AutoClustering: A Feed-Forward Neural Network Based Clustering Algorithm

Masaomi Kimura
Department of Infomation Science and Engineering, School of Engineering
Shibaura Institute of Technology
Tokyo, JAPAN
masaomi@shibaura-it.ac.jp

***Abstract*—Since a clustering process can be regarded as a map of data to cluster labels, it should be natural to employ a deep learning technique, especially a feed-forward neural network, to realize the clustering method. In this study, we discussed a novel clustering method realized only by a feed-forward neural network. Unlike self-organizing maps and growing neural gas networks, the proposed method is compatible with deep learning neural networks. The proposed method has three parts: a map of records to clusters (encoder), a map of clusters to their exemplars (decoder), and a loss function to measure positional closeness between the records and the exemplars. In order to accelerate clustering performance, we proposed an improved activation function at the encoder, which migrates a soft-max function to a max function continuously. Though most of the clustering methods require the number of clusters in advance, the proposed method naturally provides the number of clusters as the number of unique one-hot vectors obtained as a result. We also discussed the existence of local minima of the loss function and their relationship to clusters.**

I. INTRODUCTION

Clustering is a method of finding groups of similar records, termed clusters, in data. It is a useful starting point for data analysis because it provides us with information about what and how many clusters are in a target data set.

Clustering has a wide range of applications. For example, in data mining, a typical application is to find clusters of chosen options in a questionnaire, which helps the analyst grasp respondents' major opinions. In text mining, document clustering [1] is typically used to categorize documents. In the field of biology, DNA sequence clustering [2], [3] is popular.

Many clustering methods have been developed, including centroid models (e.g. K-means [4], x-means [5], Gaussian mixture model [6], affinity propagation (AP) [7]), density models (e.g DBSCAN [8]), and hierarchical models [9], [10]. All models use both a local property (closeness between data) and a global property (density higher than other spaces where data exists). Most of them (e.g. VQ and AP) also assume the existence of "exemplars", which are points or manifolds where each data record is approximately identified. In such models, each exemplar corresponds to a cluster, and each data point inside the cluster is close to it.

Recently, deep learning [11] has attracted significant research attention. A deep neural network is a very powerful and useful tool to express complex relationships between input and output. Clustering-like activity, such as the categorization of things, is done by our human brains. This suggests that a neural network can be used to realize a clustering algorithm.

In the context of neural networks, the self-organizing maps (SOM) [12]–[14] technique has been a typical method of clustering. Vector quantization is the key to SOM. Each input is assigned to the location whose vector is closest to the input. Iterating the assignments and the training of the vectors at the locations generates a map of inputs. SOM is a good method to obtain a map to visualize relative positions of input data on a two-dimensional plane. However, in case we need to separate input data into groups, we are of the opinion that a map generated by SOM hardly gives us clear borders between the groups. Moreover, SOM is a single-layer neural network, and its training method is different from the method of backpropagation and not combinable with deep learning.

In this study, we focused on a feed-forward neural network (FFNN) to realize clustering. This is because it is a basic structure of deep neural networks and it learns a functional relationship between the input and output. Since clustering process can be regarded as a map of data to cluster labels, it should be natural to employ a deep learning technique, especially a FFNN, to realize clustering method. Moreover, we should remember that clustering is a useful and important preprocessing technique to find positional structures in a data space in data mining. In other words, outputs of clustering can be inputs of other data mining techniques, which include a neural network. If a clustering process can be realized as an FFNN, it is easy to embed it into a more generic neural network.

A clustering method has to have three relationships: belongingness of each data point to a cluster, correspondence between a cluster and an exemplar, and close positional relationships between the data in the cluster and the exemplar. We propose a structure of FFNN that reproduces the relationships and a method to train it. The first relationship was realized as a map of an input vector to a one-hot vector corresponding to a cluster. The second was a map from the one-hot vector to an exemplar. The third was evaluated as a loss function.

In the language of EM algorithms, the first map corresponds to the E step and the second and the third correspond to the M step. In the proposed method, both of them are optimized through a backpropagation in an epoch.

For the majority of clustering algorithms, the number of clusters needs to be specified. In the proposed method, counting the unique one-hot vectors gives us the number of clusters.

Since the structure of the proposed FFNN is similar to AutoEncoder [11], we named it *AutoClustering*.

II. Related works

There have been many clustering methods using neural networks. The typical methods used outputs as the degrees of belonging to clusters [15].

There are also some single-layer neural-network-based clustering methods such as self-organizing maps (SOM) [12]–[14] and growing neural gas network (GNG) [16]. Particularly, SOM has been the typical clustering method used with a neural network [17]–[19] and has been expanded to its multi-layer version [20], [21]. Asadi et al. [22]–[24] used an unsupervised feed-forward neural network clustering method that utilizes Hebbian learning. There are also some studies that combined SOM and a feed-forward neural network [25]. However, their combination with deep neural networks is difficult if we regard it as a single neural network. This is because their training methods are different from backpropagation.

Some real-time clustering methods [26] have been discussed in the context of neural networks. Amin et al. [27] used adaptive threshold spiking neural networks to find clusters of user activity. The output layer of the model realized a local semi-unsupervised neural network.

Chakravarthy and Ghosh [28] used a radial basis perceptron to realize clustering, whose hidden layer nodes contain the locations of exemplars. They dealt with scaling-based clustering and discussed relationships between obtained clusters and scales. Although this provides good results, it does not clarify that normal FFNNs can realize clustering.

Xiao and Zhu [29] used a feed-forward neural network whose loss function is a sum of entropies corresponding to the abstraction process and clustering process. Hsu and Kira [30] used KL-divergence to measure the distance between the distributions of neural network softmax outputs. The same output distribution leads the same cluster. A loss function of the neural network was defined using KL-divergence. Xie et al. [31] proposed Deep Embedded Clustering, which combined soft assignment with a stacked autoencoder that mapped data into a low-dimensional feature space. Their method needs extra hyperparameters (the degrees of freedom of the student's t distribution), and the mechanism of dimensional reduction is discussed separately from clustering. These three studies utilized entropy-based evaluation applied to the output of the networks. If we regard cluster assignment as dimension reduction, the reduction mechanism and its criteria is given outside the framework of neural networks.

Simultaneous execution of similarity learning and clustering [32]–[34] is a new trend. However, we do not touch similarity learning in this study. We focus on the realization of spontaneous clustering by use of FFNN by giving a similarity between data and exemplars.

In this study, we focused on the aspect of clustering where data points are assigned to clusters ignoring the intra-cluster difference.

III. Method

A. Framework

As described in the Introduction, there are three necessary relationships to realize clustering:

1) A functional relationship of belongingness to clusters
2) A functional relationship of correspondence of clusters to their exemplars
3) Close positional relationships between the data in the clusters and their exemplars

We have illustrated the idea of our framework in Fig.1.

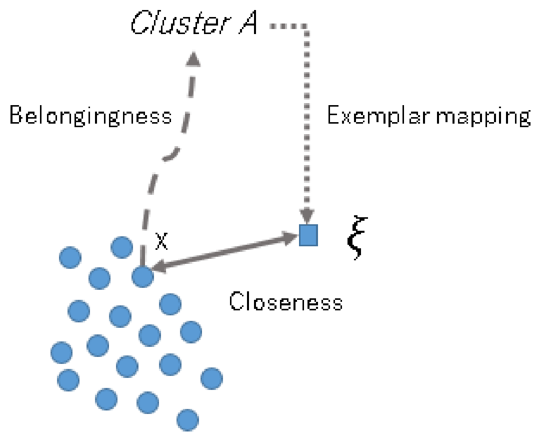

Fig. 1. The AutoClustering framework

Let $\mathbf{x}$ be an input record and $\mathbf{c}$ be a cluster label. Then, there should be a function that maps $\mathbf{x}$ to $\mathbf{c}$, namely:

$$\mathbf{c} = \mathcal{B}(\mathbf{x}) \tag{1}$$

We assume each cluster has its own exemplar ξ, which has the following relationship with $\mathbf{c}$:

$$\xi = \mathcal{C}(\mathbf{c}) \tag{2}$$

Generally, the number of clusters (or the number of unique $\mathbf{c}$) is much less than the number of records. An exemplar ξ has a one-to-one relationship with $\mathbf{c}$. Thus, we can say that ξ can be regarded as a coarse-grained $\mathbf{x}$ distribution. We should note that the exemplar can not only be a point in a data space but also a manifold, such as a curve, a circle, and so on. In order to find suitable map functions in Eq.1 and Eq.2, we propose to utilize a neural network with a loss function L. The loss function L should be decomposed into the terms that measure the closeness between $\mathbf{x}^{(i)}$ and its exemplar $\xi(\mathbf{x}^{(i)})$:

$$L = \sum_i l(\xi(\mathbf{x}^{(i)}), \mathbf{x}^{(i)}) \tag{3}$$

In order to obtain the functions, all weights are calculated by backpropagation with the loss function L.

For example, if the exemplar is a point, the simplest way to realize $l(\xi(\mathbf{x}), \mathbf{x})$ is to use their distance:

$$d(\xi, \mathbf{x}) \tag{4}$$

In this context, the distance should be short. In this framework, the clustering problem is to find the functions $\mathcal{B}$ and $\mathcal{C}$. Since deep neural networks can provide good approximate functions, we use them to get $\mathcal{B}$ and $\mathcal{C}$.

In fact, this framework can be generalized by replacing Eq.3. If we regard $\xi = \mathcal{C}(\mathcal{B}(\mathbf{x})) = \xi(\mathbf{x})$ as a cluster label, we can assume that there is a probability density function (PDF) labeled by ξ for each cluster. Let the PDF be $f_\xi(\mathbf{x}) = f(\xi, \mathbf{x})$. Here, we should note again that ξ depends on $\mathbf{x}$. The distribution of data, $\mathbf{x}^{(i)}$ $(i = 1, 2, \cdots, N)$, in a dataset can be expressed as $\rho(x) = \frac{1}{N}\sum_i^N \delta(\mathbf{x}-\mathbf{x}^{(i)})$, where $\delta(\mathbf{x})$ is a Dirac delta function. If we adapt a cross entropy as the loss function, it can be expressed as:

$$L = -\int d\mathbf{x}\, \rho(\mathbf{x}) \ln f(\xi(\mathbf{x}), \mathbf{x}) = -\frac{1}{N}\sum_i^N \ln f(\xi(\mathbf{x}^{(i)}), \mathbf{x}^{(i)}) \tag{5}$$

In the special case that $f(\xi, \mathbf{x}) = \exp(-\frac{N}{2}\|\mathbf{x}-\xi\|^2)$, this loss function can be rewritten into a quadratic loss function:

$$L = \frac{1}{2}\sum_i^N \|\mathbf{x}^{(i)} - \xi^{(i)}\|^2 \tag{6}$$

Here, $\xi^{(i)} = \xi(\mathbf{x}^{(i)})$. This means our framework has the potential to realize many types of clustering algorithms.

The idea of the abstraction process [29] is similar to the one we have proposed. However, the output of the abstraction process is a set of values (a vector) from zero to one. From the vector, it will be hard to generate one-hot vectors to specify a cluster. Moreover, it requires entropies as a loss function not only for the clustering process but also for the abstraction process. In our framework, we need a loss function just for the last part, Eq.4. This simplicity will help us design a concrete algorithm to implement this framework.

B. Formulation

In this subsection, we explain an FFNN structure that realizes the functions discussed in the framework. Here, in order to discuss the idea of implementing the framework, we limit ourselves to considering that exemplars are points. Moreover, we assume the data sits in an n-dimensional Euclidian space. Fig.2 illustrates the structure used in this study.

The first relationship is realized as a map of an input vector $\mathbf{x}$ to a one-hot vector corresponding to a cluster. The map, Belongingness encoder, is realized as an FFNN:

$$\mathbf{c} = \mathrm{smax}(\alpha \tanh(W_b\mathbf{x} - \theta_b)), \tag{7}$$

Here, W_b is a weight matrix, θ_b is a threshold vector, and smax is a softmax function:

$$\mathrm{smax}(\alpha\mathbf{v})_j = \frac{e^{\alpha v_j}}{\sum_{i=1}^n e^{\alpha v_i}}. \tag{8}$$

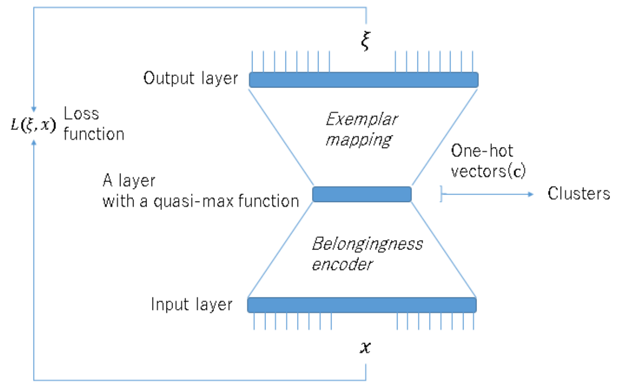

Fig. 2. The network structure of AutoClustering

In order to have a one-hot vector, Eq.7 should use a max function, whose output is 1 for a maximum-valued element and 0 for the rest. However, the max function is not differentiable and hinders the application of backpropagation. Thus, we instead approximate it with the function $\mathrm{smax}(\alpha\mathbf{v})$ with a large constant α. We call the function a quasi-maximum function. The output vector $\mathbf{c}$ is an approximate one-hot vector, whose element 1 shows a cluster that the input vector $\mathbf{x}$ belongs to.

In practice, if we take a large enough α, errors of elements in $\mathbf{c}$ from zero or one can be smaller and be in the range that floating underflow occurs. Moreover, in principle, the gap between the largest element and the second-largest element is not small for large α. This is because their ratio can be written as $\exp(\alpha \cdot \delta)$, where δ is the difference of the lower layer output. If α is large enough, the ratio is exponentially large. Thus, it is not necessary to employ a threshold to identify vectors $\mathbf{c}$ in the same cluster or to identify the largest element in $\mathbf{c}$. Figure 3 is an example to illustrate vectors $\mathbf{c}$. There are three clusters, each of which has three points. We set $\alpha = 500$ and obtained $\mathbf{c}$ in Table I. This shows that the method detected three clusters. For example, Point 1, 2 and 3 took 1 in the second elements, which indicates they belong to the same cluster.

TABLE I
AN EXAMPLE OF $\mathbf{c}$ FOR NINE POINTS. POINT 1, 2, 3 BELONG TO CLUSTER 1, POINT 4, 5, 6 BELONG TO CLUSTER 2 AND POINT 7, 8, 9 BELONG TO CLUSTER 3.

Point #	$\mathbf{c}$
1	[0. 1. 0. 0.]
2	[0. 1. 0. 0.]
3	[0. 1. 0. 0.]
4	[1. 0. 0. 0.]
5	[1. 0. 0. 0.]
6	[1. 0. 0. 0.]
7	[0. 0. 1. 0.]
8	[0. 0. 1. 0.]
9	[0. 0. 1. 0.]

The second is a map from the one-hot vector to an exemplar.

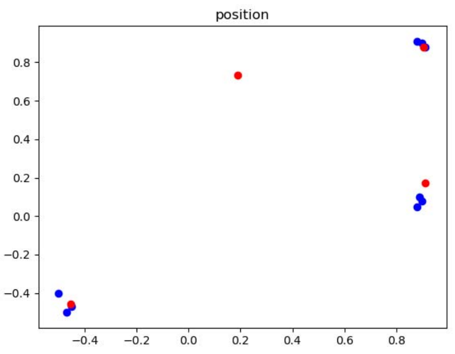

Fig. 3. A data distribution of data whose vectors,**c**, are listed in Table.I. Blue dots are target data and red dots are exemplars.

Though the map can be realized as a normal FFNN, we did not use a $\tanh$ function in the last layer to enable us to place the exemplar at any location in the data space. Thus, the map is formulated as:

$$\xi = W_o \tanh(W_h \mathbf{c} - \theta_h)) - \theta_o, \tag{9}$$

Here, W_h and θ_h respectively are a weight matrix and a threshold vector in the neurons of the hidden layer, and W_o and θ_o respectively are a matrix and a vector of an affine transformation. Since $\mathbf{c}$ is a one-hot vector, in principle, it is possible to get a result without the hidden layer, namely:

$$\xi = W_o \mathbf{c} - \theta_o. \tag{10}$$

We tried this and found that its convergence speed was extremely slow in the training phase. Therefore, we introduced the hidden layer here.

The third is realized as a loss function:

$$L = \frac{1}{2} \sum_i \|\mathbf{x}^{(i)} - \xi^{(i)}\|^2, \tag{11}$$

Here, $\mathbf{x}^{(i)}$ is the ith input vector and $\xi^{(i)}$ is the exemplar of the cluster it belongs to.

We can say that the structure is very similar to AutoEncoder. The structure for the first relationship corresponds to the encoder and the second to the decoder. AutoEncoder and AutoClustering commonly reduce unnecessary variations in data. This is the reason why AutoEncoder is used as a preprocessing step to clustering. The difference comes from the output of their encoder parts. An output vector in AutoClustering is unique to a cluster, but an output in AutoEncoder is not. This is because AutoEncoder is essentially a data compression algorithm that can approximately recover original inputs. However, AutoClustering compresses data into clusters by ignoring minor differences inside a cluster.

C. Training

Since the loss function (Eq. 11) is differentiable, backpropagation is applicable to the training of the structure. However, its direct application does not necessarily grant success in finding clusters. This is because the layer generating one-hot vectors kills the degree of freedom for exemplars to move to positions close to input vectors.

Because of this, it is effective to start the training with an ordinal softmax function and change it to a quasi-max function. In order to realize this, we used a monotonically increasing function $m(t)$, which satisfies $m(0) = 1$ and $m(T) = \alpha$ (t is an epoch parameter and T is an epoch number). Using this, we amend the quasi-max function to the following in the training phase:

$$\mathrm{qmax}(t, \mathbf{v})_j = \mathrm{smax}(m(t)\mathbf{v})_j = \frac{e^{m(t)v_j}}{\sum_i e^{m(t)v_i}} \tag{12}$$

This satisfies $\mathrm{qmax}(0, \mathbf{v}) = \mathrm{smax}(\mathbf{v})$, and $\mathrm{qmax}(T, \mathbf{v})$ coincides with the quasi-max function. We call this a softmax–hardmax transition. In practice, we used:

$$m(t) = 1 + (\alpha - 1)\left(\frac{t}{T}\right)^\gamma \tag{13}$$

Here, γ is a positive constant. In Section IV, we used $\gamma = 4$.

This model can have local minima. Let us illustrate this. Imagine there are only two points on a two-dimensional space, which are located along the X axis and far from each other. If the points belong to different clusters with two exemplars, $\xi^{[1]}$ and $\xi^{[2]}$, then the loss function is:

$$L = \frac{1}{2}\{(\xi_x^{[1]} - a)^2 + (\xi_y^{[1]})^2 + (\xi_x^{[2]} - b)^2 + (\xi_y^{[2]})^2\} \tag{14}$$

Obviously, its minimum is zero, given by $\xi_x^{[1]} = a$, $\xi_x^{[2]} = b$ and $\xi_y^{[1]} = \xi_y^{[2]} = 0$. However, if the points are assigned to the same cluster, then there is only one exemplar, ξ. In this case, the loss function is given as:

$$L = \frac{1}{2}\{(\xi_x - a)^2 + (\xi_y)^2 + (\xi_x - b)^2 + (\xi_y)^2\} \tag{15}$$

Its minimum is $\frac{1}{4}(b-a)^2 > 0$ in the case $\xi_x = (a+b)/2$ and $\xi_y = 0$.

This provides a counterexample to the idea of deep learning without local minima. In order to avoid such local minima, we utilized dropout and minibatch functions.

IV. Experiments

A. Datasets

In order to evaluate the performance of AutoClustering, we applied it to datasets in Table II. Each dataset has classes that show the ground truth of clusters.

Iris [35] is a well-known clustering benchmark dataset. Since two clusters in Iris dataset overlap, they will not be separated by the method discussed in Subsection III-B. Thus, we merged the clusters and made a new dataset, Iris (mod). Since the method in Section III-B is suitable for blobs, we used Spherical_4_3, Spherical_6_2, 2d-4c, 2d-10c, Triangle1, and Twenty [36]–[40] as the other benchmarks.

Normalization was applied to each dataset.

TABLE II
DATA SETS FOR EVALUATION

Dataset name	Dimension	# of records	# of classes (ground truth)
Iris	4	150	3
Iris (mod)	4	150	2
Spherical_4_3	3	400	4
Spherical_6_2	2	300	6
2d-4c	2	1261	4
2d-10c	2	2990	10
Triangle1	2	1000	4
Twenty	2	1000	20

B. Performance indices

We changed T in Eq.13 from 5 to 100 and measured homogeneity and completeness, attributes proposed by Rosenberg and Hirschberg [41]. Homogeneity (h) measures the entropy of classes in a cluster. Completeness (c) measures the entropy of clusters in a class. Both h and c are normalized in the range from zero to one:

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{16}$$

$$c = 1 - \frac{H(K|C)}{H(K)} \tag{17}$$

Here, $H(C)$ and $H(K)$ are entropies of classes and clusters, and $H(C|K)$ and $H(K|C)$ are conditional entropies of classes and clusters. Using the entropy of classes and clusters, $H(C,K)$, they can be expressed as $H(C|K) = H(C,K) - H(K)$ and $H(K|C) = H(C,K) - H(C)$. If the clusters have a mix of more than two classes, h is small. If classes are separately assigned to different clusters, c is small.

C. Implementation

We implemented the network with Chainer3.4.0, ran it 50 times for each T, and took their averages. In the implemented program, we increased the number of hidden layers in both Eq. 7 and Eq. 9 to three and two respectively and used Adam [42] for optimization. The number of nodes in the hidden layers were 20. In a future study, we will discuss the optimal number of layers and their nodes.

D. Results

Fig.4-10 shows the average values of h and c for each T.

We can find that homogeneity h is more than 0.8 for $T > 80$ except for the Iris dataset (Fig. 4) and Spherical_6_2 dataset (Fig.7). Completeness c also reached more than 0.8 for some ranges of T.

As for the Iris dataset (Fig.4), two overlapping clusters were identified as one. The Iris (mod) dataset (Fig.5) had $h \sim 1.0$ for $T > 20$. For both the Iris and Iris (mod) datasets, completeness was more than 0.9 only in the range $20 < T < 50$. This indicates that having more exemplars than the number of clusters can result in them taking over records for too many epoch numbers.

For the Spherical_4_3 dataset (Fig. 6), both h and c were more than 0.9 for $T > 80$. This shows that AutoClustering gave us (almost) correct clusters.

As for the Spherical_6_2 dataset (Fig. 7), h is around 0.7 for $T > 40$. This suggests that two pairs of clusters located closer than others were wrongly merged into two clusters. Completeness was almost 1 for $T > 60$, which tells us that records in each class were respectively assigned to the same cluster.

For the 2d-4c dataset (Fig.8) and 2d-10c (Fig.9) dataset, completeness was around 1.0 for $T > 10$, and homogeneity reached around 1.0 for $T \sim 100$.

For the Triangle1 dataset (Fig. 10), though h is around 1.0 for large T values, c is between 0.8 and 0.9. The Twenty dataset had results with a similar tendency but less homogeneity.

From these results, we can say that AutoClustering provides high homogeneity and completeness for datasets consisting of blobs. Particularly, if blobs are dense and clearly separable, homogeneity and completeness can reach almost 1.0 for $T \sim 100$. When $T \sim 1$, homogeneity and completeness took small values, which suggests that just using a quasi-max function caused unstable clustering. We should note that in the Iris and Spherical_6_2 datasets, AutoClustering tends to merge clusters that are closer than the other clusters. In such cases, homogeneity can saturate to less than 1.0.

In addition to evaluating AutoClustering using homogeneity and completeness, we compared it with Gaussian mixture models, k-means models, and affinity propagation. We selected them because the proposed method should be compared with other representative centroid models.

A Gaussian mixture model is an extension of the k-means clustering method, which assumes a weighted mixture of Gaussian distributions. In order to apply it to find clusters, we need to give it the number of clusters as input or use a Bayes information criterion.

Affinity propagation finds exemplars through message passing between data points. Although it does not require the number of clusters, we need suitable inputs, a damping factor, and a preference.

We applied the methods to datasets in Table II. We compared their results based on F-values, which are given by harmonic averages of homogeneity h and completeness c:

$$F = \frac{2hc}{h+c} \tag{18}$$

For Gaussian mixture models and k-means models, we calculated F-values for the cluster numbers given in Table II (Table III).

For AutoClustering, we changed T values from 5 to 100 and executed the algorithm 50 times for each T. We calculated the mean F-value (F_{mean}), the maximum F-value (F_{max}), and the minimum F-value (F_{min}) for each T value. Table IV shows the case that resulted in the largest F_{mean}-value. We can observe that F_{mean} is almost as large as F_{max}, which means that the F-values are equal to F_{max} in most of the cases.

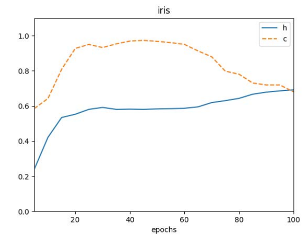

Fig. 4. Homogeneity and Completeness for Iris dataset.

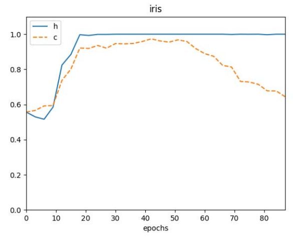

Fig. 5. Homogeneity and Completeness for Iris (mod) dataset.

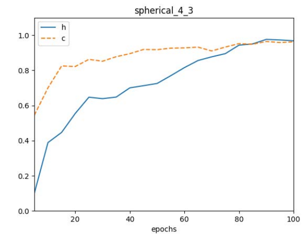

Fig. 6. Homogeneity and Completeness for Spherical_4_3 dataset.

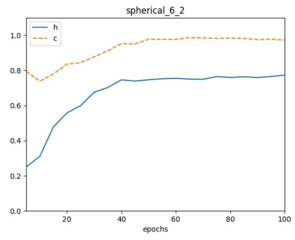

Fig. 7. Homogeneity and Completeness for Spherical_6_2 dataset.

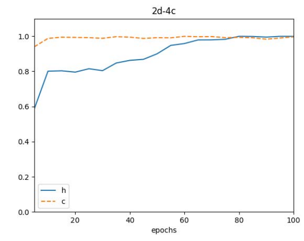

Fig. 8. Homogeneity and Completeness for 2d-4c dataset.

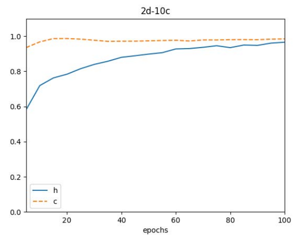

Fig. 9. Homogeneity and Completeness for 2d-10c dataset.

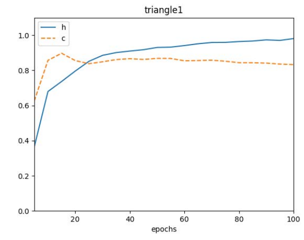

Fig. 10. Homogeneity and Completeness for Triangle1 dataset.

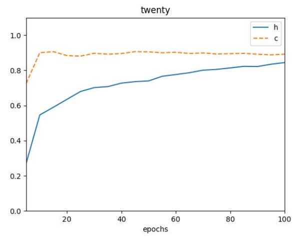

Fig. 11. Homogeneity and Completeness for Twenty dataset.

For affinity propagation, we fixed the damping factor to 0.9 and decreased preference values from 0 to -100 in steps of 5. We demonstrated the cases of the largest F-values with their preference values (Table V).

We used the scikit-learn package for Gaussian mixture models, k-means models, and affinity propagation.

Comparing the F-values in Table III with the F_{mean} values in Table IV, we can see that Gaussian mixture models provide F-values larger than those obtained by AutoClustering. This is because the loss function used for AutoClustering only measured the distances between exemplars and input data. Gaussian mixture models took account of not only the distance but also the contributions of clusters to the assignment of the inputs. Such degrees of freedom provided better results. K-means models also showed more similar tendencies to AutoClustering. For the Iris(mod) and Spherical_4_3, 2d-4c models, F-values for the k-means models and F_{max} for AutoClustering were 1.0. Although 2d-10c and Triangle1 did not result in $F = 1.0$ for AutoClustering, the F-values were close to 1.0. For Iris, Spherical_6_2 and Twenty, both F-values for the k-means models and F_{max} for AutoClustering tended to have small F-values. The similarity of results between the k-means models and AutoClustering came from the loss function, Eq.6, which assumed that data follows the distribution $(f\ \xi, \mathbf{x}) = \exp(-\frac{N}{2}\|\mathbf{x}-\xi\|^2)$. This normal distribution around an exemplar with constant variance also appears in statistical discussions of k-means algorithms. We must note that cluster numbers were given to Gaussian mixture models, although they were not given to AutoClustering. Without knowing the numbers, AutoClustering provided results close to the ones obtained using Gaussian mixture models. Even in such a situation, it is interesting that F_{max} obtained in AutoClustering is larger than F-values obtained by Gaussian mixture models for the 2d-10c dataset.

Comparing Table IV and Table V, F_{max} obtained by AutoClustering is larger than F_{max} obtained by affinity propagation for all datasets except Triangle1 and the Iris. Even for those two datasets, the variances in F_{max} were less than 0.04. This suggests that the performance of AutoClustering is comparable the performance of affinity propagation, despite the simpler structure of AutoClustering.

From these results, we can say that AutoClustering can provide comparable or better results than Gaussian mixture models and affinity propagation.

However, we must say AutoClustering is not as stable as GMM and K-means. This is because of the existence of local minima as we explained in Section III-C. It hindered reproduction of optimal clustering results. However, it might be possible to regard this as a coarse grained view of clusters. The value of the loss function at the local minima depends on the distance between clusters. Since smaller local minima will be easy to trap exemplars with, the proposed method may have the tendency to merge proximal clusters. From this, we might be able to interpret the local minima as a rough view to merge the nearest neighboring clusters like merging points in a dendrogram obtained by hierarchical clustering. Moreover, a cluster set obtained by the local minima will be able to correspond to the clusters by a high cutoff in the dendrogram. This should be studied in detail in a future study.

We found that the epoch number T affected the performance (homogeneity and completeness) of clustering. Our benchmark results suggested that T should be more than 100 for a good clustering result. However, as the clustering results for the Iris dataset showed, excessively large T values can cause less completeness. This might be related to the number of records in the Iris dataset, which was less than that of the other

TABLE III
F-VALUES OBTAINED USING GAUSSIAN MIXTURE MODELS (GMM) AND K-MEANS MODELS (KMEANS). TRUE CLUSTER NUMBERS WERE USED.

Dataset	F-values (GMM)	F-values (KMEANS)
Iris	0.8997	0.8997
Iris(mod)	1.0000	1.0000
Spherical_4_3	1.0000	1.0000
Spherical_6_2	1.0000	0.9004
2d-4c	1.0000	1.0000
2d-10c	0.9619	1.0000
Triangle1	1.0000	1.0000
Twenty	1.0000	0.9809

TABLE IV
F-VALUES OBTAINED BY AUTOCLUSTERING

Dataset	T	F_{mean}	F_{max}	F_{min}
Iris	45	0.7265	0.7337	0.6091
Iris(mod)	25	0.9784	1.0000	0.7420
Spherical_4_3	90	0.9701	1.0000	0.8635
Spherical_6_2	100	0.8599	0.9311	0.8314
2d-4c	100	0.9975	1.0000	0.9662
2d-10c	100	0.9750	0.9953	0.9309
Triangle1	70	0.9048	0.9413	0.8169
Twenty	100	0.8114	0.8795	0.7406

TABLE V
F-VALUES OBTAINED BY AFFINITY PROPAGATION (DAMPING FACTOR=0.9)

Dataset	Preference	F_{max}
Iris	-5	0.7609
Iris(mod)	-35	1.0000
Spherical_4_3	-10	1.0000
Spherical_6_2	-15	0.8520
2d-4c	-10	0.9958
2d-10c	-5	0.9294
Triangle1	-10	0.9748
Twenty	-5	0.7137

datasets, We need to study the relationship between optimal T and the number of records.

V. CONCLUSION

In this study, we discussed the possibility of a clustering method realized only using an FFNN and proposed a new clustering method named AutoClustering. The inspiration for this method was that the existing methods, SOM and GNG, do not have structures that can be suitably combined with deep learning neural networks.

The core structure of AutoClustering has three parts: an encoding part to map records to clusters (belongingness encoder), a decoding part that matches a cluster with its exemplar (exemplar mapping), and a loss function to measure positional closeness between records in the cluster and its exemplar. The activation function for the output layer of the encoding part used a quasi-max function, which is a softmax function whose argument is multiplied with a large constant. For training, we introduced a softmax–hardmax transition to accelerate clustering. It used an epoch number T as a parameter. The transition started from the softmax function and reached a quasi-max function at the end of the epochs.

The number of clusters can be obtained as the number of unique one-hot vectors returned by the encoding part.

We also discussed the existence of local minima of the loss function.

In future research, we will apply our method to large scale data. Moreover, we plan to explore the relationship between optimal T and the number of records in a dataset. The optimal number of (potential) exemplars should be discussed because unnecessary exemplars will cause poor performance. We will extend the idea to more generic exemplars such as curves. Since AutoClustering is realized as an FFNN, it will be easy to combine it with other deep learning networks. We will pursue the application that the cluster mechanism is useful for.

REFERENCES

[1] A. El-Hamdouchi. Comparison of hierarchic agglomerative clustering methods for document retrieval. 32(3):220–227, 1989.

[2] S. Meier-Ewert. Comparative gene expression profiling by oligonucleotide fingerprinting. 26:2216–2223, 1998.

[3] R. Herwig. Large-scale clustering of cdna-fingerprinting data. 9(11):1093–1105, 1999.

[4] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[5] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 727–734. Morgan Kaufmann, 2000.

[6] Javad Behboodian. On a mixture of normal distributions. *Biometrika*, pages 215–217, 1970.

[7] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

[10] Roy G. D'Andrade. U-statistic hierarchical clustering. *Psychometrika*, 4:59–67, 1978.

[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[12] Teuvo Kohonen. Clustering, taxonomy, and topological maps of patterns. In *Proceedings of the 6th International Conference on Pattern Recognition*, pages 114–128. IEEE, 1982.

[13] Teuvo Kohonen. *Self-organizing maps*. Number 30 in Springer series in information sciences. Springer, Berlin [u.a.], 1995. Literaturverz. S. [283] - 349.

[14] Teuvo Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37:52 – 65, 2013. Twenty-fifth Anniversay Commemorative Issue.

[15] Brijesh Verma, Peter McLeod, and Alan Klevansky. A novel soft cluster neural network for the classification of suspicious areas in digital mammograms. *Pattern Recognition*, 42(9):1845–1852, 2009.

[16] Bernd Fritzke. A growing neural gas network learns topologies. pages 625–632, 1995.

[17] K-L Du. Clustering: A neural network approach. *Neural networks*, 23(1):89–107, 2010.

[18] Kadim Tasdemir. Neural network based approximate spectral clustering for remote sensing images. In *2011 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2011, Vancouver, BC, Canada, July 24-29, 2011*, pages 2884–2887. IEEE, 2011.

[19] A. L. Tatuzov and N. I. Kurenkov. Neural network data clustering on the basis of scale invariant entropy. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 4912–4918, 2006.

[20] Pasi Koikkalainen and Erkki Oja. Self-organizing hierarchical feature maps. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 279–284. IEEE, 1990.

[21] Jonathan Randall, Ling Guan, Xing Zhang, and Wanqing Li. Investigations of the self organising tree map. In *Neural Information Processing, 1999. Proceedings. ICONIP'99. 6th International Conference on*, volume 2, pages 724–728. IEEE, 1999.

[22] Roya Asadi, Sameem Abdul Kareem, Mitra Asadi, and Shokoofeh Asadi. A single-layer semi-supervised feed forward neural network clustering method. *Malaysian Journal of Computer Science*, 28:189–212, 2015.

[23] Roya Asadi, Mitra Asadi, and Sameem Abdul Kareem. An efficient semisupervised feedforward neural network clustering. *AI EDAM*, 30(1):1–15, 2016.

[24] Roya Asadi, Sameem Abdul Kareem, Shokoofeh Asadi, and Mitra Asadi. A dynamic semisupervised feedforward neural network clustering. *AI EDAM*, 31(1):30–54, 2017.

[25] Vahid Nourani, Masoud Mehrvand, and Aida Hosseini Baghanam. Implication of som-ann based clustering for multi-station rainfall-runoff modeling. *Journal of Urban and Environmental Engineering*, 8(2):198–210, 2014.

[26] L. Fu. A neural network model for real-time adaptive clustering. IEEE International Conference on Neural Networks, 1993.,, San Francisco, CA, USA, USA, March 1993. IEEE, IEEE.

[27] Hesham H. Amin, Wael A. Deabes, and Kheireddine Bouazza. Clustering of user activities based on adaptive threshold spiking neural networks. In *Ninth International Conference on Ubiquitous and Future Networks, ICUFN 2017, Milan, Italy, July 4-7, 2017*, pages 1–6. IEEE, 2017.

[28] Srinivasa V Chakravarthy and Joydeep Ghosh. Scale-based clustering using the radial basis function network. *IEEE Transactions on Neural Networks*, 7(5):1250–1261, 1996.

[29] Han Xiao and Xiaoyan Zhu. Max-entropy feed-forward clustering neural network. *CoRR*, abs/1506.03623, 2015.

[30] Yen-Chang Hsu and Zsolt Kira. Neural network-based clustering using pairwise constraints. *CoRR*, abs/1511.06321, 2015.

[31] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[32] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 478–487. JMLR.org, 2016.

[33] Zhao Kang, Chong Peng, and Qiang Cheng. Kernel-driven similarity learning. *Neurocomputing*, 267:210–219, 2017.

[34] Zhao Kang, Chong Peng, and Qiang Cheng. Twin learning for similarity and clustering: A unified kernel approach. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2080–2086. AAAI Press, 2017.

[35] UCI Machine Learning Repository. Iris data set. https://archive.ics.uci.edu/ml/datasets/iris.

[36] Tomas Barton. Clustering benchmarks. https://github.com/deric/clustering-benchmark.

[37] Julia Handl. Cluster generators. http://personalpages.manchester.ac.uk/mbs/Julia.Handl/generators.html.

[38] Sanghamitra Bandyopadhyay and Ujjwal Maulik. Nonparametric genetic clustering: comparison of validity indices. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(1):120–125, 2001.

[39] Sanghamitra Bandyopadhyay and Ujjwal Maulik. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern recognition*, 35(6):1197–1208, 2002.

[40] Sanghamitra Bandyopadhyay and Sankar Kumar Pal. *Classification and learning using genetic algorithms: applications in bioinformatics and web intelligence*. Springer Science & Business Media, 2007.

[41] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.