COMPARISON BETWEEN THREE DIFFERENT CLUSTERING ALGORITHMS

Markus Törmä
markus@mato.hut.fi
Helsinki University of Technology
Institute of Photogrammetry and Remote Sensing
Espoo, Finland

Abstract

New approaches like neural networks and fuzzy sets have been used more and more in pattern recognition during recent years. In this article, neural network and fuzzy clustering algorithms are compared to the traditional clustering algorithm.

1. INTRODUCTION

Traditional statistical pattern recognition models have been studied for quite a long time. Due to difficulty of a pattern recognition task (e.g. classification) there are many different models for different tasks and problems. Common for these models are that they work well when certain conditions are fulfilled (e.g. normal distribution of dataset), otherwise not. Need for a general model which could be successful in most pattern recognition tasks have led to studying new approaches like neural networks and fuzzy sets. But the big question is, can these new approaches offer significant advantages compared to the traditional models.

Aim of this article is to compare performance of traditional clustering algorithm to similar neural network and fuzzy clustering algorithms. But first there are short introductions to neural networks, fuzzy sets and clustering. Readers interested in neural networks should see more detailed introduction from article "Introduction to Neural Networks and Their Use in Remote Sensing" from The Photogrammetric Journal of Finland, vol. 13, no. 1, 1992.

1.1 Neural networks

A neural network is a parallel, distributed signal or information processing system consisting of simple processing elements also called as nodes. Processing elements can possess a local memory and carry out localized information processing operations. Processing elements are connected via unidirectional signal channels called connections. The connections are usually weighted and those weights are adapted during training of the network. Learning of the network is based on the adaptation of the weights. A neural network can be made using special hardware technology or conventional programming languages. Neural networks try to imitate a biological nervous system and its properties like memory and learning /1//2/.

Neural networks can be used for different tasks. First, they can define which class respond best to some input. This case is a normal classification problem. Secondly, neural network can be used as an associative memory. In that case class exemplar is desired and the input pattern is used to determine which exemplar to produce. Associative memory is very useful when only a part of the input pattern is known and complete pattern is required. Third task is to divide inputs into different groups or to make vector quantization. In that case neural network is used to form clusters or compress inputs /2/.

1.2 Fuzzy sets

Fuzzy sets are generalization of conventional (Boolean) sets that have been extended to handle the concept of partial belonginess. In Boolean case, vector belongs to set completely or it does not belong to set at all, and it can belong to only one set. In fuzzy case vector belongs to set with certain degree and it can belong to several sets certain degrees. If we speak about logic, the same can be said in Boolean logic propositions are completely true (marked as 1) or completely false (marked as 0), but in fuzzy logic propositions can be partly true. The fuzzy set theory has been made for that it should be easier to handle real world vagueness like paradox in artificial intelligence /7/.

Mathematically speaking, fuzzy set is a set and a function (membership function) F mapping elements of the set into the interval [0,1]. If x is an element of a fuzzy set, then F(x) is called the membership value of x. F(x) describes the degree of compatibility of x with the concept represented by the fuzzy set (membership value 0 means total incompatibility and membership value 1 total compatibility). Fuzzy set is boolean set if all its elements have membership value 1 /7/.

There are two ways to describe uncertainty; randomness and fuzziness. Both systems use numbers in the unit interval [0,1] and both combine sets and propositions associatively, commutatively and distributively. Main distinction between them is that fuzziness describes event ambiguity and randomness describes event occurrence. So, randomness and fuzziness describe different kind of uncertainty /3/.

Kosko introduces in his book /3/ a very interesting way to examine fuzzy sets and its properties using unit hypercube. He also introduces fuzzy entropy theorem and subsethood theorem. Fuzzy entropy theorem can be used to determine uncertainty of fuzzy system and subsethood theorem to determine the degree to which set A is a subset of set B. Latter could be useful in classification. There are still some open questions. Like how should membership functions be determined? Are they same as probabilistic density functions? Is it possible to use neural networks to determine membership functions?

1.3 Clustering

In clustering, also called unsupervised classification, the classes of the available patterns (clusters) are unknown and sometimes even the number of these classes is unknown. In this kind of problems, one attempts to find classes of patterns using some measure of similarity. Usually similarity is defined as proximity of the points in the measurement space (patterns) according to a distance function, but similarity can be based on other properties such as the direction of the vectors in the measurement space. The method for finding the clusters may have a heuristic basis or may be more rigorously dependent on minimization of a mathematical clustering criterion /8/.

Clustering methods can be divided into two main groups, hierarchial and partitional methods. In the hierarchial methods separate clusterings are set up hierarchial series. Usually this kind of series are presented using tree structure. The partitional methods perform clustering minimizing or maximizing some criterion function. These methods are usually iterative /8/.

General method for partitional clustering is quite simple. We have a set of samples (Y) and class labels assigned to the samples (Ω). Also we have some criterion function $J(Y,\Omega)$ that measures the optimality of the classification. To get optimal classification we have to minimize J. Procedure /4/:

1. Choose initial classification $\Omega^{(0)}$ and evaluate J.
2. Change the classification in a way that tends to decrease J.
3. If the new classification is same as the old then stop; else go to step 2.

2. ALGORITHMS

Three different clustering algorithms were programmed and their performance compared. These algorithms were hard k-means algorithm, fuzzy k-means algorithm and similar neural network algorithm called Kohonen self-organizing feature map.

2.1 Hard k-means algorithm (HKM)

In this algorithm clusters are identified by their mean vectors. Criterion to be minimized is that distances from feature vectors to cluster mean vectors should be as small as possible.

1. Begin with an arbitrary set of cluster mean vectors and assign samples to nearest mean vectors.
2. Compute the sample mean of each cluster.
3. Reassign each sample to the cluster with the nearest mean vector.
4. If the classification of all samples has not changed, stop; else go to step 2 /4/.

2.2 Fuzzy k-means algorithm (FKM)

Differences between hard k-means and fuzzy k-means algorithms are that in latter case feature vectors belong to every clusters with certain degrees determined by membership values. Also every feature vector have effect to computing every cluster mean vectors. After iteration, correct cluster for feature vector is chosen using the highest membership value.

1. Begin with an arbitrary set of cluster mean vectors.
2. Compute the membership values. x denotes feature vector and v cluster mean vector.

$$F_i(x) = \frac{\frac{1}{d(x,v_i)}}{\sum_{j=1}^{k} \frac{1}{d(x,v_j)}},$$

where k is the number of clusters and d() is a distance function (Euclidean distance was used).

If some distance d() = 0, then

$$F_i(x) = \begin{cases} 1 \text{ if } d()=0 \\ 0 \text{ if } d()\neq 0 \end{cases}$$

3. Compute new weighted cluster mean vectors.

$$v_i = \frac{\sum_{x \in X} (F_i(x))^2 x}{\sum_{x \in X} (F_i(x))^2}$$

4. Compute maximum distance δ between old and new cluster mean vectors.
5. If δ is smaller than threshold value, stop; else go to 2 /5/.

2.3 Kohonen self-organizing feature map (SOM)

The processing elements of the network are made competitive in a self-organizing process and winning processing element whose weights are updated is chosen by some criteria. The basic idea is that the weight vectors of the processing elements approximate the probability density function of the input vectors /6/.

SOM learns a continuous topological mapping f: $B \subset \mathbb{R}^n \rightarrow C \subset \mathbb{R}^m$ using unsupervised learning. This is nonlinear mapping from n-dimensional space of input vectors to m-dimensional space of weight vectors. Weight vectors are arranged so that processing elements which are topologically near each other, are sensitive to input vectors which are physically similar. So, the weight vectors are arranged in a natural way and in clustering weight vectors correspond to cluster mean vectors /6/.

1. Initialize weights to small random values.
2. Choose input randomly from dataset.
3. Compute distance to all nodes.
4. Select output node j with minimum distance.
5. Update weights to node j and neighbors.

$$w_j(t+1) = w_j + \alpha(t)(x(t) - w_j(t)),$$

where α gain term ($0<\alpha<1$) that decreases in time. Also size of neighborhood decreases in time.

6. Go to step 2 or stop iteration if enough inputs are presented /2/.

3. RESULTS

Experiments were made using three different datasets. First, there was normal distributed dataset with two clusters. Then there was non-normal distributed dataset with two clusters. These two datasets were generated using random number generator. Final experiment was made using a Spot satellite image.

3.1 Normal distributed dataset

There were four different normal distributed datasets; 2- or 3-dimensions and 1000 or 10000 vectors per cluster. In 2-dimensional case mean vector for first cluster was m_1 = [0.3 , 0.3] and for second cluster m_2 = [0.7 , 0.7]. In 3-dimensional case mean vector for first cluster was m_1 = [0.3 , 0.3 , 0.3] and for second cluster m_2 = [0.627 , 0.627 , 0.627]. Deviations in all dimensions were 0.2. Clustering of each dataset was repeated 100 times so we could see how well algorithms could repeat themselves and what were different partitionings. Also mean value of error, its standard deviation, minimum and maximum values were computed.

Experiment with SOM was run with different parameter values. In 2-dimensional case SOM with α_0 = 0.01, linear α decrease and NE_0 = 1 was chosen. In 3-dimensional case SOM with α_0 = 0.02, linear α decrease and NE_0 = 1 was chosen. During iteration 100000 vectors were presented to SOM. Results with other parameter values were quite similar to these presented.

Table 1.

Results of normal distributed dataset in 2-dimensions.

	mean	dev	min	max
HKM	9.81%	0.0	9.81%	9.81%
FKM	10.33%	4.19	9.76%	49.85%
SOM	9.82%	0.02	9.76%	9.87%

Table 2.

Results of normal distributed dataset in 3-dimensions.

	mean	dev	min	max
HKM	8.803%	0.003	8.8%	8.805%
FKM	9.148%	4.046	8.785%	44.265%
SOM	8.805%	0.033	8.75%	8.915%

Results of experiments with different datasets were so similar that only cases with 10000 vectors per cluster are discussed. In 2-dimensional case (table 1), HKM performed in the best way. Mean value of error was smallest and standard deviation was zero. Algorithm converged every time to same solution. Mean value of error of SOM was very close to HKM and standard deviation was small. FKM performed worst. Problem with FKM in this case is that mean vectors of clusters are computed using every feature vectors. When different clusters are similar and overlapping, mean vectors of clusters move during iteration close to each other and depending on initial values they move to wrong position. In 3-dimensional case, results were similar to the 2-dimensional case (table 2).

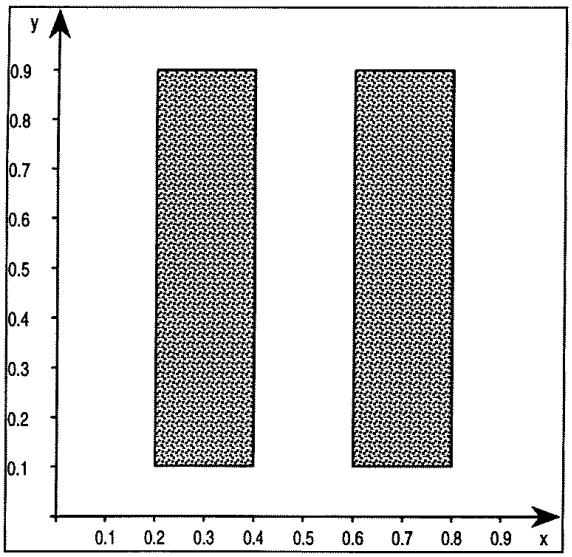
3.2 Non-normal distributed dataset

There was six different non-normal distributed datasets; 2- or 3-dimensions and 1000, 5000 or 10000 vectors per cluster. Clusters were like two linearly separable parallel blocks (see figure 1). In 2-dimensional case, values for x-axis were generated within interval [0.2 , 0.4] for first cluster and interval [0.6 , 0.8] for second cluster. Values for y-axis were generated within interval [0.1 , 0.9]. In 3-dimensional case, values for x- and y-axis were generated like in 2-dimensional case and values for z-axis were generated within interval [0.0 , 0.2]. Clustering of each dataset was repeated 100 times so we could see how well algorithms could repeat themselves and what were different partitionings. Also, number of correct clusterings were counted and mean value of error, its standard deviation and maximum values were computed. Minimum value of error were always zero.

Experiment with SOM was run with different parameter values. In both cases SOM with $\alpha_0 = 0.02$, linear α decrease and $NE_0 = 1$ was chosen. During iteration 100000 vectors were presented to SOM. Different parameter values had significant effect to results and only best results are presented.

Figure 1.

Non-normal distributed dataset in two dimensions.



Results of experiments with different datasets were so similar that only cases with 10000 vectors per cluster are discussed. Most important column in tables 3 and 4 is first one (cor), which indicates how many times algorithm converged to correct solution. In the 2-dimensional case (table 3), FKM performed in the best way. Almost every time it converged to correct solution. Although SOM was little better than HKM, SOM and HKM performed noticeably worse than FKM. They both converged to correct solution in less than half cases. 3-dimensional case was otherwise similar to the 2-dimensional case, except differences in number of correct solutions were smaller.

Table 3.

Results of non-normal distributed dataset with 2 dimensions.

	cor	mean	dev	max
HKM	41	29.16%	24.43	49.43%
FKM	96	1.39%	7.0	46.53%
SOM	46	26.64%	24.71	49.99%

Table 4.

Results of non-normal distributed dataset with 3 dimensions.

	cor	mean	dev	max
HKM	24	37.54%	21.2	49.4%
FKM	64	4.49%	11.07	49.84%
SOM	44	27.57%	24.56	49.99%

3.3 Spot image

Satellite image used in this experiment was 3-channel Spot image, size 256*256 pixels. Before clustering, pixel values were normalized between [0 , 1]. Because the correct partition of image and number of clusters were unknown, comparison between algorithms is based on quantization error and its deviations of different algorithms. Quantization error is sum of squared distances between every feature vector and its nearest cluster mean vector. Clusterings were made using number of clusters between 2 and 9. Clustering was repeated 20 times so we could see how well algorithms could repeat themselves and what were different partitionings.

If performance of different algorithms is evaluated using quantization error, SOM performs in the best manner. With different number of clusters, quantization error and its deviation (table 5) were smallest in every case using SOM. This indicates that placements of cluster centers were as good as possible and SOM was able to converge to (almost) same solution every time. Quantization error of FKM was smaller than error of HKM at first, but decrease of error of FKM was slower than of HKM. Error of FKM even increased when number of clusters were increased from six to eight. This was because every feature vector affects to evaluation of cluster mean vectors. When real data is used and different clusters are not well-separated, cluster centers move close to each other and two different cluster centers can move to same place in feature space. Error of HKM was smaller in most cases than error of FKM, but deviations were noticeably larger. This means that HKM is very sensitive to first values of cluster mean vectors evaluated by random number generator.

Clusterings were also examined visually, or original image was compared to clustering result. Ten clustering results with six clusters were chosen for comparison. Although SOM performed best, differences between SOM and FKM were surprisingly small. Main problem with FKM was that some clusters were

not "natural", or e.g. two different clusters were both mixtures of forest and grassland. Clusterings of HKM were unstable, algorithm converged on very different kind of results.

Table 5.

Mean values and deviations of quantization error of different algorithms. Most left column indicates how many clusters were formed. Abbreviations: mn = mean value, dv = deviation, HKM = hard k-means clustering algorithm, FKM = fuzzy k-means clustering algorithm, SOM = Kohonen self-organizing feature map.

	HKMmn	HKMdv	FKMmn	FKMdv	SOMmn	SOMdv
2	6461.7	3068.4	4987.6	0.0	4982.2	0.1
3	3871.0	2031.4	3806.0	0.9	2842.7	0.0
4	2355.2	883.8	3131.7	181.1	1640.2	0.9
5	1652.6	601.7	2673.6	189.5	1271.5	107.5
6	1195.3	185.7	2175.6	151.3	908.0	24.2
7	1026.5	359.7	2201.9	37.4	746.1	0.2
8	822.3	159.0	2323.7	27.7	653.6	12.8
9	726.0	144.4	2092.4	383.8	581.3	3.1

4. CONCLUSIONS

Three different algorithms based on different "philosophies" were introduced and compared using three different datasets. There was no superior algorithm, but performance of an algorithm seemed to depend heavily on the nature of dataset. In normal distributed case, HKM performed best and FKM worst. In non-normal distributed case, FKM performed best and HKM worst. When using a real satellite image, SOM performed best and HKM worst. If I should choose one algorithm to use, I would choose SOM.

REFERENCES

/1/ R. Hecht-Nielsen:
Neurocomputing
Addison-Wesley, 1989

/2/ R.P. Lippmann:
An Introduction to Computing with Neural Nets
IEEE ASSP Magazine, April 1987, pp. 4-22

/3/ B. Kosko:
Neural Networks and Fuzzy Systems
Prentice-Hall

/4/ C.W. Therrien:
Decision, Estimation and Classification
John Wiley & Sons, 1989

/5/ J.C. Dunn:
A Fuzzy Relative of the ISODATA Process and
Its Use in Detecting Compact Well-Separated Clusters
J. Cybernetics, vol. 3, no. 3, pp. 32-57, 1973

/6/ T. Kohonen:
The Self-Organizing Map
Proc. of IEEE, vol. 78, no 9, September 1990, pp. 1464-1477

/7/ Internet newsgroup Comp.ai.fuzzy Frequently Asked Questions

/8/ T.Y. Young, K.-S. Fu:
Handbook of Pattern Recognition and Image Processing
Academic Press, 1986