Computational Optimal Transport for Machine and Deep Learning

Introduction to domain adaptation

Mathurin Massias, Titouan Vayer, Quentin Bertrand.

December 11, 2024

ENS DE LYON

Table of contents

Acknowledgments

Slides adapted from those of Rémi Flamary

Euclidean to Fréchet barycenter

Let $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$ (histogram).

Standard barycenter

$$\hat{\mathbf{x}} = \sum_{i=1}^{N} \lambda_i \mathbf{x}_i = \underset{\bar{\mathbf{x}} \in \mathbb{R}^d}{\arg\min} \sum_{i=1}^{N} \lambda_i \|\bar{\mathbf{x}} - \mathbf{x}_i\|_2^2 \, . \tag{1}$$

Euclidean to Fréchet barycenter

Let $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$ (histogram).

Standard barycenter

$$\hat{\mathbf{x}} = \sum_{i=1}^{N} \lambda_i \mathbf{x}_i = \arg\min_{\overline{\mathbf{x}} \in \mathbb{R}^d} \sum_{i=1}^{N} \lambda_i \|\overline{\mathbf{x}} - \mathbf{x}_i\|_2^2 \,. \tag{1}$$

Median barycenter

$$\hat{\mathbf{x}} = \arg\min_{\overline{\mathbf{x}} \in \mathbb{R}^d} \sum_{i=1}^{N} \lambda_i \|\overline{\mathbf{x}} - \mathbf{x}_i\|_2 \,. \tag{2}$$

Euclidean to Fréchet barycenter

Let $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$ (histogram).

Standard barycenter

$$\hat{\mathbf{x}} = \sum_{i=1}^{N} \lambda_i \mathbf{x}_i = \arg\min_{\overline{\mathbf{x}} \in \mathbb{R}^d} \sum_{i=1}^{N} \lambda_i \|\overline{\mathbf{x}} - \mathbf{x}_i\|_2^2 \,. \tag{1}$$

Median barycenter

$$\hat{\mathbf{x}} = \arg\min_{\overline{\mathbf{x}} \in \mathbb{R}^d} \sum_{i=1}^{N} \lambda_i \|\overline{\mathbf{x}} - \mathbf{x}_i\|_2 \,. \tag{2}$$

Fréchet barycenter

$\mathbf{x}_1, \cdots, \mathbf{x}_N \in X^N$ where (X, d) metric space.

$$\hat{\mathbf{x}} = \arg\min_{\overline{\mathbf{x}} \in X} \sum_{i=1}^{N} \lambda_i d^2(\overline{\mathbf{x}}, \mathbf{x}_i) \,. \tag{3}$$

Wasserstein barycenter

Let $\alpha_1, \cdots, \alpha_N \in \mathcal{P}(\mathbb{R}^d)$ probability measures and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$.

Wasserstein barycenter

It is a probability measure $\hat{\mu}$ solving

$$\hat{\mu} = \underset{\overline{\mu} \in \mathcal{P}(\mathbb{R}^d)}{\arg\min} \sum_{i=1}^{N} \lambda_i W_2^2(\overline{\mu}, \alpha_i) \,. \tag{4}$$

Wasserstein barycenter

Let $\alpha_1, \cdots, \alpha_N \in \mathcal{P}(\mathbb{R}^d)$ probability measures and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$.

Wasserstein barycenter

It is a probability measure $\hat{\mu}$ solving

$$\hat{\mu} = \underset{\overline{\mu} \in \mathcal{P}(\mathbb{R}^d)}{\arg\min} \sum_{i=1}^{N} \lambda_i W_2^2(\overline{\mu}, \alpha_i)\,. \tag{4}$$

Discrete case when $N = 2$: Mccan's interpolant

When $\alpha_1 = \sum_{i=1}^n a_i \delta_{\mathbf{x}_i}$ (source), $\alpha_2 = \sum_{j=1}^m b_j \delta_{\mathbf{y}_j}$ (target) are discrete. If P is an optimal coupling $\hat{\mu} = \sum_{ij} P_{ij} \delta_{(1-t)\mathbf{x}_i + t\mathbf{y}_j}$: $n + m - 1$ points.

Wasserstein barycenter

Let $\alpha_1, \cdots, \alpha_N \in \mathcal{P}(\mathbb{R}^d)$ probability measures and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$.

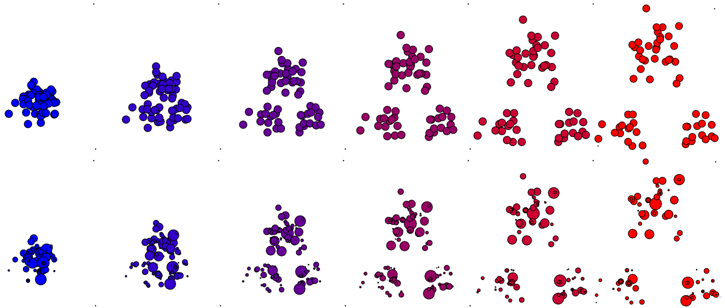
Wasserstein barycenter

It is a probability measure $\hat{\mu}$ solving

$$\hat{\mu} = \underset{\overline{\mu} \in \mathcal{P}(\mathbb{R}^d)}{\arg\min} \sum_{i=1}^{N} \lambda_i W_2^2(\overline{\mu}, \alpha_i) \,. \tag{4}$$

Discrete case when $N = 2$: Mccan's interpolant

When $\alpha_1 = \sum_{i=1}^n a_i \delta_{\mathbf{x}_i}$ (source), $\alpha_2 = \sum_{j=1}^m b_j \delta_{\mathbf{y}_j}$ (target) are discrete. If P is an optimal coupling $\hat{\mu} = \sum_{ij} P_{ij} \delta_{(1-t)\mathbf{x}_i + t\mathbf{y}_j}$: $n + m - 1$ points.



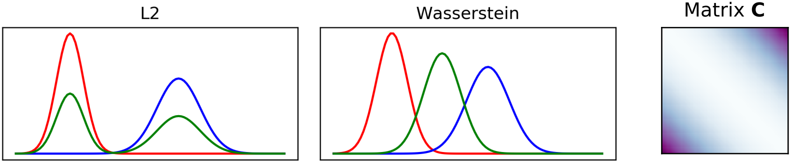
Wasserstein barycenter

Let $\alpha_1, \cdots, \alpha_N \in \mathcal{P}(\mathbb{R}^d)$ probability measures and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$.

Wasserstein barycenter

It is a probability measure $\hat{\mu}$ solving

$$\hat{\mu} = \underset{\overline{\mu} \in \mathcal{P}(\mathbb{R}^d)}{\arg\min} \sum_{i=1}^{N} \lambda_i W_2^2(\overline{\mu}, \alpha_i)\,. \tag{4}$$



Wasserstein barycenter

Let $\alpha_1, \cdots, \alpha_N \in \mathcal{P}(\mathbb{R}^d)$ probability measures and $(\lambda_1, \cdots, \lambda_N) \in \Sigma_N$.

Wasserstein barycenter

It is a probability measure $\hat{\mu}$ solving

$$\hat{\mu} = \underset{\overline{\mu} \in \mathcal{P}(\mathbb{R}^d)}{\arg\min} \sum_{i=1}^{N} \lambda_i W_2^2(\overline{\mu}, \alpha_i)\,. \tag{4}$$

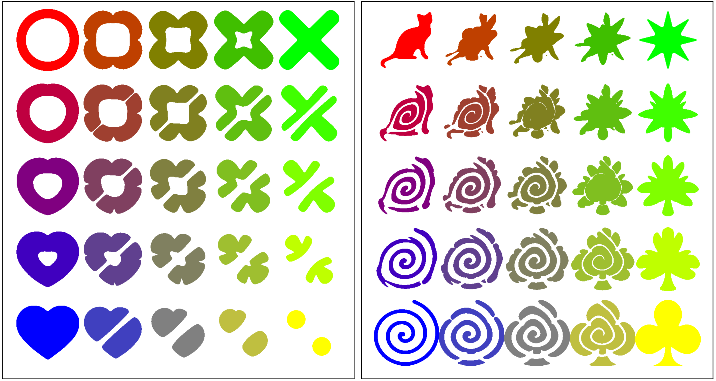


Figure: Peyré, Cuturi, et al. 2019

The barycentric mapping

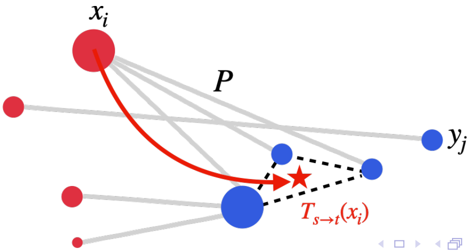
Let $\mu_s = \sum_{i=1}^{n} a_i \delta_{\mathbf{x}_i}$ (source), $\mu_t = \sum_{j=1}^{m} b_j \delta_{\mathbf{y}_j}$ (target). Let P be optimal coupling between μ_s, μ_t with cost c.

Weighted barycenter with OT plan

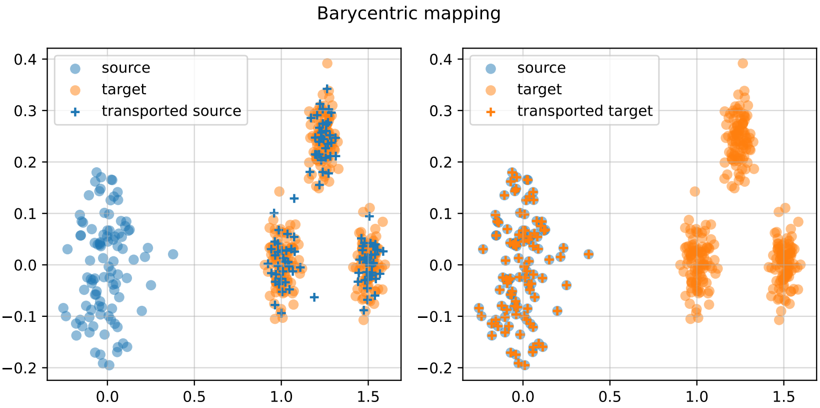
- Source to target

$$T_{s\to t} : \mathbf{x}_i \to \arg\min_{\bar{\mathbf{y}}} \sum_{j=1}^{m} P_{ij} c(\bar{\mathbf{y}}, \mathbf{y}_j) \tag{5}$$

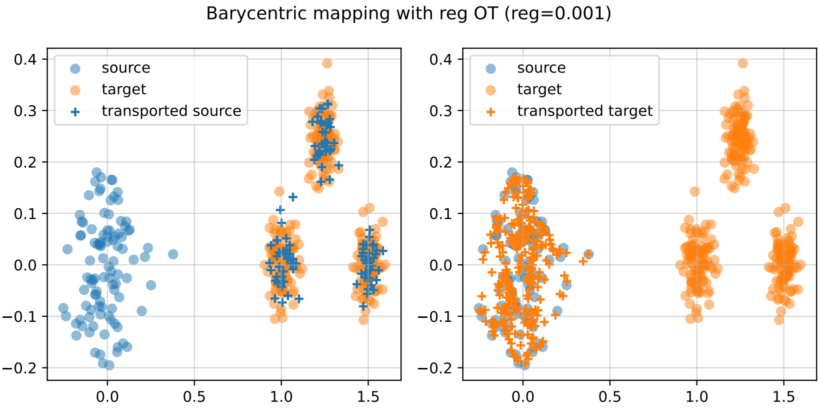
- When $c = \ell_2^2$, mapping the entire data $T_{s\to t}(\mathbf{X}) = \mathrm{diag}(P1_m)^{-1} P\mathbf{Y}$.
- If $P = ab^\top$, $T_{s\to t}(\mathbf{x}_i) = \sum_{j=1}^{m} b_j \mathbf{y}_j$.



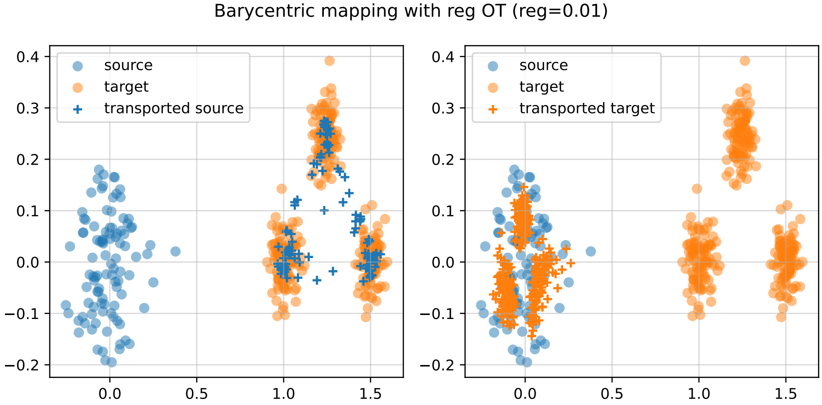
The barycentric mapping



The barycentric mapping



The barycentric mapping



The barycentric mapping

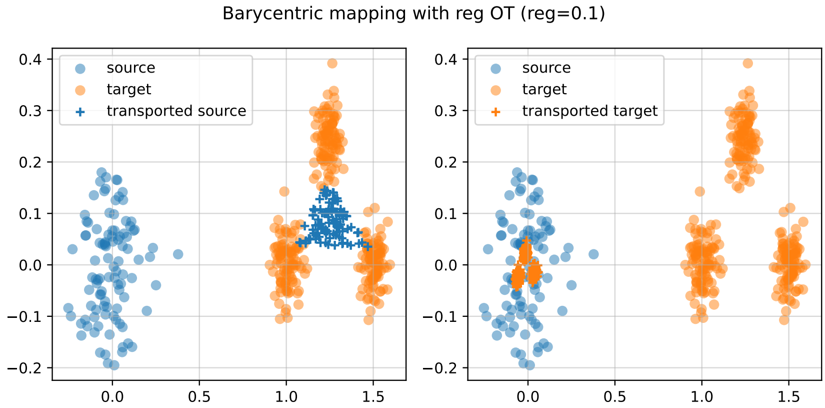


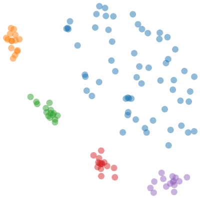
Table of contents

Supervised ML

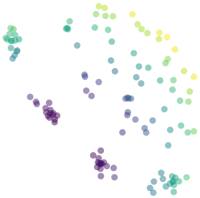
Samples + labels:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

Classification



Regression



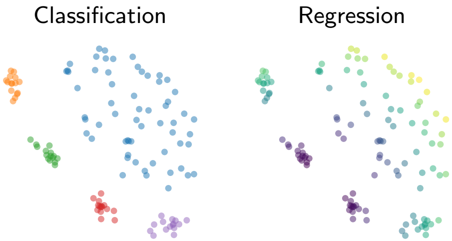
Supervised learning

- The dataset contains the samples $(\mathbf{x}_i, c_i)_{i=1}^n$ where $\mathbf{x}_i$ is the feature sample and c_i its label/class.
- The values to predict (label) can be concatenated in a vector $\mathbf{c}$

Supervised ML

Samples + labels:

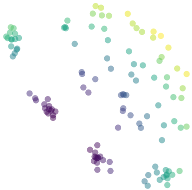
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$



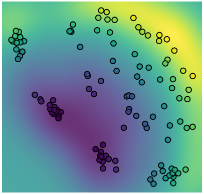
Supervised learning

- The dataset contains the samples $(\mathbf{x}_i, c_i)_{i=1}^n$ where $\mathbf{x}_i$ is the feature sample and c_i its label/class.
- The values to predict (label) can be concatenated in a vector $\mathbf{c}$
- Semi-supervised learning: few labeled points are available, but a large number of unlabeled points are given.

Regression



$\Longrightarrow$

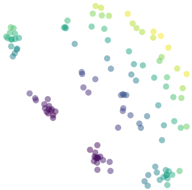


Objective

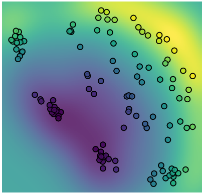
$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- Train a function $f(\mathbf{x}) = c \in \mathbb{R}$ predicting a continuous value.
- Can be extended to multi-value prediction ($\mathbb{R}^p$).

Regression



$$\Rightarrow$$



Objective

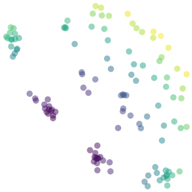
$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- Train a function $f(\mathbf{x}) = c \in \mathbb{R}$ predicting a continuous value.
- Can be extended to multi-value prediction ($\mathbb{R}^p$).

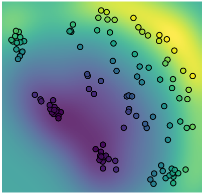
Hyperparameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Regression



$\Longrightarrow$



Objective

$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- Train a function $f(\mathbf{x}) = c \in \mathbb{R}$ predicting a continuous value.
- Can be extended to multi-value prediction ($\mathbb{R}^p$).

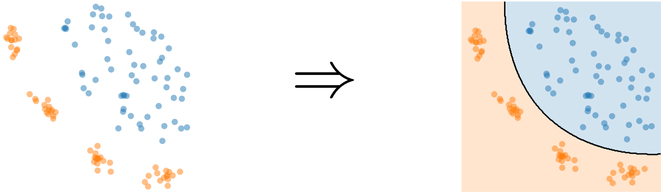
Hyperparameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Least Square (LS).
- Ridge regression, Lasso.
- Kernel regression.
- Deep learning.

Binary classification

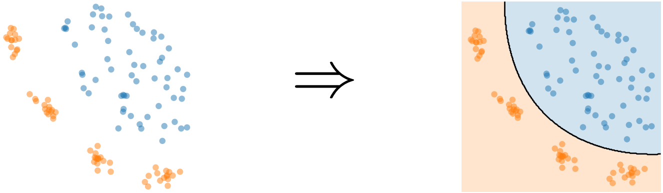


Objective

$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \to \{-1, 1\}$$

- Train a function $f(\mathbf{x}) = c \in \mathcal{C}$ predicting a binary value ($e.g.\{-1, 1\}$).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

Binary classification



Objective

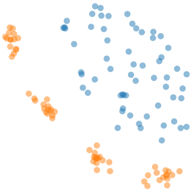
$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \{-1, 1\}$$

- Train a function $f(\mathbf{x}) = c \in \mathcal{C}$ predicting a binary value ($e.g.\{-1, 1\}$).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

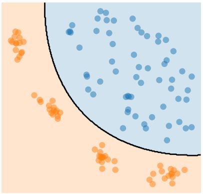
Hyperparameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Binary classification



$\Longrightarrow$



Objective

$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \{-1, 1\}$$

- Train a function $f(\mathbf{x}) = c \in \mathcal{C}$ predicting a binary value ($e.g.\{-1, 1\}$).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

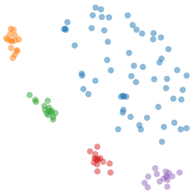
Hyperparameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

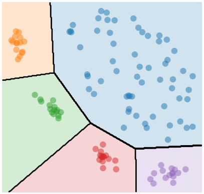
Methods

- Bayesian classifier (LDA, QDA)
- Linear and kernel discrimination
- Decision trees, random forests.
- Deep learning.

Multiclass classification



$\Rightarrow$



Objective

$$(\mathbf{x}_i, c_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \to \{1, \ldots, K\}$$

- Train a function $f(\mathbf{x}) = c \in \{1, \ldots, K\}$ predicting an integer value.

Empirical risk minimization

Minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n} \sum_{i=1}^{n} \ell(c_i, f(\mathbf{x}_i)) \tag{ERM}$$

Empirical risk minimization

Minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n} \sum_{i=1}^{n} \ell(c_i, f(\mathbf{x}_i)) \tag{ERM}$$

- ℓ is a loss function

$$\ell(\text{true value, predicted value}) = \text{how good is my prediction}$$

- It is called **empirical risk minimization (ERM)**
- Given the loss, finds the "best" f on the training data
- E.g. linear regression

Table of contents

Domain adaptation

Traditional supervised learning

- We want to learn predictor such that $c \approx f(\mathbf{x})$.
- Actual $p(x, c)$ unknown.
- We have access to training dataset $(\mathbf{x}_i, c_i)_{i=1,\ldots,n}$ $(\hat{p}(x, c))$.
- We choose a loss function $\ell(c, f(\mathbf{x}))$ that measure the discrepancy.

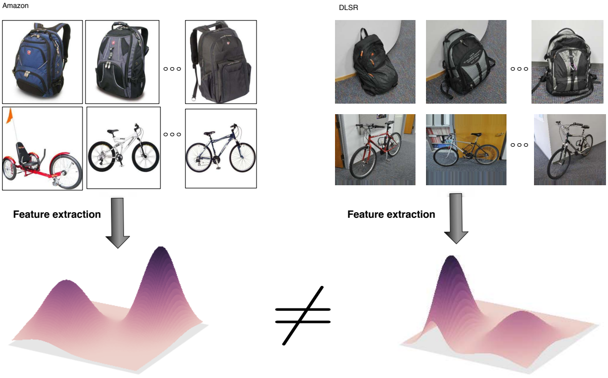
Empirical risk minimization

We week for a predictor f minimizing

$$\min_f \left\{ \mathbb{E}_{(\mathbf{x},c)\sim\hat{p}(x,c)} \; \ell(c, f(\mathbf{x})) = \sum_j \ell(c_j, f(\mathbf{x}_j)) \right\} \tag{6}$$

- Well known generalization results for predicting on new data.

Domain Adaptation problem

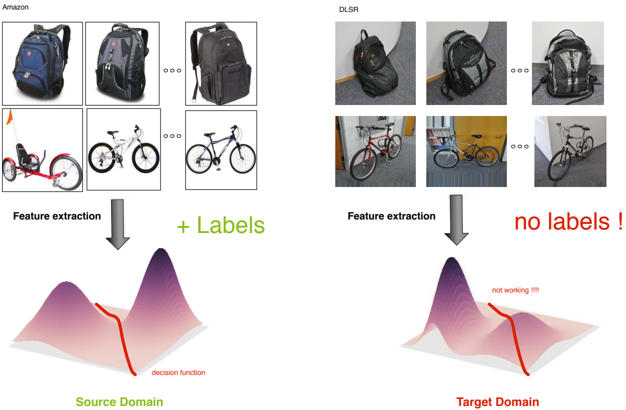


Probability Distribution Functions over the domains

Our context

- Classification problem with data coming from different sources (domains).
- Distributions are different but related.

Unsupervised domain adaptation problem

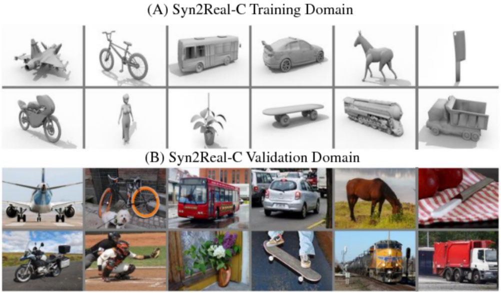


Problems

- Labels only available in the **source domain**, and classification is conducted in the **target domain**.
- Classifier trained on the source domain data performs badly in the target domain

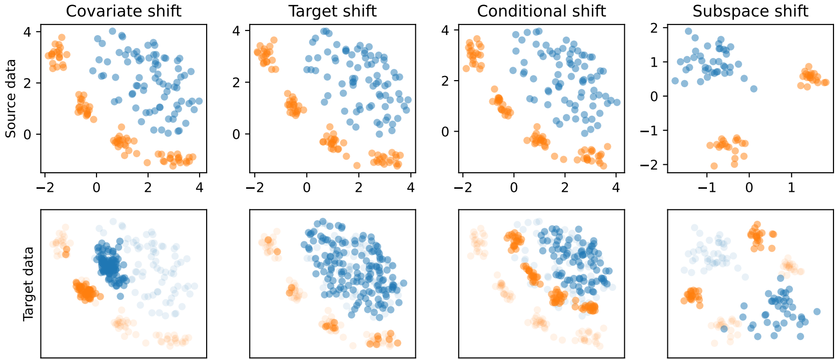
Is Domain Adaptation a real problem ?

- Ubiquitous problem in Deep Learning ! People can not afford to label billions of data for every single problems
- Novel interesting challenges if one considers learning from synthetic data



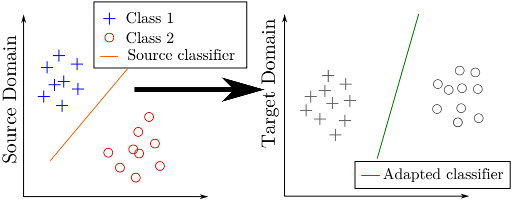
The pig picture

Many shifts are possible.



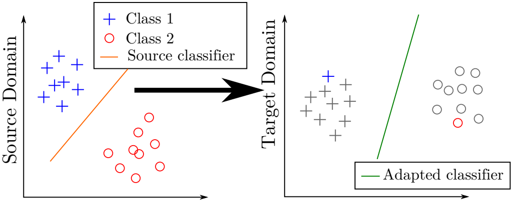
Unsupervised and semi-supervised DA

Unsupervised DA



- Source : $\{\mathbf{x}_i^s, c_i^s\}_{i=1}^{n_s}$
- Target : $\{\mathbf{x}_j^t\}_{j=1}^{n_t}$
- Requires assumptions on the shift (CS, TS, CD, SSB).

Semi-Supervised DA



- Source : $\{\mathbf{x}_i^s, c_i^s\}_{i=1}^{n_s}$
- Target : $\{\mathbf{x}_j^t\}_{j=1}^{n_t}$, $\{c_j^t\}_{j=1}^{n_l}$
- The few $n_l \ll n_t$ labeled target samples can help guide the learning on target.

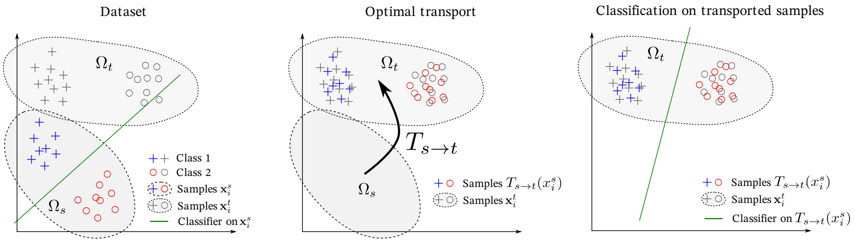
Domain adaptation

Problem: how to learn a classifier that can be good on several domains with only labels in one of the domain ?

- Theory Mansour, Mohri, and Rostamizadeh 2009 measures the difficulty of this task in terms of discrepancy of the representations of the data.
- Possible solutions include:
  - Find domain invariant representation of the data.
  - Transform data from one domain into "similar" versions in the other domain (adversarial methods).
  - At any point a notion of divergence between the distributions is involved.

Table of contents

Optimal transport for domain adaptation



Assumptions

1. There exist an OT mapping T in the feature space between the two domains.
2. The transport preserves the joint distributions:

$$P^s(\mathbf{x}, c) = P^t(T(\mathbf{x}), c).$$

3-step strategy Courty et al. 2016

1. Estimate optimal transport between distributions (use regularization).
2. Transport the training samples on target domain.
3. Learn a classifier on the transported training samples.

Label propagation

4-step strategy Redko et al. 2019

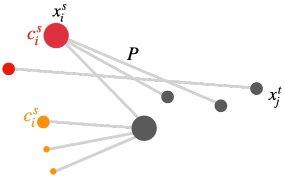
1. One-hot encoding of the classes in the source domain. E.g. if K classes $\{1, 2, \cdots, K\}$

$$c_i^s = 2 \rightarrow \mathbf{c}_i^s = \overbrace{(0, 1, \cdots, 0)}^{K}$$

2. Find a good OT plan P between source and target.
3. Propagate the labels of the source into the target.

$$\forall j \in [n_t],\ \widehat{\mathbf{c}}_j^t = \frac{1}{b_j} \sum_{i=1}^{n_s} P_{ij} \mathbf{c}_i^s = T_{t \rightarrow s}(\mathbf{c}_i^s)\,.$$

4. (optional) Find the class with maximal coordinate for prediction. E.g. $\widehat{\mathbf{c}}_j^t = (0.1, 0.8, 0.1) \rightarrow \widehat{c}_j^t = 2$.



Why it is a good idea ? (few intuitions)

Using duality theory

$$W_1(\alpha, \beta) = \sup_{f \in \mathrm{Lip}_1} \mathbb{E}_{\mathbf{x}\sim\alpha}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{y}\sim\beta}[f(\mathbf{y})]\,.$$

Let $\mathrm{error}_s(f) = \mathbb{E}_{(\mathbf{x},c)\sim P^s}[\ell(c, f(\mathbf{x}))], \mathrm{error}_t(f) = \mathbb{E}_{(\mathbf{x},c)\sim P^t}[\ell(c, f(\mathbf{x}))]$ and

$$\mathcal{F}_{L,\ell} = \{f : X \to C, \ell(\cdot, f(\cdot)) \in \mathrm{Lip}_L\}\,.$$

Take

- Best error on target $f^\star \in \arg\min_{f\in\mathcal{F}_{L,\ell}} \ \mathrm{error}_t(f)$.
- Best error on source $f_s \in \arg\min_{f\in\mathcal{F}_{L,\ell}} \ \mathrm{error}_s(f)$.

Then

$$0 \le \mathrm{error}_t(f_s) - \mathrm{error}_t(f^\star) \le 2L \cdot W_1(P^s, P^t)\,.$$

Conclusion if P^s, P^t are closed in OT then perf should be good.

Deep domain adaptation Damodaran et al. 2018

Let $P^f = \frac{1}{n_t}\sum_{j=1}^{n_t} \delta_{(\mathbf{x}_j^t, f(\mathbf{x}_j^t))}$ and $\hat{P}^s = \frac{1}{n_s}\sum_{i=1}^{n_s} \delta_{(\mathbf{x}_i^s, c_i^s)}$. Solve

$$\min_f \ W_1(\hat{P}^s, P^f)\,.$$

References I

Courty, Nicolas et al. (2016). "Optimal transport for domain adaptation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.9, pp. 1853–1865.

Damodaran, Bharath Bhushan et al. (2018). "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 447–463.

Mansour, Yishay, Mehryar Mohri, and Afshin Rostamizadeh (2009). "Domain adaptation: Learning bounds and algorithms". In: *arXiv preprint arXiv:0902.3430*.

Peyré, Gabriel, Marco Cuturi, et al. (2019). "Computational optimal transport: With applications to data science". In: *Foundations and Trends® in Machine Learning* 11.5-6, pp. 355–607.

Redko, Ievgen et al. (2019). "Optimal transport for multi-source domain adaptation under target shift". In: *The 22nd International Conference on artificial intelligence and statistics*. PMLR, pp. 849–858.