Group Meeting - September 25, 2020

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Marcus Aurelius

The universe is **change**; our life is what our thoughts make it. The soul becomes dyed with the color of its thoughts.

And Marcus Aurelius needs some colors too. Indeed, the ancient Rome was extremely colorful. It was not whiten as depicted from the Renaissance time. This is a big **white** lie. Please check: `https://www.youtube.com/watch?v=4jmMWohs1XM`

Victor Hugo (author of *Les Misérables*)

If you wish to understand what **Revolution** is, call it **Progress**; and if you wish to understand what **Progress** is, call it **Tomorrow**.

Storming of the Bastille, July 14, 1789
In the memory of Gaspard Monge

Papers – Optimal Transport (Part 1: Wasserstein)

1. **Wasserstein GAN**
   https://arxiv.org/pdf/1701.07875.pdf
2. **Wasserstein Auto-Encoders** (ICLR 2018)
   https://arxiv.org/pdf/1711.01558.pdf
3. **Optimal Transport Graph Neural Networks**
   https://arxiv.org/pdf/2006.04804.pdf
4. **Wasserstein Weisfeiler-Lehman Graph Kernels** (NeurIPS 2019)
   https://arxiv.org/pdf/1906.01277.pdf
5. **Poincaré Wasserstein Autoencoder** (NeurIPS 2018 Bayesian workshop)
   http://bayesiandeeplearning.org/2018/papers/18.pdf

Papers – Optimal Transport (Part 2: Sinkhorn)

1. **Wasserstein Variational Inference** (NeurIPS 2018)
   https://arxiv.org/pdf/1805.11284.pdf
2. **Sinkhorn Distances: Lightspeed Computation of Optimal Transport** (NIPS 2013)
   https://arxiv.org/pdf/1306.0895.pdf
3. **Sinkhorn AutoEncoders** (UAI 2019)
   https://arxiv.org/pdf/1810.01118.pdf
4. **On Unbalanced Optimal Transport: An Analysis of Sinkhorn Algorithm** (ICML 2020)
   https://arxiv.org/pdf/2002.03293.pdf

Paper 4

Wasserstein Weisfeiler-Lehman Graph Kernels (NeurIPS 2019)
Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt
`https://arxiv.org/pdf/1906.01277.pdf`

Wasserstein Weisfeiler-Lehman Graph Kernels (1)

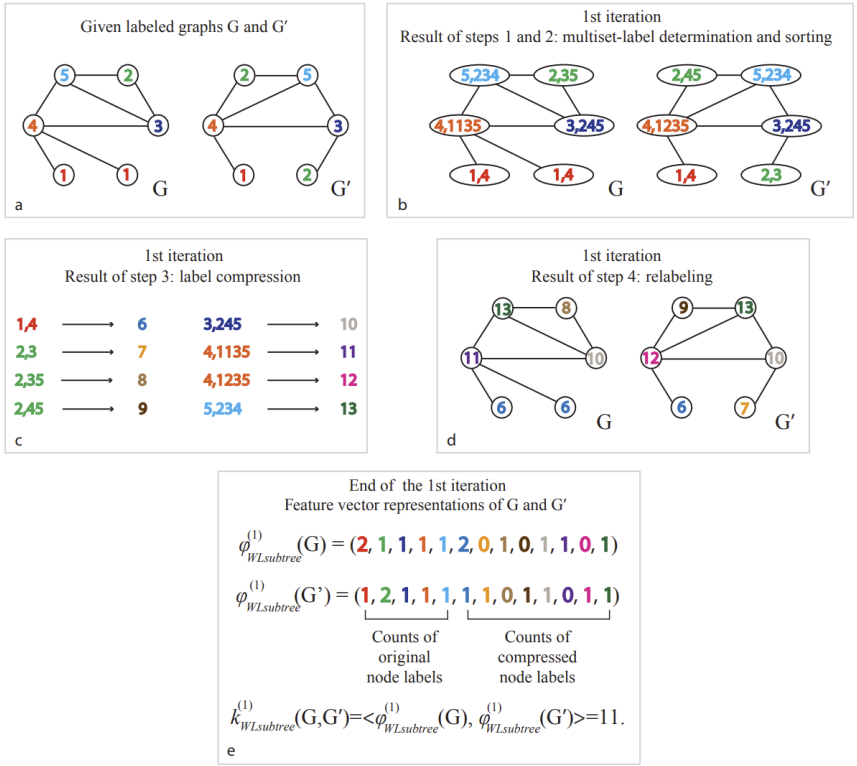


Figure 2: Illustration of the computation of the Weisfeiler-Lehman subtree kernel with $h = 1$ for two graphs. Here $\{1, 2, \ldots, 13\} \in \Sigma$ are considered as letters. Note that compressed labels denote subtree patterns: For instance, if a node has label 8, this means that there is a subtree pattern of height 1 rooted at this node, where the root has label 2 and its neighbours have labels 3 and 5.

Weisfeiler-Lehman graph kernels, Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, Karsten M. Borgwardt `https://people.mpi-inf.mpg.de/~mehlhorn/ftp/genWLpaper.pdf`

Wasserstein Weisfeiler-Lehman Graph Kernels (2)

Algorithm 1 One iteration of the 1-dim. Weisfeiler-Lehman test of graph isomorphism

1: Multiset-label determination
- For $i = 0$, set $M_i(v) := l_0(v) = \ell(v)$. [2]
- For $i > 0$, assign a multiset-label $M_i(v)$ to each node v in G and G' which consists of the multiset $\{l_{i-1}(u) | u \in \mathcal{N}(v)\}$.

2: Sorting each multiset
- Sort elements in $M_i(v)$ in ascending order and concatenate them into a string $s_i(v)$.
- Add $l_{i-1}(v)$ as a prefix to $s_i(v)$ and call the resulting string $s_i(v)$.

3: Label compression
- Sort all of the strings $s_i(v)$ for all v from G and G' in ascending order.
- Map each string $s_i(v)$ to a new compressed label, using a function $f : \Sigma^* \to \Sigma$ such that $f(s_i(v)) = f(s_i(w))$ if and only if $s_i(v) = s_i(w)$.

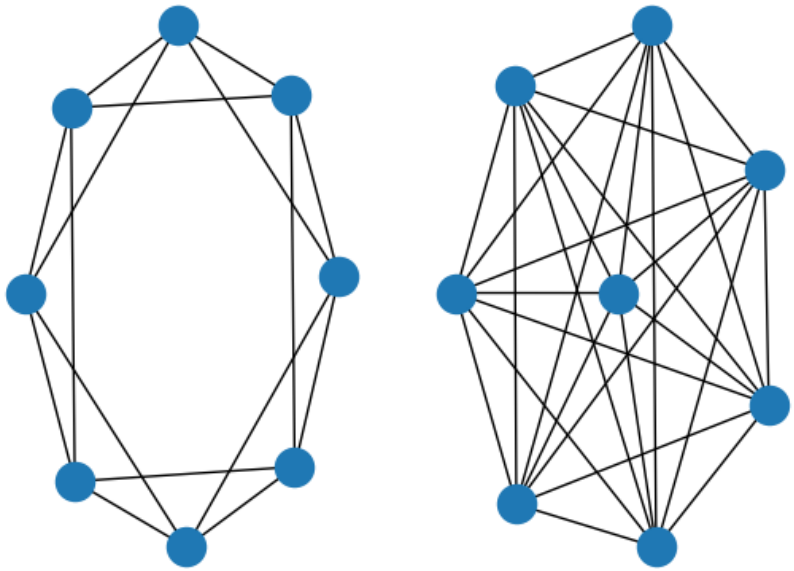
4: Relabeling
- Set $l_i(v) := f(s_i(v))$ for all nodes in G and G'.

Weisfeiler-Lehman graph kernels, Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, Karsten M. Borgwardt `https://people.mpi-inf.mpg.de/~mehlhorn/ftp/genWLpaper.pdf`

Wasserstein Weisfeiler-Lehman Graph Kernels (3)

When does the Weisfeiler-Lehman graph isomorphism test fail?

- Graph Isomorphism in Quasipolynomial Time, László Babai
  `https://arxiv.org/abs/1512.03547`
- Video lecture of Babai at UChicago:
  `https://www.youtube.com/watch?v=qYIhA3O9Nz0`
- Example: Johnson graph (8, 2) – we used to call it circular skip link graphs (left: input, right: fail reconstruction).



Wasserstein Weisfeiler-Lehman Graph Kernels (4)

Problem of existing graph kernels

Most graph kernels measure the similarity of combinatorial objects by comparing their substructures, but they use a naive aggregration of the final set of substructures (usually a sum or average) thereby potentially discrading valuable information about the distribution of individual components.

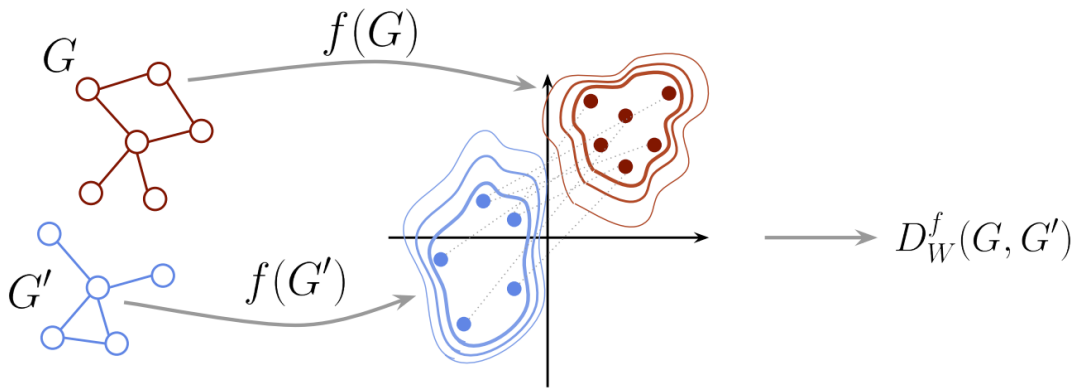


Figure 1: Visual summary of the graph Wasserstein distance. First, f generates embeddings for two input graphs G and G'. Then, the Wasserstein distance between the embedding distributions is computed.

Wasserstein Weisfeiler-Lehman Graph Kernels (5)

Proposals

1. A novel method that computes the Wasserstein distance between the node feature vector distributions of two graphs (witht the hope of finding subtler differences in data sets by considering graphs as high-dimensional objects rather than simple means).
2. Propose a Weisfeiler-Lehman-inspired embedding scheme for graphs with **continuous** node attributes and **weighted** edges, enhance it with the computed Wasserstein distance.

Wasserstein Weisfeiler-Lehman Graph Kernels (6)

Let $\mathcal{X}$ be a set and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function associated with a Hilbert space $\mathcal{H}$, such that there exists a map $\phi : \mathcal{X} \to \mathcal{H}$ with $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$. Then, $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) and k is said to be a positive definite kernel.

Graph $G = (V, E)$:

- $|V| = n_G$, $|E| = m_G$, and $|\mathcal{N}(v)| = \deg(v)$.
- A **label** on a node is a function $\ell : V \to \Sigma$ that assigns to each node v in G a value $\ell(v)$ from a finite label alphabet Σ.
- An **attribute** on a node is an associated vector $a(v) \in \mathbb{R}^m$.
- Weight on edges $w(e) \in \mathbb{R}$.

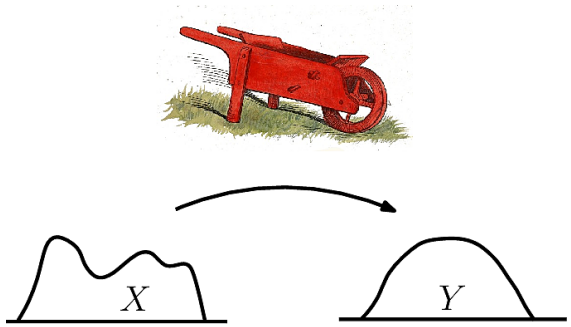
Wasserstein Weisfeiler-Lehman Graph Kernels (7)

Definition 1 – Earth mover distance (EMD)

Let σ and μ be two probability distributions on a metric space M equipped with a distance d (e.g. Euclidean distance). The L^p-Wasserstein distance for $p \in [1, \infty)$ is defined as:

$$W_p(\sigma, \mu) = \left(\inf_{\gamma \in \Gamma(\sigma, \mu)} \int_{M \times M} d(x, y)^p d\gamma(x, y) \right)^{\frac{1}{p}}$$

where $\Gamma(\sigma, \mu)$ is the set of all transportation plans $\gamma \in \Gamma(\sigma, \mu)$ over $M \times M$ with marginals σ and μ, respectively.



Wasserstein Weisfeiler-Lehman Graph Kernels (8)

Because we deal with finite sets of node embeddings (not with continuous probability distributions), we can reformulate the Wasserstein distance as a sum rather than an integral.

Definition 1(b)

Given two sets of vectors $X \in \mathbb{R}^{n \times m}$ and $X' \in \mathbb{R}^{n' \times m}$:

$$W_1(X, X') = \min_{P \in \Gamma(X, X')} \langle P, M \rangle_{\mathcal{F}}$$

where M is the distance matrix (e.g. $M_{x \in X, x' \in X'} = d(x, x')$), $P \in \Gamma$ is a transport matrix (or joint probability) such that $P\mathbf{1} = 1/n$ and $\mathbf{1}^T P = 1/n'$, and $\langle A, B \rangle_{\mathcal{F}} = \text{trace}(A^\dagger B)$ is the Frobenius dot product.

Wasserstein Weisfeiler-Lehman Graph Kernels (9)

Definition 2 – Graph embedding scheme

Given a graph embedding scheme (algorithm) $f : \mathcal{G} \to \mathbb{R}^{|V| \times m}$. For each $v_i \in V$, the i-th row of $X_G = f(G)$ is the node embedding of v_i.

Definition 3 – Graph Wasserstein Distance (GWD)

Given a graph embedding scheme $f : \mathcal{G} \to \mathbb{R}^{|V| \times m}$ and a ground distance $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, GWD is defined as:

$$D_W^f(G, G') = W_1(f(G), f(G'))$$

Wasserstein Weisfeiler-Lehman Graph Kernels (10)

- WL scheme for **discrete** node labels:
  1. Initialize $\ell^0 = \ell(v), \forall v \in V$
  2. H iterations:

$$\ell^{h+1}(v) = \text{hash}(\ell^h(v), \mathcal{N}^h(v))$$

  where

$$\mathcal{N}^h(v) = \{\ell^h(u_0), .., \ell^h(u_{\deg(v)-1})\}$$

- WL scheme for **continuous** attributes:
  1. Initialize $a^0(v) = a(v), \forall v \in V$
  2. H iterations:

$$a^{h+1}(v) = \frac{1}{2}\left(a^h(v) + \frac{1}{\deg(v)} \sum_{w \in \mathcal{N}(v)} w_{(u,v)} a^h(u)\right)$$

  where $1/2$ is the factor to ensure the same scale of features across the iterations.

Wasserstein Weisfeiler-Lehman Graph Kernels (11)

Definition 4 – WL features

Given the node embeddings of graph G at iteration H:

$$f^H : G \to \mathbb{R}^{n_G \times (m(H+1))}$$

The WL features is defined as:

$$X_G = \text{concatenate}(X_G^0, .., X_G^H)$$

in which

$$X_G^h = [x^h(v_1), .., x^h(v_{n_G})]^T$$

where $x^h(.) = \ell^h(.)$ for **discrete** labels, and $x^h(.) = a^h(.)$ for **continuous** attributes.

Wasserstein Weisfeiler-Lehman Graph Kernels (12)

We compute the ground distance between each pair of nodes:

1. For **discrete** labels, normalized Hamming distance:

$$d_{\text{Ham}}(v, v') = \frac{1}{H+1} \sum_{i=1}^{H+1} \mathbb{I}[v_i = v'_i]$$

2. For **continuous** attributes, Euclidean distance: $d_E(v, v') = ||v - v'||_2$

Definition 5 – Wasserstein Weisfeiler-Lehman

Given a set of graphs $\mathcal{G} = \{G_1, .., G_N\}$, the Wasserstein Weisfeiler-Lehman (WWL) kernel is defined as:

$$K_{WWL} = e^{-\lambda D_W^f}$$

(diffusion)

Wasserstein Weisfeiler-Lehman Graph Kernels (13)

Algorithm 1 Compute Wasserstein graph kernel

Input: Two graphs G_1, G_2; graph embedding scheme f^H; ground distance d; λ.
Output: kernel value $k_{WWL}(G_1, G_2)$.
$X_{G_1} \leftarrow f^H(G_1)$; $X_{G_2} \leftarrow f^H(G_2)$ // Generate node embeddings
$D \leftarrow \text{pairwise\_dist}(X_{G_1}, X_{G_2}, d)$ // Compute the ground distance between each pair of nodes
$D_W(G_1, G_2) = \min_{P \in \Gamma} \langle P, D \rangle$ // Compute the Wasserstein distance
$k_W(G_1, G_2) \leftarrow e^{-\lambda D_W(G_1, G_2)}$

Theorem 1

The categorial **(on discrete labels)** WWL kernel is psositive definite for all $\lambda > 0$.

Open question

The WWL kernel on **continuous attributes** is positive definite(?).

Paper 3

Optimal Transport Graph Neural Networks

Gary Bécigneul, Octavian-Eugen Ganea, Benson Chen, Regina Barzilay, and Tommi Jaakkola

`https://arxiv.org/pdf/2006.04804.pdf`

Optimal Transport Graph Neural Networks (1)

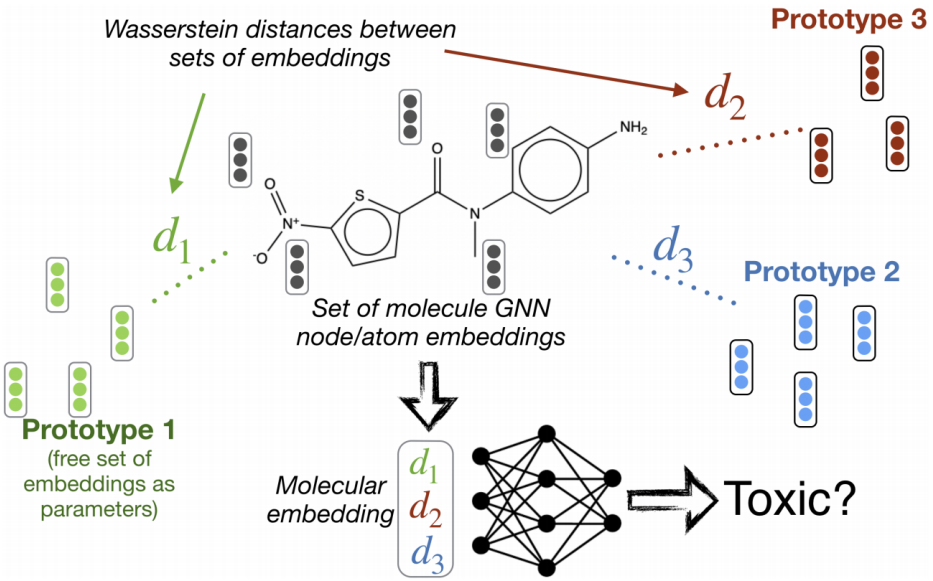


Figure 1: *Intuition for our Wasserstein prototype model. We assume that a few prototypes, e.g. some functional groups, highlight key facets or structural features of graphs in a particular graph classification/regression task at hand. We then express graphs by relating them to these abstract prototypes represented as free point cloud parameters. Note that we do not learn the graph structure of the prototypes.*

Optimal Transport Graph Neural Networks (2)

Proposal

1. OT-GNN that computes graph embeddings from optimal transport distances between the set of GNN node embeddings and **prototype** point clouds as free parameters.
2. GNN architectures naively average or sum node embeddings into an aggregated graph representation $\Rightarrow$ losing structural or semantic information. (I agree)
3. The authors argue that their function class on point clouds satisfies the universal approximation theorem, a fundamental property which was lost by sum aggregation. (I disagree: The OT is only applied on the top/graph-level, not inside the GNN!)

Optimal Transport Graph Neural Networks (3)

Wasserstein for point clouds

Let a point cloud $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ of size n to be a set of n points $\boldsymbol{x}_i \in \mathbb{R}^d$. Given a cost function c (or ground distance) on $\mathbb{R}^d$, the **Wasserstein discrepancy** between two points clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$ of sizes n and m is defined as:

$$\mathcal{W}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\boldsymbol{T} \in \mathcal{C}_{\boldsymbol{XY}}} \sum_{ij} T_{ij} c(\boldsymbol{x}_i, \boldsymbol{y}_j)$$

where $\boldsymbol{T}$ is a **transport plan** matrix that satisfies the marginal constraints (**mass preservation**):

$$\boldsymbol{T}\mathbf{1}_m = \frac{1}{n}\mathbf{1}_n, \qquad \boldsymbol{T}^T\mathbf{1}_n = \frac{1}{m}\mathbf{1}_m$$

Optimal Transport Graph Neural Networks (4)

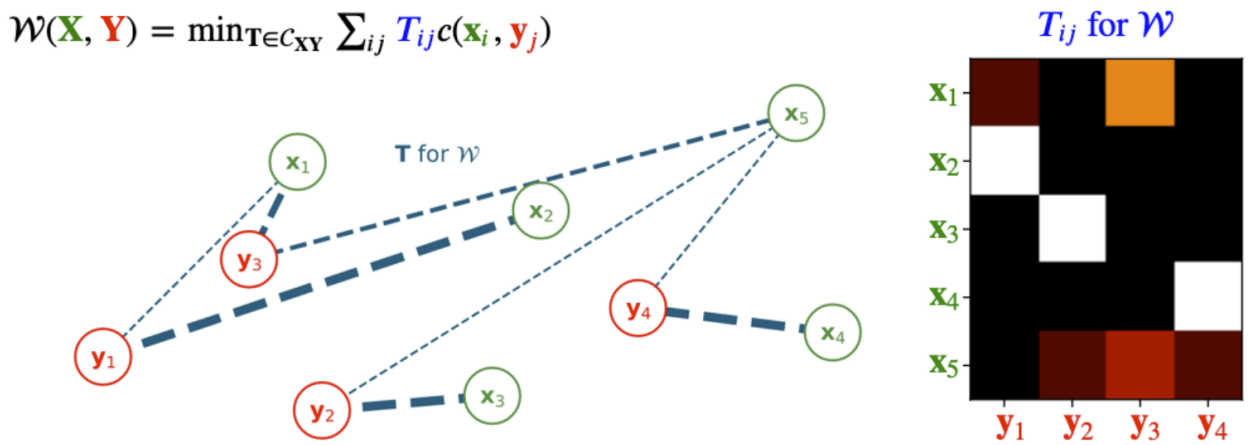


Figure 2: We illustrate, for a given 2D point cloud, the optimal transport plan obtained from minimizing the Wasserstein costs; $c(\cdot, \cdot)$ denotes the Euclidean distance. A higher dotted-line thickness illustrates a greater mass transport.

Optimal Transport Graph Neural Networks (5)

- **Before:** The graph embedding is aggregated as a sum $\boldsymbol{h} = \sum_{v \in V} \boldsymbol{h}_v$. And MLP performs matrix mulitplication $\boldsymbol{Rh}$ such that $(\boldsymbol{Rh})_i = \langle \boldsymbol{r}_i, \boldsymbol{h} \rangle$.
- **Now:** The aggregated graph embedding $\boldsymbol{h}$ is **replaced** by unaggregated point cloud $\boldsymbol{H} = \{\boldsymbol{h}_v\}_{v \in V}$, and the inner-product $\langle \boldsymbol{r}_i, \boldsymbol{h} \rangle$ is **replaced** by the **Wasserstein discrepancy**:

$$\mathcal{W}(\boldsymbol{H}, \boldsymbol{Q}_i) = \min_{\boldsymbol{T} \in \mathcal{C}_{\boldsymbol{HQ}_i}} \sum_{vj} T_{vj} c(\boldsymbol{h}_v, \boldsymbol{q}_i^j)$$

where $\boldsymbol{Q}_i = \{\boldsymbol{q}_i^j\}_j$ are point clouds and free parameters, and the cost c is some kernel.

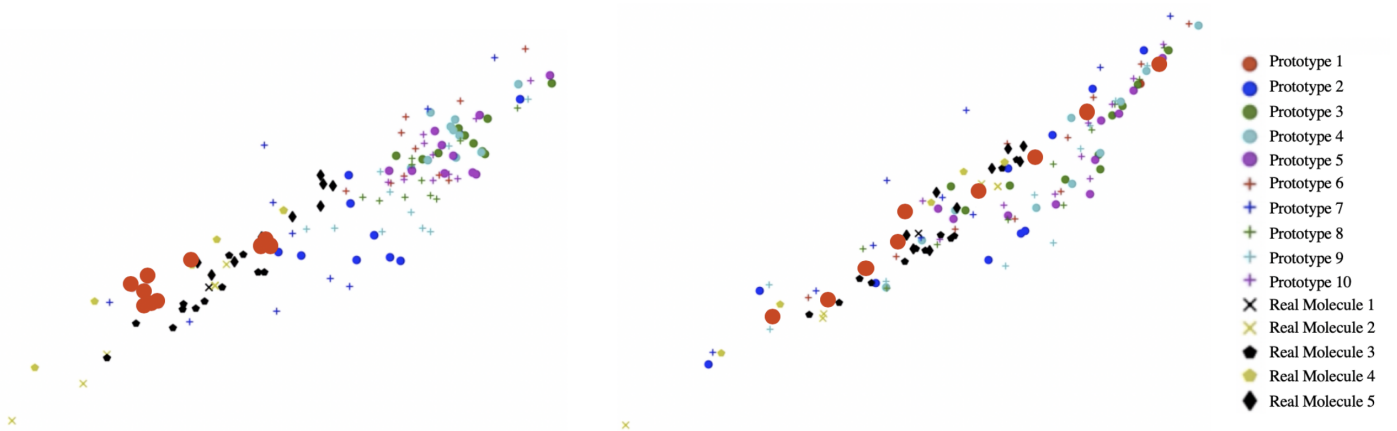
Optimal Transport Graph Neural Networks (6)

Collapse problem

What would happen to $\mathcal{W}(\boldsymbol{H}, \boldsymbol{Q}_i)$ if all points $\boldsymbol{q}_i^j$ belonging to point cloud $\boldsymbol{Q}_i$ collapse into the same point $\boldsymbol{q}_i$? For a cost $c = -\langle ., . \rangle$:

$$\boldsymbol{W}(\boldsymbol{H}, \boldsymbol{Q}_i) = -\sum_{vj} T_{vj} \langle \boldsymbol{h}_v, \boldsymbol{q}_i^j \rangle = -\langle \boldsymbol{h}, \boldsymbol{q}_i / |V| \rangle$$

that is indeed the same as the Euclidean model.



(a) No contrastive regularization (b) Using contrastive regularization (weight 0.1)

Optimal Transport Graph Neural Networks (7)

Contrastive regularization

Consider a point cloud $\boldsymbol{Y}$ of node embeddings, and let $\boldsymbol{T}_i$ be an optimal transport plan obtained in the computation of $\mathcal{W}(\boldsymbol{Y}, \boldsymbol{Q}_i)$. For each $\boldsymbol{T}_i$, we build a set $N(\boldsymbol{T}_i) \subset \mathcal{C}_{\boldsymbol{Y}\boldsymbol{Q}_i}$ of noisy/contrastive transports. The contrastive regularization consists in maximizing:

$$\sum_i \log \left(\frac{e^{-\mathcal{W}_{\boldsymbol{T}_i}(\boldsymbol{Y}, \boldsymbol{Q}_i)}}{e^{-\mathcal{W}_{\boldsymbol{T}_i}(\boldsymbol{Y}, \boldsymbol{Q}_i)} + \sum_{\boldsymbol{T} \in N(\boldsymbol{T}_i)} e^{-\mathcal{W}_{\boldsymbol{T}}(\boldsymbol{Y}, \boldsymbol{Q}_i)}} \right)$$

that can be interpreted as maximizing the **approximated** log-likelihood $\log \mathbb{P}(\boldsymbol{T}_i | \boldsymbol{Y}, \boldsymbol{Q}_i)$.

Optimal Transport Graph Neural Networks (8)

The **discrepancy** Wasserstein kernel:

$$\mathcal{W}_{L2}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\boldsymbol{T} \in \mathcal{C}_{\boldsymbol{XY}}} \sum_{ij} T_{ij} ||\boldsymbol{x}_i - \boldsymbol{y}_j||_2^2$$

The **similarity** Wasserstein kernel:

$$\mathcal{W}_{dot}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\boldsymbol{T} \in \mathcal{C}_{\boldsymbol{XY}}} \sum_{ij} T_{ij} \langle \boldsymbol{x}_i, \boldsymbol{y}_j \rangle$$

Positive definite

A kernel k is **positive definite** on $\mathcal{X}$ if for $n \in \mathbb{N}^*$, $x_1, .., x_n \in \mathcal{X}$ and $c_1, .., c_n \in \mathbb{R}$, we have:

$$\sum_{ij} c_i c_j k(x_i, x_j) \geq 0$$

or in short $c^T K c \geq 0$.

Optimal Transport Graph Neural Networks (9)

Conditionally negative definite

A kernel k is **conditionally negative definite** on $\mathcal{X}$ if for $n \in \mathbb{N}^*$, $x_1, .., x_n \in \mathcal{X}$ and $c_1, .., c_n \in \mathbb{R}$ such that $\sum_i c_i = 0$ we have:

$$\sum_{ij} c_i c_j k(x_i, x_j) \leq 0$$

or in short $c^T K c \leq 0$ for $c^T 1 = 0$.

Theorem

- The **similarity** Wasserstein kernel $\mathcal{W}_{dot}$ is **not** positive definite. (not good)
- The **discrepancy** Wasserstein kernel $\mathcal{W}_{L2}$ is **not** conditionally negative definite. (not really bad)

Nothing really good?

Optimal Transport Graph Neural Networks (10)

A subset $S \subset \mathcal{X} \subset \mathbb{R}$ is called **dense** in $\mathcal{X}$ is any real numbers can be arbitrarily well-approximated by elements of S.

Dense

A subset $S \subset \mathcal{X} \subset \mathbb{R}$ is called **dense** in $\mathcal{X}$ if for any $\epsilon > 0$ and $x \in \mathcal{X}$, there is some $s \in S$ such that $|x - s| < \epsilon$.

Dense (alternative)

A subset $S \subset \mathcal{X} \subset \mathbb{R}$ is called **dense** in $\mathcal{X}$ if for any $x \in \mathcal{X}$, there is a sequence $\{x_n\} \subset S$ such that:

$$\lim_{n \to \infty} x_n = x$$

Optimal Transport Graph Neural Networks (11)

Dense – Topological space

A subset S of a topological space $\mathcal{X}$ is called **dense** in $\mathcal{X}$ if every element $x \in \mathcal{X}$ either belongs to S or is a limit point of S.

An important property, **universality**, of a given kernel on a space $\mathcal{X}$ is whether simple functions defined on top of this kernel can **approximate** any continuous function on the same space.

Kolmogorov – Arnold representation theorem

Any multivariate continuous function can be represented as a superposition of one-dimensional functions, i.e.,

$$f(x_1, .., x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \Psi_{q,p}(x_p) \right)$$

Optimal Transport Graph Neural Networks (12)

Universal kernels

A kernel k defined on $\mathcal{X}_d^n$ is said to be **universal** if: for any compact subset $\mathcal{X} \subset \mathcal{X}_d^n$, the set of functions in the form

$$\sum_{j=1}^{m} \alpha_j \sigma(k(., \theta_j) + \beta_j), \qquad (m \in \mathbb{N}^*; \alpha_j, \beta_j \in \mathbb{R}; \theta_j \in \mathcal{X}_d^n)$$

is **dense** in the set $\mathcal{C}(\mathcal{X})$ of continuous functions from $\mathcal{X}$ to $\mathbb{R}$.

Theorem

1. The aggregation kernel (e.g. sum, averge) is **not** universal.
2. The **discrepancy** Wasserstein kernel $\mathcal{W}_{L2}$ is universal.

Paper 1

Wasserstein GAN

Martin Arjovsky, Soumith Chintala, and Léon Bottou

`https://arxiv.org/pdf/1701.07875.pdf`

Wasserstein GAN (1)

What does it mean to learn a probability distribution?

To learn a probability density. We define a parametric family of densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ and find the one that maximized the likelihood on our data $\{x^{(i)}\}_{i=1}^m$:

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

This amounts to minimizing the Kullback-Leibler divergence $\mathrm{KL}(\mathbb{P}_r || \mathbb{P}_\theta)$ where $\mathbb{P}_r$ is the real data distribution, and $\mathbb{P}_\theta$ is the distribution of the parameterized density P_θ.

Note

Our discussion about the Variational Principles:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___August_14__2020.pdf`

Wasserstein GAN (2)

Basic idea of GAN

1. Rather than estimating $\mathbb{P}_r$, we define a random variable Z with a fixed distribution $p(z)$ and pass it through a parameteric function $g_\theta : \mathcal{Z} \to \mathcal{X}$ that directly generates samples following a certain distribution $\mathbb{P}_\theta$.
2. By varying θ, we can change $\mathbb{P}_\theta$ to make it close to the real data distribution $\mathbb{P}_r$.

Advantage of GAN

1. This approach can represent distributions confined to a low dimensional manifold (Z can be low dimensional).
2. The ability to easily generate samples is often more useful than knowing the numerical value of the density.
3. It is computationally difficult to generate samples given an arbitrary high dimensional density.

Wasserstein GAN (3)

Scope of the paper

Various ways to define a distance or divergence $\rho(\mathbb{P}_\theta, \mathbb{P}_r)$ – how close the model distribution and the real distribution are.

- Earth Mover (EM) distance.
- Wasserstein GAN that minimizes an (efficient) approximation of the EM distance.
- WGANs is easier to train(!?).

Some definitions

- A sequence of distributions $(\mathbb{P}_t)_{t\in\mathbb{N}}$ converges if and only if there is a distribution $\mathbb{P}_\infty$ such that $\rho(\mathbb{P}_t, \mathbb{P}_\infty)$ tends to zero.
- A mapping $\theta \mapsto \mathbb{P}_\theta$ is called continuous, if when a sequence of parameters θ_t converges to θ, the distributions $\mathbb{P}_{\theta_t}$ also converge to $\mathbb{P}_\theta$.

Wasserstein GAN (4)

Notation

- $\mathcal{X}$: a compact metric set (e.g. the space of images $[0,1]^d$).
- Σ: the set of all the Borel subsets of $\mathcal{X}$.
- $\text{Prob}(\mathcal{X})$: the space of probability measures defined on $\mathcal{X}$.
- A probability distribution $\mathbb{P}_r \in \text{Prob}(\mathcal{X})$ admits a density $p_r(x)$ with respect to μ, that is:

$$\forall A \in \Sigma, \mathbb{P}_r(A) = \int_A p_r(x) d\mu(x)$$

iff it is absolutely continuous with respect to μ, that is:

$$\forall A \in \Sigma, \mu(A) = 0 \Rightarrow \mathbb{P}_r(A) = 0$$

Wasserstein GAN (5)

Total Variation (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

Kullback-Leibler (KL) divergence

$$\mathrm{KL}(\mathbb{P}_r || \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x)$$

- Assymetric
- Possibly infinite (e.g. when there are points such that $P_g(x) = 0$ and $P_r(x) > 0$).

Wasserstein GAN (6)

Jensen-Shannon (JS) divergence

This is a **symmetrical** extended version of KL:

$$\text{JS}(\mathbb{P}_r, \mathbb{P}_g) = \text{KL}(\mathbb{P}_r || \mathbb{P}_m) + \text{KL}(\mathbb{P}_g || \mathbb{P}_m)$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$.

f-divergences (Liese & Miescke, 2008)

$$\mathcal{D}_f(\mathbb{P}_r || \mathbb{P}_g) = \int f\left(\frac{P_r(x)}{P_g(x)}\right) P_g(x) d\mu(x)$$

where $f : (0, \infty) \to \mathcal{R}$ is any convex function satisfying $f(1) = 0$.

Wasserstein GAN (7)

Earth-Mover (EM) distance – Wasserstein-1

Kantorovich's formulation:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} ||x - y||$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively:

$$P_r(x) = \int \gamma(x, y) d\mu(y) \qquad P_g(y) = \int \gamma(x, y) d\mu(x)$$

Wasserstein GAN (8)

Theorem 1

Let $\mathbb{P}_r$ be a fixed distribution over $\mathcal{X}$. Let Z be a random variable (e.g. Gaussian) over another space $\mathcal{Z}$. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be a function, that will be denoted as $g_\theta(z)$. Let $\mathbb{P}_\theta$ denote the distribution of $g_\theta(Z)$. Then:

1. If g is continuous in θ, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.
2. If g is locally Lipschitz and satisfies regularity assumption 1, then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere, and differentiable almost everywhere.
3. Statement 1-2 are false for the Jensen-Shannon divergence $\mathrm{JS}(\mathbb{P}_r, \mathbb{P}_\theta)$ and all the KLs.

Wasserstein GAN (9)

Assumption 1

Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be locally Lipschitz between finite dimensional vector spaces. We say that g satisfies assumption 1 for a certain probability distribution p over $\mathcal{Z}$ if there are local Lipschitz constants $L(\theta, z)$ such that:

$$\mathbb{E}_{z \sim p}[L(\theta, z)] < \infty$$

Corollary 1

Let g_θ be any feedforward neural network parameterized by θ, and $p(z)$ a prior over z such that $\mathbb{E}_{z \sim p(z)}||z|| < \infty$ (e.g. Gaussian, uniform, etc.). Then assumption 1 is satisfied and therefore $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere and differentiable almost everywhere.

Wasserstein GAN (10)

Theorem 2

Let $\mathbb{P}$ be a distribution on a compact space $\mathcal{X}$ and $(\mathbb{P}_n)_{n\in\mathbb{N}}$ be a sequence of distributions on $\mathcal{X}$. Then, considering all limits as $n \to \infty$,

1. The following statements are equivalent
   - $\delta(\mathbb{P}_n, \mathbb{P}) \to 0$ with δ the total variation distance.
   - $\mathrm{JS}(\mathbb{P}_n, \mathbb{P}) \to 0$ with JS the Jensen-Shannon divergence.
2. The following statements are equivalent
   - $W(\mathbb{P}_n, \mathbb{P}) \to 0$
   - $\mathbb{P}_n \to_{\mathcal{D}} \mathbb{P}$ where $\to_{\mathcal{D}}$ represents convergence in distribution for random variables.
3. $\mathrm{KL}(\mathbb{P}_n||\mathbb{P}) \to 0$ or $\mathrm{KL}(\mathbb{P}||\mathbb{P}_n) \to 0$ imply the statements in (1).
4. The statements in (1) imply the statements in (2).

Wasserstein GAN (11)

Kantorovich – Rubinstein duality

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

$$K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

for some constant K.

Wasserstein GAN (12)

Missing a theorem?

Duality gives us:

$$K \cdot \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(x,y)\sim\gamma} ||x - y|| \equiv \sup_{||f||_L \leq K} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

Instead consider optimizing:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$$

where the parameterized family of functions $f_{w\,w\in\mathcal{W}}$ that are all K-Lipschitz for some K.

Wasserstein GAN (13)

Theorem 3

Let $\mathbb{P}_r$ be any distribution. Let $\mathbb{P}_\theta$ be the distribution of $g_\theta(Z)$ with Z a random variable with density p and g_θ a function satisfying assumption 1. Then, there is a solution $f : \mathcal{X} \to \mathbb{R}$ to the problem

$$\max_{||f||_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

and we have

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z \sim p(z)}[\nabla_\theta f(g_\theta(z))]$$

when both terms are well-defined.

Wasserstein GAN (14)

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Wasserstein GAN (15)

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Generative Adversarial Nets, Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, `https://arxiv.org/pdf/1406.2661.pdf`

Paper 2

Wasserstein Auto-Encoders (ICLR 2018)
Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf
`https://arxiv.org/pdf/1711.01558.pdf`

Wasserstein Auto-Encoders (1)

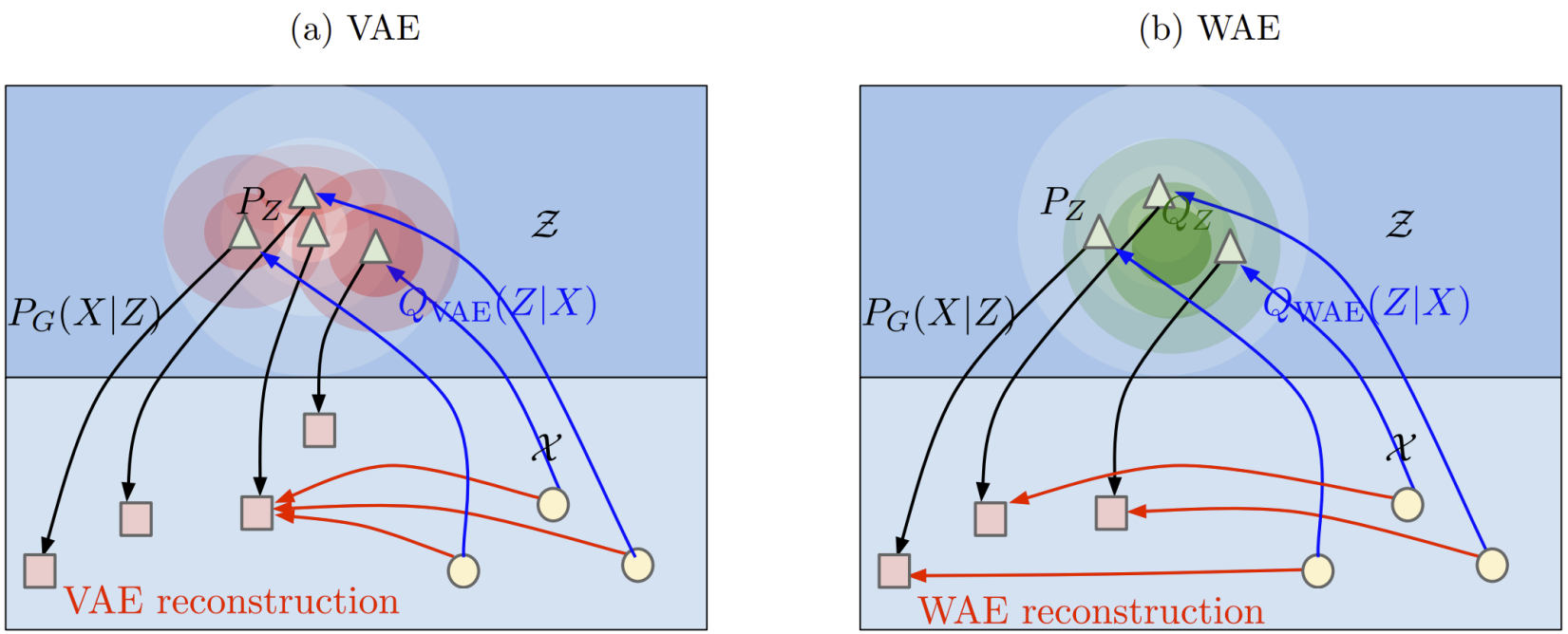


Figure 1: Both VAE and WAE minimize two terms: the reconstruction cost and the regularizer penalizing discrepancy between P_Z and distribution induced by the encoder Q. VAE forces $Q(Z|X = x)$ to match P_Z for all the different input examples x drawn from P_X. This is illustrated on picture (a), where every single red ball is forced to match P_Z depicted as the white shape. Red balls start intersecting, which leads to problems with reconstruction. In contrast, WAE forces the continuous mixture $Q_Z := \int Q(Z|X)dP_X$ to match P_Z, as depicted with the green ball in picture (b). As a result latent codes of different examples get a chance to stay far away from each other, promoting a better reconstruction.

Wasserstein Auto-Encoders (2)

A rich class of divergences between probability distributions is induced by the *optimal transport* (OT) problem **(Villani, 2003)**.

Kantorovich's formulation

$$W_c(P_X, P_G) = \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}[c(X, Y)]$$

where $c(x, y) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is any measurable cost function and $\mathcal{P}(X \sim P_X, Y \sim P_G)$ is a set of all joint distributions of (X, Y) with marginals P_X and P_G, respectively.

Kantorovich – Rubinstein duality

$$W_1(P_X, P_G) = \sup_{f \in \mathcal{F}_L} \mathbb{E}_{X\sim P_X}[f(X)] - \mathbb{E}_{Y\sim P_G}[f(Y)]$$

where $\mathcal{F}_L$ is the class of all bounded 1-Lipschitz functions on a metric space $(\mathcal{X}, d)$, and $c(x, y) = d(x, y)$.

Wasserstein Auto-Encoders (3)

P_G – latent variable models are defined by a two-step procedure:

1. a code Z is sampled from a fixed distribution P_Z on a latent space $\mathcal{Z}$.
2. Z is mapped to the image $X \in \mathcal{X} = \mathbb{R}^d$.

$$p_G(x) = \int_{\mathcal{Z}} p_G(x|z)p_z(z)dz, \qquad \forall x \in \mathcal{X}$$

Theorem 1

For any function $G : \mathcal{Z} \to \mathcal{X}$:

$$\inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}[c(X, Y)] = \inf_{Q: Q_Z = P_Z} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}[c(X, G(Z))]$$

where Q_Z is the marginal distribution of Z when $X \sim P_X$ and $Z \sim Q(Z|X)$.

Theorem 1 allows us to optimize over encoder $Q(Z|X)$ instead of optimizing over all couplings between X and Y.

Wasserstein Auto-Encoders (4)

WAE objective (plus penalty):

$$\mathcal{D}_{\text{WAE}}(P_X, P_G) = \inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}[c(X, G(Z))] + \lambda \cdot \mathcal{D}_Z(Q_Z, P_Z)$$

where $\mathcal{Q}$ is any nonparametric set of probabilistic encoders, $\mathcal{D}_Z$ is an arbitrary divergence between Q_Z and P_Z, and $\lambda > 0$ is a hyperparameter.

1. **GAN-based** $\mathcal{D}_Z$: Choose $\mathcal{D}_Z(Q_Z, P_Z) = \mathcal{D}_{\text{JS}}(Q_Z, P_Z)$ and use adversarial training.
2. **MMD-based** $\mathcal{D}_Z$ (maximum mean discrepancy):

$$\text{MMD}_k(P_Z, Q_Z) = \left\| \int_{\mathcal{Z}} k(z, .)dP_Z(z) - \int_{\mathcal{Z}} k(z, .)dQ_z(z) \right\|_{\mathcal{H}_k}$$

where $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is a positive-definite kernel, and $\mathcal{H}_k$ is the RKHS of real-valued functions mapping $\mathcal{Z}$ to $\mathbb{R}$.

Wasserstein Auto-Encoders (5)

Algorithm 1 Wasserstein Auto-Encoder with GAN-based penalty (WAE-GAN).

Require: Regularization coefficient $\lambda > 0$.

Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.

while (ϕ, θ) not converged **do**

Sample $\{x_1, \dots, x_n\}$ from the training set

Sample $\{z_1, \dots, z_n\}$ from the prior P_Z

Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$

Update D_γ by ascending:

$$\frac{\lambda}{n} \sum_{i=1}^{n} \log D_\gamma(z_i) + \log\big(1 - D_\gamma(\tilde{z}_i)\big)$$

Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) - \lambda \cdot \log D_\gamma(\tilde{z}_i)$$

end while

Algorithm 2 Wasserstein Auto-Encoder with MMD-based penalty (WAE-MMD).

Require: Regularization coefficient $\lambda > 0$, characteristic positive-definite kernel k.

Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.

while (ϕ, θ) not converged **do**

Sample $\{x_1, \dots, x_n\}$ from the training set

Sample $\{z_1, \dots, z_n\}$ from the prior P_Z

Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$

Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(z_\ell, z_j) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(\tilde{z}_\ell, \tilde{z}_j) - \frac{2\lambda}{n^2} \sum_{\ell, j} k(z_\ell, \tilde{z}_j)$$

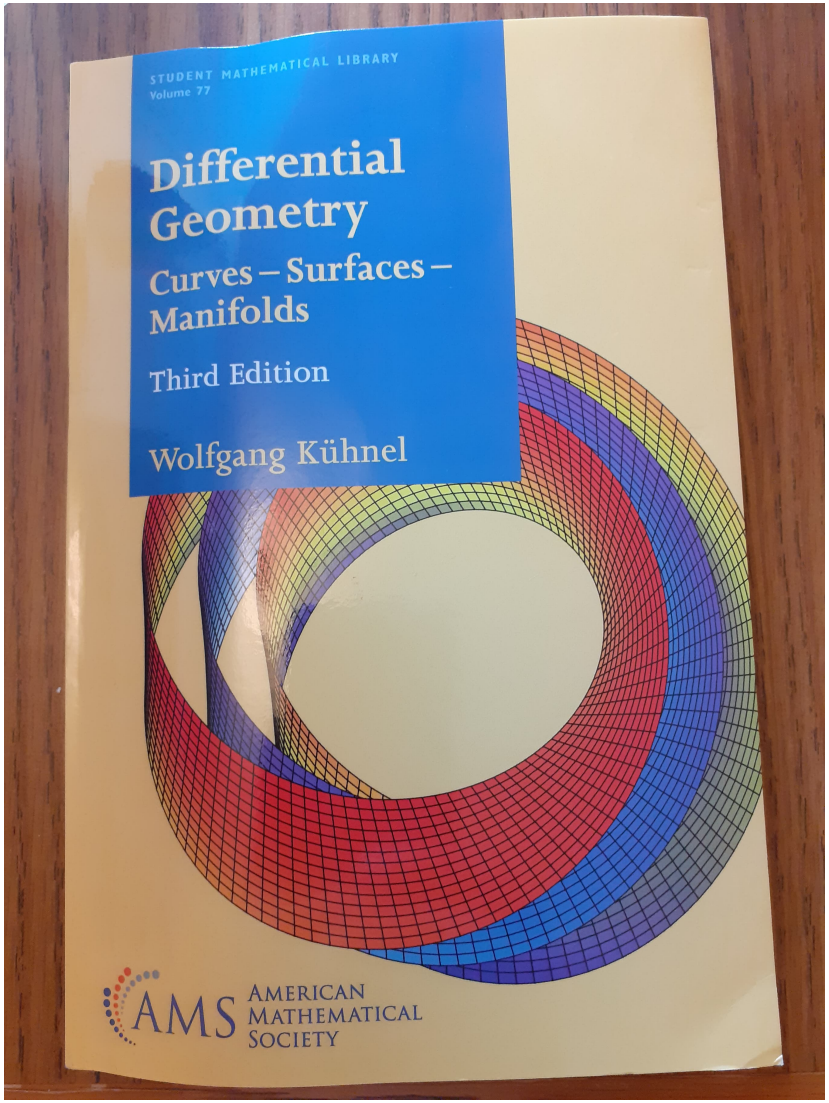
end while

We point out once again that the encoders $Q_\phi(Z|x)$ in Algorithms 1 and 2 can be non-random, i.e. deterministically mapping input points to the latent codes. In this case $Q_\phi(Z|x) = \delta_{\mu_\phi(x)}$ for a function $\mu_\phi \colon \mathcal{X} \to \mathcal{Z}$ and in order to *sample* $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ we just need to return $\mu_\phi(x_i)$.

Paper 5

Poincaré Wasserstein Autoencoder (NeurIPS 2018 Bayesian workshop)
Ivan Ovinnikov
`http://bayesiandeeplearning.org/2018/papers/18.pdf`

Poincaré Wasserstein Autoencoder (1)



Monge is also regarded as the father of Differential Geometry

Poincaré Wasserstein Autoencoder (2)

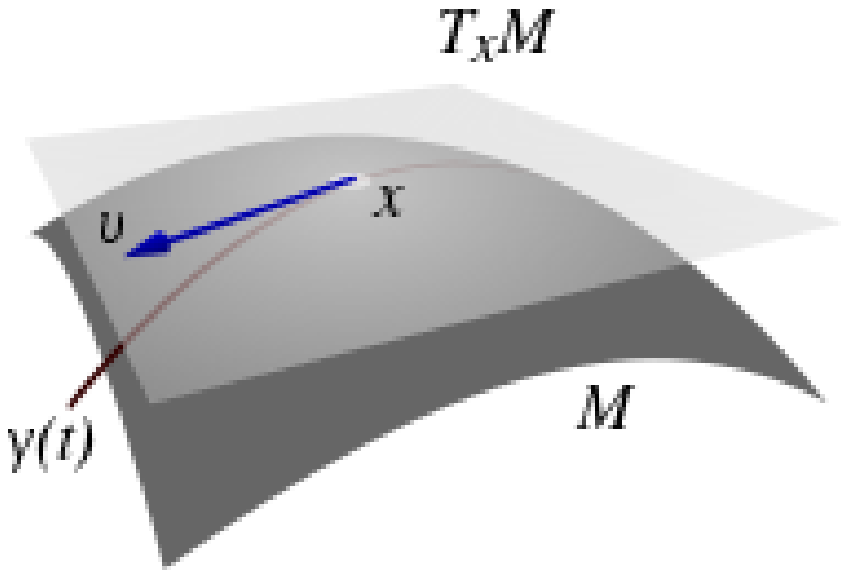
Proposals

- Reformulation of Wasserstein autoencoder on a **non-Euclidean manifold**, the Poincaré ball model of the hyperbolic space $\mathbb{H}^n$.
- The use of hyperbolic spaces has been shown to yield improved results on datasets which present a **hierarchical tree-like structure** such as word ontologies.
- By assuming the latent space to be hyperbolic, we can use its intrinsic hierarchy to impose structure on the learned latent space representations.

Poincaré Wasserstein Autoencoder (3)

Notations:

- For every point x on a given manifold $\mathcal{M}$, a **tangent space** $\mathcal{T}_x\mathcal{M}$ is defined as a first order approximation of $\mathcal{M}$ at point x. The **exponential map** $\exp_x(v)$ is the project of a vector v on the tangent space at point x into the corresponding point on the manifold.



- **Riemannian metric** g is a collection of inner products $\mathcal{T}_x\mathcal{M} \times \mathcal{T}_x\mathcal{M} \to \mathbb{R}$. A **geodesic** is a smooth curve $y(t)$ which corresponds to the shortest distance between two points on a manifold.

Poincaré Wasserstein Autoencoder (4)

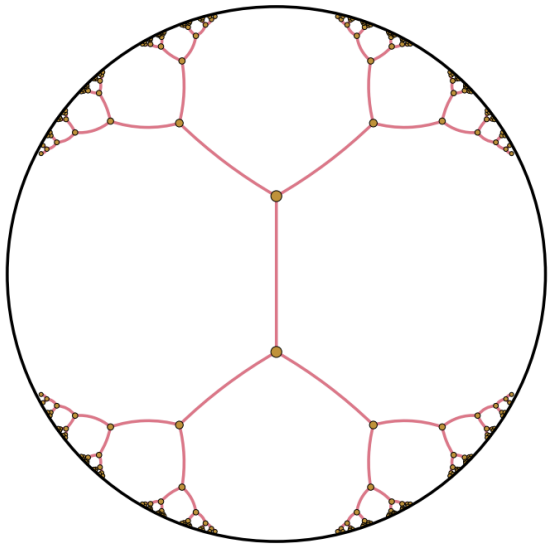


Figure 1: A regular tree isometrically embedded in the Poincaré disc. Red curves are same length *geodesics*, i.e. "straight lines".

ry

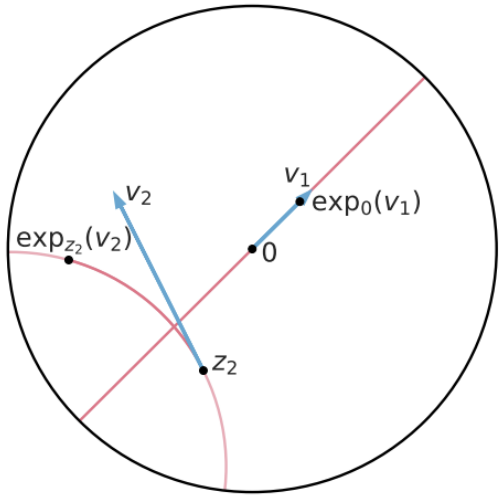


Figure 2: Geodesics and exponential maps in the Poincaré disc.

Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders (NeurIPS 2019)

`https://arxiv.org/pdf/1901.06033.pdf`

Poincaré Wasserstein Autoencoder (5)

Poincaré ball model (of hyperbolic space)

The Poincaré ball model is defined by the tuple $(\mathcal{B}_n, g_H)$:

$$\mathcal{B}_n = \{\boldsymbol{x} \in \mathbb{R}^n | ||\boldsymbol{x}|| < 1\}, \qquad g_H = \frac{4}{(1 - ||x||^2)^2}$$

Input: maximum radius r_{max}, dimensionality d, hyperbolic prior likelihood $f_{\mathcal{H}}(\mathbf{x}|\mathbf{0}, \mathbf{I}_2)$
Result: $k = nd$ samples from prior $f_{\mathcal{H}}(\mathbf{z})$
while $i < k$ **do**
  sample angle $\phi \sim \mathcal{U}(0, 2\pi)$, sample $a \sim \mathcal{U}(0, 1)$;
  get radius sample $r = \text{acosh}(1 + a(\cosh r_{max} - 1))$;
  generate pairs $\mathbf{x}_i = (\sinh r \cos \phi, \sinh r \sin \phi)$;
  evaluate $p(\mathbf{x}_i) = f_{\mathcal{H}}(\mathbf{x}_i)$;
  $M = \max(p_i)$;
  sample $u \sim \mathcal{U}(0, 1)$;
  if $u < \frac{p_i}{M}$ **then**
    accept sample $\mathbf{x}_i$;
  else
    reject sample;
  end
end
Output: stack d dimensions from $\mathbb{H}^2$-samples: $\mathbf{s} = [s_1, s_2, ..., s_d]$;

Algorithm 1: Prior sampling on Poincaré Ball

Poincaré Wasserstein Autoencoder (6)

Gaussian distribution in $\mathbb{H}^n$: The Gaussian PDF in hyperbolic space is defined analogously to the one in the Euclidean space

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\sigma}) = \frac{1}{Z(\boldsymbol{\sigma})} e^{-d^2(\boldsymbol{x},\boldsymbol{\mu})/2\boldsymbol{\sigma}^2}$$

$$Z(\boldsymbol{\sigma}) = 2\pi\sqrt{\frac{\pi}{2}} e^{\boldsymbol{\sigma}^2/2} \text{erf}\left(\frac{\boldsymbol{\sigma}}{2}\right)$$

where $d(\boldsymbol{x}, \boldsymbol{\mu})$ is the geodesic distance between two points $\boldsymbol{x}, \boldsymbol{\mu} \in \mathbb{H}^n$:

$$d(\boldsymbol{x}, \boldsymbol{\mu}) = \text{arccosh}\left(1 + 2\frac{||\boldsymbol{x} - \boldsymbol{\mu}||^2}{(1 - ||\boldsymbol{x}||^2)(1 - ||\boldsymbol{\mu}||^2)}\right)$$

Poincaré Wasserstein Autoencoder (7)

Posterior sampling (hyperbolic reparameterization) is done as a diagonal matrix-gyrovector multiplication:

$$\boldsymbol{z} = \mu_{\mathcal{H}}(\boldsymbol{x}) \oplus \text{diag}(\sigma_{\mathcal{H}}(\boldsymbol{x}))^{\otimes} \boldsymbol{z}_{\text{prior}}$$

where:

- $\oplus$ is the gyrovector addition:

$$\boldsymbol{x} \oplus \boldsymbol{y} = \frac{(1 + 2c\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c||y||^2)\boldsymbol{x} + (1 - c||x||^2)\boldsymbol{y}}{1 + 2c\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c^2||\boldsymbol{x}||^2||y||^2}$$

- $\otimes$ is the Hadamard product:

$$M^{\otimes}\boldsymbol{x} = \frac{1}{\sqrt{c}} \tanh\left(\frac{||M\boldsymbol{x}||}{||\boldsymbol{x}||} \text{arctanh}(\sqrt{c}||\boldsymbol{x}||) \right) \frac{M\boldsymbol{x}}{||M\boldsymbol{x}||}$$

- $\boldsymbol{z}_{\text{prior}}$ is sampled by rejection sampling (as mentioned in the algorithm above).

Poincaré Wasserstein Autoencoder (8)

The WAE objective is derived from the optimal transport cost by relaxing the constraint on the posterior q:

$$\mathcal{L}_{WAE} = \inf_{q_\phi(\boldsymbol{z}|\boldsymbol{x}) \in \mathcal{Q}} \mathbb{E}_{p(\boldsymbol{x})} \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})} (\log p(\boldsymbol{x}|\boldsymbol{z})) + \beta \mathcal{D}_{\text{MMD}}$$

where the Maximum Mean Discrepancy (MMD) loss function is defined over two probability meatures p and q in an RKHS unit all $\mathcal{F}$ as:

$$\mathcal{D}_{\text{MMD}}(p(\boldsymbol{z}), q_\phi(\boldsymbol{z})) = \left\| \int_{\mathcal{Z}} k(\boldsymbol{z}, .) dp(\boldsymbol{z}) - \int_{\mathcal{Z}} k(\boldsymbol{z}, .) dq(\boldsymbol{z}) \right\|_{\mathcal{F}}$$

where $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\lambda d_{\mathcal{H}}(\boldsymbol{x}, \boldsymbol{y}))$ is the Laplacian kernel.

Poincaré Wasserstein Autoencoder (9)

$\mathcal{D}_{\text{MMD}}$ can be estimated (finitely) on minibatch samples from the prior $\boldsymbol{z} \sim p(\boldsymbol{z})$ (as in the algorithm) and the posterior samples $\bar{\boldsymbol{z}} \sim q_\phi(\boldsymbol{z})$:

$$\mathcal{D}^{(B)}_{\text{MMD}}(p(\boldsymbol{z}), q_\phi(\boldsymbol{z})) = \frac{\lambda}{n(n-1)} \sum_{i \neq j} k(\boldsymbol{z}_i, \boldsymbol{z}_j) +$$

$$+ \frac{\lambda}{n(n-1)} \sum_{i \neq j} k(\bar{\boldsymbol{z}}_i, \bar{\boldsymbol{z}}_j) + \frac{2\lambda}{n^2} \sum_{i,j} k(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_j)$$

Parameters updates are performed by **Riemannian Stochastic Gradient Descent** (RSGD).

Poincaré Wasserstein Autoencoder (10)

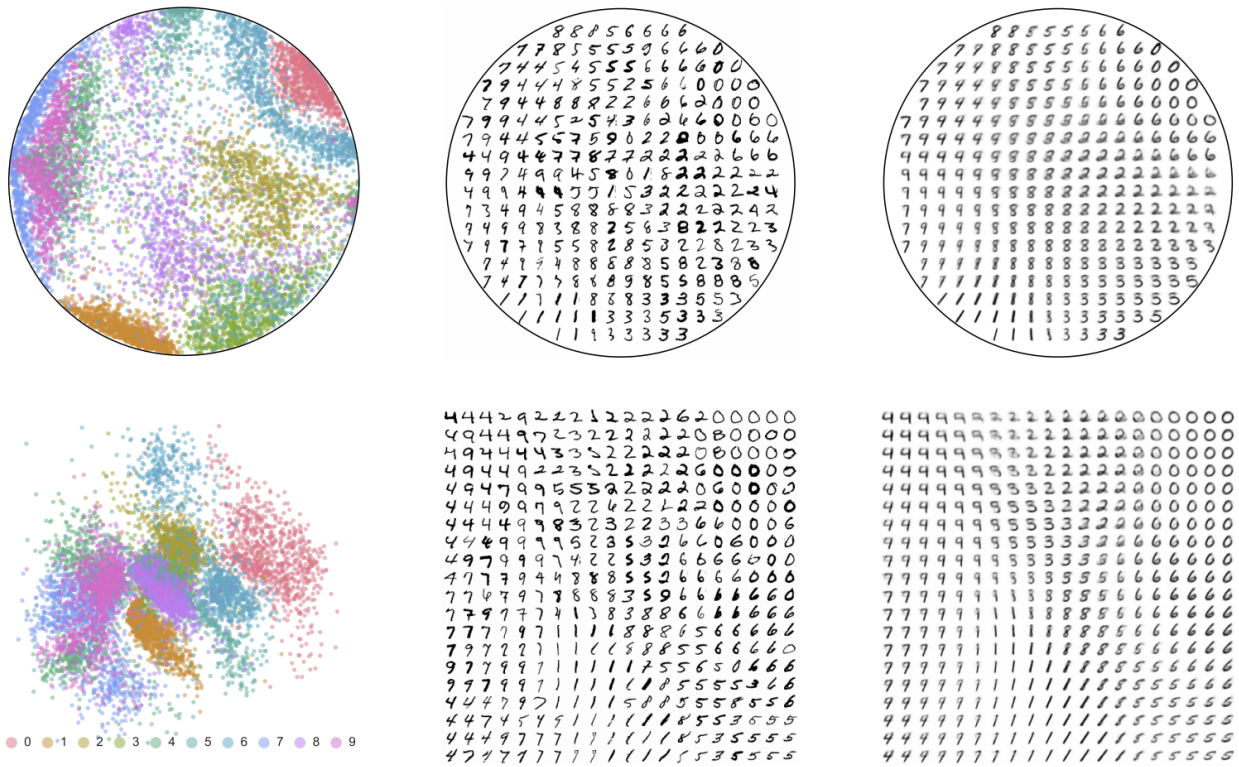


Figure 7: MNIST Posteriors mean (Left) sub-sample of digit images associated with posteriors mean (Middle) Model samples (Right) – for $\mathcal{P}^{1.4}$-VAE (Top) and $\mathcal{N}$-VAE (Bottom).

Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders (NeurIPS 2019)

https://arxiv.org/pdf/1901.06033.pdf