A New Approach in Cluster Analysis

V.M. Sineglazov
Aviation Computer-Integrated Complexes Department,
Educational & Research Institute of Information and Diagnostic Systems,
National Aviation University
Kyiv, Ukraine
svm@nau.edu.ua

O.I. Chumachenko
Technical cybernetics department,
Faculty of informatics and computer science
NTUU "Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
chumachenko@tk.kpi.ua

V.S. Gorbatiuk
Technical cybernetics department,
Faculty of informatics and computer science
NTUU "Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
vladislav.horbatiuk@gmail.com

***Abstract*—A new clustering approach that is capable of finding clusters that are separated by some complex hypersurface is proposed. The approach can be useful for performing analysis of big amounts of unlabeled images that can be nowadays easily gathered, in particular by using unmanned aerial vehicle with mounted cameras. The approach is based on "softening" the initial clustering criterion and then using nonlinear optimization to find the optimal hypersurface that separates clusters.**

Keywords—unmanned aerial vehicle; soft clustering; nonlinear optimization; artificial neural networks

I. Introduction

Nowadays, gathering big amounts of unlabeled image data is becoming much easier. However, labelling all this data is often infeasible, and so techniques from the unsupervised learning [1] paradigm should be used to analyze this data instead. Clustering analysis is probably the most well-known and studied technique from this paradigm. It is used find groups or clusters of examples that share some similarities among given set of examples.

II. Problem Statement

The clustering problem [2] that is considered in this paper can be described as follows:

Having a set of examples $X = (\vec{x}_1, ..., \vec{x}_n)$ where each example is a vector in space R^d, and given number of clusters $K \in N$, we need to set a certain cluster number $k = 1, ..., K$ for each example so that the resulting vector of cluster numbers $\vec{k} = [k_1, ..., k_n]^{\mathrm{T}}$ minimizes a certain criterion $CR(\vec{k}, X)$:

$$\vec{k}^* = \arg\left\{\min_{\vec{k}}\{CR(\vec{k}, X)\}\right\}.$$

III. An Overview of Existing Methods

Even for the very simple case, when the minimized criterion is a total Euclidean distance from cluster points to its center, the clustering problem is known to be *NP* hard [3], [4] that is, unless $P = NP$, it is impossible to define an algorithm that will precisely find the optimal vector of clusters numbers $\vec{k}^*$ and will not require an exponentially increasing number of calculations with an increase in the number of examples. Because of this, various heuristic methods have been developed to solve this problem approximately. Let us review the main existing methods, used to perform clustering, together with their pros and cons.

K-means clustering [5] is probably the most well-known and one of the simplest clustering algorithms. Informally, it can be described as follows: in the beginning, initial clusters centers are selected in certain way (the simplest way is to do this at random), and then iterations are performed until the algorithm's stopping condition is met, where each iteration consists of 2 steps: the step of finding the closest current cluster center for each example, and the step of calculating the new cluster center – as the mean value of all examples for which the current center of this cluster was the closest. Stopping condition is the equality of new and current clusters' centers. Formally, the algorithm consists of the following steps:

1. The starting centers of clusters are set as randomly selected unique K examples:

$$c_j^{(0)} := x_{r_j}, j \in \{1, .., K\}, r_j \in \{1, ..., n\}; \forall t \in \{1, ..., K\},$$
$$\forall j \in \{1, .., K\} : t \neq j \rightarrow r_j \neq r_t$$

2. $it := 0$.

3. For each example $x_i, i \in \{1, ..., n\}$ the center of the cluster closest to it is located and memorized:

$$nc_i := \arg\min_{j \in \{1, ..., K\}}\{| x_i - c_j^{(it)} |\}$$

4. New cluster centers are calculated as the mean of all examples for which the current cluster center was the closest:

$$c_j^{(it+1)} := \frac{1}{n_j} \sum_{i: nc_i = j} x_i, n_j = \sum_{i: nc_i = j} 1, j \in \{1, ..., K\}$$

5. If at least for one value $j \in \{1, ..., K\}$ the new center of the cluster is different from the current one – $c_j^{(it+1)} \neq c_j^{(it)}$ –

978-1-5386-1816-5/17/$31.00 ©

then $it := it + 1$ is assigned and a new iteration of the algorithm is performed starting from step 3. Otherwise, the algorithm stops, and the current distribution of examples between clusters along with the cluster centers are the algorithm outputs.

The algorithm tries to minimize the criterion of total mean distance between points in one cluster:

$$CR(\vec{k}, X) = \sum_{i=1}^{K} \sum_{\vec{x}_j : k_j = i} \left\| \vec{x}_j - \vec{\mu}_i \right\|^2 .$$

The main drawbacks of the algorithm are:

1) susceptible to getting stuck in a local optimum – depending on the initial cluster centers the algorithm can stop at various local optima, not finding globally optimal centers (although it is clear that no algorithm that does not require exponentially increasing number of calculations while increasing the number of examples can find the globally optimal centers unless $P = NP$);

2) the algorithm prefers clusters with approximately same number of examples;

3) it is clear that the application of the algorithm to the data set where clusters do not meet the algorithm "expectations" – i.e. is not similar to the spherical areas that are separated in space, usually gives poor results.

Hierarchical clustering algorithms [6]. Algorithms of this group are building a certain hierarchy of clusters; there are 2 main approaches for building a hierarchy: **agglomerative** when the algorithm starts with each example is in its own cluster and the clusters are gradually merged, and **divisive** when at the beginning all examples belong to one cluster and the division of one cluster into several smaller clusters is gradually done. Hierarchical clustering algorithm usually requires 2 things to be specified:

1) a metric that will be used as a measure of distance between pair of examples
2) a "linkage" criterion, that is used to calculate a measure of dissimilarity between 2 sets of examples using selected distance between pair of examples metric

The main disadvantages of this algorithms family are:

A. It is not always possible to clearly define a global criterion that is minimized by this algorithm.

B. A certain locally optimal clustering is often obtained because of the "greedy" nature of most of the algorithms of this family.

The Kohonen self-organizing map [7] (SOM). Self-organizing map is an artificial neural network [8] "suitable" for unsupervised learning and during learning the network tries to somehow approximate distribution of input examples. This approximation is achieved via "positioning" a certain number of network's output neurons in the space of input examples so that all neurons tend to be positioned closer to given input examples – often finding centers of examples' clusters in the process. Thus, after all output neurons are positioned, the clustering of input examples can be carried out by finding the closest neuron for each example, and assigning to one cluster all examples for which one certain common neuron turned out to be the closest one. The interesting distinguishing feature of SOM is a neighborhood function, that for a pair of network's output neurons u, v gives a "distance" between these neurons – and this neighborhood function is used during training stage to position neurons in the input examples' space. Namely, when an input example is "fed" to the network, the neuron whose weights vector is most similar to the input is called the best matching unit (BMU), and the weights of the BMU and neurons close to it (and "closeness" is determined by neighborhood function) are adjusted towards the input vector.

The main shortcomings of the SOM are:

1) For the original SOM training algorithm it is unclear what optimization criteria is minimized while network neurons are positioned in space of input examples.
2) The result of the SOM algorithm depends on certain parameters the values of which need to be carefully chosen.
3) In terms of clustering problem, there is no evidence that the SOM gives fundamentally better results in comparison with other clustering algorithms, such as *k*-means.

IV. Soft Clustering Algorithm Based on Separating Hypersurfaces

A new clustering algorithm with the following properties is proposed:

a) it has a clearly defined criterion which is minimized during the execution of the algorithm;

b) the optimization criterion is a differentiable function of the parameters according to which the optimization is performed;

c) it theoretically allows to find clusters that are separated by the arbitrarily complex hypersurface.

The algorithm can be applied when the criterion $CR(\vec{k}, X)$ has the following form:

$$CR(\vec{k}, X) = \sum_{i=1}^{K} f(X, \vec{k}, i) ,$$

where $f(X, \vec{k}, i)$ is the function which is either completely continuous function of X or its only discontinuity comes from using the indicator function "$1(\vec{k}, \vec{x}_j, i) = 1$, if $k_j = 1, 0$ otherwise". For example, if we consider the criterion of total mean distance between points in one cluster which is used by the *k*-means algorithm, then a corresponding $f(X, \vec{k}, i)$ can be written as follows:

$$f(X,\vec{k},i)=\frac{1}{\sum\limits_{\vec{x}_j\in X}1(\vec{k},\vec{x}_j,i)}\cdot\sum_{\vec{x}_q,\vec{x}_w\in\binom{X}{2}}1(\vec{k},\vec{x}_q,i)\cdot1(\vec{k},\vec{x}_w,i)\cdot\left\|\vec{x}_q-\vec{x}_w\right\|^2.$$

Let us consider the simple case where we have only two clusters. We perform the "softening" of the original problem [9] by introducing the continuous function $k(\vec{x})\in[0,1]$, which for each example $\vec{x}$ will return the probability of this example belonging to the cluster 1; respectively, the value $1-k(\vec{x})$ represents the probability of this example belonging to the cluster 0. In this case, the indicator functions are replaced by the value $k(\vec{x})$ and, for example, the "softened" variant of the total mean distance between points in one cluster will be as follows:

$$SCR(k,X)=\frac{1}{\sum\limits_{\vec{x}_j}[1-k(\vec{x}_j)]}\cdot\sum_{\vec{x}_q,\vec{x}_w}\left\{[1-k(\vec{x}_q)][1-k(\vec{x}_w)]\left\|\vec{x}_q-\vec{x}_w\right\|^2\right\}+\frac{1}{\sum\limits_{\vec{x}_j}k(\vec{x}_j)}\sum_{\vec{x}_q,\vec{x}_w}\left\{k(\vec{x}_q)k(\vec{x}_w)\left\|\vec{x}_q-\vec{x}_w\right\|^2\right\},$$

wherein the first component determines the contribution of the cluster with number 0 and the second – the cluster with number 1.

If we have a certain model of the hypersurface that separates the clusters in the form of a function $k(\vec{x};\vec{w})\in[0,1]$ which is a differentiable function of a certain parameters vector $\vec{w}$ – then softened criterion $SCR(k,X)$ will also be a differentiable function of vector $\vec{w}$, since the only source of discontinuity in original criterion $CR(\vec{k},X)$ will be replaced by either $k(\vec{x};\vec{w})$ or $1-k(\vec{x};\vec{w})$ (of course, this statement will only be true if the original criterion satisfies the previously stated condition that the only possible source of discontinuity should come from using and indicator function $1(\vec{k},\vec{x}_j,i)$). Thus for the minimization of softened criterion it is possible to use the entire apparatus of nonlinear continuous differentiable functions minimization which has recently been developing very rapidly. As a result, the "softened" version of clustering problem for 2 clusters can be solved as a continuous nonlinear optimization problem (if the input criteria satisfies the described above condition), for example through the use of a certain gradient descent algorithm modification – like AdaGrad [10], RMSProp [11] or Adam [12].

To solve the softened version of clustering problem into K clusters we can use the "one versus all" approach – first we divide all the examples into 2 clusters, after which we select a cluster with a larger "partial criterion" value $f(X,\vec{k},i)$, and divide it into 2 clusters and so on until we get the required number of clusters.

Perhaps the most straightforward function that can be used as a model of a hypersurface separating the clusters is a logistic sigmoid function: $k(\vec{x};\vec{w})=\frac{1}{1+e^{-\vec{w}^{\mathrm{T}}\vec{x}}}$. In essence, such a model will be a certain approximation of the separating hyperplane which is determined by the parameters vector $\vec{w}$ – for examples $\vec{x}:\vec{w}^{\mathrm{T}}\vec{x}>b, b>0$ the value of the model will be approximately equal to 1; for examples $\vec{x}:\vec{w}^{\mathrm{T}}\vec{x}<-b$ – approximately equal to 0, and for examples that are "close" to the hyperplane – i.e. those for which $-b<\vec{w}^{\mathrm{T}}\vec{x}<b$ the value of the model will smoothly grow from 0 to 1 with the "transition" from one side to the other (and for all examples $\vec{x}:\vec{w}^{\mathrm{T}}\vec{x}=0$ which are located on the hyperplane, the model value is equal to 0.5). That is, using such a model when minimizing the criterion, we will try to separate all the examples by the "almost linear" hypersurface into 2 clusters so that the total mean distance of these clusters will be minimal.

Obviously, such a model is very simple and will not work well if the clusters in the available set of examples are not linearly separate. In this case, we need more complicated models of the separating hypersurface, and we know what suits very well as such models – the neural networks. The only limitation is that the network's output should be in range $[0,1]$ but to achieve this it is enough to pass the network's output through the aforementioned logistic sigmoid function. For example, one may use a simple multilayered perceptron [13] with one hidden layer consisting of neurons with ReLU [14] activation function and one neuron with a logistic sigmoid activation function in the output layer as separating hypersurface model, appropriately choosing number of neurons in the hidden layer.

Thus, we obtain the following soft clustering general algorithm based on a certain separating hypersurface model $k(\vec{x};\vec{w})\in[0,1]$.

Algorithm inputs:

- examples set $X=(x_1,...,x_n)$, $x_i\in R^d$, $i=1,...,n$;
- number of clusters K, into which we need to split all the examples;
- some neural network that defines the model of hypersurface separating the clusters $k(\vec{x};\vec{w})\in[0,1]$;
- criterion $CR(\vec{k},X)$, which needs to be minimized and which satisfies the previously mentioned condition.

Steps of the algorithm:

1. Obtaining the softened criterion $CR(\vec{k},X)\to SCR(k,X)$.

2. The whole set of examples is split into 2 clusters. In order to do this:

– the initial vector of model's parameters $\vec{w}_0$ is randomly generated;

– a certain modification of the gradient descent is performed to minimize the value of the criterion $SCR(\vec{w}, X)$;

– as a result, we get the tuned parameters vector $\vec{w}_f$;

– all examples are divided into 2 clusters – those examples for which the model values $k(\vec{x}; \vec{w}_f) < 0.5$ are selected into cluster 0, all other examples (i.e. those for which $k(\vec{x}; \vec{w}_f) \geq 0.5$) – in cluster 1.

3. If the current number of clusters $<K$ then for both clusters the "partial" criterion value $f(X, \vec{k}, i)$ is calculated and the cluster having bigger value $f(X, \vec{k}, i)$ is chosen as a new set of examples for further separation into clusters, and the algorithm's execution continues from step 1. Otherwise, obtained K clusters are returned as the result of algorithm's execution.

V. Conclusion

A new clustering algorithm is introduced and it has the following properties:

- it allows to solve the original clustering problem by optimizing some differentiable criteria, thus making it possible to use all the recent developments in nonlinear optimization – like stochastic gradient descent methods, different regularization methods etc.;
- the algorithm is able to detect clusters of input examples that are separated by some complex hypersurface if you choose an appropriate model of this separating hypersurface;
- can be potentially applied to find clustering that is optimal under different criteria, i.e. not only total mean Euclidian distance between points in cluster.

A few important problems are not considered in this "version" of the algorithm, namely:

- in many practical problems, the number of clusters K is not known in advance, and an automatic detection of best (in some sense) number of clusters for the given set of examples is a very nice feature to have;
- using "one versus all" approach to perform clustering when $K > 2$ is basically a greedy way to find optimal clustering which is also known to produce locally optimal results; some approach to simultaneously separate more than 2 clusters by adjusting certain model's parameters could potentially greatly improve performance of suggested algorithm.

References

[1] G. Hinton, and T.J. Sejnowski, Unsupervised Learning: Foundations of Neural Computation. MIT Press, 1999.

[2] K. Bailey, Numerical Taxonomy and Cluster Analysis. Typologies and Taxonomies, 1994, 34 p.

[3] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, NP-hardness of Euclidean sum-of-squares clustering, 2009.

[4] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, The Planar k-Means Problem is NP-Hard, 2009.

[5] E.W. Forgy, "Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classifications, " Biometrics, 1965, 21, pp. 768-769.

[6] J.H. Ward, "Hierarchical Grouping to Optimize an Objective Function," Journal of the American Statistical Association, 2009, 58 (301), pp. 236-244.

[7] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," Biological Cybernetics, 1982, 43 (1), pp. 59-69.

[8] W. McCulloch, and Walter Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity, " Bulletin of Mathematical Biophysics, 1943, 5 (4), pp. 115-133.

[9] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, 1981.

[10] Duchi, John, Hazan, Elad, Singer, Yoram, "Adaptive subgradient methods for online learning and stochastic optimization," 2011, JMLR, 12, pp. 2121-2159.

[11] Hinton, Geoffrey, "Overview of mini-batch gradient descent", pp. 27-29. Retrieved 27 September 2016.

[12] Diederik, Kingma and Ba, Jimmy, "Adam: A method for stochastic optimization," 2014.

[13] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation." David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[14] R Hahnloser, R. Sarpeshkar, M A Mahowald, R. J. Douglas, and H.S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature. 405, 2000, pp. 947-951.