Topological Gradient-based Competitive Learning

Topological Gradient-based Competitive Learning

Pietro Barbiero*, Gabriele Ciravegna†‡, Vincenzo Randazzo§, Eros Pasero§, Giansalvo Cirrincione¶‖
*Cambridge University, Cambridge, United Kingdom
Email: barbiero@tutanota.com
† University of Siena, Siena, Italy
‡ University of Florence, Florence, Italy
§ Politecnico di Torino, Turin, Italy
¶ University of Picardie Jules Verne, Amiens, France
‖ University of South Pacific, Suva, Fiji

***Abstract*—Topological learning is a wide research area aiming at uncovering the mutual spatial relationships between the elements of a set. Some of the most common and oldest approaches involve the use of unsupervised competitive neural networks. However, these methods are not based on gradient optimization which has been proven to provide striking results in feature extraction also in unsupervised learning. Unfortunately, by focusing mostly on algorithmic efficiency and accuracy, deep clustering techniques are composed of overly complex feature extractors, while using trivial algorithms in their top layer. The aim of this work is to present a novel comprehensive theory aspiring at bridging competitive learning with gradient-based learning, thus allowing the use of extremely powerful deep neural networks for feature extraction and projection combined with the remarkable flexibility and expressiveness of competitive learning. In this paper we fully demonstrate the theoretical equivalence of two novel gradient-based competitive layers. Preliminary experiments show how the dual approach, trained on the transpose of the input matrix i.e. X^T, lead to faster convergence rate and higher training accuracy both in low and high-dimensional scenarios.**

***Index Terms*—Gradient-based Clustering, Competitive Learning, Deep Learning, Duality Theory, Topology, Unsupervised Learning.**

I. Introduction

From the dawn of Artificial Intelligence (AI), data clustering has always been a field of great interest in the scientific community. First approaches were mainly based on similarity measures among data. Prominent methods like k-Means [1], Gaussian Mixture Models (GMM) [2] and more recently Density Based Spatial Clustering (DBSCAN) [3] have been extensively used to uncover unknown relations in unsupervised problems. These types of approaches are capable of finding groups of samples that are similar, but they cannot detect the underlying topology. Hierarchical clustering partially solved this issue by creating a hierarchy of clusters either with an agglomerative [4], [5] or with a divisive strategy [6], [7]. Other approaches, instead, try to solve this problem by introducing a topological structure among cluster nodes. The first algorithm exploiting this concept is the Self-Organizing-Map (SOM) by Kohonen [8], where a neural network is trained to represent the input space using a grid, whose number of units and their connections, i.e. the topology, is defined in advance. Alternatively, techniques such as Neural Gas (NG) [9], Growing Neural Gas (GNG) [10] and their variants [11]–[13] apply the the Competitive Hebbian Learning (CHL) [14]–[16] for defining local topology; indeed, given an input sample, the two closest neurons, called first and second winners, are linked by an edge [17], [18]. All the previous cited methods belong to the *competitive learning* field; here, units compete to represent the input sample, i.e. they move towards it depending on their distances and the network current topology (neighbourhoods). Another issue concerning the above-cited methods is the well-known curse of dimensionality [19], [20]. Euclidean measures are no more effective when dealing with high-dimensional data such as images. To this aim, many works proposed dimensionality reduction and feature extraction methods as pre-processing before the clustering step like Principal Component Analysis (PCA) [21] and kernel functions. These methods are indeed capable of mapping row data into a feature space with a much lower dimensionality. However, the effectiveness of such techniques is limited when dealing with complex latent structures. Recently, however, Deep Neural Networks (DNN), and more specifically Convolutional Neural Network (CNN) [22], have incredibly improved processing performances when dealing with highly-dimensional data in supervised learning. As a consequence, many approaches tried to apply these methods also to the unsupervised learning field. Deep neural networks are capable to transform high-dimensional data into clustering-friendly representations. By employing DNN, clustering and feature transformation are now treated as a single task. DNN architecture may be directly trained through the optimization of a clustering loss. The choice of the learning function is particularly important when dealing with this type of architecture. As a matter of fact, straightforward employment of DNN may lead to corrupted feature transformation, where data are mapped to compact clusters that do not reflect the real data topology. In order to overcome this issue, some works proposed to exploit both unsupervised and supervised network pre-training, weight regularization, and data augmentation techniques. For what concerns unsupervised network pre-training, common strategies consider training Restricted Boltzmann Machines (RBM) [23] or AutoEncoders (AE) [24] and later fine-tune the networks (only the encoder for AE) through a clustering learning function

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

only [25], [26]. Supervised pre-training techniques are instead commonly employed when dealing with image data. Indeed, classical clustering algorithms perform well when using the feature extracted from the last layer of a CNN, pre-trained on big image dataset as ImageNet [27]. Direct approaches that do not consider any network pre-training, have been recently proposed in [28]–[30]. Otherwise, clustering learning procedures may be integrated with a network learning process. This allows the employment of more complex architectures like Autoencoders (AE), Variational-Autoencoders (VAE) or Generative Adversarial Networks (GAN). Such techniques commonly consider a double stage learning in which they first learn a good representation of the input space through a network loss function and later fine tune the network by also optimizing a clustering-specific loss.

However, the idea of joining the strength of DNN with the higher representation capabilities of competitive learning approaches has been previously considered only in a few works [31], [32]. In this work, we consider two possible variants of a neural network architecture in which competitive learning is taken into consideration by the loss function. The proposed architectures can either be employed by themselves or they can be placed on top of more complex neural architectures such as AE, CNN, VAE or GAN.

This work is organized in three main sections. The first one describes two novel methods that can be used to join competitive and gradient-based learning, namely the *vanilla competitive layer* (VCL) and the *dual competitive layer* (DCL). The following section presents preliminary experiments showing the benefits and the differences of the two approaches. Finally, the last section describes how the methods presented in this work can be further developed and extended.

II. Gradient-based competitive learning

This section describes two different approaches that join competitive and gradient-based learning. In a standard competitive layer [33]–[35], every competing neuron is described by a vector of weights w_i, representing the position of the neuron (a.k.a. *prototype*) in the input space. The inverse of the Euclidean distance between the input data x_k and the weight vector w_i represents the similarity between the input and the prototype. For every input vector x_k, the prototypes *compete* with each other to see which one of them is the most similar to that particular input vector. By following the Competitive Hebbian Learning (CHL) rule [14], [15], the two closest prototypes to x_k are connected using an edge, representing their mutual activation.

In general, competitive learning is based on more or less heuristic rules. Instead, the family of k-Means algorithms is justified by the minimization of a loss function, representing the quantization error. The first proposed approach (VCL), instead, is based directly on the minimization of this loss. This is performed by using a first-order gradient technique, for straightly estimating the prototypes. Adding this layer to the top of a deep neural network, a deep clustering can be performed by backpropagating the gradient information from the clustering to the previous layers. As a consequence, the benefits of using a powerful feature extractor and a sophisticated topological learning algorithm can be both exploited. However, the loss used by this approach is only function of the training set and the weights of the layer (the output neuron weights are the prototypes). This means the outputs of the layer are not taken into account. In this sense, the first approach is better interpreted as a straight competitive learning on the input set than a true layer to be added. The second approach (DCL) is more *neural*, because it represents a true transformation of the inputs. It is an alternative approach for the implementation of a competitive layer which is trained using the transpose of the input matrix, i.e. X^T.

A. Dual neural networks

A deep neural network can be interpreted as a nonlinear function f mapping input data $x \in \mathbb{R}^d$ into a different representation $y \in \mathbb{R}^p$ which is optimized according to an error function $\mathcal{L}$. Hence, a concise representation of a neural network is a pair $(f, \mathcal{L})$ such that:

$$f : \mathbb{R}^d \to \mathbb{R}^p, \quad \mathcal{L} : \cdot \to \mathbb{R} \tag{1}$$

The relationship between d and p and the kind of loss function used to train the model determine the kind of learning task. In most settings, deep neural networks are used to contract the input space into an interpretable codomain where the performance of the network can be easily assessed. For instance, if $p = 1$ and the loss function is the mean squared error between the output of the network ($y \in \mathbb{R}$) and a target variable ($t \in \mathbb{R}$), the learning task is called regression:

$$f : \mathbb{R}^d \to \mathbb{R}, \quad \mathcal{L} = mse(y, t) \tag{2}$$

Another common learning task is classification which can be obtained by setting $p = c$, where c corresponds to the number of classes, and by using a cross-entropy error function H:

$$f : \mathbb{R}^d \to \mathbb{R}^c, \quad \mathcal{L} = H(y, t) \tag{3}$$

In practice, the training process is performed using a dataset composed of objects $X \in \mathbb{R}^{n,d}$ and (for supervised tasks) targets $T \in \mathbb{R}^{n,p}$, where n is the number of samples, d the number of input features, and p the number of output features. Hence, the result of the training process can be seen as a projection of the input matrix X into a (usually) lower dimensional representation Y:

$$\begin{aligned} f &: \mathbb{R}^{n,d} \to \mathbb{R}^{n,p} \qquad [\text{usually } p \ll d] \\ Y &= f(X) \end{aligned} \tag{4}$$

Therefore, a deep neural network can be summarized as a nonlinear map *reducing* the number of columns of a matrix, while keeping the original number of rows. If the rows of the input matrix represent a set of samples and the columns a set of features (as it usually is), then the neural network is actually shrinking the number of features.

However, if we consider the transpose problem, where $X^T \in \mathbb{R}^{d,n}$, the neural network can still be used to transform

the input matrix into useful representations. Also in this case, the output must keep the same number of rows of the input by construction, i.e. $Y \in \mathbb{R}^{d,k}$. If $k < n$ the neural network is *contracting* the input, while if $k > n$ the neural network is *augmenting* the input. While normally neural networks are used to generate an abstract representation of the input features useful for supervised tasks like classification or regression, the transpose problem can be used to generate an abstract representation of the input samples useful for learning the topology of the input manifold. In fact, by choosing an appropriate clustering error function $\mathcal{C}$ we can define a deep neural network learning the positions of cluster centroids (a.k.a. *prototypes*) as:

$$f : \mathbb{R}^{d,n} \to \mathbb{R}^{d,k}, \quad \mathcal{L} = \mathcal{C} \tag{5}$$

where k corresponds to the number of output units of the network. Each of the k output unit returns as output a $q^T \in \mathbb{R}^d$ vector representing a position in the feature space $\mathbb{R}^d$. Hence, by optimizing such positions according to a clustering error function, the neural network can learn prototypes corresponding to cluster centroids.

B. Duality theory

The intuitions outlined in the previous section can be formalized in a general theory which considers the duality properties between a linear single-layer neural network and its dual, defined as a network which learns on the transpose of the input matrix and has the same number of neurons.

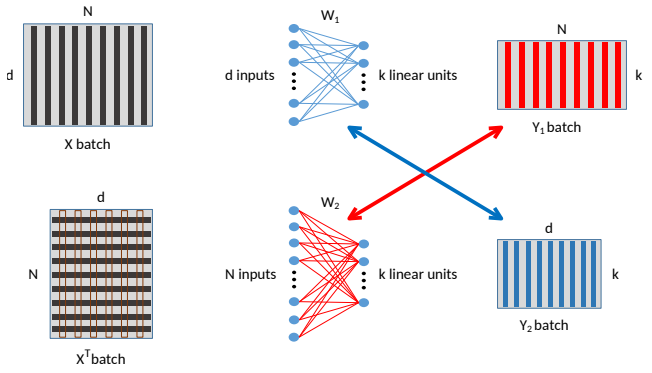

Fig. 1. Dual linear single-layer neural networks.

Consider a single layer neural network whose outputs have linear activation functions. There are d input units and k output units which represent a continuous signal in case of regression or class membership (posterior probabilities in case of cross entropy error function) in case of classification. A batch of n samples, say X, is fed to the network. The weight matrix is W_1, where the element w_{ij} represents the weight from the input unit j to the neuron i. The single layer neural network with linear activation functions in the lower scheme is here called the dual network of the former one. It has the same number of outputs and n inputs. It is trained on the transpose of the original X database. Its weight matrix is W_2 and the output batch is Y_2. The following theorems state the duality conditions of the two architectures. Figure 1 represents the two networks and their duality.

Theorem 1 (Network duality in competitive learning)**.** *Given a loss function for competitive learning based on prototypes, a single linear network (base) whose weight output neurons are the prototypes is equivalent to another (dual) whose outputs are the prototypes, under the following assumptions:*

1) *the input matrix of the dual network is the transpose of the input matrix of the base network;*
2) *the samples of the input matrix X are uncorrelated with unit variance*

Proof. Consider a loss function based on prototypes, whose minimization is required for competitive learning. From the assumption on the inputs (rows of the matrix X), it results $XX^T = I_d$. A single layer linear network is represented by the matrix formula:

$$Y = WX = \Big[\text{prototype}_1 \dots \text{prototype}_k\Big]X \tag{6}$$

By multiplying on the right by X^T, it holds:

$$WXX^T = YX^T \tag{7}$$

Under the second assumption:

$$W = \Big[\text{prototype}_1 \dots \text{prototype}_k\Big] = YX^T \tag{8}$$

This equation represents a (dual) linear network whose outputs are the prototypes W. Considering that the same loss function is used for both cases, the two networks are equivalent. □

This theorem directly applies to the VCL (base) and DCL (dual) neural networks if the assumption 2 holds for the training set. If not, a preprocessing, e.g. batch normalization, can be performed.

Theorem 2 (Impossible complete duality)**.** *Two dual networks cannot share weights as $W_1 = Y_2$ and $W_2 = Y_1$ (complete dual constraint), except if the samples of the input matrix X are uncorrelated with unit variance.*

Proof. From the duality of networks and their linearity, for an entire batch it follows:

$$\begin{cases} Y_1 &= W_1 X \\ Y_2 &= W_2 X^T \end{cases} \implies W_1 = Y_1 X^T$$
$$\implies W_1 = W_1 X X^T$$
$$\implies XX^T = I_d \tag{9}$$

$$\begin{cases} Y_1 &= W_1 X \\ Y_2 &= W_2 X^T \end{cases} \implies W_2 = Y_2 X^T$$
$$\implies W_2 = W_2 X^T X$$
$$\implies X^T X = I_n \tag{10}$$

where I_d and I_n are the identity matrices of size d and n, respectively. These two final conditions are only possible if the samples of the input matrix X are uncorrelated with unit

variance, which is not the case in (almost all) machine learning applications. □

Theorem 3 (Half duality I)**.** *Given two dual networks, if the samples of the input matrix X are uncorrelated with unit variance and if $W_1 = Y_2$ (first dual constraint), then $W_2 = Y_1$ (second dual constraint).*

Proof. From the first dual constraint (see Figure 2, right), for the second network it stems:

$$Y_2 = W_1 = W_2 X^T \tag{11}$$

Hence:

$$Y_1 = W_1 X \implies Y_1 = W_2 X^T X \tag{12}$$

under the second assumption on X^T from Theorem 1, which implies $X^T X = I_n$, the result follows (see Figure 2, left). □

Theorem 4 (Half duality II)**.** *Given two dual networks, if the samples of the input matrix X are uncorrelated with unit variance and if $W_2 = Y_1$ (second dual constraint), then $W_1 = Y_2$ (first dual constraint).*

Proof. From the second dual constraint (see Figure 2, left), for the second network it stems:

$$Y_1 = W_2 = W_1 X \tag{13}$$

From the assumption on the inputs (rows of the matrix X), it results $XX^T = I_d$. The first neural architecture yields (see Figure 2, right):

$$Y_2 = W_2 X^T \implies Y_2 = W_1 X X^T = W_1 \tag{14}$$

□

Theorem 4 justifies the use of the first single-layer neural network as a competitive layer.

C. Analysis of the learning process

The theorems illustrated in the last section establish a set of conditions under which a base competitive layer (e.g. VCL) and its dual network (e.g. DCL) are equivalent. However, this theory shows such an equivalence only in terms of the architecture of the two neural networks. By analyzing the forward and the backward pass, the learning process of the two layers is quite different. In particular, in the VCL there is no forward pass as Y_1 is not computed nor considered and the prototype matrix is just the weight matrix W_1:

$$P_1 = \left[\text{prototype}_1, \ldots, \text{prototype}_k\right] = W_1 \tag{15}$$

where $\text{prototype}_i \in \mathbb{R}^{d \times 1}$. In the dual network, instead, the prototype matrix corresponds to the output Y_2; hence, the forward pass is a linear transformation of the input X^T through the weight matrix W_2:

$$\begin{aligned} P_2 &= \left[\text{prototype}_1 \ldots \text{prototype}_k\right]^T = Y_2 = W_2 X^T = \\ &= \begin{bmatrix} \mathbf{w}_1^T \mathbf{f}_1 & \mathbf{w}_1^T \mathbf{f}_2 & \ldots & \mathbf{w}_1^T \mathbf{f}_d \\ \mathbf{w}_2^T \mathbf{f}_1 & \mathbf{w}_2^T \mathbf{f}_2 & \ldots & \mathbf{w}_2^T \mathbf{f}_d \\ \ldots & \ldots & \ddots & \vdots \\ \mathbf{w}_k^T \mathbf{f}_1 & \mathbf{w}_k^T \mathbf{f}_2 & \ldots & \mathbf{w}_k^T \mathbf{f}_d \end{bmatrix} \end{aligned} \tag{16}$$

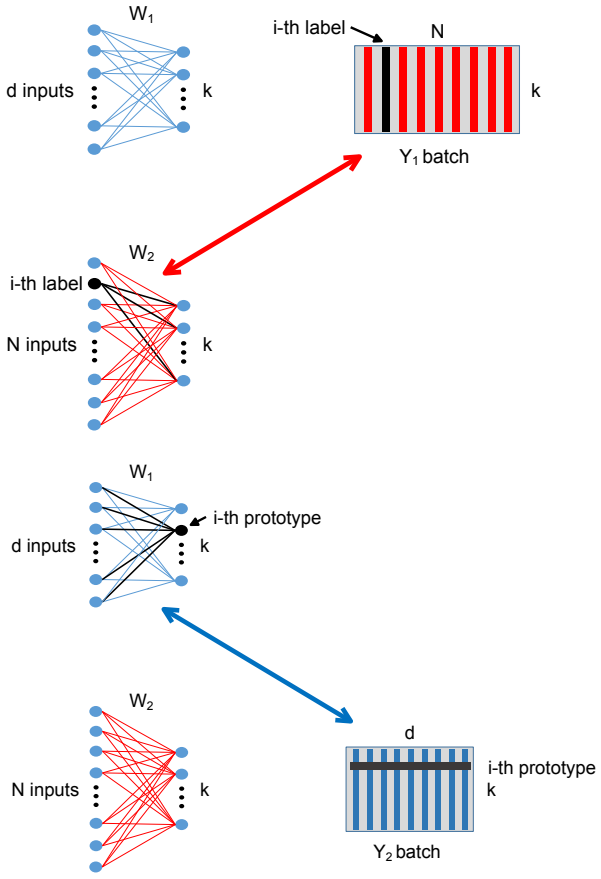

Fig. 2. Half dualities.

where $\mathbf{w}_i$ is the weight vector of the i-th output neuron of the dual network and $\mathbf{f}_i$ is the i-th feature over all samples of the input matrix X. The components of each prototype are computed using a constant weight $\mathbf{w}_i$, because P_2 is an outer product, which has rank 1. Besides, each component is computed as it were a one dimensional learning problem. For instance, the first component of the prototypes is $\left[\mathbf{w}_1^T \mathbf{f}_1 \ldots \mathbf{w}_k^T \mathbf{f}_1\right]^T$; which means that the first component of all the prototypes is computed by considering just the first feature $\mathbf{f}_1$. Hence, each component is independent from all the other features of the input matrix, allowing the forward pass to be just like a collection of d one-dimensional problems.

Such differences in the forward pass have an impact on the backward pass as well, even if the form of the loss function is the same for both systems. However, the parameters of the optimization are not the same. For the base network:

$$\mathcal{L} = \mathcal{L}(X, W_1) \tag{17}$$

while for the dual network:

$$\mathcal{L} = \mathcal{L}(X^T, Y) \tag{18}$$

where Y is a linear transformation (filter) represented by W_2. In the base competitive layer the gradient of the loss function

with respect to the weights W_1 is computed directly as:

$$\nabla \mathcal{L}(W_1) = \frac{d\mathcal{L}}{dW_1} \tag{19}$$

On the other hand, in the dual competitive layer, the chain rule is required to computed the gradient with respect to the weights W_2 as the loss function depends on the prototypes Y_2:

$$\nabla \mathcal{L}(W_2) = \frac{d\mathcal{L}}{dW_2} = \frac{d\mathcal{L}}{dY_2} \cdot \frac{dY_2}{dW_2} \tag{20}$$

As a result, despite the architecture of the two layers is equivalent, the learning process is quite different.

D. Qualitative analysis and comparison

The differences outlined in the previous subsection may have an impact in favoring one or the other layer depending on the problem. The main advantage of using the base competitive layer consists in a lower computational cost, as the forward pass is not required and the backward pass is much easier to compute. Besides, for low dimensional datasets, i.e. when $N \gg d$, the size of the weight matrix W_1 is $d \times k$, while the size of W_2 is $N \times k$. This means that the number of parameters of the dual network is much higher with respect to the base layer, leading to a even higher computational cost. However, by having more parameters, the dual layer may have an advantage in terms of flexibility and in finding better local minima. On the other hand, in high-dimensional settings, when $N \ll d$, the matrix W_1 is much larger than W_2. Hence, by having fewer parameters to optimize, the dual layer behaves like a system with a larger set of constraints, leading to smoother gradient flows and less overfitting.

Furthermore, another reason why the dual layer might be less sensitive to the number of features may depend on its learning process. Indeed, the forward pass decouples the original problem into a set of d one-dimensional problems, while the loss function and the gradient perform the coupling of such problems. Finally, by considering how the two layers build their prototypes, the dual network seems more suitable for joining with deep neural networks. Indeed, the base network is an atypical layer as it does not perform a forward pass at all. The dual network, instead, is more similar to a regular layer as it applies a linear transformation to its input. This latter linear map could also be generalized to a nonlinear transformation by stacking a set of dense layers with nonlinear activation functions.

E. Deep dual clustering

The fact that the dual layer is designed for using the gradient of the loss function for training allows to back-propagate to other previous layers in order to preprocess implicitly the training set. The same can be said for the base network. However, the latter is not a true layer. Indeed, it is simply a minimization process in which the weights are directly estimated. For this reason, the output has no meaning. Instead, the dual one has meaningful outputs and, so, has the same nature of the blocks composing a deep neural network.

The deep dual network is composed of a stack of fully-connected layers. The first layer is fed with the transpose of the input matrix X and the last layer is the dual linear layer. All the hidden layers have non linear activation functions (*tanh*), but the output layer is linear. This approach allows the invariance of the feature dimension at each layer. Instead, it is the number of samples that changes at each step. In this way, clustering centroids are directly estimated in the original input space, despite the fact that they are pre-processed in the hidden layers.

F. Extension to topological clustering

Topological clustering refers to a class of techniques where cluster centroids are connected during the learning process such that a Delaunay triangulation of the data manifold is induced. One of the most common approaches employs CHL at this purpose: If two prototypes are the two closest centroids for the same sample, an edge is created between them, representing their mutual relationship outlined by the common neighborhood.

The framework developed in the previous section can be easily adapted to accommodate for this kind of learning task. In this sense, the loss function main term represents a clustering index, the quantization error or the ratio between inter- and intra-cluster distances. However, in order to learn the minimal topological relationship, the loss function can be augmented by a Lagrangian term accounting for the complexity of the network connecting prototypes. At the end of each epoch, the adjacency matrix E, which represents the connections between prototypes using CHL, is computed and its norm is also included in the loss function. The gradient of the resulting loss can be computed in order to optimize prototypes' positions such that the complexity of the connections is minimized. The overall loss function looks like:

$$\mathcal{L} = \mathcal{Q} + \lambda ||E||_2 \tag{21}$$

where $\mathcal{Q}$ is the quantization error (average squared Euclidean distance between samples and corresponding centroids) and E is the adjacency matrix representing the connections between prototypes. At the end of the learning process, prototypes without connections and with an empty Voronoi set can be pruned. Hence, the number of output units in the last layer represents an upper bound of the number of valid prototypes, as the neural network will automatically prune redundant centroids.

III. Experimental evaluation

In order to validate the theory with non-trivial experiments and to analyze the differences of the two learning approaches, the base competitive layer and its dual network are compared on three synthetic datasets containing clusters of different shapes and sizes. Table I summarizes the main characteristics of each experiment. The first dataset is composed of samples drawn from a two-dimensional Archimedean spiral (*Spiral*). The second dataset consists of samples drawn from two half semicircles (*Moons*). The last one is composed of two concentric circles (*Circles*). Each dataset is normalized by removing

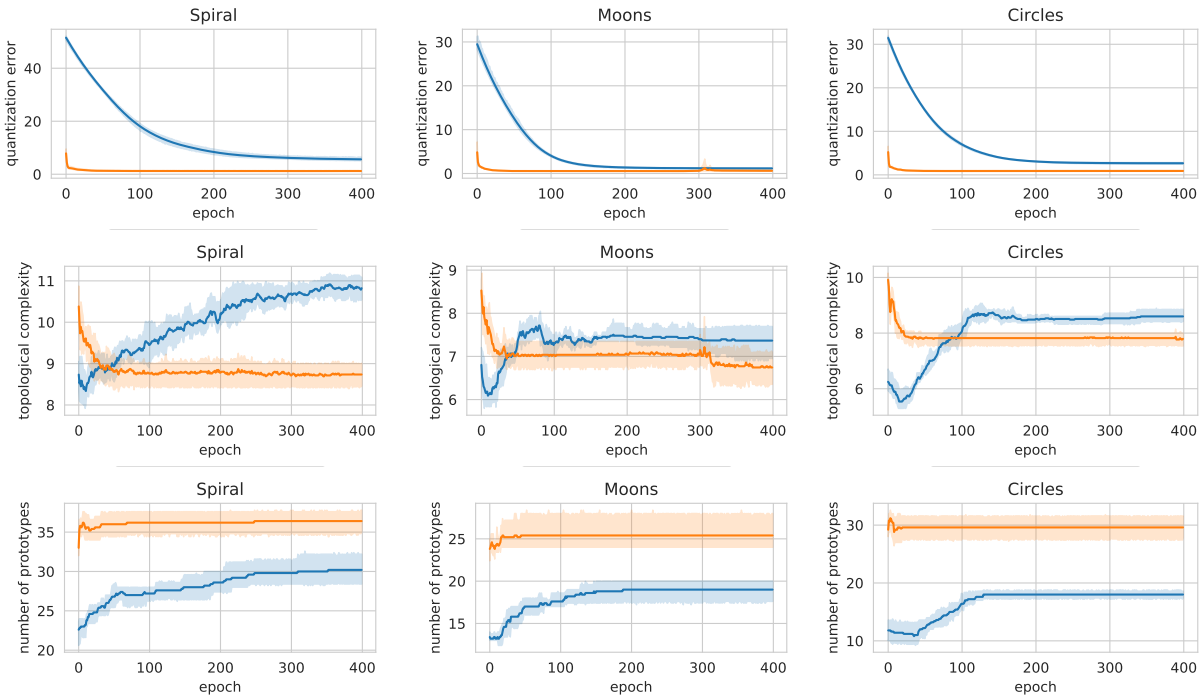

Fig. 3. Comparison VCL (**blue line**) and DCL (**red line**) over 10 runs on three metrics: the quantization error (**top row**), the norm of the matrix of the edges (**middle row**), and the number of valid prototypes (**bottom row**). The metrics are computed on three different datasets: *Spiral* (**left column**), *Moons* (**middle column**), and *Circles* (**right column**).

the mean and scaling to unit variance before fitting neural models. For all the experiments, the number of output units k of the dual network is set to 30. A grid-search optimization is conducted for tuning the hyperparameters. The learning rate is set to $\epsilon = 0.008$ for the base competitive layer and to $\epsilon = 0.0008$ for its dual layer. Besides, for both networks, the number of epochs is equal to $\eta = 400$ while the Lagrangian multiplier to $\lambda = 0.01$. For each dataset, both networks are trained 10 times using different initialization seeds in order to statistically compare their performance.

TABLE I
SYNTHETIC DATASETS USED FOR THE EXPERIMENTS.

DATASET	SAMPLES	FEATURES	CLUSTERS
SPIRAL	500	2	1
MOONS	500	2	2
CIRCLES	500	2	2

Qualitative results are presented in Figure 4. The solutions provided by the base competitive layer are shown in the first row, while the dual network ones are in the second row. Nodes (prototypes) belonging to the same connected component are linked with edges according to their neighborhood. Samples are represented with different colors depending on the cluster they belong to. Qualitative considerations considering the location of prototypes suggest that good clustering performance can be obtained using both networks. However, as shown in Figure 3, the dual network tends to propose solutions using a slightly higher number of prototypes, thus finding better connections between them and providing a superior representation of the underlying topology, especially considering the *Spiral* and the *Circles* datasets, where clusters are well separated.

In order to assess the main characteristics of the learning process, several metrics are evaluated while training the two networks on the three benchmark datasets. Figure 3 shows for each dataset a comparison between the base layer and its dual on three key metrics: the quantization error, the topological complexity of the solution (i.e. the norm of the edge matrix), and the number of valid prototypes (i.e. the ones with a non-empty Voronoi set). The main differences between the two approaches are outlined by the quantization error. Both networks seem to converge to similar local minima

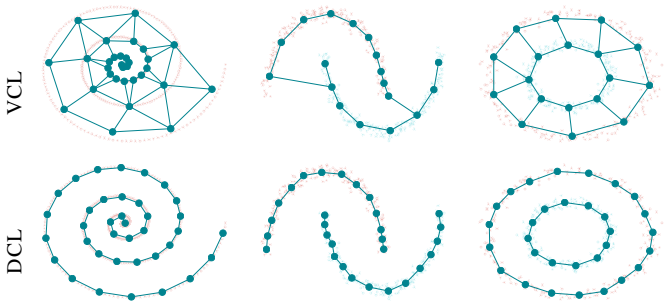

Fig. 4. Experiments on synthetic datasets. From left to right: *Spiral*, *Moons*, and *Circles* dataset.

in all scenarios, thus validating their theoretical equivalence. Nonetheless, the single-layer dual network exhibits a much faster rate of convergence compared to a standard competitive layer. The training of the dual network appears much more stable as outlined by a much lower variance of the quantization error.

A. An application to high-dimensional clustering

Here the performance of the standard competitive layer and its dual network in tackling high dimensional problems is assessed. Sure enough, standard distance-based algorithms generally suffer the well-known curse of dimensionality when dealing with high-dimensional data. Therefore, the intuition described in previous Section about dual-layer performances in this scenario is evaluated by working with an increasing number of features and a fixed number of samples. The MADELON algorithm proposed in [36] is used to generate the high-dimensional datasets. This algorithm creates clusters of points normally distributed about vertices of an n-dimensional hypercube. An equal number of cluster and data is assigned to two different classes. Both the number of samples (n_s) and the dimensionality of the space (n_f) in which they are placed can be defined programmatically. More precisely, the number of samples is set to $n_s = 100$ while the number of features ranges in $n_f \in [1000, 2000, 3000, 5000, 10000]$. The number of required centroids is fixed to one tenth the number of input samples. Three different networks are compared: the base network (VCL), a single dual layer network (DCL), and a deep variant of the dual network with two hidden layers of 10 neurons each (deep-DCL). Results are averaged over 10 repetitions on each dataset. Accuracy for each cluster is calculated by considering true positive those samples belonging to the class more represented and false positive the remaining data. As shown in the top plot of Fig. 5, VCL accuracy already drops when the number of feature is higher than 1000. DCL and deep-DCL, instead, are more capable to deal with high-dimensional data and their accuracy remains near 100% until 2000 and 3000, respectively. Nevertheless, all the proposed methods struggle when dealing with higher-dimensional data.

A further experiment is also performed in order to check whether the opposite scenario holds true - i.e. that the DBGC layer was not suitable for working with a high number of samples (corresponding to a high number of network inputs). In order to do that we repeated the experiment on the MADELON dataset by setting a fixed number of features $n_f = 100$, while working with an increasing number of samples $n_s \in [10^2, 10^3, 10^4]$. In the bottom plot of Fig.5, it is shown that notwithstanding a higher computational complexity, DBGC and deep-DBGC are still capable to find a perfect quantization even when dealing with a very high number of samples.

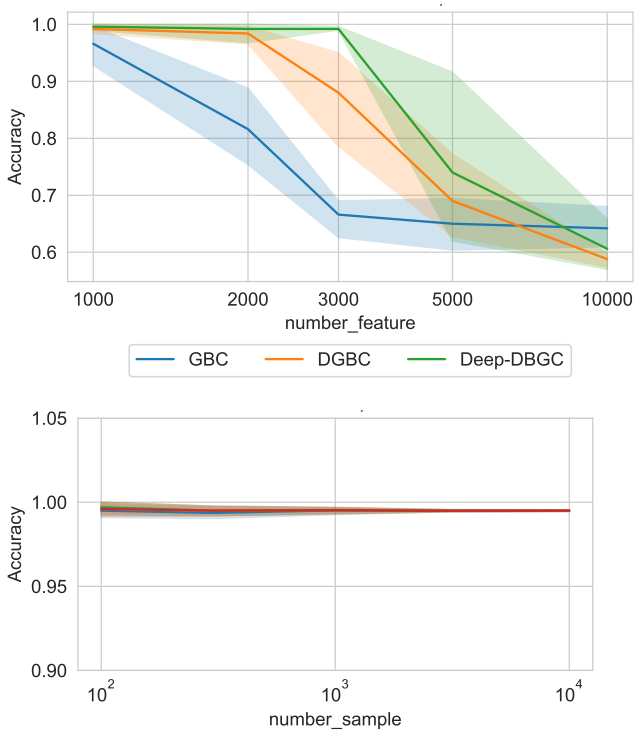

Fig. 5. Accuracy of the VCL and the DCL layer, and the deep-DCL network tested while working with fixed number of samples and an increasing number of features (**top**) and a fixed number of features and an increasing number of samples (**bottom**) on the synthetic MADELON dataset [36]. Error bands represent the standard error of the mean.

IV. Conclusion

This work sketches a novel interpretation of topological competitive learning using backpropagation. The foundation of a new theory is provided bridging two research fields which are usually thought as disjointed: gradient-based learning and unsupervised competitive neighborhood-based learning. This theory may represent the basis for a comprehensive reinterpretation of supervised and unsupervised learning with neural networks. Besides, as outlined in the experimental section, the framework can be easily extended to integrate complex topological structures and relationships among prototypes. The two novel competitive layers presented in this work represent the first steps towards the integration of competitive and topological learning with deep neural architectures, outlining the power and flexibility of the approach paving the way towards more advanced and challenging learning tasks such as: topological nonstationary clustering , hierarchical clustering , core set discovery , incremental and attention-based approaches, or multi-objective optimization of a latent space with topological constraints.

Software

All the code has been implemented in Python 3, relying upon open-source libraries [37], [38]. All the experiments have been run on the same machine: Intel® Core™ i7-8750H 6-Core Processor at 2.20 GHz equipped with 8 GiB RAM.

To enable code reuse, the Python code for the mathematical models including parameter values and documentation will be freely available under Apache 2.0 Public License from a GitHub repository. Unless required by applicable law or agreed to in writing, software will be distributed on an ”as is” basis, without warranties or conditions of any kind, either express or implied.

REFERENCES

[1] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[2] G. Mclachlan and K. Basford, *Mixture Models: Inference and Applications to Clustering*, 01 1988, vol. 38.

[3] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[4] R. Sibson, "Slink: an optimally efficient algorithm for the single-link cluster method," *The computer journal*, vol. 16, no. 1, pp. 30–34, 1973.

[5] D. Defays, "An efficient algorithm for a complete link method," *The Computer Journal*, vol. 20, no. 4, pp. 364–366, 01 1977. [Online]. Available: https://doi.org/10.1093/comjnl/20.4.364

[6] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[7] G. Cirrincione, G. Ciravegna, P. Barbiero, V. Randazzo, and E. Pasero, "The gh-exin neural network for hierarchical clustering," *Neural Networks*, vol. 121, pp. 57–73, 2020.

[8] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[9] T. Martinetz, K. Schulten *et al.*, "A" neural-gas" network learns topologies," 1991.

[10] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in neural information processing systems*, 1995, pp. 625–632.

[11] ——, "A self-organizing network that can follow non-stationary distributions," in *International conference on artificial neural networks*. Springer, 1997, pp. 613–618.

[12] E. J. Palomo and E. López-Rubio, "The growing hierarchical neural gas self-organizing neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 2000–2009, 2017.

[13] E. J. Palomo, M. A. Molina-Cabello, E. López-Rubio, and R. M. Luque-Baena, "A new self-organizing neural gas model based on bregman divergences," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[14] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[15] T. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," in *International conference on artificial neural networks*. Springer, 1993, pp. 427–434.

[16] R. H. White, "Competitive hebbian learning," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. ii, 1991, pp. 949 vol.2–.

[17] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Networks*, vol. 7, no. 3, pp. 507–522, 1994.

[18] B. Fritzke, "Some competitive learning methods," *Artificial Intelligence Institute, Dresden University of Technology*, 1997.

[19] P. Barbiero, G. Squillero, and A. Tonda, "Modeling generalization in machine learning: A methodological and computational study," 2020.

[20] N. Altman and M. Krzywinski, "The curse (s) of dimensionality," *Nature Methods*, vol. 15, pp. 399–400, 2018.

[21] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[23] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," Colorado Univ at Boulder Dept of Computer Science, Tech. Rep., 1986.

[24] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[25] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[26] G. Chen, "Deep learning with nonparametric clustering," *arXiv preprint arXiv:1501.03084*, 2015.

[27] C.-C. Hsu and C.-W. Lin, "Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 421–429, 2017.

[28] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," *arXiv preprint arXiv:1702.08720*, 2017.

[29] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[30] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5879–5887.

[31] P. Hartono, P. Hollensen, and T. Trappenberg, "Learning-regulated context relevant topographical map," *IEEE transactions on neural networks and learning systems*, vol. 26, 12 2014.

[32] P. Hartono, "Mixing autoencoder with classifier: conceptual data visualization," *IEEE Access*, vol. 8, pp. 105 301–105 310, 2020.

[33] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive science*, vol. 9, no. 1, pp. 75–112, 1985.

[34] H. B. Barlow, "Unsupervised learning," *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[35] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice-Hall, Inc., 2007.

[36] I. Guyon, "Design of experiments of the nips 2003 variable selection benchmark," in *NIPS 2003 workshop on feature extraction and feature selection*, vol. 253, 2003.

[37] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.