ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Self-Organized Chains for Clustering

Hassan Ghaziri

Swiss Federal Institute of Technology
I&C-LANOS
Station 14, CH-1015 Lausanne - Switzerland
E-Mail: hassan.ghaziri@epfl.ch

Abstract. This paper presents a new algorithm for clustering. It is an generalisation of the K-means algorithms . Each cluster will be represented by a chain of prototypes instead of being represented by one prototype like for the K-means. The chains are competing together to represent clusters and are evolving according to Kohonen maps adaptation rule. It is well known that K-means performs very well with hyper-spherical data and has difficulties in dealing with irregular data. We have shown on special artificial data that the new algorithm we are presenting performs very well for different types of data sets . In addition, it shows robustness regarding initial conditions.

1 Introduction

Clustering consists of finding a partition of a data set such that similar data points are grouped together. This definition is vague on purpose. In fact, there is no standard definition to clustering and it remains at the end a subjective procedure allowing the end user to detect some hidden structure or regularities inside the data set. Clustering has a wide variety of applications such as data mining, pattern recognition, knowledge discovery, text mining, and many others [1]. Clustering algorithms can be divided into two big families. The hierarchical and the partitioning algorithms. The algorithm we are presenting in this paper belongs to the partitioning family of algorithms. k-Means and Self-Organizing Maps (SOM) are among the most popular algorithms for clustering. K-means aims at minimizing a quantization error. Although it can be proved that the procedure will always converge, there is no guarantee that the optimal solution for finding the global minimum of the quantization error can be obtained. The SOM introduces the concept of neighborhood influence allowing the algorithm to escape from local minima. In order to implement this concept SOM relies on a topological order of the neurons that is not always optimal for the given data. It leads to frequent mismatches and hence a deterioration of the performance [2]. Our objective is to overcome the deficiencies of k-Means and SOM by designing an algorithm that generalizes k-Means and takes advantage of the topology preserving principle of SOM. This paper is organized as follows. In the next section, we describe briefly the k-Means and the SOM algorithms and then show how K-Means can be generalized through the Self-Organized Chains (SOC) algorithm. In section 3 we discuss the implementation and then we show the results obtained on artificial data sets. Finally we draw some conclusions based on the performance comparison with k-Means.

2 The self-organized Chain Algorithm

2.1 K-Means prototype based algorithm

Let us first present quickly the K-means algorithm. K-means is an unsupervised learning algorithm designed to solve clustering problems. Let us consider a data set $\chi = \{x_1, x_2, ...x_N\}$ in the Euclidean space $\Re^d$ where N is the number of data points and d the dimension of the data set. Let us consider that we want to find a partition of χ into k clusters where is k is given. Formally, the objective is to find the position $\mathbf{c}_j$,where $1 \leq j \leq k$, of the k clusters centroids (prototypes) that minimize the objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \parallel V_{ij}(\mathbf{x}_i - \mathbf{c}_j) \parallel^2$$

where $\parallel \mathbf{x}_i - \mathbf{c}_j \parallel^2$ is the distance measure representing the level of similarity between a data point $\mathbf{x}_i$ and the cluster centre $\mathbf{c}_j$. V_{ij} takes the value 1 if the point x_i is assigned to cluster c_j and 0 otherwise.
The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated

The algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. It might get stuck in a local minimum. The algorithm is also significantly sensitive to the initial position of the centroids. One way to escape from local minimum or at least to improve the quality of the solution obtained, is to run the algorithm from different initial conditions.

2.2 Self-Organized Map (SOM) for clustering

In this section we introduce briefly the SOM concepts leading to solve clustering problems. The SOM is based on unsupervised learning, which means that no human intervention is needed during the learning and that little needs to be known about the characteristics of the input data. It provides a topology preserving mapping from the high dimensional space to the neurons. Neurons, usually form

a two dimensional lattice, and thus the mapping is a mapping from high dimensional space onto a plane. The property of topology preservation means that the mapping preserves the relative distance between the points. Points that are near each other in the input space are mapped to close neurons in the SOM. The SOM can thus serve as a cluster analysis tool for high-dimensional data [3,4].The neighborhood relationship between inputs is defined according to the Euclidean distance in this paper. Several architectures have been designed in order to solve a wide variety of problems. The ring architecture has been first introduced by Fort in 1988[5] to solve the Traveling salesman problem. This approach was extensively analyzed in [6]. This concept was further extended to other routing problems by Ghaziri in [7,8].

2.3 SOC Clustering

In this new approach we replace the centroid of the K-means algorithm by a chain of prototypes. Each cluster will be represented by one chain, and each point in the data set will be assigned to one chain. The assignment procedure is a competitive one i.e. the different chains will compete in order to represent a certain point. All points assigned to the same chain will be considered as belonging to the same cluster. The formation of the clusters is following an iterative on-line procedure.

From a clustering perspective, the concept of neighborhood is very powerful. In fact our objective in a clustering problem is to find a partition such that the similarity among points belonging to the same clusters is maximized and similarity among points of different clusters is minimized. The concept of similarity is formalized by the concept of neighborhood in the data space. Two similar inputs are represented by two neighboring neurons. The chain architecture is extending the concept of prototype. A prototype is a point in the data space that is representing a whole cluster. From a coding perspective the prototype is very useful because it is leading to a reduction in data representation. From a clustering perspective this is not a major concern. Therefore the fact that we have a whole chain of neurons to represent a cluster is not a drawback on the contrary it allows more flexibility in representation and generalizes the prototype approach.

Designing a neural algorithm consists of the following three procedures: 1) Define the architecture giving the way neurons are connected together, 2) The competition procedure gives the way neurons are competing together in order to represent an input 3) the adaptation rule according to which neurons are adjusting their state. In this paper, it means adjusting their position in the Euclidean space $\Re^d$.

The Architecture. Usually, in SOM algorithms neurons are placed on a grid in which a certain neighborhood relationship is defined. For SOC, neurons are placed on a chain. In this architecture, each neuron is linked to two neighboring neurons the antecedent and the next neuron. For the head and tail neurons of the chain, they are connected to the next neuron and antecedent one respectively. In some cases, the head and tail of the chain are linked we obtain a ring

that could be more appropriate for some data sets. In general, we have k chains corresponding to the number of clusters desired. Each chain will be formed of m neurons. Although clusters might not have the same sizes, we are considering homogenous chains because we do not know which chain will be assigned to which cluster. The number of neurons in each chain is empirically defined as the total number of data points divided by the number of clusters. $m = \lceil N/k \rceil$. Each neuron $\mathbf{n}_{ij}$ is a vector in the data space and is identified by to indexes, with $1 \leq i \leq m$ where m is the number of neurons in each chain; and j is the index of the cluster to which the neuron is belonging, where $1 \leq j \leq k$. The lateral distance between two neurons $\mathbf{n}_{ij}$ and $\mathbf{n}_{lj}$ in the same chain j is $d_L(\mathbf{n}_{ij}, \mathbf{n}_{lj}) = |l - i|$. It is equal to a great number if neurons are not in the same chain.

Competition Procedure. The competition in this network occurs on two levels . There is a competition at the level of the chains and a competition between the neurons inside a chain. The way we implement this competition is as follows. We start by presenting a data point (an input) and we compare its position with the position of all neurons . The winning neuron, meaning the nearest neuron to the input, will allow us to select also the winning chain to which it belongs. This procedure corresponds to the on-line algorithm. The winning neuron will be denoted $\mathbf{n}_{i^*j^*}$ where j^* is the index of the winning chain and i^* is the index of the winning neuron in the winning chain. A batch algorithm could be designed following the Heskes [9] procedure. We did not not consider this perspective since the objective of such procedure is to reduce the CPU time. In this paper our objective is to enhance the performance of the solution obtained in terms of accuracy and not CPU time.

The Adaptation rule. Suppose that the point $\mathbf{x}_p$ has been presented and that the corresponding winning neuron is n_{i^*,j^*}. The adaptation rule applied, is the same as the one used for SOM:

$$\mathbf{n}_{ij}(t+1) = \mathbf{n}_{ij}(t) + \eta V_{jj^*} \Gamma(d_L(n_{ij}, n_{i^*,j^*}, \sigma)(\mathbf{x}_p - \mathbf{n}_{ij}(t))$$

the only difference is the term V_{jj^*} that takes the value 1 if $j = j^*$ and zero otherwise. It guarantees that only the neurons in the winning chain are adapted. The other neurons in the remaining chains will not be adapted. Γ is gaussian function controlling the cooperation procedure between the winning neuron and its neighbors. σ is a parameter controlling the standard deviation of the gaussian . η allowing the algorithm to converge by reducing its value at each iteration. is another control parameter
Here are the steps of the SOC algorithm.

1. Initialization. Placing the K chains in their initial position. Fix the number k of chains, the initial values of the parameters and number of epochs
2. Select a data point $\mathbf{x}_p$
3. Compare the positions of all neurons with the position of the presented point. Select the nearest neuron. The chain to which the winning neuron is considered as the winning neuron.

4. Apply the SOM adaptation rule

5. Stop when the number of epochs is reached

3 Implementation

In this section various aspects of the implementation are discussed. Concerning the initialization of the parameters and their update rule, we have in fact two parameters η and σ those two parameters are usual parameters of the SOM algorithm . We used default values for these parameters without a major change during experimentations. In addition we fixed the number of neurons in each chain. The number of neurons used was $\lfloor n/2k \rfloor$. Increasing beyond this value the number of neurons was not improving the results but increasing drastically the CPU time. The order of presenting the data points was the random order without considering any other prior information. Finally , the most important issue is the initialization the chains. It is well known that the k-Means algorithm is very sensitive to initial conditions. Given the SOM parameters that we used, the initial position of the chains did not have a major impact on the results. In fact the parameter σ was chosen such that the influence of the winning neuron in the first iterations was very important on its neighbors in the chain, leading to a kind of aggregation of all neurons in a small area of the data space. And then iteration after iteration, the chain unfold to approximately fit the shape of the cluster.

4 Experiments

We have considered some artificial data extracted from [10] to test the quality of the SOC algorithm in terms of robustness and performance. The performance measure we used is the accuracy of the clustering. Since the data are artificial we know how many clusters we should have and to what clusters each point should belong. Hence the accuracy is the number of assignments to the right cluster over the total number of points. The following table shows that the SOC outperforms the k-Means algorithms for these artificial data . The only situation where the results obtained by SOC where not clearly outperforming the k-SOM is the Atom dataset. This dataset is formed of two concentric clusters of different densities. It shows that SOC is very good for irregular shapes but needs to be improved for clusters of different densities. In terms of robustness to parameters and most importantly to initial positions, SOC showed that it is not sensitive to initial positions . Although we performed experiments on each dataset 10 times and selecting the best result there was no big discrepancy between different initialization for SOC while K-means as expected was quite sensitive to this aspect. The data sets are extracted from the following website http://www.mathematik.uni-marburg.de/ databionics/en//?q=data where all details on these sets can be found. The results of the experiments conducted are given by Table 1.

Dataset	K-means	SOC
Hepta	1	1
Lsun	0.5	1
Tara	1	1
Chainlink	0.5	0.95
Atom	0.5	0.47
Two diamands	1	1
Wing Nut	0.8	0.95

Table 1: Comparing the performance of SOC and k-Means. The accuracy of the clustering solution is between 0 and 1, 1 meaning that the accuracy is perfect.

5 Conclusion

In this paper we have proposed a new clustering algorithm generalizing k-means by using a competitive set of SOMs. The maps considered were reduced to simple chains of neurons. We compared this SOC algorithm to k-Means on artificial data sets representing different types of difficulties for clustering, varying from clusters with different densities to irregular shapes . SOC shows excellent performance across the different types of datasets. Another important advantage of SOC over k-means is its robustness to initial conditions.

References

[1] M. N. Murty, A.K. Jain, & P.J. Flynn, Data clustering: A review, *ACM Computing Surveys*, 31:264-323, 1999.

[2] T. Villman, R. Der, M.Herrmann, & T. Martinetz, Topology preservation in self-organizing feature maps: Exact definition and measurement, *IEEE Transactions on Neural Networks*, 2:256-266, 1994.

[3] T.Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin, 1995.

[4] J. Vesanto and E. Alhoniemi (2000). *Clustering of the Self-Organizing Map,* IEEE Transactions on Neural Networks, Volume 11, Number 3, pp. 586-600, 2000.

[5] J.C. Fort, Solving a combinatorial problem via self-organizing process: An application to the traveling salesman problem, *Biological Cybernetics*, 59:33-40, 1988.

[6] K.A. Smith, Neural network for combinatorial optimization: A review of more than a decade of research, *INFORMS Journal on Computing*, 11:15-34, 1999.

[7] H.Ghaziri, Supervision in the self-organizing feature map. application to the vehicule routing problem. In. I.H. Osman and J.P. Kelly, *Meta-Heuristics: Theory and applications*,Kluwer Academic publishers, Boston 651-660, 1996.

[8] H.Ghaziri, & I.Osman, A neural network algorithm for travelling sales man problem With backhauls, *Computers & Industrial Engineering*, 44:267-281, 2003.

[9] T. Heskes, Self-organizing maps, vector quantization, and mixture modeling, *IEEE Transactions on Neural Networks*, 12:1299-1305, 2001.

[10] A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data,* Prentice Hall, New York, 1988.