\mathbb{R}}(0, 1) \quad \text{for } b = 1, \dots, B$$

$$t \leftarrow \frac{\min_{b,b'} \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_{b'}})\|_2}{2 \times J_{\max}}$$

while $\max_b \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_b} + t \times \mathcal{F}(\mathbf{x}_{i_b}))\|_2 \leq \frac{1}{2} \min_{b,b'} \|\phi(\mathbf{x}_{i_b}) - \phi(\mathbf{x}_{i_{b'}})\|_2$ is not satisfied **do**

$$t \leftarrow \frac{t}{2}$$

end while

$$\mathbf{y}_b \leftarrow \mathbf{x}_{i_b} + t \times \mathcal{F}(\mathbf{x}_{i_b}) \quad \text{for } b = 1, \dots, B$$

$$\tilde{\mathbf{x}}_b \leftarrow u_b \times \mathbf{x}_{i_b} + (1 - u_b) \times \mathbf{y}_b \quad \text{for } b = 1, \dots, B$$

$$\tilde{\mathbf{x}}_b^0 = \tilde{\mathbf{x}}_b \text{ and } \tilde{\mathbf{x}}_b^{\ell} = \text{Layer}(\tilde{\mathbf{x}}_b^{\ell-1}) \quad \text{for } b = 1, \dots, B \text{ and } \ell = 1, \dots, K$$

$$\kappa \leftarrow \kappa + \frac{1}{T+B} \times \left(\left(\sum_{b=1}^{B} \sum_{\ell=1}^{L} s_{\ell}(\tilde{\mathbf{x}}_{b,[:d]}^{\ell}) \right) - \kappa \right) \right) \quad \text{for } b = 1, \dots, B$$

$$T \leftarrow T + B$$

$$\mathbf{a}_b \leftarrow \exp(\frac{-\kappa}{D}) \times \psi(\mathbf{x}_{i_b}) \quad \text{for } b = 1, \dots, B$$

$$\mathbf{b}_b \leftarrow \exp(\frac{-\kappa}{D}) \times \psi(\mathbf{y}_b) \quad \text{for } b = 1, \dots, B$$

$$\mathcal{L} \leftarrow \frac{1}{t \times B} \sum_{b=1}^{B} (\mathcal{C}(\mathbf{a}_b) - \mathcal{C}(\mathbf{b}_b))$$

Perform a gradient ascent step with $\mathcal{L}$ over θ

end while

Figure 20: Unsupervised Feature Importance Algorithm

3.5 POSSIBLE COMPUTER VISION APPLICATIONS

For the specific computer vision scientific field, pixels are made of one or three coordinates (for gray levels or color images respectively), so showing relevant pixels in images of the same category could lead to an unsupervised foreground/background segmentation where the only supervision is the fact that all images belong to the same semantic category. A similar problem was tackled by Joulin et al. [2010] a few years ago but with a little more supervision: we have access to several images categories labels and they call the problem cosegmentation. As they stated:

> Purely bottom-up, unsupervised segmentation of a single image into foreground and background regions remains a challenging task for computer vision.

This remains true for all kind of data but without different meanings. Fundamentally, if one has a dataset, one could interpret relevance measurements this kind of algorithms could provide. In computer vision, the core idea of co-segmentation is that the availability of multiple images that contain instances of the same "object" classes makes up for the absence of detailed supervisory information. Some research has been efficiently conducted for interactive foreground / background segementation [Rother et al., 2004] but here we would want to avoid user interaction and benefit from a whole dataset: a class of data sharing a common pattern that we want to highlight. The only form of supervision is knowing that data share some information of interest without knowing what precisely.

3.6 FUTURE WORK AND CONCLUSION

Investigating Wasserstein distances with varying metric seems promising for future work. This sketch of contribution can be a stepstone answer to metric invariance pointed out by Kleinberg [2015] for clustering (which is an unsupervised task like feature importance extraction in this work). This shows that this work can be improved in terms of machine learning and optimization and also engineering on real world data.

Indeed, computer vision in general and foreground/background unsupervised segmentation (in the way we present it) in particular are ways to provide a better understanding between unsupervised metric learning and feature importance extraction thanks to large cardinality datasets.

4

PREDICTION WITH UNCERTAINTY

Les espoirs que suscite l'informatique quantique annoncent des améliorations révolutionnaires en matière d'intelligence artificielle (IA) embarquée dans les produits du quotidien. Bien que la recherche en IA évolue rapidement vers l'informatique quantique avec des personnes clés comme Pr. Julia Kempe (succédant à un pionnier de l'IA, le Pr. Yann LeCun à la tête du Centre de Data Science de l'Université de New York depuis quelques années).

Néanmoins, des changements plus imminents et plus radicaux sont déjà en cours grâce à l'électronique de pointe : par exemple, l'accélération matérielle amplifiée par les "processeurs neuronaux" d'Amazon et d'Apple (c'est-à-dire des processeurs dédiés contrairement aux processeurs traditionnels "CPU" généralistes). En particulier, l'électronique de faible puissance est cruciale, car la non-stockage de l'électricité reste un problème malgré les efforts de recherche d'Apple, de Tesla (et de nombreux autres).

ABSTRACT

Uncertainty estimation with neural networks predictions is not yet well studied (except in the recent and controversial field called Bayesian deep learning). Moreover, in the academic and business worlds, some misinterpretations persist about the probabilities as outputs of a supervised classifier: a probability of belonging to a category or its associated vector across all categories does not fully express information on uncertainty. Indeed, most practitioners extract the maximum entry of a probability vector, which is hacky and not sufficient for a more thorough interpretation of uncertainty. At the same time, in many sensitive applications where security, health or even justice issues are involved, it seems that uncertainty estimation is crucial.

The core idea of our Hypothesis of an Uncertainty Model (HUM) contribution is predicting parameters of an output law instead of predicting an output estimate. On the one hand, putting a parametrized probabilistic law on the output side makes our approach Bayesian. On the other hand, using neural networks to produce these parameters makes it less principled but more pragmatic. By combining the best of these two worlds in a small number of additional lines of code is convenient in practice to estimate uncertainty for automatic or assisted decisions in supervised learning in an engineeringly feasible but yet theoretically principled fashion.

KEYWORDS

Bayesian deep learning, Bayesian neural networks, Uncertainty, Confidence, Reparametrization Trick, Neural Networks, Distributions

4.1 INTRODUCTION

There is a large class of supervised learning problems that can be cast in an optimization of this form:

$$\begin{aligned} &\min_{\boldsymbol{\theta}_{\mathcal{F}}} \mathcal{L}(\boldsymbol{\theta}_{\mathcal{F}}) \\ &\mathcal{L}(\boldsymbol{\theta}_{\mathcal{F}}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim \text{Nature}} \left(\ell(\mathbf{y}, \mathcal{F}(\mathbf{x})) \right) \end{aligned} \tag{122}$$

where the quantity $\ell(\mathbf{y}, \mathcal{F}(\mathbf{x}))$ expresses how much one tolerates error for confusing the prediction $\mathcal{F}(\mathbf{x})$ with the groundtruth output $\mathbf{y}$ for input $\mathbf{x}$. Our notations abide by the deliberate starting epistemological choices of this dissertation in section 1.2.2 especially for the Nature idealized distribution. The prediction function $\mathcal{F}$ is implemented here by a neural network (whose parameters are $\theta_{\mathcal{F}}$) but we insist on the fact that it could be implemented by other tools that we can pick from the enormous machine learning zoo of supervised learning of algorithms [Bishop, 2006] while still keeping this current contribution relevant.

We will describe how to modify this Eq. (122) and how to create the right optimization scheme while still being both mathematically principled and engineeringly friendly in a general case with many applications. But first, we want to debunk some widespread misunderstanding among practitioners in supervised classification about how to interpret an output classification probability vector with respect to uncertainty (which was our engineering starting point). Typically, for an image classification problem (say with an AlexNet convolutional neural network [Krizhevsky et al., 2012] trained on ImageNet [Fei-Fei, 2010]), the output is a 1-sum positive vector of high dimensions (for $K \simeq 10^3$ categories for ImageNet) in the form of conditional probability estimates $(\mathbb{P}(c = k|\mathbf{x}))_{k=1,\dots,K}$ for an image $\mathbf{x}$ belonging to which category $k \in [\![1, K]\!]$. Unfortunately, there is a detail that is often overlooked concerning to the "knowing part $|\mathbf{x}$" of the notation $\mathbb{P}(c = k|\mathbf{x})$ which states in an implicit (and sometimes forgotten) fashion that the input $\mathbf{x}$ should be taken from the same theoretical distribution both at training and testing times. Indeed, at training time, the formulas are by construction true because the images are taken from Nature (better approximated by large cardinality dataset). At test time, the input examples must be coming from the same Nature distribution for the knowing part of the probability being true which is just a reasonable hypothesis in theory but that is difficult (and rather impossible) to guarantee in practice (i. e. in every real-world and practical industrial applications). The best proof of it is that if we give a completely random input $\mathbf{x}$ taken from an arbitrary distribution, the system would still answer a 1-sum positive vector output although the input does not belong to any of the categories corresponding to these output vector entries due to its independently random nature.

From these considerations, we draw 2 conclusions for each input $\mathbf{x}$ at test time: (i) taking the maximum entry index k^* of such outputs $(\mathbb{P}(c = k|\mathbf{x}))_{k=1,\dots,K}$ has indeed some significance in a *maximum a posteriori* multinomial perspective

in order to make a concrete decision but nevertheless (ii) the corresponding maximum entry estimating $\mathbb{P}(c = k^*|\mathbf{x})$ has no or little uncertainty significance. If uncertainty is a matter of interest, then the output should be considered as a whole K-dimensional point (and not just the maximum of its entries) living on the unit simplex and uncertainty should be a possibly blurry zone around one predicted point in that space. Now that we have described some practical issues related to uncertainty in classification, we present a general solution to tackle supervised learning problems that takes into account some uncertainty information.

The spectacular come back of deep learning techniques can be dated back to 2012, when Krizhevsky et al. [2012] dramatically improved image classification scores. The suprise was considerable because one particular reason (among others): beyond larger experimental scales, there was almost no conceptual difference between the initial LeNet [LeCun and Bengio, 1995] in 1995 and that implementation except: the activation function (from the Sigmoid function to the positive part function) for marginal accuracy improvements, impressive implementation improvements with the original usage of GPU hardware (Graphics Processing Units) for speed (otherwise, training are infeasible for this kind of year-long-type optimization with classic CPUs of traditional computers at that time) and DropOut regularization [Srivastava et al., 2014] to cope with overparametrization of larger neural networks than usual. Indeed, DropOut consists in injecting artificial random noise to learning paramaters during training with the intuition that without that noise, testing accuracy will be more robustly improved for generalization purposes.

The originality of what Gal [2016] proposed consists in continuing DropOut at test time which implicitly provides (dependent) several outputs for each input and then estimating law parameters on those (say mean and variance of a Gaussian law). Averaging correlated outputs (the learned parameters are very close) has unclear theoretical implications for the Monte Carlo estimation but this method has the merit of pointing to a practical need to investigate sound statistical approaches to cope with the *black-box* deserved reputation of neural networks (outputs without "as-is" possible interpretations).

In the past, Nix and Weigend [1994] proposed to predict the mean and covariance of each prediction by minimizing the Kullback-Leibler divergence between the output empirical data distribution and the Gaussian output prediction (i. e. maximizing the likelihood). It is based on the assumption that there is a sufficiently large data set, i. e. , that their is no risk of overfitting and that the neural network finds the correct regression.

In our work, we chose to suggest a lighter machinery (than the ones of Nix and Weigend [1994] and Gal [2016]): instead of changing learning parameters to implicitly produce an ensemble effect on the prediction side, we directly estimate the parameters of a law on the prediction side. We removed the randomness on the weights that the approach of Gal [2016] implies to put it entirely on the

prediction side giving a beneficial *out of local minimum escape* effect to our system while still keeping a valid Kullback-Leibler interpretation like the approach of Nix and Weigend [1994] and even Bishop [1994]. We highly recommend careful readers about uncertainty in deep learning to see the video of Zoubin Ghahramni[1] to get an hindsighful overview of that kind of literature.

As recently describes by Detlefsen et al. [2019], estimating uncertainty is desirable goal in several contexts:

- for time series with Gaussian processes [Seeger, 2004] where means and variances of predictions are estimated at each timestamps;
- the historical work of Bishop [1994] and [Nix and Weigend, 1994] using the seminal idea of predicting the parameters of a law instead of a value that we build upon;
- meanwhile the probabilistic framework of Bayesian techniques was also useful thanks to MacKay [1992] and revisited by Kingma and Welling [2013];
- Monte Carlo Dropout of Gal [2016] with unclear theoretical consequences as predictors are correlated;
- active learning while uncertainty prioritizes choice of informative training examples [Huang et al., 2010].

In this work, we aim to generalize the inspiration of Nix and Weigend [1994] for a larger part of supervised deep learning problems.

4.2 REVISITING DEEP SUPERVISED LEARNING

As described in the state of the art chapter at section 1.2.3, classification and regression are two important fields applications of supervised learning and are thus both applications of this current uncertainty estimation contribution.

4.2.1 *Classification*

Neural Networks are a fundamentally continuous technology that progressively train algorithms by taking into account its own errors on mini-batches of data in order to improve thanks to stochastic gradient descent and back-propagation throughout all the parameters [Goodfellow et al., 2016].

In fact, even plain classification with neural networks is a special case of regression where a degenerated histograms encode categories (a clean category corresponding to a vector with zero entries except for one entry one, hence the name *one-hot encoding*) which has the advantage of naturally manipulating

1 `https://www.youtube.com/watch?v=FD8l2vPU5FY`

probabilities with float real numbers and of letting us access to the continuous regression tools. For potentially K different categories, the supervised classification literature tends to encourage the negative log-likelihood loss corresponding to:

$$\ell(\mathbf{y}, \mathbf{z}) = -\frac{1}{K}\mathbf{y}^\top \log(\mathbf{z}) = -\frac{1}{K}\sum_{k=1}^{K} \mathbf{y}^{(k)} \times \log(\mathbf{z}^{(k)}) \tag{123}$$

where the label $\mathbf{y}$ adopts the one-hot encoding which gives naturally access to its relaxed probabilistic interpretations.

Depending on the scientific culture of the reader, Eq. (123) can also be seen as the cross-entropy loss or even the Kullback-Leibler divergence between the two discrete densities represented in 1-sum positive vectors $\mathbf{z}$ and $\mathbf{y}$. Indeed, the empirical likelihood of given predictions $\mathbf{z} = \mathcal{F}(\mathbf{x})$ measured on (input, output) pairs $(\mathbf{x}, \mathbf{y})$ from training data is

$$\begin{aligned} \text{Likelihood} &= \Big(\prod_{i=1}^{N}\prod_{k=1}^{K} (\mathbf{z}_i^{(k)})^{\mathbf{y}_i^{(k)}}\Big)^{\frac{1}{NK}} \\ &= \Big(\prod_{i=1}^{N}\prod_{k=1}^{K} (\mathcal{F}(\mathbf{x}_i)^{(k)})^{\mathbf{y}_i^{(k)}}\Big)^{\frac{1}{NK}} \end{aligned} \tag{124}$$

and the empirical negative log-likelihood gives:

$$\begin{aligned} -\log(\text{Likelihood}) &= \frac{-1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{y}_i^{(k)} \times \log(\mathbf{z}_i^{(k)}) \\ &= \frac{-1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{y}_i^{(k)} \times \log(\mathcal{F}(\mathbf{x}_i)^{(k)}) \end{aligned} \tag{125}$$

which justifies Eq. (123).

At the same time, if we put the Kullback-Leibler divergence $\mathrm{KL}(\mathbf{y}_i, \mathbf{z}_i)$ definition between two discrete densities $\mathbf{y}_i$ and $\mathbf{z}_i = \mathcal{F}(\mathbf{x}_i)$:

$$\mathrm{KL}(\mathbf{y}_i, \mathbf{z}_i) = \frac{1}{K}\sum_{k=1}^{K} \mathbf{y}_i^{(k)} \times \log\left(\frac{\mathbf{y}_i^{(k)}}{\mathbf{z}_i^{(k)}}\right) = \left(\frac{1}{K}\sum_{k=1}^{K} \mathbf{y}_i^{(k)} \log(\mathbf{y}_i^{(k)})\right) - \log(\text{Likelihood}) \tag{126}$$

but the first term $\frac{1}{K}\sum_{k=1}^{K} \mathbf{y}_i^{(k)} \log(\mathbf{y}_i^{(k)})$ does not depend on the input $\mathbf{x}_i$ nor the prediction $\mathbf{z}_i$ which makes it removable towards a predictor optimization scheme

which also justifies the minimization of Eq. (123) in this classification memory aid:

$$\begin{aligned} & \text{Maximizing the classification likelihood} \\ \iff & \text{Minimizing the classification cross-entropy loss} \\ \iff & \text{Minimizing the mean Kullback-Leibler divergence} \\ & \text{between groundtruth labels and predictions discrete densities} \end{aligned} \tag{127}$$

4.2.2 *Regression*

For regression, $\mathbf{y}$ is continuous (floats) and possibly multivariate, e. g. the price of rent corresponding to each location in a city, supply chain forecasts, weather forecast maps. A stable and well-studied optimization tool is the quadratic regression:

$$\ell(\mathbf{y}, \mathbf{z}) = \|\mathbf{y} - \mathbf{z}\|_2^2 = \sum_{k=1}^{K} \left(\mathbf{y}^{(k)} - \mathbf{z}^{(k)}\right)^2 \tag{128}$$

another loss is the Manhattan one:

$$\ell(\mathbf{y}, \mathbf{z}) = \|\mathbf{y} - \mathbf{z}\|_1 = \sum_{k=1}^{K} |\mathbf{y}^{(k)} - \mathbf{z}^{(k)}| \tag{129}$$

to focus on small errors (as big errors are considerably more punished in quadratic rather then Manhattan loss).

In a way that is similar to establishing a likelihood for cross-entropy classification, it is possible to draw the links between regression and likelihood. As a matter of fact, the likelihood of a Gaussian model on the output side centered on the predictions $\mathcal{F}(\mathbf{x})$ with an unknown but fixed diagonal homothety covariance matrix $\sigma \mathbf{I}_K$ (for the sake of understanding and lack of any sophisticated hypothesis, it turns out we do not loose generality with $\sigma = 1$ in our particular case):

$$\text{Likelihood} = \left(\prod_{i=1}^{N} \frac{\exp\left(-\frac{1}{2}\|\mathbf{y} - \mathcal{F}(\mathbf{x})\|_2^2\right)}{(2\pi)^{\frac{K}{2}}} \right)^{\frac{1}{N}} \tag{130}$$

as the neg-log-likelihood gets:

$$-\log(\text{Likelihood}) \propto \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathcal{F}(\mathbf{x}_i)\|_2^2 \tag{131}$$

and likewise for a the Kullback-Leibler divergence between that Gaussian density q centered at $\mathcal{F}(\mathbf{x})$ (with identity covariance without loss of generality) and the distribution p of ground-truth labels $\mathbf{y} \mid \mathbf{x}$ (when $(\mathbf{x}, \mathbf{y}) \sim$ Nature), we eventually have:

$$\begin{aligned} \mathrm{KL}(p, q) &= \mathbb{E}_{\mathbf{y}\sim q} \log\left(\frac{q(\mathbf{y})}{p(\mathbf{y})}\right) \\ &= \left(\mathbb{E}_{\mathbf{y}\sim q} \log\left(q(\mathbf{y})\right)\right) - \left(\mathbb{E}_{\mathbf{y}\sim q} \log\left(p(\mathbf{y})\right)\right) \qquad (132) \\ & \qquad (133) \end{aligned}$$

but the first term is once again useless for optimizing $\mathcal{F}$, which justifies the least squares approach because:

$$\begin{aligned} -\left(\mathbb{E}_{\mathbf{y}\sim q} \log\left(p(\mathbf{y})\right)\right) &= -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{Nature}}\left[\log\left(\frac{\exp\left(-\frac{1}{2}\|\mathbf{y}-\mathcal{F}(\mathbf{x})\|_2^2\right)}{(2\pi)^{\frac{K}{2}}}\right)\right] \\ &\propto \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{Nature}}\left(\|\mathbf{y}-\mathcal{F}(\mathbf{x})\|_2^2\right) \end{aligned}$$

In the end we re-establish a regression memory aid:

Maximizing the regression likelihood (134)

$\iff$ Minimizing the regression quadratic loss

$\iff$ Minimizing the mean Kullback-Leibler divergence between the groundtruth outputs distribution and the gaussianized predictions distribution

Please note that one could do exactly the same for Laplacians and Manhattan L_1 loss instead of Gaussians and quadratic (eucidlean) L_2 criterion.

4.3 HUM: HYPOTHESIS FOR AN UNCERTAINTY MODEL

The aim of this work is to propose a way to introduce some uncertainty modelization in supervised machine learning presented above focusing on deep neural networks for the sake of recent research attention. We want to build a model that provides a *probabilistic law* prediction instead of a *value* prediction which is following a Bayesian tradition. At test time, we would use for example the main mode of our output law as the output value (i. e. in lieu of a value prediction) but with some additional uncertainty information depending e. g. on how big the support of the distribution is around that mode.

To put this in simple layman's terms and with some abuse of terminology, traditional approaches try to predict a specific value $\mathcal{F}(\mathbf{x})$ hoping to match the label $\mathbf{y}$ in order to ideally reach even at test time some sort of conformity

$$\mathbf{y} \simeq \mathcal{F}(\mathbf{x}) \tag{135}$$

(approximately) as the loss function evaluation $\ell(\mathbf{y}, \mathcal{F}(\mathbf{x}))$ role is to avoid too much discrepancy during training between prediction $\mathcal{F}(\mathbf{x})$ and groundtruth label $\mathbf{y}$.

In contrast, we propose to focus on uncertainty thanks to outputs corresponding to interpretable parameters of a smooth distribution $\mathcal{G}(\mathbf{x})$ hoping that we reach another kind of conformity

$$\mathbf{y} \approx \mathcal{G}(\mathbf{x}) \tag{136}$$

(approximately). Indeed, the graphic slight difference of notation between the symbols "$\approx$" and "$\sim$" inspired us. More mathematically, this work relies on two statements. First, thanks to the definition of Dirac distributions, we easily establish that we always have:

$$\begin{aligned}(\forall(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^D \times \mathbb{R}^K) \;\; \ell\big(\mathbf{y}, \mathcal{F}(\mathbf{x})\big) &= \int_{\mathbb{R}^K} \ell\big(\mathbf{y}, \mathbf{z}\big) \times \delta_{\mathcal{F}(\mathbf{x})}(\mathbf{z}) d\mathbf{z} \\ &\triangleq \mathbb{E}_{\mathbf{z} \sim \delta_{\mathcal{F}(\mathbf{x})}} \left(\ell\big(\mathbf{y}, \mathbf{z}\big)\right)\end{aligned} \tag{137}$$

where $\delta_{\mathbf{a}}$ is the Dirac distribution located at $\mathbf{a}$ that gives that same $\mathbf{a}$ value for when integrated throughout space. Second, as recalled in section 1.5.1 with the Robbins-Monro theorem, we do not need to have access at precisely the exact evaluation of the minimization objective or of its gradient in order to minimize it by stochastic gradient descent: only a biased-free estimator of the gradient is required.

The core idea of this chapter is to ask: *Why not taking a more convenient and more interpretable distribution than the Dirac distribution?* in Eq. (137). This intuition is motivated by the fact that using sum Dirac distributions is considering Nature as just a set of scattered points whereas a smooth distribution gives some consistency to our so-called Nature modelization. Now, if we are emancipated from the previous uncertain-free formulation of section 4.2, then in terms of modelling combining Eq. (137) and Eq. (122) gets:

$$\min_{\mathcal{G}} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \text{Nature}} \left[\mathbb{E}_{\mathbf{z} \sim \mathcal{G}(\mathbf{x})} \left[\ell\big(\mathbf{y}, \mathbf{z}\big) \right] \right] \tag{138}$$

and $\mathcal{G}(\mathbf{x})$ is a prediction law whereas $\mathcal{F}(\mathbf{x})$ previously was a prediction value. A discretized version of this approach has been extensively applied to supply chain forecasts by the Lokad company [Vermorel, 2018] since few years ago. In this chapter, dealing with a wider range of applications (especially suitable with

neural networks as we will see) in a more continuous fashion is our contribution to the original and seminal idea of Vermorel [2018].

In terms of optimization of the parameters $\boldsymbol{\theta}_{\mathcal{G}}$ of function $\mathcal{G}$ predicting the output law $\mathcal{G}(\mathbf{x})$, the good news is that the same Robbins-Monro theorem [Robbins and Monro, 1951] as in section 1.5.1 justifies the usual Hebbian learning rule of stochastic gradient descent. For the sake of clarity we admit that what is true to the standard stochastic gradient descent algorithms is also true for modern stochastic algorithmical routines like Adam [Kingma and Ba, 2014] which benefits from the rest of regular contemporary deep learning tools [Abadi et al., 2015, Paszke et al., 2017].

For the usual uncertainty-free settings, we usually had:

$$\boldsymbol{\theta}_{\mathcal{F}}^{t+1} = \boldsymbol{\theta}_{\mathcal{F}}^{t} - \alpha^t \hat{\mathbf{f}}^t \tag{139}$$

where α^t is the learning rate and $\hat{\mathbf{f}}^t = \frac{1}{B}\sum_{b=1}^{B} \nabla_{\boldsymbol{\theta}_{\mathcal{F}}} \left(\ell\left(\mathbf{y}_{i_b}, \mathcal{F}(\mathbf{x}_{i_b})\right)\right)$ with $i_b \sim \mathcal{U}_{\mathbb{N}}(1, N)$ is a random uniform index parsing the N-cardinality training dataset in mini-batches of size B totalling B gradient evaluations per mini-batch.

In our proposed uncertainty modelling, we get:

$$\boldsymbol{\theta}_{\mathcal{G}}^{t+1} = \boldsymbol{\theta}_{\mathcal{G}}^{t} - \alpha^t \hat{\mathbf{g}}^t \tag{140}$$

with same kind of learning rate α^t as before. The main difference is in the bias-free gradient estimation that needs a little Monte Carlo estimation (as M stands for *Monte Carlo number of iterations*):

$$\hat{\mathbf{g}}^t = \frac{1}{MB}\sum_{m=1}^{M}\sum_{b=1}^{B} \nabla_{\boldsymbol{\theta}_{\mathcal{G}}} \left(\ell\left(\mathbf{y}_{i_b}, \mathbf{z}_{m,i_b}\right)\right) \tag{141}$$

where we still have $i_b \sim \mathcal{U}_{\mathbb{N}}(1, N)$ the same kind of random uniform index as before but we also have M sampled outputs from the same prediction law for each input $\mathbf{x}_{i_b}$ totalling MB gradient evaluations per mini-batch: $\mathbf{z}_{m,i_b} \sim \mathcal{G}(\mathbf{x}_{i_b})$ which is M times more computations than before but there is no guarantee confirming or infirming that a big B or a big M could improve convergence or results as the two sources of stochasticity are related (maybe even a simple $M = B = 1$ scenario could give good resuls).

There is a strong link between our technique and the *Reparametrization Trick* [Kingma and Welling, 2013] for initially variational auto-encoders problematics. Indeed, for the authors, several probabilistic laws, the law parameters can be represented in the gradient calculus from its normalized version (e. g. without mean nor unusual standard deviation for the Gaussian law) of the same simplified probabilistic law accordingly distorted to fit the desired parametrized law. Thanks to three special cases, we will now precisely see how this is all working thanks to pseudo-random generators of our computers.

4.3.1 *Uncertain Logistic Regression for Classification*

In logistic regression classification, we recall from section 4.2.1 that for the general optimization $\min_{\mathcal{F}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\text{Nature}}[\ell(\mathbf{y},\mathcal{F}(\mathbf{x}))]$ is specified for logistic classification by:

$$\ell(\mathbf{y},\mathbf{z}) = -\frac{1}{K}\mathbf{y}^\top \log(\mathbf{z}) \tag{142}$$

and to maintain the simplex-constraint we propose a uniform law (as segments always remain inside a convex set) for applying Eq. (141):

$$\begin{aligned} & \mathbf{z}_{m,i_b} \sim \mathcal{U}_{\mathbb{R}^K}\left(\mathcal{A}(\mathbf{x}_{i_b}),\mathcal{B}(\mathbf{x}_{i_b})\right) \\ \iff \quad & t_{m,i_b} \sim \mathcal{U}_{\mathbb{R}}(0,1) \text{ and } \mathbf{z}_{m,i_b} = t_{m,i_b}\times\mathcal{A}(\mathbf{x}_{i_b}) + (1-t_{m,i_b})\times\mathcal{B}(\mathbf{x}_{i_b}) \end{aligned} \tag{143}$$

where functions $\mathcal{A}$ and $\mathcal{B}$ give class membership probabilities and are implemented with regular neural networks with a last SoftMax layer and it all becomes:

$$\begin{aligned} \hat{\mathbf{g}}^t &= \frac{1}{MB}\sum_{m=1}^{M}\sum_{b=1}^{B}\nabla_{\theta_\mathcal{G}}\left(\ell\left(\mathbf{y}_{i_b}, t_{m,i_b}\times\mathcal{A}(\mathbf{x}_{i_b}) + (1-t_{m,i_b})\times\mathcal{B}(\mathbf{x}_{i_b})\right)\right) \\ &= \frac{1}{MB}\sum_{m=1}^{M}\sum_{b=1}^{B}\nabla_{\theta_\mathcal{G}}\left(-\mathbf{y}_{i_b}^\top\log\left(t_{m,i_b}\times\mathcal{A}(\mathbf{x}_{i_b}) + (1-t_{m,i_b})\times\mathcal{B}(\mathbf{x}_{i_b})\right)\right) \end{aligned} \tag{144}$$

to be compared to Eq. (141).

In these settings, the prediction uncertainty can be measured in $0 \leq \frac{1}{2}\|\mathcal{A}(\mathbf{x}_{i_b}) - \mathcal{B}(\mathbf{x}_{i_b})\|_1 \leq 1$ which justifies our probabilistic model. In terms of Kullback-Leibler divergence, minimizing the mean divergence between the fixed discrete groundtruth density $\mathbf{y}$ and the random discrete prediction density $\mathbf{z} \sim \mathcal{G}(\mathbf{x})$ this way, is useful even at a category level by superposing both $\mathcal{A}(\mathbf{x})$ and $\mathcal{B}(\mathbf{x})$ histograms (i. e. positive 1-sum vectors).

Out of curiosity, we can explore what is happening for more than 2 histograms for the sake of algorithms.

$$\begin{aligned} & \mathbf{z}_{m,i_b} \sim \mathcal{U}_{\mathbb{R}^K}\left(\mathcal{A}_1(\mathbf{x}_{i_b}),\ldots,\mathcal{A}_c(\mathbf{x}_{i_b}),\ldots,\mathcal{A}_C(\mathbf{x}_{i_b})\right) \\ \iff \quad & \mathbf{t}_{m,i_b} \sim \mathbf{\Delta}_C \text{ and } \mathbf{z}_{m,i_b} = \sum_{c=1}^{C}\mathbf{t}_{m,i_b}^{(c)}\times\mathcal{A}_c(\mathbf{x}_{i_b}) \end{aligned} \tag{145}$$

where $\mathbf{\Delta}_C$ is the uniform distribution over the C-order simplex (i. e. $\sum_{c=1}^{C}\mathbf{t}_{m,i_b}^{(c)} = 1$). To uniformly sample inside that C-order simplex, first, we first sample uniformly $C-1$ values between 0 and 1 $u_{m,i_b,c} \sim \mathcal{U}_{\mathbb{R}}(0,1)$ (and we rearranged the indices corresponding to c such that they are re-ordered in increasing order), second we recursively build:

$$\begin{aligned} \mathbf{t}_{m,i_b}^{(1)} &= u_{m,i_b,1} \\ \mathbf{t}_{m,i_b}^{(c)} &= u_{m,i_b,c+1} - u_{m,i_b,c} \text{ for } c \in [\![2, C-1]\!] \\ \mathbf{t}_{m,i_b}^{(C)} &= 1 - u_{m,i_b,C-1} \end{aligned} \tag{146}$$

that cancels out in a sum of 1 and finally we get:

$$\hat{\mathbf{g}}^t = \frac{1}{MB}\sum_{m=1}^{M}\sum_{b=1}^{B}\nabla_{\boldsymbol{\theta}_{\mathcal{G}}}\left(\ell\left(\mathbf{y}_{i_b}, \sum_{c=1}^{C}\mathbf{t}_{m,i_b}^{(c)} \times \mathcal{A}_c(\mathbf{x}_{i_b})\right)\right) \tag{147}$$

which implies an output law that fires in a convex set drawn by the polygon made of $(\mathcal{A}_c(\mathbf{x}))_{c=1,\dots,C}$ for each input $\mathbf{x}$. Although this formulation is theoretically pleasing, it did not provide, in our experience any additional interpretation ease nor accuracy improvements, yet.

4.3.2 *Uncertain Least Square for Regression*

Likewise, in least square regression:

$$\ell(\mathbf{y}, \mathbf{z}) = \|\mathbf{y} - \mathbf{z}\|_2^2 \tag{148}$$

and we choose the Gaussian distribution to obtain:

$$\begin{aligned} & \mathbf{z}_{m,i_b} \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}_{i_b}), \boldsymbol{C}(\mathbf{x}_{i_b}) \times \boldsymbol{C}(\mathbf{x}_{i_b})^\top\right) \\ \iff \quad & \boldsymbol{\epsilon}_{m,i_b} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K) \text{ and } \mathbf{z}_{m,i_b} = \boldsymbol{\mu}(\mathbf{x}_{i_b}) + \boldsymbol{C}(\mathbf{x}_{i_b}) \times \boldsymbol{\epsilon}_{m,i_b} \end{aligned} \tag{149}$$

where function $\boldsymbol{\mu}$ is a vanilla neural network and $\boldsymbol{C}$ is a lower-triangular matrix (with possible bounded diagonal values thanks to affine Sigmoids see Table 1 at page 14). For the Hebbian rule we update with:

$$\hat{\mathbf{g}}^t = \frac{1}{BM}\sum_{b=1}^{B}\sum_{m=1}^{M}\nabla_{\boldsymbol{\theta}_{\mathcal{G}}}\left(\|\mathbf{y}_i - \left(\boldsymbol{\mu}(\mathbf{x}_{i_b}) + \boldsymbol{C}(\mathbf{x}_{i_b}) \times \boldsymbol{\epsilon}_{m,i_b}\right)\|_2^2\right) \tag{150}$$

In these settings, the uncertainty information is contained in the $\boldsymbol{C}(\mathbf{x}_{i_b}) \times \boldsymbol{C}(\mathbf{x}_{i_b})^\top$ covariance matrix as we have fitted a Gaussian to our prediction outputs. Obviously, for the sake of completeness, we have modelled a Gaussian distribution with a full covariance matrix but a diagonal covariance matrix or a simple proportional to the identity covariance matrix is most of the times enough in real-world applications.

In terms of Kullback-Leibler divergence, we still have the same probabilistic interpretations as we minimize the mean over input $\mathbf{x}$ of the divergence between the deterministic groundtruth labels $\mathbf{y}$ Dirac distribution and the smooth Gaussian distribution $\mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{C}(\mathbf{x}) \times \boldsymbol{C}(\mathbf{x})^\top\right)$ which means we did manage not to sacrifice interpretation for uncertainty. The mean $\boldsymbol{\mu}(\mathbf{x})$ plays the role of the good old-fashioned value prediction that is tainted by incertainty $\boldsymbol{C}(\mathbf{x}) \times \boldsymbol{C}(\mathbf{x})^\top$.

4.3.3 *Uncertain Mixtures for Regression and Classification*

Sometimes, beyond mono-modal distribution for regression (e. g. often Gaussian) and uniform law for classification, one can imagine a non-continuous spectrum of

outcomes like in supply chain forecasts [Vermorel, 2018] to handle several discrete scenarios at the same time. This way, one can benefit from the computational power of machines to process newly predictable outcomes. This is the reason why for each input $\mathbf{x}$ we tackle here one finite mixture $q(\mathbf{x})$ of C distributions with proportions $\boldsymbol{\pi}(\mathbf{x})$ and components $(g_k(\mathbf{x}))_{k=1}^K$ (that are themselves distributions) which can written as:

$$q(\mathbf{x}) = \sum_{c=1}^{C} \boldsymbol{\pi}_c(\mathbf{x}) \times g_c(\mathbf{x}) \tag{151}$$

Just by the continuous integral definition of the expectation, we obtain:

$$\mathbb{E}_{\mathbf{z}\sim q}\left(\ell(\mathbf{y},\mathbf{z})\right) = \sum_{c=1}^{C} \boldsymbol{\pi}_c(\mathbf{x}) \times \mathbb{E}_{\mathbf{z}_c\sim g_c(\mathbf{x})}\left[\ell(\mathbf{y},\mathbf{z}_c)\right] \tag{152}$$

which facilitates our mixture use because we can directly use the formulas of the previous sections 4.3.1 and 4.3.2 depending on the applications we have:

$$\hat{\mathbf{g}}^t = \frac{1}{MB}\sum_{m=1}^{M}\sum_{b=1}^{B}\sum_{c=1}^{C} \nabla_{\boldsymbol{\theta}_\mathcal{G}}\left(\boldsymbol{\pi}_c(\mathbf{x}) \times \ell\big(\mathbf{y}_{i_b}, \mathbf{z}_{m,i_b,c}\big)\right) \quad \text{with} \quad \mathbf{z}_{m,i_b,c} \sim \mathcal{G}_c(\mathbf{x}_{i_b}) \tag{153}$$

where the previous technique of reparametrization as previously can re-applied to generate samples from $\mathcal{G}_c(\mathbf{x}_{i_b})$.

In this mixture scenario, the parameters set $\boldsymbol{\theta}_\mathcal{G}$ contains the parameters θ_π of the neural network $\boldsymbol{\pi}$ that gives the C proportions and each of the parameters $\boldsymbol{\theta}_{\mathcal{G}_c}$s that parametrize the laws $\mathcal{G}_c(\mathbf{x})$s. Two special cases of mixtures attract our attention:

GAUSSIAN MIXTURE FOR REGRESSION Interestingly, this corresponds to the Mixture Density Networks designed by Bishop [1994] in the 1990s;

DIRAC MIXTURE FOR CLASSIFICATION For a high number of possible categories ($K \simeq 10^3$ in ImageNet [Fei-Fei, 2010]), allowing a mixture of a much reduced number $C \sim 10^1$ of components is useful to handle both uncertainty and close categories in a more interpretable way than in the usual supervised deep learning apparatus.

e

4.3.4 *Links with Other techniques*

Interestingly, our straightforward yet efficient approach has some connections with other techniques. Let us take into account that the discrepancy function ℓ in Eq. (138), if we recall that in classification and regression cases, we chose $\ell(\mathbf{y},\mathbf{z}) = -\mathbf{y}^\top \log(\mathbf{z})$ and $\ell(\mathbf{y},\mathbf{z}) = \|\mathbf{y}-\mathbf{z}\|_2^2$ respectively. In both cases, for any

label $\mathbf{y}$, we know that the function $\mathbf{z} \mapsto \ell(\mathbf{y}, \mathbf{z})$ is convex and thus, thanks to Jensen inequality for any distribution q of predicted output (e. g. $q = \mathcal{G}(\mathbf{x})$):

$$\mathbb{E}_{\mathbf{z}\sim q}\left(\ell(\mathbf{y}, \mathbf{z})\right) \geq \ell\left(\mathbf{y}, \mathbb{E}_{\mathbf{z}\sim q}(\mathbf{z})\right) \tag{154}$$

which implies that:

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim \text{Nature}}\left[\mathbb{E}_{\mathbf{z}\sim\mathcal{G}(\mathbf{x})}\left[\ell\big(\mathbf{y}, \mathbf{z}\big)\right]\right] \geq \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim \text{Nature}}\left[\ell\Big(\mathbf{y}, \mathbb{E}_{\mathbf{z}\sim\mathcal{G}(\mathbf{x})}\big(\mathbf{z}\big)\Big)\right] \tag{155}$$

which means that for a non-Dirac global minimum $\mathcal{G}^*$ of Eq. (138) corresponding to the lower bounded right hand side of Eq. (155), we know that the global minimum is also reached at

$$\mathcal{G}_d : \mathbf{z} \mapsto \mathbb{E}_{\mathbf{z}\sim\mathcal{G}^*(\mathbf{x})}(\mathbf{z}) \tag{156}$$

One possible interpretation of that could be *The deterministic optimization and our optimization share some global minima if they exist*. What is interesting to mention is that in practice, first, space zones that is very close to the training data manifold have roughly the same result for with or without uncertainty model and second, *in-between* zoom have better results and interpretation with out uncertainty model. Another way to look at it is to consider our technique like a DropOut [Srivastava et al., 2014] technique consisting in adding some randomness to avoid being stuck in local minima. Our approach also looks like the one of **?** but we do linear combination on the output instead of doing it at the input like them. Our method needs further investigation beyond showing good results because of intringuing self-regularized optimization phenomena thanks to randomness. Although, it is rigorous to separate model from optimization, in our case, it is possible that the additional randomness of our optimization technique actual makes our model more robust and it is unclear to attribute improvements between model and optimization as we will see in the next section.

4.4 ALGORITHM AND PRACTICAL IMPLEMENTATION DETAILS

Based on the updated Hebbian rules, we previously described in normal and out uncertain contexts, the stochastic gradient descent algorithm optimizes the parameters of our neural networks. Using the (pseudo)-random number generators of our computers, we were able to generalize the Robbins-Monro theorem allowing us to calculate our bias-free estimators of gradients at each stochastic gradient iteration.

4.4.1 *Safe Computations*

Some numerical overflow instabilities have been studied and avoided thanks to the so-called *logsumexp* trick. This technique must be revisited in our case.

Let's first briefly describe what it usually is and then how we can adapt it for our implementations.

Even though the *logsumexp* trick can be used in many scenarios, we choose to describe it in the special example of the previouly presented logistic regression of classification for the sake of pragmatism. In neural networks usual implementations, the cross-entropy loss is applied after the computation of a SoftMax layer on what we call logits $\mathbf{c} = [\mathbf{c}_1, \ldots, \mathbf{c}_k, \ldots, \mathbf{c}_K]$ (see the previously described in Table 1 at page 14). At some point, there is a need to compute the gradient of the quantity $\mathbf{v} \in \mathbb{R}^K$:

$$\mathbf{v} = \log(\text{SoftMax}(\mathbf{c})) \tag{157}$$

with respect to the neural network parameters involved in the calculus of $\mathbf{c}$. If we take a closer look to $\mathbf{v}$, we expand it to:

$$\mathbf{v}_k = \log\left(\frac{\exp(\mathbf{c}_k)}{\sum_{k'=1}^{K}\exp(\mathbf{c}_{k'})}\right) = \log(\exp(\mathbf{c}_k)) - \log\left(\sum_{k'=1}^{K}\exp(\mathbf{c}_{k'})\right) \tag{158}$$

The direct computation of $\exp(\mathbf{c}_k)$ where $\mathbf{c}_k \geq 30$ is already bigger than 10^{13} which makes it all unfeasible for a modern computer even though we know that $\log(\exp(\mathbf{c}_k)) \simeq \log(10^{13}) \simeq 30$ is a reasonable number. For any value $M \in \mathbb{R}$, we can use the fact that:

$$\begin{aligned}\mathbf{v}_k &= M + \log(\exp(\mathbf{c}_k - M)) - M - \log\left(\sum_{k'=1}^{K}\exp(\mathbf{c}_{k'} - M)\right) \\ &= \log(\exp(\mathbf{c}_k - M)) - \log\left(\sum_{k'=1}^{K}\exp(\mathbf{c}_{k'} - M)\right)\end{aligned} \tag{159}$$

and for the specific choice $M = \max_k \mathbf{c}_k$, our numerical problems disappear because all exponential arguments $\mathbf{c}_k - M \leq 0$ are more suitable for the computation of $\exp(\mathbf{c}_k - M) \leq 1$ without approximation.

Nowadays deep learning tools such as PyTorch [Paszke et al., 2017], Tensorflow [Abadi et al., 2015] and MXNet [Chen et al., 2015] applies that *logsumexp* trick implicitly *behind the scene* but this can reused in this present work. We can remark that the idea consisted in avoiding big exponentials by guaranteeing negativity of their arguments thanks to the maximum entry that is substracted for safer computations.

Likewise, in our uncertainty estimation context, we have to compute at some point the quantity $\mathbf{w} \in \mathbb{R}^K$ for $t \in [0, 1]$:

$$\mathbf{w} = \log\left(t \times \text{SoftMax}(\mathbf{a}) + (1 - t) \times \text{SoftMax}(\mathbf{b})\right) \tag{160}$$

where $\mathbf{a} \in \mathbb{R}^K$ and $\mathbf{b} \in \mathbb{R}^K$ come from neural networks. With that negative argument exponentials technique in mind, we understand that the SoftMax function is invariant to uniform shifting:

$$(\forall M \in \mathbb{R}) \;\; \text{SoftMax}(\mathbf{c}) = \text{SoftMax}(\mathbf{c} - M) \tag{161}$$

thus we propose for any $P \in \mathbb{R}$ and $Q \in \mathbb{R}$:

$$\begin{aligned}\mathbf{w}_k &= \log\left(t \times \frac{\exp(\mathbf{a}_k)}{\sum_{k'=1}^{K}\exp(\mathbf{a}_{k'})} + (1-t) \times \frac{\exp(\mathbf{b}_k)}{\sum_{k'=1}^{K}\exp(\mathbf{b}_{k'})}\right) \\ &= \log\left(t \times \frac{\exp(\mathbf{a}_k - P)}{\sum_{k'=1}^{K}\exp(\mathbf{a}_{k'} - P)} + (1-t) \times \frac{\exp(\mathbf{b}_k - Q)}{\sum_{k'=1}^{K}\exp(\mathbf{b}_{k'} - Q)}\right)\end{aligned}$$

and to guarantee the same negative exponentials properties as before, we choose: $P = \max_k \mathbf{a}_k$ and $Q = \max_k \mathbf{b}_k$ which provides much more numerical stability when it comes to computing $\mathbf{w}$ and its gradient with respect to the parameters *a fortiori*.

In the special case where the number of classes $K = 2$, practioners use a Sigmoid function applied on a logit value c and the 2 estimated probabilities get $[\text{Sigmoid}(c), 1 - \text{Sigmoid}(c)]$ but the SoftMax function is a generalization of the Sigmoid for that $K = 2$ binary case thanks to that relationship:

$$\begin{aligned}\text{SoftMax}\left([0, -c]\right) &= [\text{Sigmoid}(c), \text{Sigmoid}(-c)] \\ &= [\text{Sigmoid}(c), 1 - \text{Sigmoid}(c)]\end{aligned} \tag{162}$$

which gives us Eq. (162) in that $K = 2$ binary special case for no supplementary effort.

4.4.2 *Re-Using Uncertainty-Free Models*

In 2018, Apple and Amazon reached $\$10^{12}$ of market capitalization and in 2019 Microsoft followed in 2020 by Alphabet that owns Google also did reach that financial milestone while Facebook is at roughly $\$0.6 \times 10^{12}$. So we humbly imagine a way to use all these powerful open-source means to better train our uncertainty models just to *stand on the shoulders of giants*.

In order to re-use all that wealth of trained models, we use the idea that an uncertainty-free model is a model that has no uncertainty: for example, a Gaussian density with no covariance becomes a Dirac distribution, a uniform density with equal bounds also becomes a Dirac distribution. We can briefly explain this strategy in two supervised instances:

REGRESSION In our Gaussian regression case, $\boldsymbol{\mu}$ can be reasonably well-initialized by its uncertainty-free already trained counterpart with a zero-initialized Cholesky function $\mathbf{C}$ for the covariance matrix.

$$\mathbf{C} = \mathbf{0} \iff \text{no uncertainty} \tag{163}$$

CLASSIFICATION In our uniform classification case, both $\mathcal{A}$ and $\mathcal{B}$ can be initialized by the same weights (while the random generator of t will give

different gradients starting from the very first mini-batch gradient computation).

$$\mathcal{A} = \mathcal{B} \iff \text{no uncertainty} \tag{164}$$

In our experiments, approximately 10% more of training epochs is enough to get relevant additional uncertainty information. We must insist on the fact that we did not want to improve accuracy of already impressively sophisticated algorithms (at the very least with an engineering perspective and for means of distributed computational power) but only we only wanted to get some new uncertainty information.

4.4.3 *Algorithms*

There exists many ways to implement the neural networks from our HUM (Hypothesis of Uncertainty Model). In this section, we provide a binary classification pseudo-code and another pseudo-code for regression.

Neural Networks are often built in layers which correspond to a composition of elementary functions (even in deep residual learning [He et al., 2016], we can still separate blocks of layers). This is good news for cheap learning of heavy neural networks with uncertainty because they can be initialized with worldwide companies means but without uncertainty. For example, in classification, a neural network corresponding to a function $\mathcal{F}$ can be thus separated it into $\mathcal{F} = \mathcal{G} \circ \mathcal{H}$ and we create 2 functions, namely $\mathcal{A} = \mathcal{G}_{\mathcal{A}} \circ \mathcal{H}$ and $\mathcal{B} = \mathcal{G}_{\mathcal{B}} \circ \mathcal{H}$ where function $\mathcal{H}$ is shared whereas both $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$ are equally initialized by the previous training output $\mathcal{G}$.

Algorithm 5 Classification with Uncertainty

1: **Input:** Data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1,\dots,N}$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0,1\}^K \cap \Delta_K$, a mini-batch size B and a Monte Carlo parameter $M \in \mathbb{N}^*$

2: **Initialization:** $\theta = \{\theta_{\mathcal{H}}, \theta_{\mathcal{G}_{\mathcal{A}}}, \theta_{\mathcal{G}_{\mathcal{B}}}\}$ initialized from a regular uncertainty-free classification optimization scheme for respectively $\mathcal{H}, \mathcal{A}$ and $\mathcal{B}$;

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

4: Free all gradients accumulators

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

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

6: Sample BM values from a pseudo-random generator

$$t_{b,m} \sim \mathcal{U}_{\mathbb{R}}(0,1) \quad \text{for } b = 1, \dots, B \text{ and } m = 1, \dots, M$$

7: Compute

$$\mathbf{h}_b \leftarrow \mathcal{H}(\mathbf{x}_{i_b}), \mathbf{a}_b \leftarrow \mathcal{G}_{\mathcal{A}}(\mathbf{h}_b) \text{ and } \mathbf{b}_b \leftarrow \mathcal{G}_{\mathcal{B}}(\mathbf{h}_b) \quad \text{for } b = 1, \dots, B$$

8: Apply the *logsumexp trick* $\mathbf{a}_b \leftarrow \mathbf{a}_b - \max_k \mathbf{a}_b^{(k)}$ and $\mathbf{b}_b \leftarrow \mathbf{b}_b - \max_k \mathbf{b}_b^{(k)}$

9: And finally

$$\mathcal{L} \leftarrow \frac{-1}{BM} \sum_{m=1}^{M} \sum_{b=1}^{B} \mathbf{y}_{i_b}^{\top} \log \left(t_{b,m} \times \text{SoftMax}(\mathbf{a}_b) + (1 - t_{b,m}) \times \text{SoftMax}(\mathbf{b}_b) \right)$$

10: Perform a gradient descent step of $\mathcal{L}$ to update θ

11: **end while**

Algorithm 6 Regression with Uncertainty

1: **Input:** Data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1,\dots,N}$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \mathbb{R}^K$, a mini-batch size B and a Monte Carlo parameter $M \in \mathbb{N}^*$
2: **Initialization:** $\theta_{\boldsymbol{\mu}}$ for a mean function $\boldsymbol{\mu}$ initialized from a regular uncertainty-free regression optimization scheme; $\theta_{\mathbf{C}}$ for a lower triangular matrix function $\mathbf{C}$ initialized by a diagonal of training standard deviations; $\theta = \{\theta_{\boldsymbol{\mu}}, \theta_{\mathbf{C}}\}$
3: **while** θ has not converged **do**
4: Free all gradients accumulators
5: Sample a mini-batch of size B from the dataset

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

6: Sample BM values from a pseudo-random generator

$$\boldsymbol{\epsilon}_{b,m} \sim \mathcal{N}(\mathbf{0}_K, \mathbf{I}_K)$$

7: Compute

$$\mathbf{m}_b \leftarrow \boldsymbol{\mu}(\mathbf{x}_{i_b}) \text{ and } \mathbf{S}_b \leftarrow \mathbf{C}(\mathbf{x}_{i_b})$$

8: The loss gets:

$$\mathcal{L} \leftarrow \frac{1}{BM} \sum_{m=1}^{M} \sum_{b=1}^{B} \|\mathbf{m}_b + \mathbf{S}_b \times \boldsymbol{\epsilon}_{b,m} - \mathbf{y}_{i_b}\|_2^2$$

9: Perform a gradient descent step of $\mathcal{L}$ to update θ
10: **end while**

4.5 EXPERIMENTS

4.5.1 *Synthetic data*

In order to go deeper with our approach, we study a toy dataset called "Two Moons" with 1000 points for each of the 2 groups (blue and red 2D dots) in 2 dimensions as presented in Fig. 21 in two scenarios: an easily separable one and another much noisier one. In this supervised classification exercize, the easy

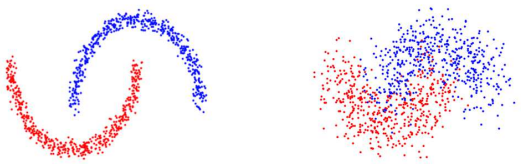

Figure 21: "Two Moons" Dataset in two Scenarios: a clear and easy case (left) and a noisy and difficult case (right)

case wihtout noise in not very interesting from an uncertainty perspective which makes our approach unnecessary on the left. On the contrary, the difficult case is more relevant because some space regions imply legitimate doubt levels on the right.

We trained a small multi-layer perceptron on this dataset to investigate what was empirically at stake: for each input $[x_1, x_2]^\top$ we predict 2 probability vectors $p = [a, b]^\top$ and $q = [c, d]^\top$ as bounds of a uniform prediction law $\hat{\mathbf{y}} \sim \mathcal{U}_{\mathbb{R}}(p, q)$ following section 4.3.1 (page 99). Thus, it is fair to consider $u = \frac{\|p-q\|}{2} \in [0, 1]$ as an evaluation of uncertainty. Our goal is not to provide any classification accuracy improvements but only some more additional information about uncertainty. So we empirically checked on this synthetic problem if our technique was able to preserve good classification results which is the case in Fig. 22 with yellow for the *blue class*, a purple for the *red class* and small region of green for the blurry boundary with a thickness that is understandable when considering the mixed classes. Now we can see if it is also providing a useful uncertainty information in Fig. 23.

Here we see an interesting phenomenon: the prediction is highly confident almost everywhere in space (purple low level of uncertainty) except in the boundary which can be split in two: (i) low uncertainty boundary in purple at the center of the figure and high uncertainty far from where the points are (whereas the boundary is green in the previous Fig. 22 wihtout distinction of having a 0.5 probability of being blue.)

These encouraging synthetic results exhibits two ways of *not knowing*: with uncertainty in yellow in Fig. 23 and with certainty in purple in Fig. 23. The Dirac phenomenon we suspected in section 4.3.4 is occuring: close to the training

manifold, the uniform distribution we get is a Dirac but going further from the training manifold grows some thickness away from the Dirac distribution.

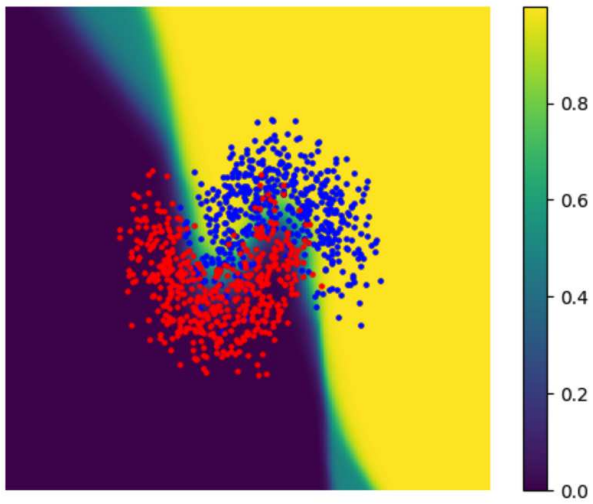

Figure 22: "Two Moons" Map of Prediction Probability of *being blue* $\frac{a+c}{2}$

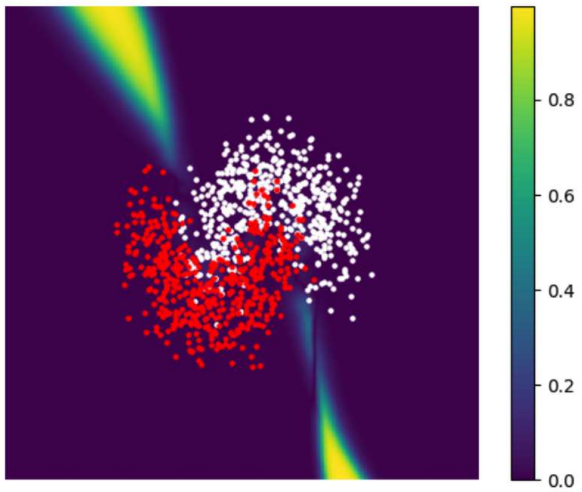

Figure 23: "Two Moons" Map of Uncertainty Probability $\frac{\|p-q\|_1}{2}$ (the higher, the less certain). Blue became white for readibility reasons.

4.5.2 *Dogs and Wolves*

To illustrate our approach, we took a image classification task that is difficult even for us as human beings: distinguishing domesticated canide animals (say mainly dogs) and wild canide animals (say wolves, foxes etc.) that we took from the ImageNet dataset [Fei-Fei, 2010] (and its tree-structured label ontology). We formed a dataset[2] of a balanced 800 images per class training set and 200 images per class validation set on which we measure accuracy. We also formed a test set from the ImageNet test dataset (annotated by the ResNet-152 available via PyTorch) with 100 cats, 100 dogs and 100 wolves to evaluate test error and extrapolation uncertainty (i. e. out of distribution) as cats are not wolves nor dogs [3].

(a) Dog (b) Wolves

Figure 24: Two examples of images from our "Dogs and Wolves" dataset taken from Imagenet

As Greenspan et al. [2016] explains, using a convolution neural networks (CNN) trained from a *natural* images classification problem with plenty of labeled data (e. g. ImageNet [Fei-Fei, 2010]) is useful for a *medical* images classification problem that suffered from a notorious lack of large scale labeled datasets until recently. Indeed, *chopping the head off* a CNN is relevant as a feature extractor even for a very different problem which is understandable because early computer vision cues such as the relationship between contours and gradients in the first layers seems to be independent from the image domain and thus shared among all computer vision tasks. Meanwhile, some empirical studies tend to show that the first layers of a convolutional network often correspond to Gabor functions for general-purpose computer vision applications and the deeper layers can be interpreted in more sophisticated ways [Zeiler and Fergus, 2014].

Inspired by these successful ideas and intuitions, first we build a convolutional neural network like in Fig. 25 close to AlexNet [Krizhevsky et al., 2012] but we double the number of ending layers and second we make all the first layers shared for two separate ending dense neural networks. Indeed for our "Dogs

2 `http://harchaoui.org/warith/dogs_wolves.zip`
3 `http://harchaoui.org/warith/imagenet_test_resnet152_pytorch.zip`

and Wolves" images classification problem, we need two outputs with the belief that some features are shared among these virtually two neural networks. We do not entirely double the number of parameters with two separate convolutional neural networks because we want to prevent our learning procedure from some bad overfitting effects of over-parametrization and gain some more speed.

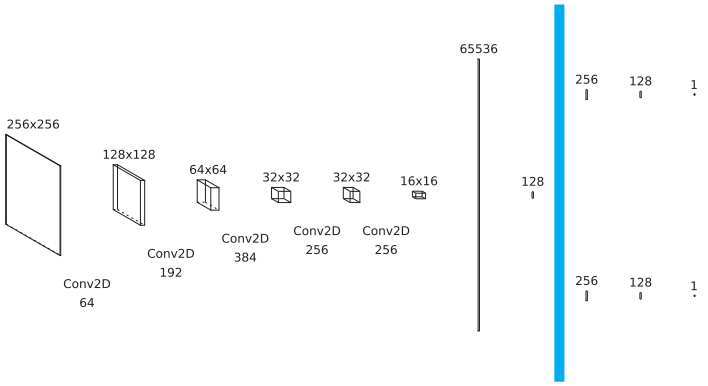

Figure 25: Our Convolutional Neural Network in two parts separated by the tick blue line: (i) some shared convolutional and fully-connected layers on the left and (ii) a few fully-connected layers on the right to finally output the two bounds of our uniform prediction law.

Our "Dogs and Wolves" images classification problem looks like our "Two Moons" noisy case with much higher dimensionality and where even human beings might disagree sometimes. In fact, the very notion of being domesticated is human-related for example. Numerically, we observe a slight improvement of less than 1% of validation accuracy from a 76% in normal uncertainty-free training and 77% with our HUM technique. In fact, we initialized as previously stated in section 4.4.2 our HUM convolutional neural network with a classic uncertainty-free regular approach. The goal here was not to improve the validation score but only to check if we are not loosing some which is valid as Fig. 26 is showing little overlaid differences. The real benefit of our HUM approach is the uncertainty estimation. Thus, it seems that our HUM approach on a difficult computer vision task is efficient.

In Table. 3 we provide some of the most uncertain examples according to our HUM CNN: we took the top-8 uncertain (considering $\frac{\|p-q\|_1}{2}$ as an evaluation of uncertainty) images of our testing set (that includes cats that were not seen during training). Qualitatively, we see that just taking the entropy of a regular CNN as an uncertainty criterion does not perform as well in Table. 4.

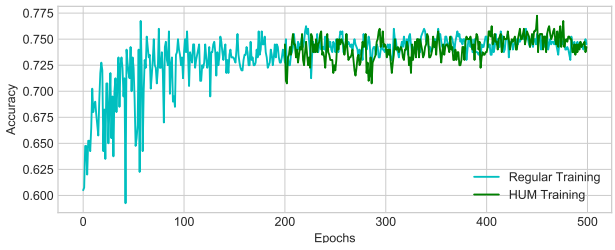

Figure 26: Validation score w.r.t. the training epochs comparing our HUM CNN with its initialized regular CNN counterpart (the higher, the better, 1.0 means 100% of good classification on the validation set)

Table 3: Top-8 of most uncertain testing images of our "Dogs and Wolves" dataset according to our HUM CNN

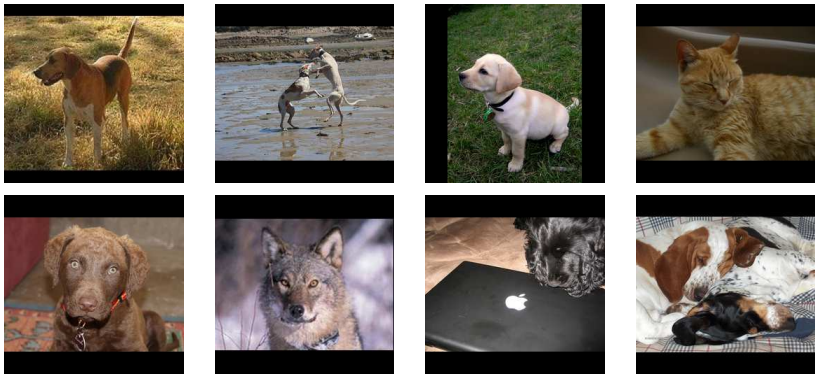

Table 4: Top-8 of most entropic testing images of our "Dogs and Wolves" dataset according to a regular CNN

4.6 FUTURE WORK AND CONCLUSION

In this work, we revisited an old problematic in Machine Learning and more generally in Statistics about dealing with estimation error that is inherent to real-world studied phenomena. An interesting avenue of research could be to refine the "far from dataset" kind of uncertainty: revisiting the Wasserstein distance between the query Dirac and the training dataset would pragmatically give a geometric aspect to uncertainty.

This field of research has a wide of applications: active learning would choose examples based on predicted uncertainty which is intuitively (at least) reasonable: *to discover what we are unsure of first*. There is also another promising application: reinforcement learning without reward to learn how to navigate in a new environement without task which would give some complementary understanding to the notion of curiosity in the scientific effort of Burda et al. [2018] and we could consider that diminishing an uncertainty estimation would be the cornerstone of a reinforcement learning policy. Indeed, even in regular reinforcement learning, training inescapably boils down to distentangling an exploration / exploitation dilemma [Sutton and Barto, 2018]: on the one hand, exploration is testing unusual (or risky) patterns of actions with the hope to expect possibly higher rewards and on the other hand, exploitation is benefiting from known (or conservative) patterns of actions in order to better guarantee a certain level of rewards. This contribution about uncertainty estimation could help breaking the dilemma. The rationale behind could consist in saying that one should explore where uncertainty is high and exploit where uncertainty is low and value estimation is high.

It seems interesting to also investigate the blurry prediction provided by uncertainty models to help for more robustness to adversarial examples (example designed to fool supervisedly trained algorithms, see the survey of Yuan et al. [2019] for details). At least, this kind of systems would be fooled but the predictions would have larger standard deviation for example for this deceiving points.

CONCLUSION

In this dissertation, learning representations using neural networks and optimal transport provided promising statistical and algorithmic tools for clustering, for an attempt towards unsupervised feature importance and some practical supervised prediction with uncertainty. Three axes have been present throughout these contributions in large scale settings for thrice: clustering for simplifying the cardinality axis, unsupervised feature importance for simplifying the dimensionality axis and finally uncertainty estimation for the output space axis.

For further research work, some interactions between our three contributions can be drawn: (i) optimizing over a set of Wasserstein metrics as we suggest in unsupervised feature importance extraction can be investigated for clustering in order to tackle the metric-invariance, (ii) unsupervised feature importance extraction and clustering can be done in a simultaneous fashion combining both feature selection and clustering which is desirable for high-dimensional data and (iii) out of distribution uncertainty estimation can be revisited with Wasserstein distances.

Away from traditional supervised algorithms, we have shown that neural networks are appealing tools especially when empowered by optimal transport and data analysis purposes. A lot of programming has been done to tackle these issues and interacting between real-world pragmatism and theoretical reasoning back and forth has been key to conduct this work.

APPENDIX: GENERATIVE ADVERSARIAL NETWORKS PRETRAINING

INTRODUCTION

For dramatic computational speed improvements reasons, some pretraining techniques from 1990s are less popular nowadays in neural networks training. Recently, the rise of Generative Adversarial Networks (GANs) initiated by Goodfellow et al. [2014] is revisiting a new kind of optimization: not an usual minimization problem but a min-max problem hence the adjective *adversarial* which is thus more difficult to monitor. In practice, for same domain (or medium) data, it is possible to use the same structure and hyper-parameters as other authors e. g. the DCGAN structure by Radford et al. [2015] for images. Unfortunately, for custom data such as genomics data, there is a need for practical rule of thumbs to begin working for new kinds of data and neural networks and particularly wirh GANs.

Historically, neural networks experts used to train an unsupervised auto-encoder, in order to keep the encoder as an initialization of the neural network now trained in a supervised fashion. This pre-training technique seems to save a lot of time during an era of expensive and long computations. The aim of this appendix is to follow this kind of reasoning to initialize GANs that are notoriously painful to train and monitor without further data knowledge.

SIMILAR OBJECTIVES

Let's study the Wasserstein GAN [Arjovsky et al., 2017] objective function with a generator $\mathcal{G}$, a real data distribution p living in $\mathbb{R}^D$, and noise generator distribution n living in $\mathbb{R}^d$

$$\min_{\mathcal{G}} W(p, q_{\mathcal{G}}) \tag{165}$$

where the Wasserstein distance $W(p, q_{\mathcal{G}})$ is defined between the distributions of real data p and generator data $q_{\mathcal{G}}$ (defined by $\mathbf{y} \sim q_{\mathcal{G}} \iff \mathbf{z} \sim n$, and $\mathbf{y} = \mathcal{G}(\mathbf{z})$)

$$W(p, q_{\mathcal{G}}) = \min_{\gamma \in \Gamma(p, q_{\mathcal{G}})} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \left[\|\mathbf{y} - \mathbf{x}\|_2\right] = \min_{\gamma' \in \Gamma(p, n)} \mathbb{E}_{(\mathbf{x},\mathbf{z})\sim\gamma'} \left[\|\mathcal{G}(\mathbf{z}) - \mathbf{x}\|_2\right] \tag{166}$$

where $\Gamma(p, q)$ stands for the set of couplings γ based on the two marginal distributions p and q. In the GAN literature, this is reformulated by thanks to the Kantorovich-Rubinstein duality but in this appendix, we will still use the Wasserstein formulation.

Indeed, if we look at that auto-encoder objective function made of an encoder $\mathcal{E}$ and decoder $\mathcal{D}$:

$$\min_{\mathcal{E},\mathcal{D}} AE(p, \mathcal{E}, \mathcal{D}) \tag{167}$$

where the AE operator is simply the mean distance between the original data from p and the same data distorted by an encoder $\mathcal{E}$ followed by a decoder $\mathcal{D}$:

$$AE(p, p_{\mathcal{E},\mathcal{D}}) = \mathbb{E}_{\mathbf{x}\sim p}\left[\|\mathcal{D} \circ \mathcal{E}(\mathbf{x}) - \mathbf{x}\|_2\right] \tag{168}$$

We believe that equations Eq. (166) and Eq. (168) are similar. Training an auto-encoder and training a generative adversarial network are both unsupervised but auto-encoder are easier to train because there is only a one-way minimization which is not the case for GANs that have two-ways type of minimization (min-max) w.r.t. different parameters.

By training an auto-encoder, we get a reasonable reconstruction loss for output functions $\mathcal{E}$ encoder and $\mathcal{D}$ decoder. Now, it is possible to estimate the mean $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{E}(\mathbf{x}_i)$ and covariance $\boldsymbol{\sigma} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(\mathcal{E}(\mathbf{x}_i) - \boldsymbol{\mu}\right)^2}$ (with element-wise square root and square functions). The intuition is that for a Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$, then the decoded points $\mathcal{D}(\boldsymbol{\mu} + \boldsymbol{\sigma} \times \boldsymbol{\epsilon})$ (with element-wise product) has a distribution that is close to decoded codes $\mathcal{D} \circ \mathcal{E}(\mathbf{x})$ which is close to the original data $\mathbf{x}$. This way, we made a reasonable initialization for a generator made of a Gaussian noise into a linear layer from empirical mean $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ followed by the decoder function $\mathcal{D}$.

Early on since the beginning of the GAN literature in 2014, deep learning scientists saw the potential of seeing the min-max optimization scheme as the minimization of a discrepancy between real and generated data where that discrepancy required and adversarial optimization for estimation [Goodfellow, 2016] (i. e. a maximization of an objective to be minimized *in fine*). Back to our work, at this point, the auto-encoder training was a traditional *one-way* minimization but now the adversarial network (or critic against the generator) is needed to measure the discrepancy we just mentioned between real and generated data. In our Wasserstein discrepancy case, this correponds to only maximization via the Kantorovich-Rubinstein duality of the Wasserstein distance. Once both the generator from the auto-encoder minimization and next the critic from the discrepancy maximization for estimation are done. We actually not only know how far our generator (i. e. decoder) combined with our noise generator is from the real data distribution but we also get a convincing initialization for regular GANs training.

To sum up, we propose three steps:

1. Auto-Encoder Minimization for the euclidean reconstruction error to be low;

2. Critic Maximization until real v. s. generated discrepancy has plateaued out;

3. Regular GAN Min-Max optimization with a generator initialized by a Gaussian fitting the encoded codes followed by the decoder on the one side and the critic previously maximizing the real v. s. generated discrepancy on the other side;

which is much easier for monitoring and prototyping as getting rid off an unpredictable objective variations behavior especially at the first epochs contributes to acceleration of development and robustness w.r.t. uncontrolled objective variations. In these conditions, finding hyper-parameters is much more simplified.

EXPERIMENTS TRAINING DCGAN ON CIFAR-10

Generative Adversarial Networks [Goodfellow et al., 2014] really attracted the research community attention when image rendering started to get convincing especially with the spark ignition DCGAN work of Radford et al. [2015] with a recent research and engineering accomplishment in extremely large scale settings of BigGAN [Brock et al., 2019] and mainstream press demos[4]. Following this trend, we use the same neural network structure as DCGAN [Radford et al., 2015] shown in Fig. 27 for a Wasserstein GAN trained on CIFAR-10 images dataset [**?**].

Figure 27: DCGAN encoder structure, the critic has same structure (with an additional 100-1 perceptron layer) and the decoder/generator has a symmetric structure

In this section, we use Adam [Kingma and Ba, 2014] default parameter with 10^{-6} for generator learning rate and 10^{-5} for critic learning rate and also 10^{-5} the one-way minimization learning rate. Moreover, in min-max optimizations, we do 10 max iterations followed by 1 min iteration for each data mini-batch. The first step according to our section 4.6 consists in training an auto-encoder. Fig. 28 is showing the evolution of the minimized loss w.r.t. the number of iterations.

Second step consists in estimating the mean μ and standard deviation σ of the codes coordinates (codes are encoded version of data). Then a critic maximization is taking place so that we can estimate the Wassertein distance between the real data and the decoded Gaussian noise associated with $\mathcal{N}(\mu, \sigma)$ as shown in Fig. 29. Here, the Gaussian parameter (μ, σ) and the decoder playing now the role of generator are all remaining frozen (i. e. with zero learning rate) so that the critic neural network can be considered as just a tool to estimate the Wasserstein distance between fixed paramaters noise and generator data distribution and real data distribution in a Kantorovich-Rubinstein maximization procedure.

4 `https://www.thispersondoesnotexist.com`

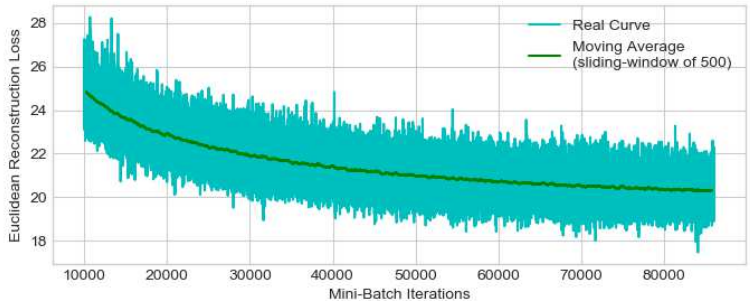

Figure 28: Autoencoder mini-batch loss w.r.t. the number of iterations (the lower the better, 50 iterations for 1 epoch)

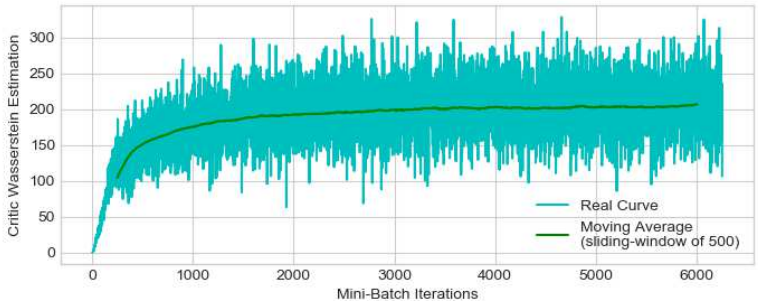

Figure 29: Critic Mini-Batch Wasserstein Distance Estimate w.r.t. the number of iterations. (the higher the better, 50 iterations for 1 epoch)

Third and final step is doing the original optimization scheme but now from a relevant starting point (which is crucial in non-convex deep learning optimization notoriously prone to spurious extrema). Fig. 30 is showing a direct minimization in spite of the adversarial min-max learning procedure. Here all parameters are allowed to vary meaning that the first two steps are just initialization steps following the *end-to-end* recommendations.

All these steps (except the first and easy auto-encoder step) are greatly simplified thanks to the ease of using the Lipschitz property for the critic neural network offered by Miyato et al. [2018] (this would not be reasonable with clipping [Arjovsky et al., 2017] nor regularization term [Gulrajani et al., 2017]). Compared to direct GAN training shown in Fig. 32, with fewer iterations and less tedious selection of learning rates, our approach gets better results. Indeed, one-way optimization schemes are easier than two-ways min-max optimization ones to handle. With mastered tools described by LeCun et al. [1998] and decades of shared good practice, one-way optimization (say minimization and not min-max optimization) is well studied nowadays. This is to be put in contrast with the two opposite learning rates required to the direct stochastically alternated method. To be honest, in our case, the final step has the same min-max GAN-like algorithm

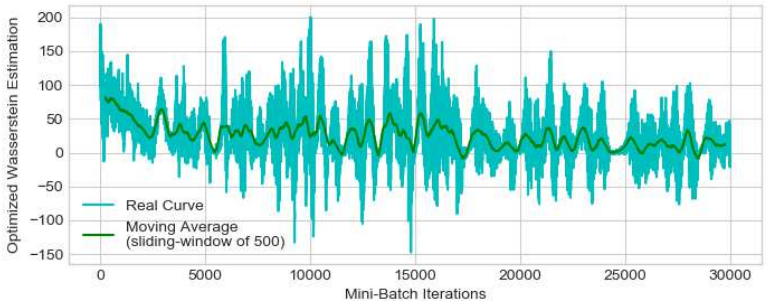

Figure 30: Our pre-trained GAN mini-batch objective w.r.t. the number of iterations. (the lower the better, 550 iterations for 1 epoch as each data mini-batch is seen 10 times for the critic and once for the generator)

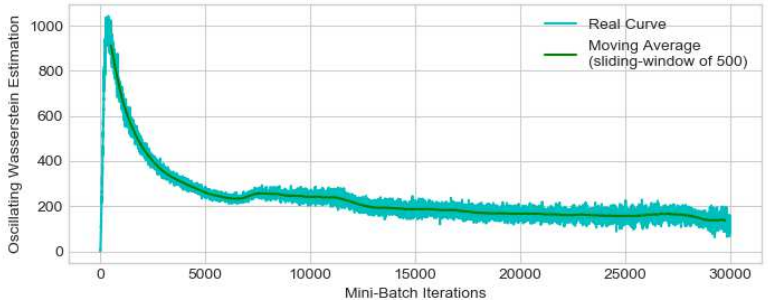

Figure 31: Oscillating GAN mini-batch objective w.r.t. the number of iterations. (the lower the better, 550 iterations for 1 epoch as each data mini-batch is seen 10 times for the critic and once for the generator)

but the variations are better controlled in practice because the system is better initialized.

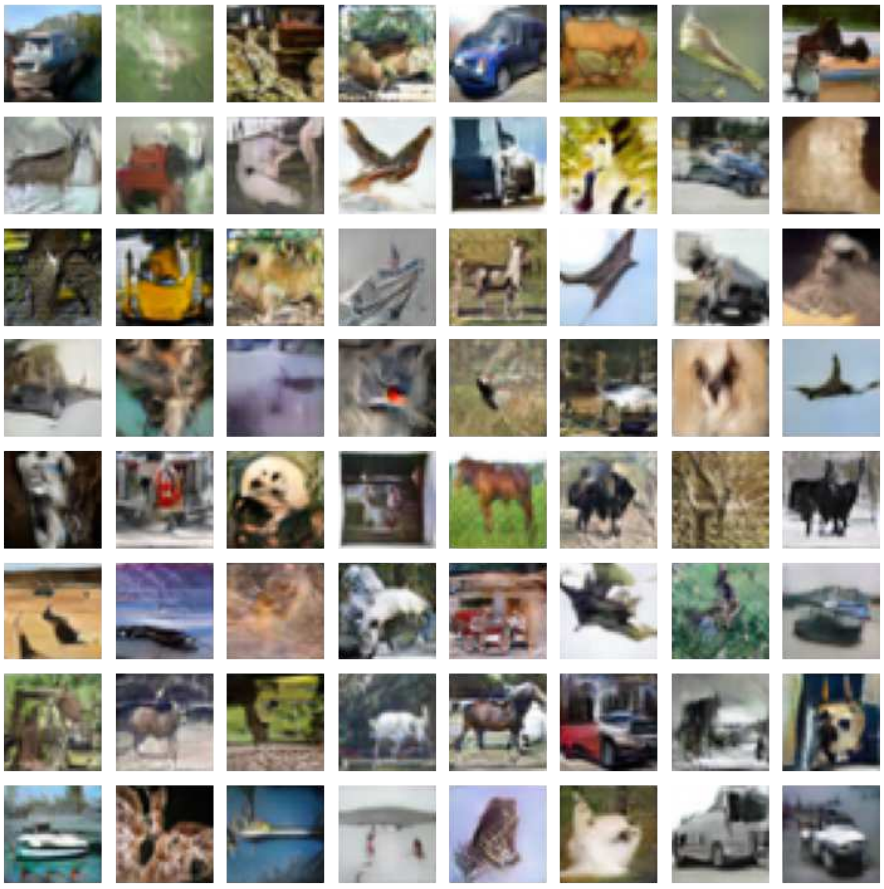

Figure 32: Generated samples of our Wasserstein GAN trained on CIFAR-10

BIBLIOGRAPHY

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`.

S. Abbott and C. A. Rogers. *Hausdorff Measures*, volume 83. Cambridge University Press, 1999. doi: 10.2307/3619107.

A. Abouchar. Air Transport Demand, Congestion Costs, and the Theory of Optimal Airport Use. *The Canadian Journal of Economics*, 3(3):463, 1970. ISSN 00084085. doi: 10.2307/133661.

S. Abu-El-Haija, N. Kothari, J. Lee, A. P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. YouTube-8M: A Large-Scale Video Classification Benchmark. *Google Research*, 2016. URL `https://research.google/pubs/pub45619`.

Z. Ahmad. *The epistemology of Ibn Khaldūn*. Routledge, 2003. ISBN 020363389X.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows. *Handbooks in Operations Research and Management Science*, 1(C):211–369, 1989. ISSN 09270507. doi: 10.1016/S0927-0507(89)01005-4.

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

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

L. Ambrosio, N. Gigli, and G. Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

A. M. Andrew. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, volume 30. Cambridge University Press, 2001. doi: 10.1108/k.2001.30.1.103.6.

C. Archambeau and M. Verleysen. Manifold constrained variational mixtures. In *International Conference on Artificial Neural Networks*, pages 279–284. Springer, 2005.

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

S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In *34th International Conference on Machine Learning, ICML 2017*, volume 1, pages 322–349. JMLR, 2017. ISBN 9781510855144.

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

F. Bach. Beyond stochastic gradient descent for large-scale machine learning, 2016. URL http://ecmlpkdd2014.loria.fr/wp-content/uploads/2014/09/fbach{_}ecml{_}2014.pdf.

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

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

D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2011. doi: 10.1017/cbo9780511804779.

E. Barillot, L. Calzone, P. Hupe, J.-P. Vert, and A. Zinovyev. *Computational systems biology of cancer*. CRC Press, 2012.

R. E. Bellman. *Dynamic Programming*. Rand Corporation Research Study. Princeton University Press, 1957.

J. D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000. ISSN 0029599X. doi: 10.1007/s002110050002.

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

C. Bernard. *Introduction à l'étude de la médecine expérimentale*. Librairie Joseph Gilbert, 1898.

D. P. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997. ISSN 01605682. doi: 10.4018/978-1-4666-5202-6.ch147.

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

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

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

C. Bishop. Mixture density networks. Technical report, January 1994.

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

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

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

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

M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *Thirtieth Conference on Neural Information Processing Systems*, abs/1604.07316, 2016. URL `https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf`.

J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical optimization: theoretical and practical aspects*, volume 41. Springer Science and Business Media, 2003. doi: 10.5860/choice.41-0357.

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

H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

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

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

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

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

C. Bouveyron, S. Girard, and C. Schmid. High-Dimensional Data Clustering. *Computational Statistics and Data Analysis*, 2007.

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

S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014. ISBN 978-0-521-83378-3. doi: 10.1017/CBO9780511804441. URL `https://web.stanford.edu/%7Eboyd/cvxbook/`.

L. Breiman. Probability, volume 7 of classics in applied mathematics. *Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA*, 2:6, 1992.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, oct 2001. ISSN 08856125.

L. Breiman. *Classification and regression trees*. Routledge, 2017. ISBN 9781351460491.

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

Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

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

M. Castelluccio. AI rising. *Strategic Finance*, 2017.

G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14(3):315–332, 1992. ISSN 01679473. doi: 10.1016/0167-9473(92)90042-E. URL `http://www.sciencedirect.com/science/article/pii/016794739290042E`.

A. Chakraborty, S. Ghosh, P. Mukhopadhyay, S. M. Dinara, A. Bag, M. K. Mahata, R. Kumar, S. Das, J. Sanjay, S. Majumdar, and D. Biswas. Trapping effect analysis of AlGaN/InGaN/GaN Heterostructure by conductance frequency measurement. *MRS Proceedings*, XXXIII(2):81–87, 2014. ISSN 0717-6163. doi: 10.1007/s13398-014-0173-7.2.

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

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

G. Chen. Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084*, 2015.

T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1724–1734, 2014. doi: 10.3115/v1/d14-1179.

A. Choromanska, T. Jebara, H. Kim, M. Mohan, and C. Monteleoni. Fast spectral clustering via the nyström method. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 367–381, 2013. ISBN 9783642409349. doi: 10.1007/978-3-642-40935-6_26.

N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal Transport for Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017.

N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid. MARS: Motion-Augmented RGB Stream for Action Recognition. In *CVPR*, 2019.

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

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

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2008.

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

F. De la Torre and T. Kanade. Discriminative cluster analysis. pages 241–248, 2006.

H. De March. *Multidimensional martingale optimal transport.* Theses, Université Paris-Saclay, June 2018. URL `https://pastel.archives-ouvertes.fr/tel-01973279`.

C. R. de Sá. Variance-Based Feature Importance in Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11828 LNAI, pages 306–315. Springer, 2019. ISBN 9783030337773. doi: 10.1007/978-3-030-33778-0_24.

A. Defazio, F. R. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 1646–1654, 2014.

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

N. S. Detlefsen, M. Jørgensen, and S. Hauberg. Reliable training and estimation of variance networks. In *Advances in Neural Information Processing Systems*, pages 6323–6333, 2019. URL `http://arxiv.org/abs/1906.03260`.

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

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

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

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

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

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

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

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

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

E. S. Epstein. Stochastic dynamic prediction. *Tellus*, 21(6):739–759, 1969. doi: 10.3402/tellusa.v21i6.10143.

W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. *arXiv preprint arXiv:1710.08446*, 2017.

L. Fei-Fei. Imagenet: crowdsourcing, benchmarking and other cool things. In *CMU VASC Seminar*, volume 16, pages 18–25, 2010.

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

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

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

Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

J. K. Galbraith. The pragmatism of John Kenneth Galbraith. *Acta Oeconomica*, 69 (s1):195–213, 2019. ISSN 15882659. doi: 10.1556/032.2019.69.S1.12.

T. Gao and V. Jojic. Degrees of freedom in deep neural networks. *32nd Conference on Uncertainty in Artificial Intelligence 2016, UAI 2016*, 2016.

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

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

T. Glasmachers. Limits of end-to-end learning. *arXiv preprint arXiv:1704.08305*, 2017.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/glorot10a.html`.

A. V. Goldberg and R. E. Tarjan. Finding Minimum-Cost Circulations by Canceling Negative Cycles. *Journal of the ACM (JACM)*, 36(4):873–886, 1989. ISSN 1557735X. doi: 10.1145/76359.76368.

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

G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

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

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

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

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

H. Greenspan, B. Van Ginneken, and R. M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

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

B. Hanin and M. Sellke. Approximating continuous functions by relu nets of minimal width. *CoRR*, abs/1710.11278, 2017. URL `http://arxiv.org/abs/1710.11278`.

Z. Harchaoui. Large-scale learning for image classification, 2013. URL `https://harchaoui.org/zaid/cvml13.pdf`.

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

P. J. Hayes and S. P. Weinstein. Construe-TIS: A System for Content-based Indexing of a Database of News Stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, volume 90, pages 49–64, 1990. ISBN 0-262-68068-8.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. URL `https://arxiv.org/abs/1512.03385`.

H. Hendrikx, F. Bach, and L. Massoulié. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives. In K. Chaudhuri and M. Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 897–906. PMLR, 2019. URL `http://proceedings.mlr.press/v89/hendrikx19a.html`.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning discrete representations via information maximizing self-augmented training. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1558–1567. PMLR, 06–11 Aug 2017.

P. Huang, Y. Huang, W. Wang, and L. Wang. Deep embedding network for clustering. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1532–1537. IEEE, 2014.

S.-J. Huang, R. Jin, and Z.-H. Zhou. Active Learning by Querying Informative and Representative Examples. In *Advances in Neural Information Processing Systems*, 2010.

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

R. Jenatton. *Structured sparsity-inducing norms: Statistical and algorithmic properties with applications to neuroimaging*. PhD thesis, 2011.

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

M. I. Jordan. On statistics, computation and scalability. *Bernoulli Society for Mathematical Statistics and Probability*, abs/1309.7804, 2013. URL `http://arxiv.org/abs/1309.7804`.

A. Joulin, F. R. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1943–1950. IEEE, 2010. ISBN 9781424469840. URL `https://doi.org/10.1109/CVPR.2010.5539868`.

O. Kallenberg. *Random measures, theory and applications*, volume 77. Springer, 2017.

A. Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks, 2015. URL `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001. ISSN 1369-7412. doi: 10.1111/1467-9868.00294.

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

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

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

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

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

J. M. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems*, pages 463–470, 2015.

R. T. Knight. Neural networks debunk phrenology. *Science*, 316(5831):1578–1579, 2007. ISSN 00368075. doi: 10.1126/science.1144677.

N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, J. Reynolds, A. Melnikov, N. Lunova, and O. Reblitz-Richardson. Pytorch captum. `https://github.com/pytorch/captum`, 2019.

M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

Y. LeCun. What's Wrong With Deep Learning? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. URL `http://yann.lecun.com`.

Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *MIT Press, Cambridge*, 1995.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989a. ISSN 0899-7667.

Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning. *IEEE Communications Magazine*, 27(11):41–46, 1989b.

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems 2, NIPS 1989*, pages 396–404. Morgan Kaufmann Publishers, 1990.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

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

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 1998.

B. Lévy. A Numerical Algorithm for L2 Semi-Discrete Optimal Transport in 3D. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(6):1693–1715, 2015. ISSN 12903841. doi: 10.1051/m2an/2015055.

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

J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong. *Fundamentals of speech recognition*. Pearson Education India, 2016. doi: 10.1016/b978-0-12-802398-3.00002-7.

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

Y. Liu, M. Yamada, Y. H. Tsai, T. Le, R. Salakhutdinov, and Y. Yang. Lsmi-sinkhorn: Semi-supervised squared-loss mutual information estimation with optimal transport. *AAAI*, abs/1909.02373, 2020. URL `http://arxiv.org/abs/1909.02373`.

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

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

D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

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

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

S. Mallat. *A wavelet tour of signal processing: the sparse way*. Academic Press, Inc., 2008.

V. G. Maz'ya and T. O. Shaposhnikova. *Jacques Hadamard: a universal mathematician*. Number 14. American Mathematical Soc., 1999.

Q. Mérigot. A multiscale approach to optimal transport. In *Eurographics Symposium on Geometry Processing*, volume 30, pages 1583–1592. Wiley Online Library, 2011. doi: 10.1111/j.1467-8659.2011.02032.x.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations ofwords and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

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

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

G. Monge. Mémoire sur la théorie de déblais et de remblais. Histoire de l'Académie Royale des science de Paris, avec les Mémoires de Mathématiques et de Physique pour la même année, 1781. URL `https://gallica.bnf.fr/ark:/12148/bpt6k35800/f1.image`.

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

K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *Conference on Human Factors in Computing Systems - Proceedings*, pages 237–242. ACM, 1991. ISBN 0897913833. doi: 10.1145/108844.108900.

K. Murphy. *Machine Learning, a Probabilistic Perspective*. MIT press, 2012.

R. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

S. Newman. *Descartes' epistemology*. Routledge, 2018.

A. Ng. Deep learning, 2013. URL `https://www.youtube.com/watch?v=n1ViNeWhC24`.

A. Ng. The state of artificial intelligence, 2018. URL `https://www.youtube.com/watch?v=19IXayufFv4`.

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

D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng. Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access*, 7:36322–36333, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2905015.

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

K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, 1901.

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.

B. Peters. The Age of Big Data. *Forbes*, 11(2012):4–9, 2012. URL `http://www.forbes.com/sites/bradpeters/2012/07/12/the-age-of-big-data/`.

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

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

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

J. Rabin, G. Peyré, J. Delon, and M. Bernot. Wasserstein barycenter and its application to texture mixing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 435–446. Springer, 2012.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

L. Roberts. Picture Coding Using Pseudo-Random Noise. *IRE Transactions on Information Theory*, (2):145–154, 1962. ISSN 21682712.

F. Rosenbaltt. The perceptron–a perciving and recognizing automation. *Report 85-460-1 Cornell Aeronautical Laboratory, Ithaca, Tech. Rep.*, 1957.

F. Rosenblatt. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Technical Report 4, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM SIGGRAPH*, 23(3):309–314, 2004.

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

F. Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

C. Schmid. Active Large-scale Learning for Visual Recognition, 2013. URL `https://lear.inrialpes.fr/allegro`.

B. Scholkopf and A. J. Smola. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2001.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

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

M. Seeger. *Gaussian processes for machine learning.*, volume 14. MIT press Cambridge, MA, 2004. doi: 10.1142/S0129065704001899.

J. Shawe-Taylor, N. Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

D. Shu, J. Cunningham, G. Stump, S. W. Miller, M. A. Yukish, T. W. Simpson, and C. S. Tucker. 3D Design Using Generative Adversarial Networks and Physics-based Validation. *Journal of Mechanical Design*, 142(7):1–51, 2019. ISSN 1050-0472. doi: 10.1115/1.4045419.

K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The Hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, 2010.

L. Smolinski. The Scale of Soviet Industrial Establishments. *The American Economic Review*, 52(2):138–148, 1962. URL `https://about.jstor.org/terms`.

R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML*, pages 129–136. Omnipress, 2011. ISBN 9781450306195. URL `https://icml.cc/2011/papers/125_icmlpaper.pdf`.

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

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

N. Srivastava. Improving neural networks with dropout. *University of Toronto*, 182(566):7, 2013.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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

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

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

K. Sung, T. Poggio, H. Rowley, S. Baluja, and T. Kanade. MIT+ CMU frontal face dataset a, b and c, 1998.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 1–9, 2015. ISBN 9781467369640.

J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade. Invariances and Data Augmentation for Supervised Music Transcription. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

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

G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

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

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

J. Vermorel. Probabilistic forecasting, 2018. URL https://youtu.be/KXC-hPCojGQ.

J.-P. Vert. Learning from ranks, learning to rank. URL http://members.cbio.mines-paristech.fr/~jvert/talks/200114turing/turing.pdf.

R. Vidal, J. Bruna, R. Giryes, and S. Soatto. Mathematics of deep learning, 2017. URL https://www.youtube.com/watch?v=eEPXTMHnBJA.

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

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec): 3371–3408, 2010.

P. A. Viola and M. J. Jones. Robust real-time face detection. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), Vancouver, British Columbia, Canada, July 7-14, 2001 - Volume 2*, page 747. IEEE Computer Society, 2001.

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

Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang. Learning a task-specific deep architecture for clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 369–377. SIAM, 2016.

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

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

J. Yang, D. Parikh, and D. Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. 2016b. URL http://arxiv.org/abs/1604.03628.

X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30 (9):2805–2824, 2019.

M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010*, 2010.

M. Zaslavskiy. *Graph matching and its application in computer vision and computational biology R ´ esum ´ e*. PhD thesis, Mines de Paris, 2010.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, 2014. ISBN 9783319105895.

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

X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.

P. Zikoupoulos and C. Eaton. *Understanding big data: Analytics for Enterprise Class Hadoop and Streaming*, volume 11. McGraw-Hill Osborne Media, 2016. ISBN 9780071790536.

V. Zue, S. Seneff, and J. Glass. Speech database development at MIT: Timit and beyond. *Speech Communication*, 1990. ISSN 01676393.