Ideas from Metric Learning

Statistical ML report

Francesco Maria Guadagnuolo
Luca Raffo
Erasmo Santini

Statistical Machine Learning
EPFL Department of Mathematics

1 Introduction

Metric Learning is a branch of machine learning that focuses on learning distance functions tailored to specific tasks. Mathematically speaking, learning a distance on a space X corresponds to learning a function $d : X \times X \to \mathbb{R}$ that satisfies **simmetry**, **positivity** and the **triangle inequality**. The metric learning procedure can be summarized as follows:

1. We are given a bunch of points, each couple of them can be **similar** or **dissimilar**.
2. We embed data points in a metric space according to the supervised information.

Learning a task-specific metric can be powerful for a variety of tasks, such as:

- improving recognition tasks: we want to learn a metric that maps images of the same person closer together while separating different individuals, as shown in Cao et al. [2013].
- improving clustering tasks: we want to learn a metric tailored to the dataset to enhance clustering algorithms, as shown in Xiang et al. [2008].
- graph matching: metric learning can enhance matching of coupled graphs by embedding the topological structure onto an Euclidean space via Wasserstein-Gromov distance, as shown in Xu et al. [2019], and Cheng et al. [2024].

In the following chapters we will give a touch of classical metric learning with an overview of how the Mahalanobis distance learning can improve clustering tasks; and then we will move to the Wasserstein-Gromov metric learning, which will be the main topic of our project.

2 Mahalanobis Metric Learning

Given two objects x and y, and an input distance function $d(x, y)$, along with supervised information (e.g., similarity/dissimilarity constraints or relative distances), the goal is to construct a new and improved distance function $d'(x, y)$.
In **Mahalanobis distance learning**, the distance is restricted to linear transformations:

$$d'(x, y) = d(Gx, Gy) = \sqrt{(x - y)^\top G^\top G(x - y)},$$

where G is the learned linear transformation.
We define $A := G^\top G$, so that the distance function can be written as: $d_A(x, y) = \sqrt{(x - y)^\top A(x - y)}$.
The objective is to learn the matrix A, under the constraint that it must be **positive semi-definite**.

Learning the Matrix A

How do we learn A? Given $\mathcal{S}$ and $\mathcal{D}$, the sets of similar and dissimilar point pairs respectively, we aim to optimize the following:

$$\begin{aligned} \min_{A \succeq 0} \quad & \sum_{(x_i, x_j) \in \mathcal{S}} d_A(x_i, x_j) \\ \text{such that} \quad & \sum_{(x_i, x_j) \in \mathcal{D}} d_A(x_i, x_j) \geq 1, \end{aligned}$$

2.1 Enhancing Kmeans

The objective of this experiment is to test whether the Mahalanobis distance can improve k-means clustering performance on a synthetic dataset, where Euclidean-based clustering struggles.
We create a synthetic dataset in 3D with three ellipsoidal clusters. These clusters are closely aligned along

their elongated axes, leading to significant overlap near their tips. Gaussian noise is added to increase complexity. Standard k-means clustering with Euclidean distance is first applied, but due to noise and overlapping clusters, it is expected to perform poorly.
To overcome this, we adopt a metric learning approach to learn the Mahalanobis distance:

1. Randomly sample pairs of points from the dataset.
2. Label the pairs as: *Must-link* (if from the same cluster) or *Cannot-link* (if from different clusters).
3. Solve an optimization problem to: Minimize intra-cluster distances and maximize inter-cluster separation.

The learned Mahalanobis distance is used to transform the data into a new feature space, aligning clusters by their covariance structure. k-means is then reapplied to the transformed data. Performance is evaluated using clustering accuracy.

The critical challenge in learning the Mahalanobis distance is the **limited number of pairs included during learning**. If the number of *must-link* and *cannot-link* pairs is too small relative to the number of data points, the model may overfit to the provided constraints. This overfitting leads to poor generalization performance on the data that were not included in the similarity/dissimilarity couples. On the other hand, including too many pairs during learning would make this approach pointless, as the key feature is that we want to enhance our clustering accuracy by knowing only a small percentage of pairs in advance.
This issue was addressed in our research, and we found empirically that Mahalanobis enhanced Kmeans increases the accuracy as the number of pairs included in the similarity measures increases, until it surpasses the Euclidean-based Kmeans, and then it stabilizes.
We show below a 3D representation of the Mahalanobis improvement on distancing the clusters, and the plot of the accuracies against the number of pairs included during learning (where the number of total data points generated is fixed to 1000).

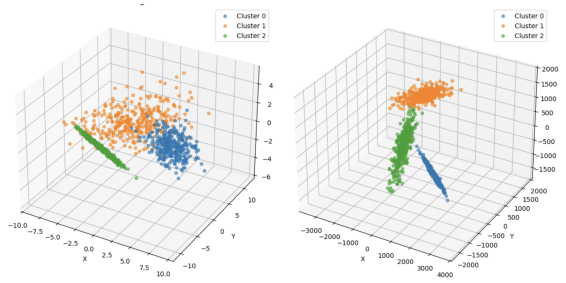

Figure 1: Visual representation

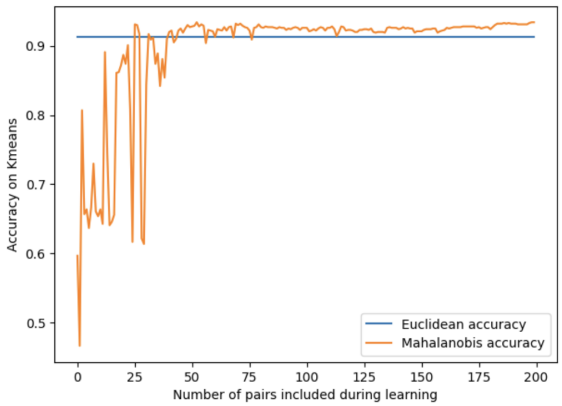

Figure 2: Accuracies vs Number of pairs included

3 Graph Matching and Node Embeddings via Optimal Transport

During recent years the metric learning framework has been extended to more abstract settings, where the latent space is not necessarily Euclidean. From now on we will focus on Wasserstein related distances, which have been shown to have interesting applications in macro-molecules analysis (Wu et al. [2023]) and trajectory inference (Scarvelis and Solomon [2023]) among all. In particular, following Xu et al. [2019], we

will address learning Gromov-Wasserstein distances for graph matching and node embeddings.
Given two measurable metrix spaces (X, d_X, μ_X), (Y, d_Y, μ_Y), we can define the **Gromov Wasserstein distance** as:

$$d_{GW}(\mu_X, \mu_Y) := \inf_{\Pi \in \Gamma(\mu_X, \mu_Y)} \int_{X \times Y} \int_{X \times Y} L(x, y, x', y') \; d\pi(x, y) \; d\pi(x', y')$$

where $L(x, y, x', y') := |d_X(x, x') - d_Y(y, y')|$. This is a metric used to compare two metric spaces by analyzing their structural similarity, regardless of how their points correspond to each other. For this reason, it is widely used in the context of graph matching.
Given two graphs $G = (V, E)$ and $G' = (V', E')$, **graph matching** is the problem of finding a function $f : V \to V'$ that sets correspondences between the two sets of vertices while preserving complex relational information among them. A problem related to graph matching is the learning of **node embeddings**, which aims to map each node of a graph into a low-dimensional space; the collection of embeddings approximates the topology of the graph, with similar/related nodes nearby in embedding space.
While these two problems are typically treated and solved independently, learning node embeddings can be beneficial to graph matching, as two graphs can be matched according to the node structure associated with their node embeddings. To achieve the goal of tackling both the problems together we will use a Gromov-Wasserstein learning framework. This consists in using the Gromov-Wasserstein discrepancy (a relaxation of the GW distance where d_y and d_X can be dissimilarity measures and the loss function can be more flexible, e.g. KL divergence or MSE) to measure the dissimilarity between two graphs comparing the distance matrices in a relational manner, and then learning an optimal transport between the nodes of different graphs. At the same time, the embeddings of the nodes are learned, in a way such that the distance between the embeddings within the same graph approaches the distance matrix, and the distance between the embeddings across different graphs reflects the correspondence found by the learned optimal transport. This approach leads to the following optimization problem:

Gromov-Wasserstein distance learning

$$\min_{X_s, X_t} \min_{T \in \Pi(\mu_s, \mu_t)} \underbrace{\langle L(C_s(X_s), C_t(X_t), T), T \rangle}_{\text{Gromov-Wasserstein discrepancy}} + \alpha \underbrace{\langle K(X_s, X_t), T \rangle}_{\text{Wasserstein discrepancy}} + \beta \underbrace{R(X_s, X_t)}_{\text{prior information}}$$

where

- C_s and C_t are the dissimilarity matrices respectively of the source graph and target graph derived from the data;
- μ_s and μ_t are the empirical distribution of nodes, calculated based on the normalized degree of a graph;
- X_s, X_t are the embeddings of the two graphs;
- α and β are hyperparameters for the structural regularizers terms of the embeddings.

We refer you to Xu et al. [2019] for the definition of K and R, but we notice that the first term measures the **relational dissimilarity** between the two graphs, where the distance matrices consider both the information of data and that of embeddings; the second term measures the **absolute dissimilarity** between the embeddings of the two graphs, and therefore links optimal transport with node embeddings; the last term represents a **regularization** of the embeddings which aims to avoid overfitting.

This double optimization problem can be solved by alternatively learning the optimal transport and the embeddings. In particular, the method used in Xu et al. [2019] applies **nested iterative optimization**. Given current embeddings, the optimal transport problem is solved via a Sinkhorn-Knopp algorithm. Give the optimal transport, the embeddings are updated by stochastic gradient descent. Also, since α controls the influence of node embeddings on the Wasserstein discrepancy which are unreliable at the beginning, it

is initialized with a small value and increased linearly with respect to the number of outer iterations.
The algorithm has been implemented in https://github.com/HongtengXu/gwl. For this project, we wrote from scratch a simplified version. We refer to our Github for details https://github.com/FranceGuad2002/Statistical-Machine-Learning-Fall-2024-.

3.1 Experiments and Discussion

The procedure we have exposed so far is relatively new; therefore, we decided to test it on many datasets where we knew the ground truth: how the graph matching should be done. We considered several synthetic networks, here we enumerate the most meaningful ones:

1. graphs with completely random weights
2. $\mathbb{R}^3$-embedded point clouds, that can be tread as graphs, with weights being distances among points
3. Barabasi-Albert graphs, with randomly re-assigned weights
4. Barabasi-Albert graphs, where we considered the shortest path length as similarity measure

We implemented the algorithm trying both Euclidean and Cosine-based metric in the Wasserstein-discrepancy term, $K(X_s, X_t)$. Following the paper we initially chose the hyper-parameters as: $\beta = 0.5$, $\gamma = 0.5$, $\lambda = 0.001$, $N = 200$ (inner iterations) , $M = 30$ (outer iterations); moreover, for interpretation and visualization purposes, we only considered embeddings in $\mathbb{R}^3$.
After several trials, we observed that the original algorithm leads to completely inconsistent results, as it is too sensitive with respect to the initial random embeddings. To solve this issue, we decided to initialize the embedding of the first graph at random, and then initialize the second by adding random noise to the first embedding. For more general cases, when the second graph has more nodes than the first, we decided to initialize the second embedding by adding noise to the corresponding nodes of the first embedding, and embed the remaining nodes at random. Moreover, to gain robustness we compute the average embedding over 100 different realizations. In (Fig: 3) we report an example of our procedure. We outline that heuristically it seems that considering the shortest paths as weights and taking the average embedding over several realizations improves the robustness of the algorithm. We consider the Euclidean distance in the Wasserstein discrepancy term.

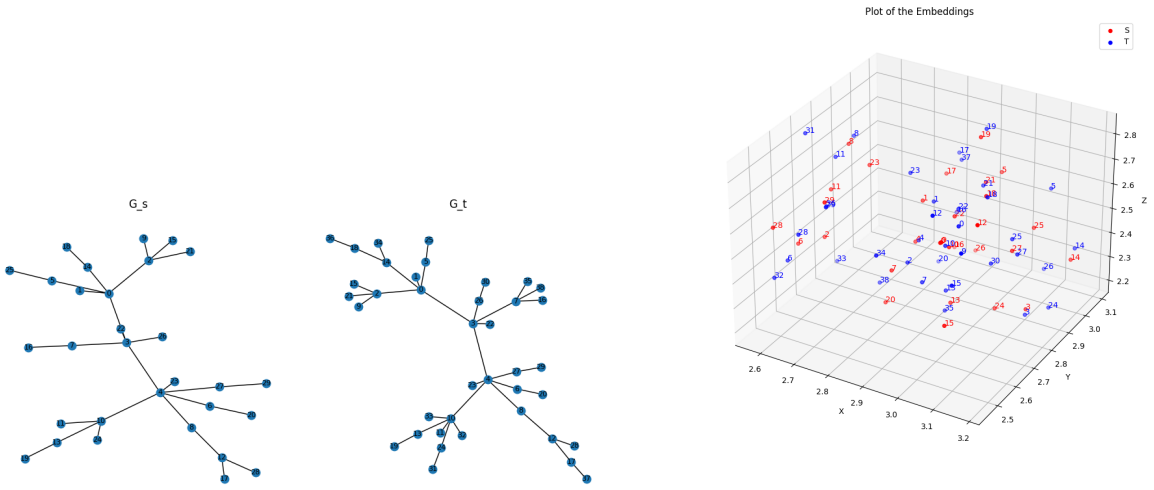

Figure 3: In the left, the original graph, G_s and its noised version, G_t, we then compute their shortest path matrices and the embeddings, averaged over 100 realization, which are plotted in the right.

We also tested the algorithm over real datasets. In real world applications network structure is often reconstructed from similarity or dissimilarity measure: in this way, one can for example obtain a network from time series, considering the correlations as weights. This idea allows to build networks in different contexts, and consequently, we decided to test the algorithm on a network built in this way, in order to exploit the full potential of the proposed procedure in different contexts. We compared two networks: the structural connectivity and the functional connectivity neuronal networks, averaged over 100 individuals. We thank Andrea Santoro (`https://andresantoro.github.io/`) for the data. Details can be found in Griffa et al. [2023].

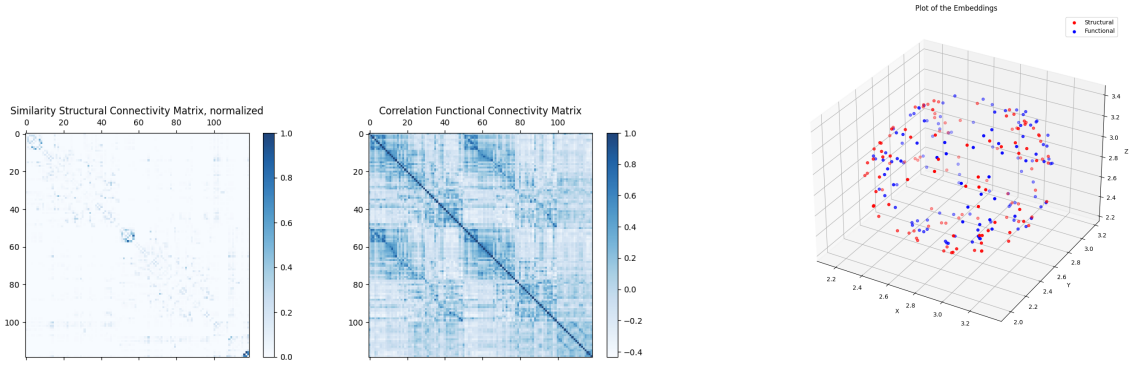

Figure 4: In the left, the structural and functional connectivity matrices. We build weighted networks from them and we apply the algorithm to match the brain regions. We consider only one random initialization with noise. We plot the results in the right.

3.2 Conclusion

From all the tests we have performed, we have drawn the following conclusions:

1. initialization strategies have to be implemented to guarantee consistent results
2. the algorithm is not robust in general, the convergence is very steep, and we believe that many local minima prevent the procedure to reach to optimal matching. Nonetheless, it seems that averaging over many tests is a good way to avoid this "overfitting".
3. as suggested in the original paper, the embeddings converge to a manifold. For the Euclidean distance this seems to be the sphere. We do not observe this behavior for the cosine metric.

To directly observe the dynamics of learning, we realized the following video: dynamics of learning. Finally, we outline that the authors have recently proposed a variation of the algorithm to address these issues in Cheng et al. [2024]. However, this novel procedure uses extra information and is based on the message-passing algorithm.

References

Qiong Cao, Yiming Ying, and Peng Li. Similarity metric learning for face recognition. In *2013 IEEE International Conference on Computer Vision*, pages 2408–2415, 2013. doi: 10.1109/ICCV.2013.299.

Haoran Cheng, Dixin Luo, and Hongteng Xu. Robust graph matching using an unbalanced hierarchical optimal transport framework, 2024. URL https://arxiv.org/abs/2310.12081.

Alessandra Griffa, Mathieu Mach, Julien Dedelley, Daniel Gutierrez-Barragan, Alessandro Gozzi, Gilles Allali, Joanes Grandjean, Dimitri Van de Ville, and Enrico Amico. Evidence for increased parallel information transmission in human brain networks compared to macaques and male mice. *Nature Communications*, 14, 2023. URL https://api.semanticscholar.org/CorpusID:248700044.

Christopher Scarvelis and Justin Solomon. Riemannian metric learning via optimal transport, 2023. URL https://arxiv.org/abs/2205.09244.

Fang Wu, Nicolas Courty, Shuting Jin, and Stan Z. Li. Improving molecular representation learning with metric learning-enhanced optimal transport, 2023. URL https://arxiv.org/abs/2202.06208.

Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600–3612, 2008. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2008.05.018. URL https://www.sciencedirect.com/science/article/pii/S0031320308002057.

Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin. Gromov-wasserstein learning for graph matching and node embedding, 2019. URL https://arxiv.org/abs/1901.06003.