Clustering Through Hybrid Network Architecture With Support Vectors

Emrah Ergul, Nafiz Arica, *Member, IEEE*, Narendra Ahuja, *Fellow, IEEE*, and Sarp Erturk, *Senior Member, IEEE*

***Abstract*— In this paper, we propose a clustering algorithm based on a two-phased neural network architecture. We combine the strength of an autoencoderlike network for unsupervised representation learning with the discriminative power of a support vector machine (SVM) network for fine-tuning the initial clusters. The first network is referred as prototype encoding network, where the data reconstruction error is minimized in an unsupervised manner. The second phase, i.e., SVM network, endeavors to maximize the margin between cluster boundaries in a supervised way making use of the first output. Both the networks update the cluster centroids successively by establishing a topology preserving scheme like self-organizing map on the latent space of each network. Cluster fine-tuning is accomplished in a network structure by the alternate usage of the encoding part of both the networks. In the experiments, challenging data sets from two popular repositories with different patterns, dimensionality, and the number of clusters are used. The proposed hybrid architecture achieves comparatively better results both visually and analytically than the previous neural network-based approaches available in the literature.**

***Index Terms*— Autoencoder (AE) network, clustering neural networks, greedy layerwise learning, prototype encoding (PE) network, support vector machine (SVM).**

I. Introduction

DATA clustering deals with the problem of grouping a set of unlabeled data into subsets, such that the samples within the same subset are more similar to each other than the ones that belong to the other groups [1], [2]. The main objective is to find latent patterns reflected by the underlying parameters, which maximize the likelihood of the input data to their assigned clusters. Clustering is widely used before the ultimate goal of classification or regression and can be implemented for various purposes such as feature extraction, image segmentation, dimension reduction, and function approximation.

Although clustering techniques in the literature are diverse, we can group them into hierarchical and partitional methods based on the task-specific applications. Hierarchical clustering methods aim to build an agglomerating or divisive hierarchy of patterns. Partitional algorithms, on the other hand, determine independent clusters at once. However, both the methods are subject to a common goal of optimizing the parameters of the data model. The parameters represent the hidden patterns of each cluster, and are basically the data statistics calculated by some metric. Based on the type of statistics, distance measures can be used for (dis)similarity computation in a feature space, or the probability is computed for density estimation of posterior distributions in the same manner. Thereafter, association criteria, which are called objective functions, are employed to find the optimum solution to the clustering problem.

Among graph- and tree-based structures, neural networks have justifiable popularity because they are flexible to simulate any learning algorithm for hierarchical or partitional clustering. Neural networks can be explained as the multilayer structures that are formed by the composition of successive nonlinear transformations of the input data. They aim to implicitly achieve intermediate feature representations of the original data in deeper layers. Although neural networks are initially designed for supervised learning, recent research reveals their strength in the unsupervised learning. Autoencoder (AE) and restricted Boltzmann machines can be referred as the building blocks of unsupervised representation learning in neural networks. The details about such architectures, their variants, and implementations are available in [3]–[7].

In the literature, neural network-based clustering methods mainly build a two-layer network with lateral inhibitive connections in the output layer. They typically use the prototype/centroid vectors directly as the weight parameters between the input and output layers, since each neuron in the output layer is assumed to be a cluster. Inhibition in the output layer simply determines the characteristics of the cluster updating procedure, where for each input, only the closest centroid or all the centroids in a receptive field are updated in a weighting scheme. In general, gradient-based algorithms are used to minimize the cost of assigning data samples to their predicted prototypes iteratively.

In this paper, we propose a hybrid network architecture for partitional clustering that combines two complementary networks with different objective functions to optimize the solution. The proposed architecture embodies both the unsupervised representation learning for the initial clustering and the discrimination power of the support vector machine (SVM) for cluster fine-tuning. In the first network, we introduce the novel concept of a prototype encoding (PE) network that

Manuscript received December 29, 2014; revised December 30, 2015 and February 21, 2016; accepted March 10, 2016. This work was supported by the Scientific and Technological Research Council of Turkey under Grant 2214-B.14.2.TBT.0.06.01-214-83.

E. Ergul and S. Erturk are with the Electronics and Communication Engineering Department, Kocaeli University, İzmit 41380, Turkey (e-mail: 106103002@kocaeli.edu.tr; sertur@kocaeli.edu.tr).

N. Arica is with the Computer Engineering Department, Bahçeşehir University, Istanbul 34353, Turkey (e-mail: nafiz.arica@eng.bahcesehir.edu.tr).

N. Ahuja is with the Computer Vision and Robotics Laboratory, Beckman Institute, University of Illinois at Urbana–Champaign, Champaign, IL 61801 USA (e-mail: n-ahuja@illinois.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2542059

attempts to map each input sample to its closest prototype vector in an unsupervised manner. The encoding weights of the PE network are initialized as orthogonal vectors to Voronoi-like hyperplanes that separate the clusters in a pairwise manner. To update the clusters, we utilize a topology preserving grid structure with a neighboring function defined in a latent feature space. The second network with two layers is based on the SVM that basically tries to map the input samples to their currently assigned cluster labels. Each neuron in the output layer represents a cluster, and the network parameters are optimized using a one-vs-all scheme so as to maximize the margins between clusters. The SVM network uses the current prototype vectors as landmarks in the kernel function.

In the last part of our work, a hybrid network structure is established in a greedy layerwise clustering. Basically, we have two by-products of the first clustering network, i.e., PE. The hidden layer activations represent the data in a new and probably more discriminative feature space, and they are used as input to the SVM network. Furthermore, the cluster assignments of the PE network are utilized in the SVM network as *a priori* knowledge. To conclude, the SVM network fine-tunes the previously established clusters in a supervised manner, and this greedy layerwise scheme enhances the clustering performance remarkably.

Contributions: The contributions of this paper are twofolded. First, we introduce a PE network scheme for data clustering. The PE maps the input data to the prototype vectors in the output layer and updates the clusters in a latent space, established at the hidden layer. PE is clearly different from AE [3]–[5], where the input is associated with itself in the output layer. AE approaches may increase the variance in the output layer, and identity functions can be learned when the number of neurons in the hidden layer is more than the input dimensionality. In addition, we initialize the encoding weights of the PE network in a Voronoi-like manner, which separates the clusters in a pairwise manner. This initialization not only achieves a better starting point for the encoding weights than random selection for PE network but also it dynamically computes the number of hidden neurons with respect to the randomly initialized prototypes. Second, a greedy layerwise clustering scheme is proposed where an SVM network is used for fine-tuning. After optimizing the PE network, the input data in a new feature space are passed through the SVM network where the cluster assignments of PE are used as the initial supervision. Finally, only the current centroid vectors are utilized as the landmarks in the kernel function to reduce the complexity of traditional SVM learning.

The rest of this paper is outlined as follows. In Section II, we summarize the related studies based on clustering network architectures. Section III gives the details of the PE and SVM networks, and explains the method they are combined for the greedy layerwise clustering in a hybrid topology. The experimental results performed on the public data sets are given both visually and numerically in Section IV. Finally, we conclude this paper with the clear findings of the experiments and possible future work about the clustering network algorithms.

II. Related Work

The initial work on clustering neural networks starts with a simple competitive learning (SCL) network [2], [8], [9]. SCL is based on a two-layer fully connected network that implements the nearest neighbor paradigm. Basically, the input data, $X = \{\boldsymbol{x^{(i)}} | i = 1, 2, 3, \ldots, N\}$ where $\boldsymbol{x^{(i)}} \epsilon R^d$ are approximated by a finite number of prototypes and $C = \{\boldsymbol{c^{(j)}} | j = 1, 2, 3, \ldots, K\}$ where $K \ll N$ and $\boldsymbol{c^{(j)}} \epsilon R^d$, in the output layer. The SCL objective is to minimize the mean squared error (MSE)

$$E = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} \frac{1}{2} \mu_{ij} \|x^{(i)} - c^{(j)}\|^2; \quad \mu_{ij} \in \{0, 1\} \tag{1}$$

where μ_{ji} is the Kronecker delta that takes the value of 1 when $\boldsymbol{c^{(j)}}$ is the closest (i.e., winning) prototype to the data sample $\boldsymbol{x^{(i)}}$ in the L_2-norm metric and 0 otherwise. This nearest prototype condition is called the winner-takes-all (WTA) process, where Kronecker delta prevents updating the other prototypes and only the winning prototype $\boldsymbol{c^{(w)}}$ is updated by each input $\boldsymbol{x^{(i)}}$. The weights between input–output layers represent the prototype vectors C, and the objective function is optimized by the gradient descent method. Thus, the SCL network updates the prototypes iteratively as

$$\forall x; \quad c^{(w)} = c^{(\text{argmin}_{j=1,2,3,\ldots,K} \|x^{(i)} - c^{(j)}\|)} \tag{2}$$

$$\frac{\partial E}{\partial c^{(w)}} = -[x^{(i)} - c^{(w)}(t)] \tag{3}$$

$$c^{(w)}(t+1) = c^{(w)}(t) + \eta(t)[x^{(i)} - c^{(w)}(t)] \tag{4}$$

$$c^{(j)}(t+1) = c^{(j)}(t); \quad j \neq w \tag{5}$$

where $\eta(t)$ is a positively small learning rate, usually chosen to be decreasing gradually in time. The SCL network may be run either in the batch mode where the whole training data set is available or in the incremental mode for the online data. The number of clusters is predefined and cluster prototypes are randomly initialized as the network weight vectors. One of the drawbacks of the SCL network is that it is highly dependent on the randomly initialized cluster prototypes, since it updates only the winning one each time. The data sequence also impacts the updates in incremental learning.

Another popular clustering network is the Kohonen network, also called self-organizing map (SOM). The SOM has mainly the same network structure and objective as in the SCL [10]–[12]. However, unlike the SCL network where only the winning prototype is updated by each data sample, SOM introduces an excitation or neighborhood function that makes all the prototype vectors in a field centered at the winning prototype to be updated. First, an imaginary surface of a specific grid configuration is randomly initialized as the weight vectors C between input–output layers [24]. The grid configuration reflects static neighboring bindings between neurons at the output layer by a predefined metric such as Euclidean distance. Once the winning prototype of a sample $\boldsymbol{c^{(w)}}$ is selected for

the input sample, the update rule for all the prototype vectors in the neighborhood is

$$c^{(j)}(t+1) = c^{(j)}(t) + \eta(t) h_{jw}(t)[x^{(i)} - c^{(j)}(t)] \quad (6)$$

$$h_{jw}(t) = h_0 e^{\frac{-\|c^{(j)} - c^{(w)}\|^2}{\sigma^2(t)}} \quad \forall x \quad (7)$$

$$\sigma(t) = \sigma_0 e \frac{-t}{T}; \quad \sigma_0 \in (0, 1] \quad (8)$$

where $h_{jw}(t)$ is the neighborhood Gaussian function, which decreases as the distance from the winning prototype $\boldsymbol{c^{(w)}}$ increases. $\sigma(t)$ is the scale factor of the Gaussian that determines the excitation magnitude while $\eta(t)$ is the learning rate, decreasing in time, and h_0 is the initial neighboring constant in the range (0, 1]. The cluster update procedure in the SOM is not based on the minimization of an objective function. Therefore, it suffers from unguaranteed convergence, and it is often dependent on the data sequence.

Given the aforementioned disadvantages of SOM and SCL, a learning vector quantization (LVQ) network is proposed in [12] to fine-tune the unsupervised clustering networks with supervision. It is based on a supervised learning in that the previously found prototype vectors and their data assignments are assumed to reflect the priori density function of the input data for the LVQ network. It minimizes (1) as

$$c^{(w)}(t+1) = c^{(w)}(t) - (-1)^{y_w^{(i)}} \eta(t)[x^{(i)} - c^{(w)}(t)] \quad y_w^{(i)} \in \{0, 1\} \quad (9)$$

$$c^{(j)}(t+1) = c^{(j)}(t); \quad j \neq w \quad (10)$$

where the input data are now given by the pattern pairs of $P = \{(\boldsymbol{x^{(i)}}, y^{(i)})\}$, $\boldsymbol{x^{(i)}} \in R^d$, and $y^{(i)} \in \{0, 1\}^K$. The binary labels $y^{(i)}$ indicate the initial cluster assignments and they are fixed throughout the clustering procedure. Here, $\eta(t)$ is again the scalar learning rate (i.e., $0 < \eta(t) < 1$), which is decreasing in time. When we look in (9), LVQ simply resembles the SCL network in which only the winning prototype $\boldsymbol{c^{(w)}}$ is updated by each sample. But the minus sign in (9) the first equation is set to correct the winning prototype in the opposite direction if the sample $\boldsymbol{x^{(i)}}$ is clustered differently from the prior assumption. This helps to decrease the density around the prototypes on the Bayesian decision surfaces but it also triggers vector divergence [2].

For comparison, LVQ and our proposed SVM network are both supervised and used for fine-tuning. On the other hand, they mainly differ from each other in which the SVM network continues to update the cluster assignments in a latent space iteratively and it uses a max-margin objective function. Thus, the SVM network fine-tuning improves the clustering performance notably.

So far, the SCL and LVQ networks are introduced as WTA clustering algorithms where only the winning prototypes are updated in the unsupervised and supervised manners, respectively. SOM, on the other hand, excites all the prototype vectors for each sample with a neighboring function that updates the vicinity of the winning prototype. Although such a neighborhood updating scheme holds inspiration, SOM predefines static neighborhood relations between cluster prototypes. For this perspective, neural gas network (NGN) is proposed in [13]. The soft-max rule is employed as an extension to the standard k-means like clustering used in SOM. The earlier works also use the maximum entropy algorithm [14] in the updating step. NGN takes a dynamic neighborhood ranking into account among the prototype vectors with respect to each sample.

For the data sample $\boldsymbol{x^{(i)}}$, the neighboring order is established by using the Euclidean distance to all the prototype vectors C. Then, each prototype $\boldsymbol{c^{(j)}}$ is assigned to an integer rank value, $r_{ji} = 0, 1, 2, \ldots, K-1$, where K is the predefined number of prototypes and 0 indicates the winning prototype $\boldsymbol{c^{(j)}}$. The gradient descent-based clustering update rule is conducted as

$$c^{(j)}(t+1) = c^{(j)}(t) + \eta(t) h(r_{ji}, t)[x^{(i)} - c^{(j)}(t)] \quad (11)$$

$$h(r_{ji}, t) = e^{\frac{-r_{ji}}{\rho(t)}} \quad (12)$$

$$\eta(t) = \eta_0 \left(\frac{\eta_f}{\eta_0}\right)^{\frac{t}{T}}, \quad \rho(t) = \rho_0 \left(\frac{\rho_f}{\rho_0}\right)^{\frac{t}{T}} \quad (13)$$

where $h(r, t)$ is the ordering factor that takes rank r_{ji} and the characteristic decay constant $\rho(t)$ is used to update every prototype $\boldsymbol{c^{(j)}}$ with respect to a specific sample $\boldsymbol{x^{(i)}}$. Like $\eta(t)$, the decay constant $\rho(t)$ decreases gradually in time, in which η_0 and ρ_0 are the initial decay parameters, and η_f and ρ_f are the final decay parameters, respectively. NGN resembles the SOM network except the neighboring function where NGN uses integer ranks for the prototypes with respect to each sample instead of utilizing the distances between prototype vectors. By using this kind of dynamic ordering r_{ji} and the clustering shape modulator $\rho(t)$, NGN is assumed to reach a lower distortion error and be more robust to local minima than the others, without the problem of underutilization [2].

Other popular clustering networks in the literature are adaptive resonance theory (ART) [15] and fuzzy clustering (FC) networks [16]. The idea behind the ART network is to create a two-layer recurrent structure where the input layer $\mathbf{F_1}$, and the output layer $\mathbf{F_2}$ are fully connected in both directions with different weights. The feedforward weights $\boldsymbol{w^{(i)}}$ represent short-term memory that is used to find the winning prototype $\boldsymbol{c^{(w)}}$ in the output layer. The feedback weights $\boldsymbol{c^{(j)}}$ introduce long-term memory that actually indicates the prototype vector C. Although the ART network tends to get robust clusters in rapidly changing input sequences, the input signal and the weights are binary, it is sensitive to the feeding order of the input patterns, and its complexity is high due to differential equations. The FC network, on the other hand, uses the same structure of the SCL network. But it partitions the data set in a fuzzy manner, meaning that each data sample belongs to multiple clusters with a membership function, $\mu(\boldsymbol{x^{(i)}}$ and $\boldsymbol{c^{(j)}})$. Because the memberships of a sample for all the clusters correspond to the probabilities in the fuzzy set theory, they are restricted to the sum of unity. An alternating optimization procedure is established by first using the soft-max rule for updating memberships, thereafter taking fuzzy means of the data samples for updating the prototype vectors [2]. The memberships and cluster prototypes are initialized randomly. FC is not suitable for large-scale data sets because of high computational and storage requirements.

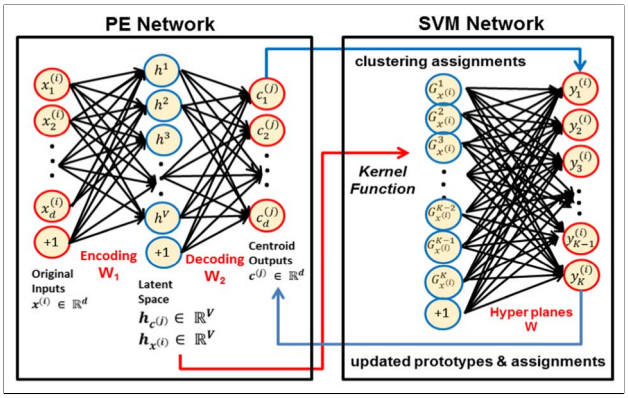


Fig. 1. Proposed hybrid neural network architecture for clustering.

With regard to the SVM-based approaches in clustering problems, there are two main concepts in the literature. The approaches in the first category basically focus on adopting their max-margin objective function to the one that helps finding spherical regions in the original feature space as clusters [25], [26]. The Lagrangian method is introduced into the objective function, which computes support vectors, representing hypothetical boundaries. Thereafter, the connected components are extracted using graph based algorithms for cluster labeling in an unsupervised manner. The approaches in the second category use SVM as a complementary tool to the main clustering algorithm. For example, in [27], k-means is implemented to discover the topical clusters of textual documents in the upper level. Meanwhile, an SVM classifier is constructed with the help of k-means supervision for each cluster by using other interrelated data sets. As the assignments change, SVM classifiers are updated in addition to the clusters. Nevertheless, our proposed SVM clustering network is completely different from the aforementioned methods in that it has a network structure that updates the clusters in a latent space, and it is used as a cluster fine-tuning scheme with its simple max-margin objective function.

III. Hybrid Network Architecture for Clustering

In this paper, we propose a hybrid network for clustering that embodies unsupervised representation learning in a new feature space and uses the power of SVM discriminants in an additional layer for cluster fine-tuning. The proposed architecture is shown in Fig. 1 and is composed of two parts. In the first part, we construct a three-layer network that maps the input data to their closest prototype vector. Encoding weights of the network are initialized by Voronoi-like hyperplanes that separate the randomly selected prototypes in a pairwise manner. This process also determines the number of neurons in the hidden layer implicitly. In addition, the hidden layer activations of each sample are restricted to its closest prototype activations in the training stage. The samples are assigned to their closest cluster in the latent feature space and not in the original domain. We call this PE network, and the details are given in Section III-A.

The second part is designed for fine-tuning the clusters found in a previous clustering method, such as a PE network. It gets the final assignments of the PE network as the initial cluster labels and the hidden layer activations of the data to be clustered as the new input signals. This network employs the concept of SVM. It takes the current prototype vectors as landmarks in its kernel function, and maximizes the margins between the cluster boundaries. This second part, called the SVM network in Section III-B, improves the separation between clusters. In both the networks, we use a SOM-like approach to iteratively update the clusters in the latent spaces where the hidden layer and the output layer are used for the PE network and the SVM network, respectively. Finally, a greedy layerwise algorithm is applied for optimizing the parameters of both the networks as a whole in Section III-C.

A. Prototype Encoding Network

PE is based on an idea similar to that of the AE network [3], [5], [17], where the input signal is simply mapped to itself in the output layer by the latent patterns in a hidden layer. However, unlike AE, the PE network is forced to iteratively couple each input sample to its closest prototype, instead of itself. Each cluster is represented by a prototype vector that is initialized randomly. The Euclidean metric is used for cluster assignments in a nearest neighbor fashion as in k-means [18].

Given the data sample-cluster pairs $X = \{(\boldsymbol{x^{(i)}}, \boldsymbol{c^{(j)}})\}$; $\boldsymbol{x^{(i)}}, \boldsymbol{c^{(j)}} \epsilon R^d$, $i = 1, 2, 3, \ldots, N$; $j = 1, 2, 3, \ldots, K$, where K is a user-defined parameter defining the number of clusters and N is the number of samples in the data set; the objective function is

$$\forall x;\, c^{(j)} = c^{(\text{argmin}_{k=1,2,3,\ldots,K}\|x^{(i)}-c^{(k)}\|)} \tag{14}$$

$$J_X(W,b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\|h_{W,b}(x^{(i)}) - c^{(j)}\|^2 + \beta\sum_{v=1}^{V}\left(h^v_{c^{(j)}}\log\frac{h^v_{c^{(j)}}}{h^v_{x^{(i)}}} + \left(1-h^v_{c^{(j)}}\right)\log\frac{1-h^v_{c^{(j)}}}{1-h^v_{x^{(i)}}}\right) + \frac{\lambda}{2}\sum_{W_1,W_2}\|W\|^2. \tag{15}$$

The objective function $J_x(\mathbf{W}, \boldsymbol{b})$ involves three terms: MSE, divergence, and weight regularization. MSE penalizes the differences between the predictions of the network $h_{\mathbf{W},\boldsymbol{b}}(\boldsymbol{x})$ and the desired output signals (i.e., the closest prototypes) $\boldsymbol{c^{(j)}}$. The divergence term forces both the input data and their closest prototypes to be similar in the latent space, and the latent space is now represented by the hidden-layer activations. This is achieved by restricting the hidden-layer activations h^v with an entropy-based formula, i.e., Kullback–Leibler (KL) divergence, $\text{KL}(h_c||h_x)$. Because we try to assign each sample to a cluster in this latent space instead of the original domain, it is a good intuition to add an approximation term to the objective function for adjusting the basis vectors of the target feature space. Weight regularization term, on the other side, tries to avoid over-fitting. β and λ are the tradeoff constants in KL divergence and the weight regularization, respectively.

Another important point in the PE network is the initialization of encoding weights (i.e., $\mathbf{W_1}$ in Fig. 2). We initialize the encoding weights using the Voronoi approach, which mainly

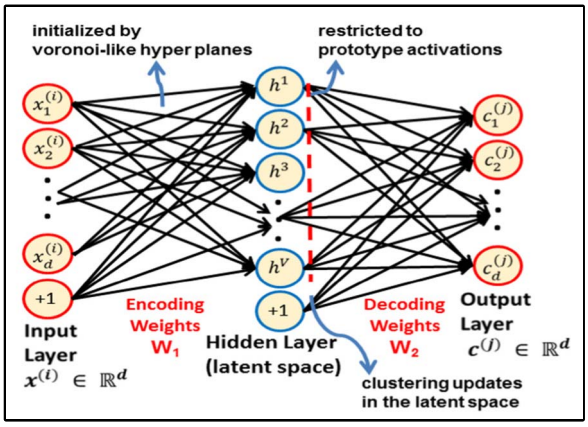


Fig. 2. PE network.

partition the feature space into a finite number of regions by the hyperplanes [2], [13]. Voronoi hyperplanes are established by taking the cluster centers as seed points. After selecting the prototypes randomly, each hyperplane is found by a vector orthogonal to the connecting plane between a pair of prototypes, and it intersects at the center point on this plane. However, such combinational hyperplanes lead to increase the computational complexity exponentially, since they represent the neurons in the hidden layer, many of which may be redundant. Therefore, we first sort the remaining prototype points in the increasing order of their Euclidian distances for each selected prototype. A hyperplane is then produced between the selected prototype and its closest one in a couplewise fashion. Since we already accept the computed hyperplane for the selected prototype with a minimum margin, those prototypes that are separated from the selected prototype vector by this hyperplane with even larger margin can be intuitively eliminated from the pairwise hyperplane construction. This procedure continues iteratively until there is no coupled prototype. This greedy method reduces the number of hyperplanes dramatically. Note that we not only initialize the encoding weights rationally but also determine the number of neurons in the hidden layer dynamically with respect to the relative positions of the randomly selected prototypes. The decoding weights $\mathbf{W_2}$ are randomly selected like the prototype vectors. The sigmoid function is used both in the hidden and output layers as a nonlinear activation function.

The optimization of network parameters is performed by the stochastic gradient descent algorithm. Partial error derivations are calculated as in the traditional feedforward–back propagation algorithm. Hence, the weight and bias parameters are updated in mini batches while keeping the prototype vectors, input data-cluster assignments, and the hidden layer restrictions fixed. After each passes through the whole data set to be clustered, the prototypes are updated. We use (six to eight) of the SOM network to update the prototype vectors where the winning prototype vector for each sample $\boldsymbol{c^{(w)}}$ and the factor parameter of the neighboring Gaussian function $h_{jw}(t)$ are computed in the latent space. The prototype update procedure, on the other hand, is still conducted in the original domain, since we use the same data in the input layer iteratively. Also note that updating the prototype vectors will also affect

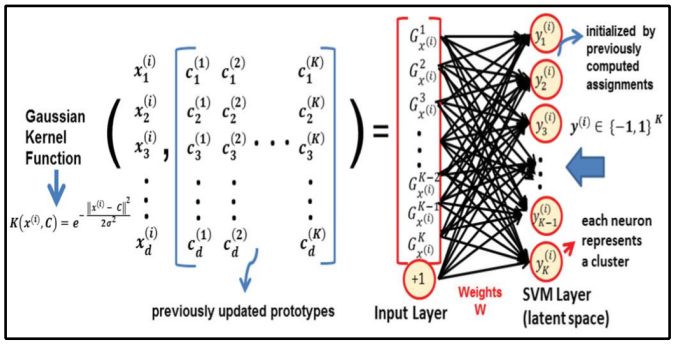


Fig. 3. SVM network for fine-tuning.

the data-cluster assignments and the hidden-layer restrictions for the next epoch. The intuition behind this procedure is that encoding weights represent hyperplanes, which are estimated as the cluster boundaries and the hidden layer activations for each sample are the distances from those hyperplanes to either positive or negative side. Because the samples are represented in a higher dimensional space by hidden-layer activations, they can be discriminated better. The update procedure continues iteratively until no further cluster changes are achieved or we reach an iteration bound.

B. SVM Network for Cluster Fine-Tuning

SVM is a powerful tool for supervised learning, which separates the feature space linearly into two categories: positive and negative. It tries to maximize the margin between the positive and negative sides [18], [20]. The margin in the SVM represents the gap between support vectors of both the sides, which are data samples acceptably close in limits to the opposite ones. The main idea is to find the optimum hyperplane that achieves the total minimum distance between the support vectors and the hyperplane. After training the SVM, a new sample is classified simply as either positive or negative by the result of the dot product with the optimized weight vector.

In fact, SVM is a linear discriminant function and it obviously does not handle nonlinearly separable data sets with a satisfactory accuracy. Kernel functions (Gaussian, polynomial, chi-square, histogram intersection, and so on) are introduced in the literature to establish the nonlinear SVM classifiers and they achieve a justifiable popularity with SVM. Actually, SVM still preserves its linearity in this perspective, but the input data are transferred into a new feature space by nonlinear kernel functions beforehand. Hence, we get more complex feature spaces with higher dimensionality instead of complicating the discriminant function itself.

In the second part of the introduced hybrid architecture, we propose a cluster fine-tuning concept that is based on the one-vs-all SVM topology for multiple clusters. The two-layer supervised clustering network is shown in Fig. 3. Given the input data $X = \{\boldsymbol{x^{(i)}}\}$ and the cluster prototypes $C = \{\boldsymbol{c^{(j)}}\}$, input signals for the SVM network are first achieved by the Gaussian kernel function (GKF) $K(\boldsymbol{x^{(i)}}, C)$

$$K(x^{(i)}, c) = e^{\frac{-\|x^{(i)} - c\|^2}{2\sigma^2}}; \quad c \in C \tag{16}$$

where σ is the scale parameter that factors the neighborhood. Therefore, each data sample is now represented by its GKF responses to the current prototypes (i.e., landmarks). Also note that the input dimensionality now equals to the number of clusters K. Because the SVM is a supervised learning algorithm, the sample cluster assignments obtained in a previous application are used as the initial data labels in the SVM layer for supervision. The unconstrained objective function of the SVM network is

$$J_{G_X,y}(W,b) = P\frac{1}{N}\sum_{i=1}^{N}\max(1-w^T G_{x^{(i)}}y^{(i)},0)^2 + \frac{1}{2}\sum\|W\|^2 \quad y^{(i)} \in \{-1,1\}^K \tag{17}$$

where P is the tradeoff constant, penalizing data points that violate the margin requirements. G_x represents the GKF output vector of (16) for each sample $\boldsymbol{x^{(i)}}$, which is the new input signal to the SVM network. $\mathbf{W}$ is the matrix, which embeds the parameter vectors, including biases. They are assumed to be orthogonal to the hyperplanes that separate K clusters and randomly initialized like the decoding weights of the PE network. As aforementioned, SVM simply separates the space into two parts. To extend the SVM network for a multicluster problem, we use the one-vs-all approach. For the K-cluster problem, K linear SVMs are independently trained, where the data samples from the other clusters form the negative cases [20]. Therefore, the desired output signal for each input $y^{(i)}$ is achieved by assigning 1 to the assigned cluster and -1 for the rest. The stochastic gradient descent algorithm is then employed as

$$h_W(G_x) = W^T G_x \tag{18}$$

$$\delta w = \frac{\delta J(W,b)}{\delta h_W} = -2Py\max(1-h_W(G_x)y,0) \tag{19}$$

$$\Delta w = \alpha\left(\frac{1}{N}\sum\left(G_x^T\delta w\right)+w\right) \tag{20}$$

$$w_{\text{new}} = w_{\text{old}} - \Delta w \tag{21}$$

where δw is the back-propagated derivation of the error signal per data sample and Δw is the average weight correction that includes L_2 regularization without bias terms. Also note that α is the learning rate and h_w is the hypothesis function of the network. The prediction procedure used by the SVM network is similar to that of the soft-max classifier. But unlike soft-max, where the derivative error is handled in a combined form by matrix multiplication, SVM neurons propagate the error back independently. This leads to computing the derivative errors by weight vectors $\boldsymbol{w^{(j)}}$ instead of using the whole matrix $\mathbf{W}$.

Once the SVM network is set up, we optimize the weight parameters iteratively in mini batches. The hyperplanes are updated with the max-margin objective function to separate the samples of each cluster based on the current sample-cluster assignments. After each epoch, we follow the cluster updating procedure similar to that of the PE network. Hence, the output layer activations of the SVM network are now used as the new latent space. Then the SOM-like updating scheme of (six to eight) for both the prototype vectors and the assignments are achieved in this latent space. Once we change the data sample-cluster assignments, the desired output signals $y^{(i)}$ are redefined for the next epoch accordingly. Also note that the kernel outputs are recomputed, since we have updated the prototype vectors as the *new landmarks*. Therefore, the input signal to the SVM network changes in each epoch although we still input the same data sequence. The termination conditions are the same as the PE network.

Algorithm 1 The proposed hybrid network architecture for clustering

Input:
Data to be clustered: $X = \{\boldsymbol{x^{(i)}}\}$, where $\boldsymbol{x^{(i)}} \in R^d$; i=1,2,3,…, N
Randomly initialized *prototype vectors:* $C = \{\boldsymbol{c^{(j)}}\}$, where $\boldsymbol{c^{(j)}} \in R^d$; j=1,2,3,…, K
Output:
Finalized data-cluster assignments: $D_c = \{\boldsymbol{x^{(i)}}, \boldsymbol{c^{(j)}}\}$

BEGIN PE network
  Initialize the network parameters, encoding & decoding weights: $\mathbf{W_1}$ & $\mathbf{W_2}$
    2.1. Initialize $\mathbf{W_1}$ by Voronoi-like hyper planes as in section *3.a*
      2.1.1. ($\mathbf{W_1}$, #NeuronsOfHiddenLayer) ← *InitW1(C)*
    2.2. Initialize $\mathbf{W_2}$ randomly
  Compute D_c in the hidden layer activations (i.e. latent space)
  FOR each epoch
    Update $\mathbf{W_1}$ & $\mathbf{W_2}$ by eq. *(6)* while keeping D_c fixed
    Update C and D_c in the latent space by eq. *(3)* while keeping $\mathbf{W_1}$ & $\mathbf{W_2}$ fixed
  END FOR
END PE network

BEGIN SVM network
  Initialize the network parameters, $\mathbf{W}$, randomly
  FOR each epoch
    Compute hidden layer activations of the PE network for X and C
    Compute Gaussian Kernel Function by eq. (7) for input signals to the SVM network
    Update $\mathbf{W}$ by eq. *(8) and* (9) while keeping D_c fixed
    Update C and D_c in the output layer activations by eq. *(3)* while keeping $\mathbf{W}$ fixed
  END FOR
END SVM network

Fig. 4. Pseudocode of the greedy layerwise clustering network.

C. Combining PE and SVM Networks in Hybrid Architecture

So far, two networks are independently presented for partitional (i.e., competitive) clustering and they have different subobjectives. The lateral purpose of the PE network is to learn a latent space in an unsupervised manner, which introduces more discriminative data representations for clustering. The SVM network, on the other hand, aims to fine-tune the prior clusters by maximizing the margins between these clusters. It also uses output layer activations as the new latent space for the cluster update procedure. Note that the number of neurons in each layer of both the networks is determined implicitly. We combine these two networks in a greedy layerwise clustering fashion, and the algorithm is pseudocoded in Fig. 4.

We first extract Voronoi-like initialized patterns in the encoding part of the PE network that leads to unsupervised feature learning in the hidden layer while we update the clusters iteratively. This process can be viewed as the expectation-maximization algorithm, where the network parameters are first updated when the prototype vectors are kept fixed during the expectation step. Thereafter, the clusters are updated in the latent space during the maximization step with the fixed encoding weights, representing the basis vectors of this new feature space.

In the SVM network part, instead of using the input data in its original domain, the hidden layer activations of the PE network with respect to both the input data $\boldsymbol{x^{(i)}}$ and the

TABLE I
SUMMARY OF DATA SETS USED IN THE EXPERIMENTS

	Synthetic Datasets					UCI Datasets			
	S_1	S_2	S_3	S_4	Aggregation set	wine_red	wine_white	letter recognition	clave_vectors
# samples	5000				788	1599	4898	20000	10800
# clusters	15				7	6	7	26	4
# dims	2				2	11	11	16	16
ref. #	[22, 23]				[22, 29]	[30, 31]		[33]	[32]

current prototype vectors $\boldsymbol{c}^{(j)}$ are computed as

$$H(\varphi) = \mathrm{Sigm}\left(W_1^T \varphi\right) = \frac{1}{1 + e^{-W_1^T \varphi}} \tag{22}$$

$$H_x = \{h(x^{(i)}) = h_{x^{(i)}} | i = 1, 2, 3, \ldots, N; \quad h_x \in R^v\} \tag{23}$$

$$H_c = \{h(c^{(j)}) = h_{c^{(j)}} | j = 1, 2, 3, \ldots, K; \quad h_c \in R^v\}. \tag{24}$$

The final data-cluster assignments of the PE network are also used in the SVM network for the initial supervision. In addition, the active input data for the SVM network are computed by the GKF that handles current prototype vectors as the *landmarks*. Afterward, the cluster fine-tuning procedure is implemented with the max-margin objective. Once prototype vectors and data-cluster assignments are updated with the one-vs-all SVM network in its latent space for each epoch, the new assignments are then fed to the output layer of the SVM network. $\boldsymbol{H_c}$ is recomputed in the encoder part of the PE network, since we update the prototype vectors. Note that we do not change the parameters of the PE network because we have finalized learning it already. This procedure continues until equilibrium is achieved with a given error threshold or the iteration bound is reached.

IV. EXPERIMENTAL RESULTS

We compare the clustering results of the proposed network architectures with the widely used clustering networks: SCL, LVQ, SOM, and NG over two types of performance: visual and analytical. To do so, we use nine different data sets that are summarized at Table I. Note that they have varying number of data samples, clusters, and dimensions in wide ranges, and different repositories (i.e., UCI machine learning, and speech and image processing) are also explored. Therefore, we can achieve an idea of how the algorithms behave comparatively in such diverse inputs throughout the experiments. Basically, we use the synthetic data sets to enable the visualization of the clustering results, while the others are handled for analytical performance comparisons, which are among the popular data sets in the UCI machine learning repository.

It is a common practice to perform several preprocessing steps on the raw data before clustering. In this paper, we apply the whitening method of [3] to get uncorrelated features with the same variance, and the input is normalized into the range of (0, 1] at the preprocessing stage. The clustering network algorithms to be compared with our proposed works are implemented in accordance with their original papers [2], [11]–[13] and the formulas in Section III. Since all the methods have different number of parameters to be set, we apply grid search in varying ranges to find the optimum configurations on a held-out validation set, and they are kept fixed during experiments. Except LVQ and SVM networks, which are supervised methods, the clustering algorithms start with the same cluster prototypes that are randomly selected. Meanwhile, we use SCL results for the LVQ network initialization, while the updated clusters of the PE network are fed into the SVM fine-tuning network as proposed.

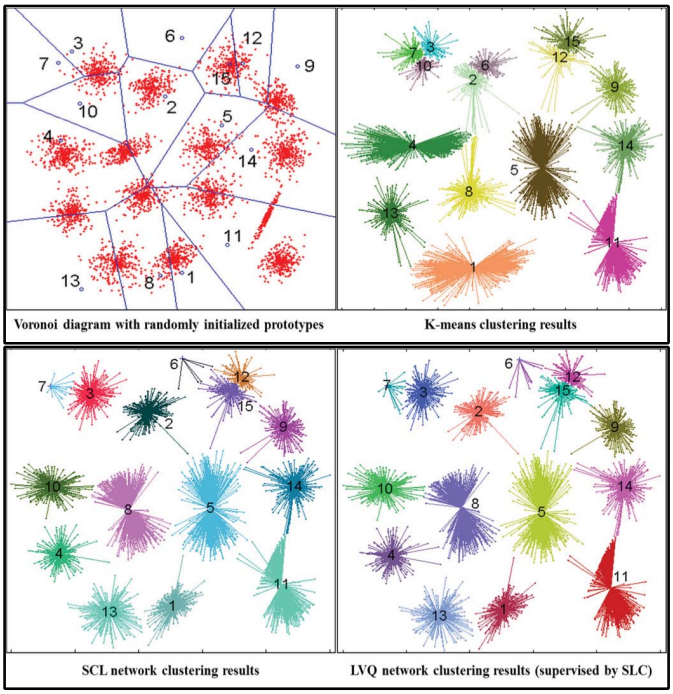


Fig. 5. Clustering results of the WTA networks on S_1-set.

We repeat the experiments 50 times, and the mean and the standard deviations in the numerical results are also noted for each data set to justify the visual results in consistency. They will be covered in Section IV-B. With regards to visual results, we select the common and significant clustering performances among these trials to display the differences between clustering algorithms. For Tables II–VII, only the best results are displayed in bold font for a better comparison.

A. Visual Clustering Analysis

The aggregation and S ($S_1 - S_4$) data sets, which we use in the visual experiments, have individual challenges, since the cluster characteristics (i.e., shape, size, relative closeness, and scatter) vary in 2-D spatial distribution. Aggregation set is more compact than S sets, such that the clusters in the visual domain can be easily selected.

The same prototype vectors that we randomly initialize are used in all the algorithms at the start, and the final results are achieved after 100 iterations. We mainly group the clustering networks into the WTA and the neighborhood function-based (NFB) methods of which the visual results are shown in Figs. 5 and 6, respectively.

In Fig. 5, we first give Voronoi tessellation of the randomly selected prototypes to have a clear idea about initialization.

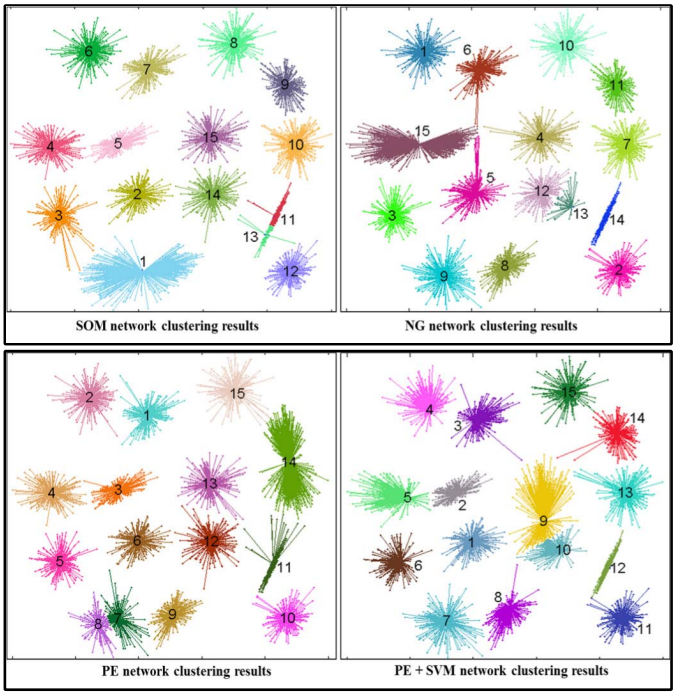


Fig. 6. Clustering results of the NFB networks on S_1 set.

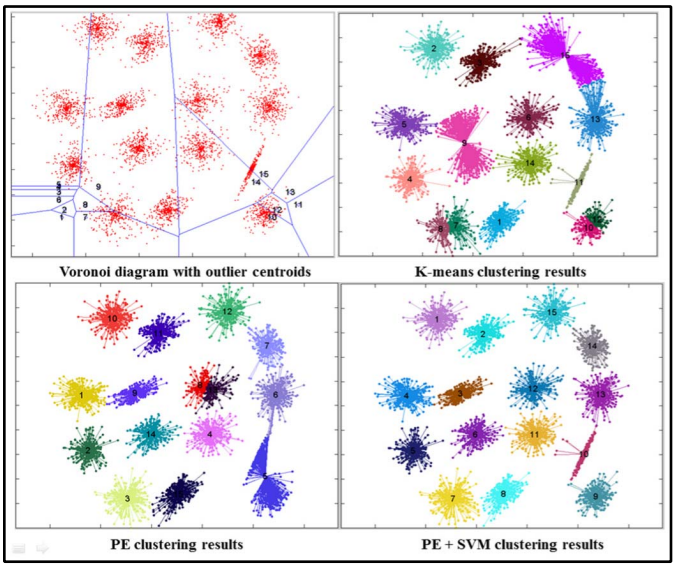


Fig. 7. Clustering results with noisy initialization on S_1 set.

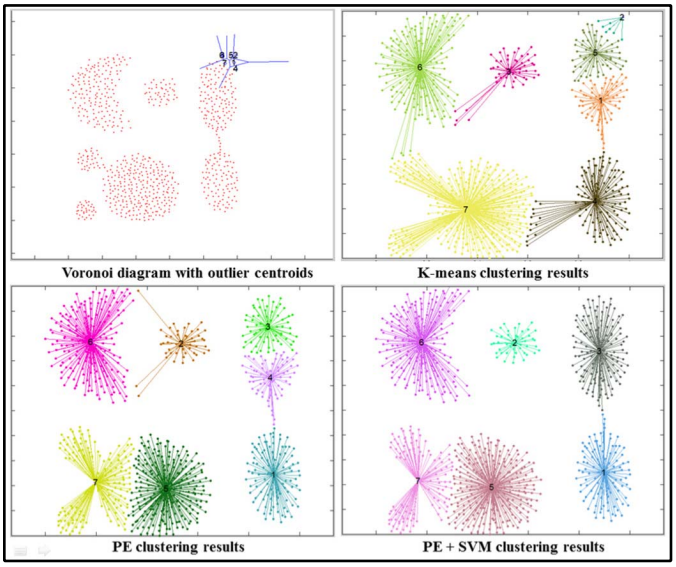


Fig. 8. Clustering results with noisy initialization on aggregation-set.

K-means results are also displayed, as it is a popular WTA clustering algorithm. Overall, the WTA networks perform much worse than the NFB clustering methods, because they only update the winning prototypes each time. Besides, the WTA networks are highly dependent on the initial prototype vectors, and relative positions (i.e., spatial configuration) are preserved through the iterations. They generally converge to the final clusters faster than the NFB algorithms with a great error rate, since the sample-cluster assignments do not change after a low iteration level. SCL and LVQ networks, on the other hand, achieve better performances than the classical k-means. The incremental cluster updates with a learning rate and enhances the results, instead of simply taking the average among cluster members. Because the LVQ network is a supervised algorithm, we use the clustering results of the SCL network as a prior for supervision with the purpose of fine-tuning. However, no obvious performance increase is observed when compared with the SCL results. The possible reason is that the SCL assignments $y(i)$ are kept fixed during the LVQ network iterations and it is simply forced to simulate the SCL behaviors in the cluster update procedure.

In the NFB networks, we visually achieve more meaningful results due to the neighboring functions when we compare them with the WTA-based methods, as shown in Fig. 6. In general, each sample updates all the prototype vectors proportional to their neighborhood degree. This helps to use the feature space more reasonably and to avoid the dependenc on the initial prototypes. In addition, we implement the gradient descent method to update the clusters, which may lead to spatial configuration changes. SOM, NG, and our first proposed clustering method, PE, have similar visual results. Nevertheless, the SVM fine-tuning network that is supervised by PE (i.e., PE + SVM network in Fig. 6) boosts the clustering performance remarkably. Unlike the LVQ, the SVM algorithm updates the clusters in a latent space iteratively, and the initial assignments are dynamically changed. Note that both the PE and SVM networks try to optimize their own weight parameters in their latent spaces during the cluster update procedure.

Next, for the visual analysis, we assume that the clustering results of the PE network, which is sequentially fed into the SVM network, are not satisfactory. Therefore, we try to find experimentally to what extent the SVM network may compensate for this initially unsatisfactory clustering. To address this issue, we first add Gaussian noise to the data samples in the input layer of the PE network. Second, we initialize cluster centroids for the PE network, such that they are deployed in the outlier regions in the features space. The clustering results are shown in Figs. 7 and 8.

In Fig. 7, we initialize the centroids in two distinct regions on the S_1 set. As expected, the PE network outperforms the k-means algorithm and it captures almost all of the clusters. On the other hand, although the PE + SVM network mainly follows the same routine of the PE network, it performs better in that cluster #10 and 12 (i.e., in the bottom-right frame) are successfully discriminated through the PE + SVM network.

We repeat the experiments on aggregation-set with a different configuration where the centroids are initialized in a small outlier region in Fig. 8 (bottom right). Surprisingly, in the PE + SVM network, we see that two clusters (#3 and 4 of the PE network) are combined in cluster #3 in the PE + SVM network as it is the ground-truth condition, and one out of seven centroids is implicitly eliminated. We conclude that the PE + SVM network is not strictly sensitive to the PE network clustering results where it mostly changes the configuration of the PE network. Furthermore, it may even sacrifice some centroids for better clustering results in the iterative cluster updating procedure.

Finally, it would be informative to explore the nature of transformations that have been learned through the proposed approach by visualizing the hidden-layer patterns. In addition, the improved clustering results can be further investigated as we adopt different number of hidden layers and neural nodes in each layer. To make things more clear, we adopt two PE networks successively such that once the first PE network is learned with the data samples in the original feature space, its hidden layer activations will be, then, the new input to the next one. Note that the number of nodes in the hidden layers is also changed implicitly, because the dimensionality of the input data differs.

In detail, S_1 set and randomly initialized prototypes are used for comparison with the previous results. Once the first PE network parameters are optimized, we project both the input data and the random prototypes into a new (i.e., higher dimensional) feature space by using the encoder part of the network. Thereafter, we implement the second PE network in the new feature space with its own setup. We visualize the hidden layer outputs of both the PE networks in Fig. 9, which represent the transformed features in terms of their first two principal components derived from the principal component analysis.

As usual, the randomly selected prototypes and the input data are plotted with their initial assignments in the original feature space. We select this initialization specifically, such that cluster #3 has no member at the beginning, and the initial prototypes are gathered on some locations spatially. After the first PE implementation, we observe that the original feature space is transformed in a way where the between-cluster separations are increased. This also holds for the second PE network that achieves a better result (i.e., cluster #2 is exploited additionally, while clusters #4 and 14 of the first PE network are unified in a single cluster #3 in the second PE network) by projecting the data samples into another latent space. It is seen that the proposed approach obtains more discriminative feature spaces in its hidden layers, which help to get improved clustering performance.

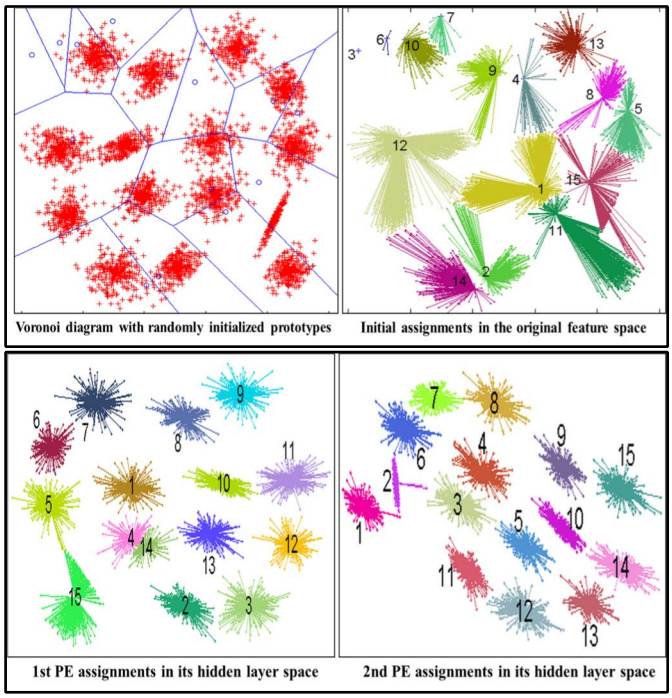


Fig. 9. Clustering in the latent spaces of successive PE networks on S_1 set.

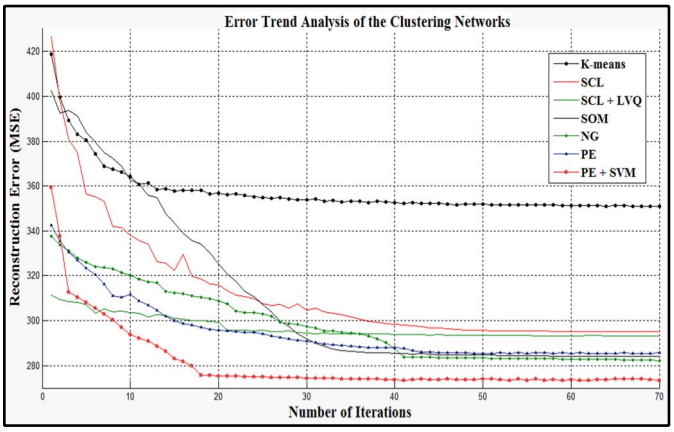


Fig. 10. Iterative error trend plotting on white-wine preference data set.

B. Analytical Performance Analysis

We first analyze the clustering results graphically based on the MSE trends of clustering networks on the white-wine preference data set in Fig. 10. Thereafter, MSE and time analysis results of all the data sets are given at Tables II and III for compactness. Also note that the same prototypes with random initializations are used for all the algorithms, and the iteration bound is set to 100 in the experiments. Basically, in Fig. 10, the k-means algorithm is the fastest converging method with the worst error rate. Although the LVQ that is supervised by the SCL network has much lower errors at the first iterations, the error rate slightly decreases in the final iterations. Apart from the WTA networks, the SOM, NG, and PE networks achieve similar performances because of their neighborhood functions. The proposed SVM fine-tuning network, on the other hand, gets the best

TABLE II
MSE AND TIME ANALYSIS ON SYNTHETIC DATA SETS

Algorithms	Synthetic Datasets							
	S_1		S_2		S_3		S_4	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time
K-means	28.71 ±12.44	**13.32 ±0.46**	26.70 ±4.86	**14.79 ±0.71**	29.99 ±3.15	**14.83 ±0.76**	31.18 ±1.19	**14.36 ±0.76**
SCL	21.28 ±7.11	23.44 ±0.62	21.82 ±4.91	24.59 ±0.41	25.71 ±1.60	26.41 ±0.49	26.29 ±2.07	16.55 ±0.53
SCL+LVQ	20.24 ±6.20	89.77 ±4.58	20.80 ±3.66	93.33 ±0.26	24.93 ±1.38	83.63 ±0.22	23.12 ±1.11	84.10 ±0.25
SOM	12.27 ±2.69	97.35 ±5.76	18.46 ±2.89	90.42 ±2.32	22.70 ±1.04	92.64 ±1.48	20.19 ±0.99	90.81 ±0.20
NG	14.01 ±1.09	53.93 ±0.79	17.93 ±1.44	55.15 ±0.90	23.98 ±0.63	58.25 ±0.92	20.95 ±0.36	56.49 ±1.52
PE	12.46 ±3.12	68.98 ±1.87	19.19 ±1.31	66.76 ±0.76	22.83 ±1.16	76.15 ±0.52	20.41 ±0.58	65.43 ±1.50
PE+SVM	**10.32 ±1.44**	122.69 ±6.86	**16.86 ±2.45**	128.69 ±6.41	**17.71 ±1.50**	137.16 ±4.18	**18.34 ±1.31**	134.82 ±4.65

TABLE III
MSE AND TIME ANALYSIS ON UCI DATA SETS

Algorithms	UCI Datasets							
	wine_red		wine_white		letter_ recognition		clave_ vectors	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time
K-means	276.38 ±4.41	7.36 ±0.06	408.57 ±12.39	**18.26 ±0.61**	3306.28 ±44.89	**61.49 ±0.46**	41297 ±201	**31.05 ±0.56**
SCL	234.21 ±4.62	**7.26 ±0.13**	346.85 ±19.96	21.14 ±0.88	3172.38 ±67.24	64.61 ±0.57	39812 ±329	36.14 ±0.76
SCL+LVQ	221.18 ±4.08	24.21 ±1.89	328.52 ±12.16	43.90 ±0.30	3033.88 ±63.51	180.06 ±18.87	39270 ±24	64.16 ±0.42
SOM	180.43 ±3.24	26.41 ±1.48	303.40 ±6.50	53.44 ±0.29	2865.60 ±45.35	213.77 ±5.35	37010 ±159	109.49 ±0.53
NG	179.84 ±3.46	21.27 ±0.65	291.95 ±2.56	42.14 ±0.11	2845.62 ±35.86	170.11 ±3.26	37208 ±124	73.67 ±1.19
PE	176.56 ±0.73	24.41 ±1.87	296.99 ±6.11	60.34 ±0.52	2847.76 ±32.82	204.53 ±33.29	37364 ±134	93.53 ±1.85
PE+SVM	**168.50 ±1.19**	44.73 ±1.93	**284.24 ±1.85**	119.61 ±0.51	**2746.91 ±12.74**	316.70 ±37.98	**36350 ±87**	144.94 ±2.83

result. This is regarded to be due to the PE supervision and the cluster updates in the latent space during iterations.

Next, we display the MSE (in Euclidean metric) and the timing results (in seconds) for all the data sets at Tables II and III. Because we repeat the experiments 50 times, standard deviations and mean values are computed. Overall, we observe that the PE + SVM network achieves the best performance in all the situations. On the other hand, the hybrid algorithm seems to take more time, almost doubling the nearest one for clustering the data samples. It is because two network architectures are implemented sequentially and they update the network parameters iteratively in mini batches. One may assume that these networks are individually comparable with the others in timing, and they can also be used together if priority is given to the clustering performance.

Compactness within a cluster and *separation* between clusters are two important measurements for cluster validation in the literature [2]. The validity of clustering results is defined as

$$\text{Validity} = \frac{1}{K}\sum_{j=1}^{K} \max_{j\neq m}\left\{\frac{d_{\text{WCS}}(c^{(j)}) + d_{\text{WCS}}(c^{(m)})}{d_{\text{BCS}}(c^{(j)}, c^{(m)})}\right\} \quad (25)$$

TABLE IV
COMPACTNESS AND SEPARATION-BASED CLUSTER VALIDITY

Algorithms	Synthetic Datasets				UCI Datasets			
	S_1	S_2	S_3	S_4	wine_ red	wine_ white	letter recog- nition	clave_ vectors
K-means	0.85 ±0.09	0.63 ±0.09	0.85 ±0.04	0.82 ±0.03	2.86 ±0.12	2.88 ±0.33	2.95 ±0.17	7.84 ±0.10
SCL	0.61 ±0.12	0.62 ±0.08	0.78 ±0.03	0.76 ±0.04	2.85 ±0.08	2.73 ±0.08	2.77 ±0.04	6.83 ±0.13
SCL+LVQ	0.60 ±0.11	0.62 ±0.05	0.76 ±0.04	0.73 ±0.03	2.84 ±0.07	2.71 ±0.07	2.74 ±0.04	6.81 ±0.11
SOM	0.42 ±0.06	0.55 ±0.06	0.73 ±0.05	0.71 ±0.04	2.07 ±0.31	1.94 ±0.03	1.70 ±0.02	3.72 ±0.16
NG	0.41 ±0.06	0.52 ±0.10	0.74 ±0.04	0.72 ±0.02	2.04 ±0.18	1.90 ±0.09	1.72 ±0.03	3.48 ±0.13
PE	0.41 ±0.05	0.55 ±0.04	0.74 ±0.03	0.71 ±0.03	2.04 ±0.11	1.96 ±0.16	1.71 ±0.04	3.52 ±0.18
PE+SVM	**0.39 ±0.06**	**0.49 ±0.03**	**0.70 ±0.03**	**0.68 ±0.02**	**1.96 ±0.05**	**1.67 ±0.11**	**1.68 ±0.03**	**3.22 ±0.14**

$$d_{\text{WCS}}(c^{(j)}) = \frac{\sum_{i=1}^{P} \|x^{(i)} - c^{(j)}\|}{P} \quad \forall x^{(i)} \in C_j \quad (26)$$

$$d_{\text{BCS}}(c^{(j)}, c^{(m)}) = \|c^{(j)} - c^{(m)}\|_{L_2} \quad (27)$$

where within-cluster scatter d_{WCS} indicates the average distance between data points $\boldsymbol{x^{(i)}}$ and their assigned prototype vector $\boldsymbol{c^{(j)}}$. Between-cluster separation for clusters j, m, and d_{BCS} is simply the distance between cluster prototypes. The best clustering method minimizes the validity measure in this manner. Table IV summarizes the validity results of the clustering algorithms on all the data sets for comparison. Although the PE network follows the similar validity results to those of the SOM and NG networks, the proposed PE + SVM network achieves again the lowest validity in all the situations, and this also supports the previous visual results. Also note that the NFB networks are well ahead of the WTA ones as the dimensionality and the number of samples increase.

Silhouette metric refers to another method of interpretation and validation for the clustering problems [28]. Given the already clustered data, it simply gives a matching rate for each sample with respect to its assigned and the nearest neighbor cluster. The Silhouette $S(\boldsymbol{x^{(i)}})$ is computed as

$$a(x^{(i)}) = \frac{\sum_{n=1}^{N} \|x^{(i)} - x^{(n)}\|}{N} \quad \forall x^{(i)} \in C_j \quad (28)$$

$$b(x^{(i)}) = \frac{\sum_{m=1}^{M} \|x^{(i)} - x^{(m)}\|}{M} \quad \forall x^{(i)} \in C_j \ \forall x^{(m)} \in C_z \quad (29)$$

$$C_z = c^{(\text{argmin}_{j=1,2,3,\ldots,K}\|x^{(i)} - c^{(j)}\|)} \quad (30)$$

$$S(x^{(i)}) = \begin{cases} 1 - \dfrac{a(x^{(i)})}{b(x^{(i)})}; & a(x^{(i)}) < b(x^{(i)}) \\ 0; & a(x^{(i)}) = b(x^{(i)}) \\ \dfrac{b(x^{(i)})}{a(x^{(i)})} - 1; & a(x^{(i)}) > b(x^{(i)}) \end{cases} \quad (31)$$

where $a(\boldsymbol{x^{(i)}})$ can be interpreted as how well the sample $\boldsymbol{x^{(i)}}$ is matched to the cluster it is assigned (i.e., the smaller value and

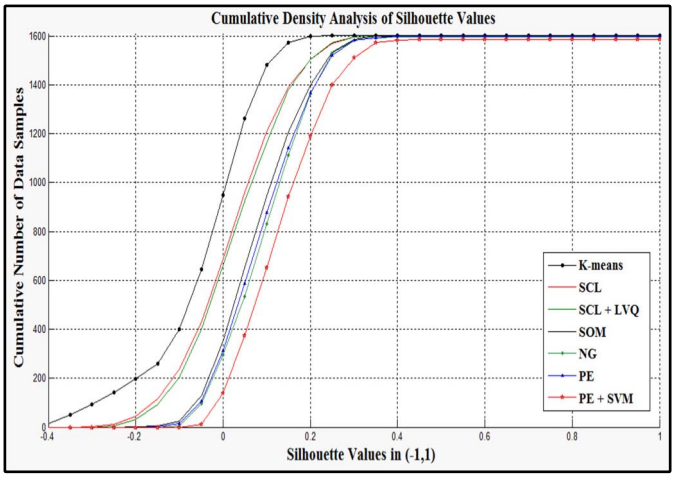


Fig. 11. Silhouette analysis on red-wine preference data set.

TABLE V
CLUSTERING COMPARISONS ON SILHOUETTE VALUES

Algorithms	Synthetic Datasets				UCI Datasets			
	S_1	S_2	S_3	S_4	wine_ red	wine_ white	letter recog- nition	clave_ vectors
K-means	0.61 ±0.04	0.56 ±0.04	0.44 ±0.01	0.41 ±0.03	0.04 ±0.01	0.06 ±0.01	0.09 ±0.01	0.03 ±0.01
SCL	0.62 ±0.04	0.56 ±0.04	0.45 ±0.01	0.42 ±0.03	0.07 ±0.01	0.07 ±0.01	0.10 ±0.01	0.04 ±0.01
SCL+LVQ	0.62 ±0.03	0.57 ±0.04	0.46 ±0.01	0.42 ±0.03	0.07 ±0.01	0.07 ±0.01	0.10 ±0.01	0.04 ±0.01
SOM	0.70 ±0.02	0.59 ±0.02	0.49 ±0.02	0.43 ±0.01	0.11 ±0.01	0.12 ±0.01	0.14 ±0.01	0.06 ±0.01
NG	0.69 ±0.02	0.62 ±0.01	0.51 ±0.01	0.43 ±0.01	0.14 ±0.01	0.12 ±0.01	0.14 ±0.01	0.05 ±0.01
PE	0.69 ±0.02	0.61 ±0.02	0.50 ±0.02	0.43 ±0.01	0.12 ±0.01	0.11 ±0.01	0.14 ±0.01	0.05 ±0.01
PE+SVM	**0.72 ±0.02**	**0.62 ±0.02**	**0.52 ±0.02**	**0.44 ±0.02**	**0.17 ±0.01**	**0.14 ±0.01**	**0.15 ±0.01**	**007 ±0.01**

the better matching). $b(\boldsymbol{x}^{(i)})$, on the other hand, represents the average dissimilarity of the sample to the neighboring cluster in which, aside from the previously assigned cluster, it fits best. From the above definitions, it is obvious that $S(\boldsymbol{x}^{(i)})$ falls in a range of $(-1, 1)$, and the best clustering algorithm tries to maximize it. We evaluate the clustering methods on the Silhouette values in Fig. 11 and in Table V. Because each sample has its own Silhouette value, we first produce a histogram that quantifies the amount of data points in bins, setting the bin width to 0.2. Thereafter, the histogram is transformed into cumulative densities for each clustering method to get better intuitions overall. Like in other comparisons so far, the PE + SVM network has the best performance in terms of the Silhouette metric, since its values start increasing last on red-wine preference data set in Fig. 11, and it achieves the highest mean Silhouette value in Table V.

Finally, we evaluate the clustering performances of the networks on all the Data sets by means of *accuracy* and *purity* metrics, given in percentages in Tables VI and VII. Given that we have the real cluster labels of the data samples, we can calculate the accuracy by first finding the best combinations between real labels and the cluster assignments. Accuracy is

TABLE VI
ACCURACY AND PURITY ANALYSIS ON SYNTHETIC DATA SETS

Algorithms	Synthetic Datasets							
	S_1		S_2		S_3		S_4	
	acc.	pur.	acc.	pur.	acc.	pur.	acc.	pur.
K-means	81.20 ±9.42	93.47 ±1.18	81.90 ±9.18	92.33 ±0.98	66.99 ±6.21	71.14 ±1.26	65.40 ±1.43	67.46 ±1.43
SCL	84.56 ±9.03	93.25 ±1.10	82.77 ±9.96	92.56 ±1.18	68.99 ±6.67	72.90 ±1.72	66.52 ±2.93	68.33 ±1.93
SCL+LVQ	84.79 ±9.16	93.49 ±1.24	83.56 ±9.01	92.63 ±1.11	69.95 ±6.72	74.83 ±1.33	67.94 ±2.04	69.22 ±1.46
SOM	91.61 ±4.38	95.20 ±0.25	90.03 ±5.77	94.12 ±1.39	77.70 ±5.74	81.00 ±1.18	74.07 ±2.81	78.48 ±1.03
NG	93.74 ±3.63	96.00 ±0.12	92.65 ±3.47	**94.89 ±0.55**	78.69 ±6.42	81.77 ±1.19	75.12 ±1.04	78.44 ±1.04
PE	91.85 ±3.56	94.91 ±0.08	91.20 ±2.04	94.15 ±1.00	77.65 ±5.50	81.38 ±1.42	73.94 ±2.83	78.12 ±0.68
PE+SVM	**95.73 ±1.18**	**96.73 ±0.45**	**93.86 ±2.95**	94.73 ±0.46	**79.66 ±3.32**	**83.26 ±0.96**	**77.84 ±2.27**	**80.29 ±1.02**

TABLE VII
ACCURACY AND PURITY ANALYSIS ON UCI DATA SETS

Algorithms	UCI Datasets							
	wine_red		wine_white		letter_ recognition		clave_ vectors	
	acc.	pur.	acc.	pur.	acc.	pur.	acc.	pur.
K-means	15.50 ±3.87	29.57 ±2.17	15.68 ±2.20	20.79 ±1.52	19.50 ±1.57	27.15 ±1.10	37.07 ±4.63	38.39 ±3.39
SCL	17.55 ±4.28	32.26 ±1.77	16.08 ±2.24	22.30 ±2.96	22.57 ±1.40	29.50 ±0.66	39.59 ±5.51	43.98 ±4.80
SCL+LVQ	17.70 ±3.86	33.36 ±2.42	16.83 ±2.10	23.36 ±1.17	22.94 ±1.69	30.43 ±1.13	39.47 ±5.50	43.46 ±3.96
SOM	20.72 ±1.45	46.50 ±4.49	20.05 ±1.26	25.92 ±1.03	24.93 ±1.49	33.32 ±0.82	42.82 ±3.35	44.90 ±2.67
NG	21.19 ±2.65	46.70 ±3.36	21.11 ±2.48	26.28 ±1.47	25.89 ±1.43	35.15 ±0.61	47.23 ±4.36	47.78 ±4.17
PE	22.14 ±2.06	46.92 ±4.39	21.96 ±2.55	26.34 ±1.69	26.88 ±1.56	36.29 ±1.23	46.43 ±3.97	47.58 ±4.45
PE+SVM	**23.31 ±2.72**	**48.12 ±3.84**	**23.40 ±1.80**	**29.84 ±1.26**	**27.50 ±1.57**	**37.13 ±1.10**	**47.44 ±3.63**	**48.07 ±3.68**

then the ratio of true positives to the number of samples. With regard to purity, each cluster is initially assigned to a class label, which is established by the most frequent (i.e., majority voting) label in that cluster. Thereafter, purity can be addressed to the accuracy formula in this setup. Note that real label-cluster pairs may have one-to-many relationships. Although the PE network usually performs worse than the SOM and/or NG algorithms in general, the PE + SVM network achieves better results by fine-tuning the PE clusters in its own latent space. The accuracy and purity results clearly support the previous statistical and visual clustering assessments.

V. CONCLUSION

In this paper, we propose a hybrid neural network that combines two complementary networks for the clustering problem. The first network is referred as PE, since it maps the input data samples to their closest prototypes successively in the output layer. While optimizing the network, we get unsupervised features in a latent space, which holds for the hidden layer of the PE. The clustering update procedure is carried out in this domain unlike previous works where the original feature space is used instead. Encoding weights (i.e., $\mathbf{W}_1$) of the PE network are initialized with respect to Voronoi-like hyperplanes in a pair-wisely greedy manner instead of random selection. Please note that the number

of neurons in the hidden layer is also found implicitly, and we eliminate this free parameter in the three-layer network structure. In addition, the Gaussian neighboring function is implemented in the latent space to update not only the winning but also the other prototypes for each sample. This also contributes to the performance increase as confirmed by the experimental results.

In the second part of the hybrid network, SVM is used to fine-tune the previously established clusters of the PE network. The margins between cluster boundaries are maximized using a quadratic hinge loss function as the objective. Like PE, the SVM network also updates the clusters iteratively in a latent space, which is represented by its output layer activations. Another important implementation detail is the utilization of the kernel function. This idea fits in our proposed network quite well, because the prototype vectors are now activated as landmarks for the GKF. Hence, we avoid the common application of treating all data samples as landmarks in which case all the samples are used to compute the kernel. Therefore, the computational complexity is remarkably reduced in this manner. Furthermore, instead of the original data, we feed the hidden layer activations of the optimized PE network successively to the GKF, which makes our proposed work more cooperative.

Our purposed work is compared with similar clustering networks in the literature using different evaluation metrics. We use a wide range of data sets from popular clustering repositories that are divided into two subsets to analyze the experimental results in each performance spectrum. The synthetic 2-D data sets are mainly utilized for visual analysis while the real-world UCI data sets are explored for the analytical experiments. Both the visual and analytical results clearly reveal that the PE + SVM clustering network outperforms the other approaches with the same experimental setup.

In addition, the parameter complexity of the proposed method is almost equal to its contemporaries, SOM and NG networks. Because we implicitly compute the number of neurons in all the layers, the weight parameters (except $\mathbf{W}_1$ in the PE network) are initialized randomly in the proposed work. Although the hybrid algorithm seems to take more time, two network architectures are sequentially implemented and they update the network parameters iteratively in mini batches. One may assume that these networks are individually comparable with the others in timing. Also note that we update the clusters in the latent spaces while optimizing the proposed networks, PE and SVM. Especially for the PE network, the latent space is produced in an unsupervised manner, and this directly impacts the clustering quality.

We also try to find experimentally to what extent the SVM network may compensate for initially unsatisfactory clustering of the PE network. To address this issue, we add noise to the input data samples, and we initialize cluster centroids, so that they are deployed in the outlier regions of the features space. We conclude that the SVM network is not strictly sensitive to the PE network clustering results where it mostly changes the configuration of the PE network. Furthermore, it may even sacrifice some centroids for better clustering results in the iterative cluster updating procedure.

Finally, we explore the nature of transformations that have been learned through the proposed approach by visualizing the hidden layer patterns. In addition, the improved clustering results are further investigated as we adopt different number of hidden layers and neural nodes in each layer. To do so, we adopt two PE networks successively, such that once the first PE network is learned with the data samples in the original feature space, its hidden layer activations are then fed into the next one. The visual analysis shows that the proposed approach obtains more discriminative feature spaces in its hidden layers, which help to get improved clusters.

For the future work, we plan to find the optimum number of clusters and initial prototype vectors for a given unlabeled data set in the network architectures. Obviously, network architectures have very elastic configurations, which can help to simulate any objective function inside. In addition, new latent spaces are achieved in the hidden layers, and each layer can be used for a specific task in a greedy layerwise manner. Therefore, it is envisioned that neural networks can solve complex clustering problems dynamically, and traditional supervision limitations may be eliminated in this manner.

REFERENCES

[1] Y. Stekh, F. M. E. Sardieh, and M. Lobur, "Neural network based clustering algorithm," in *Proc. 5th MEMSTECH*, Apr. 2009, pp. 168–169.

[2] K.-L. Du, "Clustering: A neural network approach," *Neural Netw.*, vol. 23, no. 1, pp. 89–107, 2010.

[3] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2011, pp. 215–223.

[4] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2013, pp. 8595–8598.

[5] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," M.S. thesis, DTU Inform., Tech. Univ. Denmark, Kongens Lyngby, Denmark, 2012.

[6] G. Hinton, "A practical guide to training restricted Boltzmann machines," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. UTML TR 2010-003, Aug. 2010. [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf

[7] N. Srivastava, "Improving neural networks with dropout," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2013.

[8] J. P. F. Sum, C.-S. Leung, P. K. S. Tam, G. H. Young, W. K. Kan, and L.-W. Chan, "Analysis for a class of winner-take-all model," *IEEE Trans. Neural Netw.*, vol. 10, no. 1, pp. 64–71, Jan. 1999.

[9] E. Yair, K. Zeger, and A. Gersho, "Competitive learning and soft competition for vector quantizer design," *IEEE Trans. Signal Process.*, vol. 40, no. 2, pp. 294–309, Feb. 1992.

[10] M. Mishra and H. S. Behera, "Kohonen self organizing map with modified K-means clustering for high dimensional data set," *Int. J. Appl. Inf. Syst.*, vol. 2, no. 3, pp. 34–39, 2012.

[11] T. Kohonen, *Self-Organizing Maps* (Springer Series in Information Sciences), vol. 30, 3rd ed. Berlin, Germany: Springer, 2001.

[12] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[13] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 558–569, Jul. 1993.

[14] K. Rose, F. Gurewitz, and G. C. Fox, "Statistical mechanics and phase transitions in clustering," *Phys. Rev. Lett.*, vol. 65, no. 8, pp. 945–948, 1990.

[15] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vis. Graph. Image Process.*, vol. 37, no. 1, pp. 54–115, 1987.

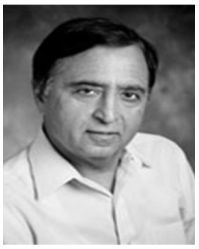
[16] J. C. Bezdek, "Modified objective function algorithms," in *Pattern Recognition With Fuzzy Objective Function Algorithms*. Norwell MA, USA: Kluwer, 1981, pp. 155–201.
[17] A. Y. Ng. *Autoencoders and Sparsity*, accessed on Mar. 8, 2016. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Sparsity
[18] E. Alpaydin, "Support vector machines," in *Introduction to Machine Learning*. London, U.K.: MIT Press, 2004, pp. 218–225.
[19] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric data structure," *Int. J. ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, Sep. 1991.
[20] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[21] Y. Tang, "Deep learning with linear support vector machines," in *Proc. Workshop Represent. Learn., Int. Conf. Mach. Learn. (ICML)*, 2013.
[22] Speech and Image Processing Unit. Clustering datasets. School of Computing, University of Eastern Finland, accessed on Mar. 8, 2016. [Online]. Available: http://cs.joensuu.fi/sipu/datasets/
[23] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognit.*, vol. 39, no. 5, pp. 761–765, 2006.
[24] P. Dostál and P. Pokorný, "Cluster analysis and neural network," in *Proc. Mezinárodní Konference Tech. Comput.*, Prague, Czech Republic, 2008, accessed on Mar. 8, 2016. [Online]. Available: http://dsp.vscht.cz/konference_matlab/MATLAB08/prispevky/025_dostal.pdf
[25] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *J. Mach. Learn. Res.*, vol. 2, pp. 125–137, Mar. 2002.
[26] J. Lee and D. Lee, "Dynamic characterization of cluster structures for robust and inductive support vector clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1869–1874, Nov. 2006.
[27] L. Bolelli, S. Ertekin, D. Zhou, and L. C. Giles, "K-SVMeans: A hybrid clustering algorithm for multi-type interrelated datasets," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Nov. 2007, pp. 198–204.
[28] M. Scarpiniti, "Neural networks and cluster analysis," Lecture Notes on Neural Networks, Dept. INFOCOM, Sapienza Univ. Rome, Rome, Italy, Tech. Rep., 2009, accessed on Mar. 8, 2016. [Online]. Available: http://ispac.diet.uniroma1.it/scarpiniti/files/NNs/Less5.pdf
[29] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, pp. 1–27, 2007.
[30] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Syst.*, vol. 47, no. 4, pp. 547–553, 2009.
[31] UCI Machine Learning Repository. *Wine Quality Data Sets*, accessed on Mar. 8, 2016. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Wine+Quality
[32] UCI Machine Learning Repository. *Firm-Teacher_Clave-Direction_Classification Data Set*, accessed on Mar. 8, 2016. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Firm-Teacher_Clave-Direction_Classification
[33] UCI Machine Learning Repository. *Letter Recognition Data Set*, accessed on Mar. 8, 2016. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

Emrah Ergul received the B.S. degree from the Electrical and Electronics Engineering Department, Turkish Naval Academy, Istanbul, Turkey, in 2001, the M.S. degree in computer engineering from the Naval Science and Engineering Institute, Istanbul, in 2009, and the Ph.D. degree from the Electronics and Communication Engineering Department, Kocaeli University, İzmit, Turkey, in 2016. He studied with the Computer Vision and Robotics Laboratory, Beckman Institute, The University of Illinois at Urbana–Champaign, Champaign, IL, USA, from 2013 to 2014, with a scholarship by The Scientific and Technological Research Council of Turkey.

Nafiz Arica (M'98) received the B.S. degree from the Turkish Naval Academy, Istanbul, Turkey, in 1991, and the M.S. and Ph.D. degrees in computer engineering from Middle East Technical University (METU), Ankara, Turkey.

He spent four years as a Communications and Combat Officer in the Turkish Naval Forces, Kocaeli, Turkey. In 1995, he joined METU. He conducted research as a Post-Doctoral Associate with the Beckman Institute, University of Illinois at Urbana–Champaign, Champaign, IL, USA, and the Department of Information Science, Naval Postgraduate School, Monterey, CA, USA. From 2004 to 2013, he was a Faculty Member with the Computer Engineering Department, Turkish Naval Academy. He is currently an Associate Professor with the Faculty of Engineering and Natural Sciences, Bahçeşehir University, Istanbul. His current research interests include object detection, recognition and tracking, path planning for unmanned vehicles, and facial expression analysis.

Associate Prof. Arica's thesis received the Thesis of the Year Award at METU in 1998.



Narendra Ahuja (F'92) received the Ph.D. degree from the University of Maryland at College Park, College Park, MD, USA, in 1979.

He was the Founding Director of the International Institute of Information Technology, Hyderabad, India, where he continues to serve as the International Director. He is currently the Donald Biggar Willet Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Champaign, IL, USA. His current research interests include next-generation cameras, 3-D computer vision, video analysis, image analysis, pattern recognition, human–computer interaction, image processing, image synthesis, and robotics.

Prof. Ahuja is a fellow of the American Association for Artificial Intelligence, the International Association for Pattern Recognition, the Association for Computing Machinery, the American Association for the Advancement of Science, and the International Society for Optical Engineering. He is on the Editorial Boards of several journals.

Sarp Erturk (M'99–SM'15) received the B.S. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 1995, and the M.S. degree in telecommunication and information systems and the Ph.D. degree in electronic systems engineering from the University of Essex, Colchester, U.K., in 1996 and 1999, respectively.

He carried out his compulsory service with the Army Academy, Ankara, Turkey, from 1999 to 2001. He is currently appointed as a Full Professor with Kocaeli University, İzmit, Turkey, where he was an Assistant Professor from 2001 to 2002, and an Associate Professor from 2002 to 2007. He currently leads the Kocaeli University Laboratory of Image and Signal Processing that has a full-time researcher capacity of about 20 researchers. His current research interests include digital signal and image processing, video coding, embedded systems, remote sensing, and digital communications.