End-to-End Similarity Learning and Hierarchical Clustering for unfixed size datasets

Leonardo Gigli, Beatriz Marcotegui, Santiago Velasco-Forero

End-to-End Similarity Learning and Hierarchical Clustering for unfixed size datasets

Leonardo Gigli, Beatriz Marcotegui, Santiago Velasco-Forero

Centre for Mathematical Morphology - Mines ParisTech - PSL University

Abstract. Hierarchical clustering (HC) is a powerful tool in data analysis since it allows discovering patterns in the observed data at different scales. Similarity-based HC methods take as input a fixed number of points and the matrix of pairwise similarities and output the dendrogram representing the nested partition. However, in some cases, the entire dataset cannot be known in advance and thus neither the relations between the points. In this paper, we consider the case in which we have a collection of realizations of a random distribution, and we want to extract a hierarchical clustering for each sample. The number of elements varies at each draw. Based on a continuous relaxation of Dasgupta's cost function, we propose to integrate a triplet loss function to Chami's formulation in order to learn an optimal similarity function between the points to use to compute the optimal hierarchy. Two architectures are tested on four datasets as approximators of the similarity function. The results obtained are promising and the proposed method showed in many cases good robustness to noise and higher adaptability to different datasets compared with the classical approaches.

1 Introduction

Similarity-based HC is a classical unsupervised learning problem and different solutions have been proposed during the years. Classical HC methods such as Single Linkage, Complete Linkage, Ward's Method are conceivable solutions to the problem. Dasgupta [1] firstly formulated this problem as a discrete optimization problem. Successively, several continuous approximations of the Dasgupta's cost function have been proposed in recent years. However, in the formulation of the problem, the input set and the similarities between points are fixed elements, i.e., the number of samples to compute similarities is fixed. Thus, any change in the input set entails a reinitialization of the problem and a new solution must be found. In addition to this, we underline that the similarity function employed is a key component for the quality of the solution. For these reasons, we are interested in an extended formulation of the problem in which we assume as input a family of point sets, all sampled from a fixed distribution. Our goal is to find at the same time a "good" similarity function on the input space and optimal hierarchical clustering for the point sets. The rest of this paper is organized as follows. In Section 2, we review related works on hierarchical clustering, especially the hyperbolic hierarchical clustering. In Section 3, the proposed method is

described. In section 4, the experimental design is described. And finally, section 5 concludes this paper.

2 Related Works

Our work is inspired by [2] and [3] where for the first time continuous frameworks for hierarchical clustering have been proposed. Both papers assume that a given weighted-graph $\mathcal{G} = (V, E, W)$ is given as input. [2] aims to find the best ultrametric that optimizes a given cost function. Basically, they exploit the fact that the set $\mathcal{W} = \{w : E \to \mathbb{R}_+\}$ of all possible functions over the edges of $\mathcal{G}$ is isomorphic to the Euclidean subset $\mathbb{R}_+^{|E|}$ and thus it makes sense to "*take a derivative according to a given weight function*". Along with this they show that the min-max operator function $\Phi_{\mathcal{G}} : \mathcal{W} \to \mathcal{W}$, defined as $(\forall \tilde{w} \in \mathcal{W}, \forall e_{xy} \in E) \quad \Phi_{\mathcal{G}}(\tilde{w})(e_{xy}) = \min_{\pi \in \Pi xy} \max_{e' \in \pi} \tilde{w}(e')$, where Π_{xy} is the set of all paths from vertex x to vertex y, is sub-differentiable. As insight, the min-max operator maps any function $w \in \mathcal{W}$ to its associated subdominant ultrametric on $\mathcal{G}$. These two key elements are combined to define the following minimization problem over $\mathcal{W}$, $w^* = \arg\min_{\tilde{w} \in \mathcal{W}} J(\Phi_{\mathcal{G}}(\tilde{w}), w)$, where the function J is a differentiable loss function to optimize. In particular, since the metrics $\tilde{w}$ are indeed vectors of $\mathbb{R}^{|E|}$, the authors propose to use the L^2 distance as a natural loss function. Furthermore, they come up with other regularization functions, such as a cluster-size regularization, a triplet loss, or a new differentiable relaxation of the famous Dasgupta's cost function [1]. The contribution of [3] is twofold. On the one hand, inspired by the work of [4], they propose to find an optimal embedding of the graph nodes into the Poincaré Disk, observing that the internal structure of the hierarchical tree can be inferred from leaves' hyperbolic embeddings. On the other hand, they propose a direct differentiable relaxation of the Dasgupta's cost and prove theoretical guarantees in terms of clustering quality of the optimal solution for their proposed function compared with the optimal hierarchy for the Dasgupta's function. As said before, both approaches assume a dataset $\mathcal{D}$ containing n datapoints and pairwise similarities $(w_{ij})_{i,j \in [n]}$ between points are known in advance. Even though this is a very general hypothesis, unfortunately, it does not include cases where part of the data is unknown yet or the number of points cannot be estimated in advance, for example point-cloud scans. In this paper, we investigate a formalism to extend previous works above to the case of n varies between examples. To be specific, we cannot assume anymore that a given graph is known in advance and thus we cannot work on the earlier defined set of functions $\mathcal{W}$, but we would rather look for optimal embeddings of the node features. Before describing the problem, let us review hyperbolic geometry and hyperbolic hierarchical clustering.

2.1 Hyperbolic Hierarchical Clustering

The Poincaré Ball Model $(\mathbb{B}^n, g^{\mathbb{B}})$ is a particular hyperbolic space, defined by the manifold $\mathbb{B}^n = \{x \in \mathbb{R}^n \mid \|x\| < 1\}$ equipped with the following Riemannian metric $g_x^{\mathbb{B}} = \lambda_x^2 g^E, \quad$ where $\lambda_x \coloneqq \frac{2}{1-\|x\|^2}$, where $g^E = \mathbf{I}_n$ is the Euclidean

metric tensor. The distance between two points in the $x, y \in \mathbb{B}^n$ is given by $d_{\mathbb{B}}(x,y) = \cosh^{-1}\left(1 + 2\frac{\|x-y\|_2^2}{(1-\|x\|_2^2)(1-\|y\|_2^2)}\right)$. It is thus straightforward to prove that the distance of a point to the origin is $d_o(x) \coloneqq d(o,x) = 2\tanh^{-1}(\|x\|_2)$. Finally, we remark that $g_x^{\mathbb{B}}$ defines the same angles as the Euclidean metric. The angle between two vectors $u, v \in T_x\mathbb{B}^n \setminus \{\mathbf{0}\}$ is defined as $\cos(\angle(u,v)) = \frac{g_x^{\mathbb{B}}(u,v)}{\sqrt{g_x^{\mathbb{B}}(u,u)}\sqrt{g_x^{\mathbb{B}}(v,v)}} = \frac{\langle u,v\rangle}{\|u\|\|v\|}$, and $g^{\mathbb{B}}$ is said to be *conformal* to the Euclidean metric. In our case, we are going to work on the Poincaré Disk that is $n = 2$. The interested reader may refer to [5] for a wider discussion on Hyperbolic Geometry. Please remark that the geodesic between two points in this metric is either the segment of the circle orthogonal to the boundary of the ball or the straight line that goes through the origin in case the two points are antipodal.The intuition behind the choice of this particular space is motivated by the fact that the curvature of the space is negative and geodesic coming out from a point has a *"tree-like"* shape. Moreover, [3] proposed an analog of the Least Common Ancestor (LCA) in the hyperbolic space. Given two leaf nodes i, j of a hierarchical T, the LCA $i \vee j$ is the closest node to the root r of T on the shortest path π_{ij} connecting i and j. In other words $i \vee j = \arg\min_{k \in \pi_{ij}} d_T(r,k)$, where $d_T(r,k)$ measures the length of the path from the root node r to the node k. Similarly, the *hyperbolic lowest common ancestor* between two points z_i and z_j in the hyperbolic space is defined as the closest point to the origin in the geodesic path, denoted $z_i \rightsquigarrow z_j$, connecting the two points: $z_i \vee z_j \coloneqq \arg\min_{z \in z_i \rightsquigarrow z_j} d(o,z)$. Thanks to this definition, it is possible to decode a hierarchical tree starting from leaf nodes embedding into the hyperbolic space. The decoding algorithm uses a union-find paradigm, iteratively merging the most similar pairs of nodes based on their hyperbolic LCA distance to the origin. Finally [3] also proposed a continuous version of Dasgupta's cost function. Let $Z = \{z_1, \ldots, z_n\} \subset \mathbb{B}^2$ be an embedding of a tree T with n leaves, they define their cost function as:

$$C_{\text{HYPHC}}(Z; w, \tau) = \sum_{ijk}(w_{ij} + w_{ik} + w_{jk} - w_{\text{HYPHC},ijk}(Z; w, \tau)) + \sum_{ij} w_{ij}, \quad (1)$$

where $w_{\text{HYPHC},ijk}(Z; w, \tau) = (w_{ij}, w_{ik}, w_{jk}) \cdot \sigma_\tau(d_o(z_i \vee z_j), d_o(z_i \vee z_k), d_o(z_j \vee z_k))^\top$, and $\sigma_\tau(\cdot)$ is the scaled softmax function $\sigma_\tau(w)_i = e^{w_i/\tau} / \sum_j e^{w_j/\tau}$. We recall that w_{ij} are the pair-wise similarities, which in [3] are assumed to be known, but in this work are learned.

3 End-to-End Similarity Learning and HC

Let us consider the example of k continuous random variables that take values over an open set $\Omega \subset \mathbb{R}^d$. Let $\mathcal{X}_t = \{x_1^{(t)}, \ldots, x_{n_t}^{(t)}\}$ a set of points obtained as realization of the k random variables at step t. Moreover, we assume to be in a semi-supervised setting. Without loss of generality, we expect to know the associated labels of the first l points in $\mathcal{X}_t$, for each t. Each label takes value

in $[k] = \{1, \ldots, k\}$, and indicates from which distribution the point has been sampled. In our work, we aim to obtain at the same time a good similarity function $\delta : \Omega \times \Omega \to \mathbb{R}_+$ that permits us to discriminate the points according to the distribution they have been drawn and an optimal hierarchical clustering for each set $\mathcal{X}_t$. Our idea to achieve this goal is to combine the continuous optimization framework proposed by Chami [3] along with deep metric learning to learn the similarities between points. Hence, we look for a function $\delta_\theta : \Omega \times \Omega \to \mathbb{R}_+$ such that

$$\min_{\theta, Z \in \mathcal{Z}} C_{\text{HYPHC}}(Z, \delta_\theta, \tau) + \mathcal{L}_{\text{triplet}}(\delta_\theta; \alpha). \tag{2}$$

The second term of the equation above is the sum over the set $\mathcal{T}$ of triplets:

$$\mathcal{L}_{\text{triplet}}(\delta_\theta; \alpha) = \sum_{(a_i, p_i, n_i) \in \mathcal{T}} \max(\delta_\theta(a_i, p_i) - \delta_\theta(a_i, n_i) + \alpha, 0), \tag{3}$$

where a_i is the anchor input, p_i is the positive input of the same class as a_i, n_i is the negative input of a different class from a_i and $\alpha > 0$ is the margin between positive and negative values. One advantage of our formalism is that it allows us to use deep learning approach, i.e., backpropagation and gradient descend optimization to optimize the model's parameters. As explained before, we aim to learn a similarity function and at the same time find an optimal embedding for a family of point sets into the hyperbolic space which implicitly encodes a hierarchical structure. To achieve this, our idea is to model the function δ_θ using a neural network whose parameters we fit to optimize the loss function defined in (2). Our implementation consists of a neural network NN_θ that carries out a mapping $\text{NN}_\theta : \Omega \to \mathbb{R}^2$. The function δ_θ is thus written as:

$$\delta_\theta(x, y) = \cos(\angle(\text{NN}_\theta(x), \text{NN}_\theta(y))), \tag{4}$$

We use the cosine similarity for two reasons. The first comes from the intuition that points belonging to the same cluster will be forced to have small angles between them. As a consequence, they will be merged earlier in the hierarchy. The second reason regards the optimization process. Since the hyperbolic metric is *conformal* to the Euclidean metric, the cosine similarity allows us to use the same the Riemannian Adam optimizer [6] in (2). Once computed the similarities, the points are all normalized at the same length to embed them into the Hyperbolic space. The normalization length is also a trainable parameter of the model. Accordingly, we have selected two architectures. The first is a Multi-Layer-Perceptron (MLP) composed of four hidden layers, and the second is a model composed of three layers of Dynamic Graph Egde Convolution (DGCNN) [7].

4 Experiments

Experiments setup: Inspired by the work of [2] we took into account four sample generators in Scikit-Learn to produce four datasets as it is illustrated in Fig. 1.

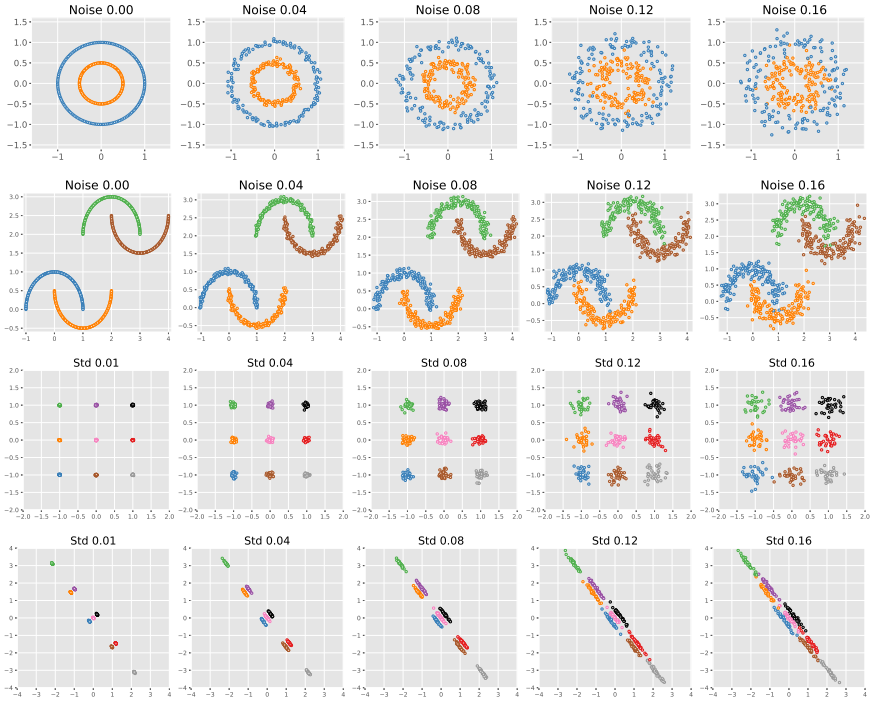


Fig. 1. Different examples of *circles*, *moons*, *blobs* and *anisotropics* that have been generated varying noise value to evaluate robustness and performance of proposed method.

For each dataset we generate, the training set is made of 100 samples and the validation set of 20 samples. The test set contains 200 samples. In addition, each sample in the datasets contains a random number of points. In the experiments we sample from 200 to 300 points each time and the labels are known only for the 30% of them. In *circles* and *moons* datasets increasing the value of noise makes the clusters mix each other and thus the task of detection becomes more difficult. Similarly, in *blobs* and *anisotropics*, we can increase the value of standard deviation to make the problem harder. Our goal is to explore how the models embed the points and separate the clusters. Moreover, we want to investigate the robustness of the models to noise. For this reason, in these datasets, we set up two different levels of difficulty according to the noise/standard deviation used to generate the sample. In *circles* and *moons* datasets, the easiest level is represented by samples without noise, while the harder level contains samples whose noise value varies up to 0.16. In Blobs and Anisotropic datasets, we chose two different values of Gaussian standard deviation to generate the sample. In the easiest level, the standard deviation value is fixed at 0.08, while in the harder level is at 0.16. For each level of difficulty, we trained the models and compared the architectures. In addition, we used the harder level of difficulty to test all the models.

Architectures: The architectures we test are MLP and DGCNN. The dimension of hidden layers is 64. After each Linear layer we apply a LeakyReLU defined as $\text{LeakyReLU}(x) = \max\{x, \eta x\}$ with a negative slope $\eta = 0.2$. In addition, we use Batch Normalization [8] to speed up and stabilize convergence.

Metrics: Let k the number of clusters that we want to determine. For the evaluation, we consider the partition of k clusters obtained from the hierarchy and we measure the quality of the predictions using Average Rand Index (ARI), Purity, and Normalized Mutual Information Score (NMI). Our goal is to test the ability of the two types of architectures selected to approximate function in (4).

Visualize the embeddings: Let first discuss the results obtained on *circles* and *moons*. In order to understand and visualize how similarities are learned, we first trained the architectures at the easiest level. Fig. 2 illustrates the predictions carried out by models trained using samples without noise. Each row in the figures illustrates the model's prediction on a sample generated with a specific noise value. The second column from the left of sub-figures depicts hidden features in the feature space $\mathcal{H} \subset \mathbb{R}^2$. The color assigned to hidden features depends on points' labels in the ground truth. The embeddings in the Poincaré Disk (third column from the left) are obtained by normalizing the features to a learned scale. Furthermore, the fourth column of sub-figures shows the prediction obtained by extracting flat clustering from the hierarchy decoded from leaves embedding in the Poincaré Disk. Here, colors assigned to points come from predicted labels. The number of clusters is chosen in order to maximize the average rand index score. It is interesting to remark how differently the two architectures extract features. Looking at the samples without noise, it is straightforward that hidden

features obtained with MLP are aligned along lines passing through the origin. Especially in the case of *circles* (Fig. 2), hidden features belonging to different clusters are mapped to opposite sides with respect to the origin, and after rescaling hidden features are clearly separated in the hyperbolic space. Indeed, picking cosine similarity in (4) we were expecting this kind of solution. On the other hand, the more noise we add, the closer to the origin hidden features are mapped. This leads to a less clear separation of points on the disk. Unfortunately, we cannot find a clear interpretation of how DGCNN maps points to hidden space. However, also in this case, the more noise we add, the harder is to discriminate between points of different clusters[1].

Comparison with classical HC methods: In Fig. 3 we compare models trained at easier level of difficulty against classical methods such as Single, Complete, Average and Ward's method Linkage on *circles*, *moons*, *blobs* and *anisotropics* respectively. The plots show the degradation of the performance of models as we add noise to samples. Results on *circles* say that Single Linkage is the method that performs the best for small values of noise. However, MLP shows better robustness to noise. For high levels of noise, MLP is the best method. On the other hand, DGCNN exhibits a low efficacy also on low levels of noise. Other classical methods do not achieve good scores on this dataset. A similar trend can also be observed in the *moons* dataset. Note that, in this case, MLP is comparable with Single Linkage also on small values of noise, and its scores remain good also on higher levels of noise. DGCNN and other classical methods perform worse even in this data set. Results obtained by MLP and DGCNN on *blobs* dataset are comparable with the classical methods, even though the performances of models are slightly worse compared to classical methods for higher values of noise. On the contrary, MLP and DGCNN achieve better scores on the *anisotropics* dataset compared to all classical models. Overall, MLP models seem to act better than DCGNN ones in all the datasets.

Benchmark of the models: Table 1 reports the scores obtained by the trained models on each dataset. Each line corresponds to a model trained either at an easier or harder level of difficulty. The test set used to evaluate the results contains 200 samples generated using the harder level of difficulty. Scores obtained demonstrate that models trained at the harder levels of difficulty are more robust to noise and achieve better results. As before, also in this case MLP is, in general, better than DGCNN in all the datasets considered.

5 Conclusion

In this paper, we have studied the metric learning problem to perform hierarchical clustering where the number of nodes per graph in the training set can vary. We have trained MLP and DGCNN architectures on five datasets by using

[1] Supplementary Figures are available at `https://github.com/liubigli/similarity-learning/blob/main/GSI2021_Appendix.pdf`

our proposed protocol. The quantitative results show that overall MLP performs better than DGCNN. The comparison with the classic methods proves the flexibility of the solution proposed to the different cases analyzed, and the results obtained confirm higher robustness to noise. Finally, inspecting the hidden features, we have perceived how MLP tends to project points along lines coming out from the origin. To conclude, the results obtained are promising and we believe that it is worth testing this solution also on other types of datasets such as 3D point clouds.

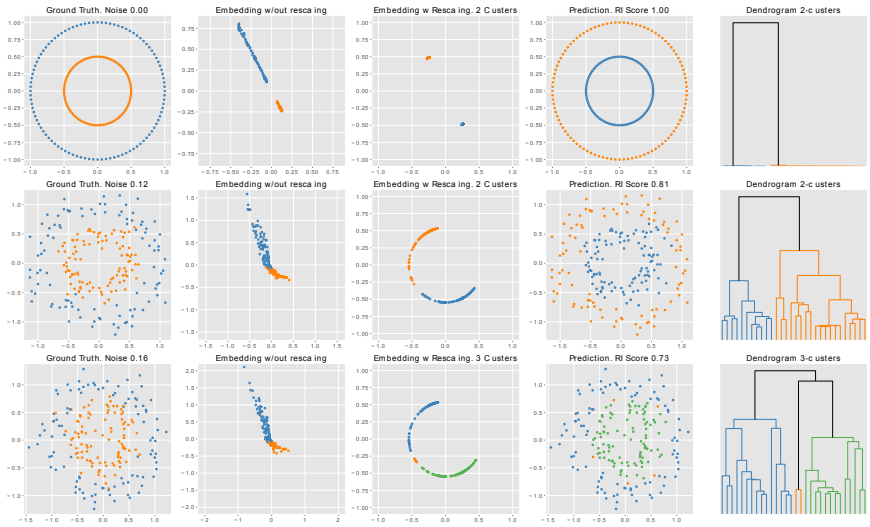
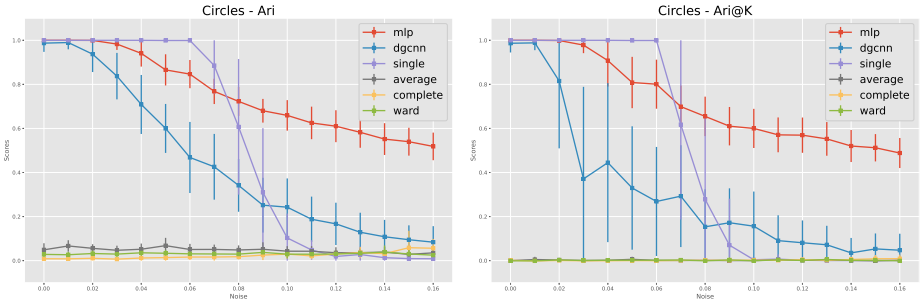


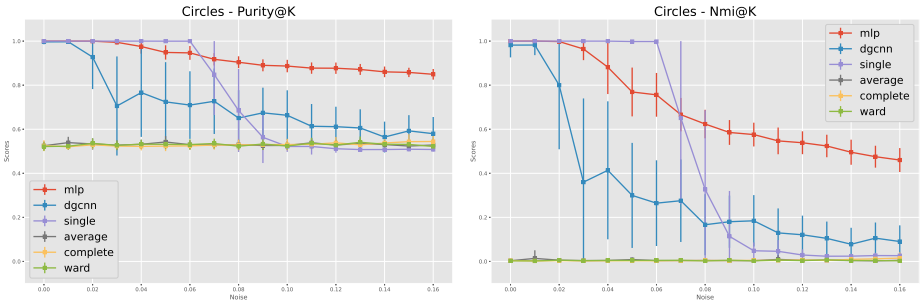
Fig. 2. Effect of noise on predictions in the *circles* database. The model used for prediction is an MLP trained without noise. From top to bottom, each row is a case with an increasing level of noise. In the first column input points, while in the second column we illustrate hidden features. Points are colored according to ground truth. The third column illustrates hidden features after projection to Poincaré Disk. The fourth column shows predicted labels, while the fifth column shows associated dendrograms.

References

1. Dasgupta, S.: A cost function for similarity-based hierarchical clustering. In: Proceedings of the 48 annual ACM symposium on Theory of Computing. (2016) 118–127
2. Chierchia, G., Perret, B.: Ultrametric fitting by gradient descent. In: NIPS. (2019) 3181–3192
3. Chami, I., Gu, A., Chatziafratis, V., Ré, C.: From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. NIPS **33** (2020)
4. Monath, N., Zaheer, M., Silva, D., McCallum, A., Ahmed, A.: Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In: 25th ACM SIGKDD Conference on Discovery & Data Mining. (2019) 714–722



(a) Best ARI (b) ARI @ K



(c) Purity @ K (d) Purity @ K

Fig. 3. Robustness to the noise of models on *circles*. We compare trained models against classical methods as Single Linkage, Average Linkage, Complete Linkage, and Ward's Method. The models used have been trained on a dataset without noise. Test sets used to measure scores contain 20 samples each. Plots show the mean and standard deviation of scores obtained.

Dataset	k	Noise/Cluster std	Model	Hidden	Temp	Margin	Ari@k ± s.d	Purity@k ± s.d	Nmi@k ± s.d	Ari ± s.d
circle	2	0.0	MLP	64	0.1	1.0	0.871 ± 0.153	0.965 ± 0.0427	0.846 ± 0.167	0.896 ± 0.123
circle	2	$[0.0 - 0.16]$	MLP	64	0.1	1.0	0.919 ± 0.18	0.972 ± 0.0848	0.895 ± 0.187	0.948 ± 0.0755
circle	2	0.0	DGCNN	64	0.1	1.0	0.296 ± 0.388	0.699 ± 0.188	0.327 ± 0.356	0.408 ± 0.362
circle	2	$[0.0 - 0.16]$	DGCNN	64	0.1	1.0	0.852 ± 0.243	0.947 ± 0.116	0.826 ± 0.247	0.9 ± 0.115
moons	4	0.0	MLP	64	0.1	1.0	0.895 ± 0.137	0.927 ± 0.108	0.934 ± 0.0805	0.955 ± 0.0656
moons	4	$[0.0 - 0.16]$	MLP	64	0.1	1.0	0.96 ± 0.0901	0.971 ± 0.0751	0.972 ± 0.049	0.989 ± 0.017
moons	4	0.0	DGCNN	64	0.1	1.0	0.718 ± 0.247	0.807 ± 0.187	0.786 ± 0.191	0.807 ± 0.172
moons	4	$[0.0 - 0.16]$	DGCNN	64	0.1	1.0	0.917 ± 0.123	0.942 ± 0.0992	0.941 ± 0.0726	0.966 ± 0.0455
blobs	9	0.08	MLP	64	0.1	0.2	0.911 ± 0.069	0.939 ± 0.057	0.953 ± 0.025	0.958 ± 0.022
blobs	9	0.16	MLP	64	0.1	0.2	0.985 ± 0.0246	0.992 ± 0.0198	0.99 ± 0.0115	0.992 ± 0.00821
blobs	9	0.08	DGCNN	64	0.1	0.2	0.856 ± 0.0634	0.891 ± 0.0583	0.931 ± 0.025	0.921 ± 0.0401
blobs	9	0.16	DGCNN	64	0.1	0.2	0.894 ± 0.0694	0.92 ± 0.0604	0.95 ± 0.0255	0.948 ± 0.0336
aniso	9	0.08	MLP	64	0.1	0.2	0.86 ± 0.0696	0.904 ± 0.0631	0.922 ± 0.0291	0.925 ± 0.0287
aniso	9	0.16	MLP	64	0.1	0.2	0.952 ± 0.0503	0.968 ± 0.044	0.972 ± 0.0189	0.976 ± 0.0133
aniso	9	0.08	DGCNN	64	0.1	0.2	0.713 ± 0.0835	0.793 ± 0.0727	0.844 ± 0.0401	0.795 ± 0.0652
aniso	9	0.16	DGCNN	64	0.1	0.2	0.84 ± 0.0666	0.879 ± 0.0595	0.922 ± 0.0274	0.914 ± 0.0436

Table 1. Scores obtained by MLP and DGCNN on four datasets: *circle*, *moons*, *blobs*, and *anisotropics*. In each dataset the models have been tested on the same test set containing 200 samples.

5. Brannan, D.A., Esplen, M.F., Gray, J.: Geometry. Cambridge University Press, Cambridge ; New York (2011)
6. Bécigneul, G., Ganea, O.E.: Riemannian adaptive optimization methods. arXiv preprint arXiv:1810.00760 (2018)
7. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5) (2019) 1–12
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)