CLUSTERING WITH MEAN FIELD ANNEALING AND UNSUPERVISED LEARNING

Yizhou Yu
Applied Math. Dept., Zhejiang University 310027
Hangzhou, PRC.

ABSTRACT

When neural networks are used to solve a clustering problem, there is often no precise measure. But in such fields as pattern recognition, a clustering problem is often with an objective function. In this paper, MFT neural nets are taken to tackle such a problem. Even when the number of clusters is unknown, an unsupervised neural network with gradient descent can evaluate it. The experimental results is satisfactory.

INTRODUCTION

Clustering is an important problem in many fields, such as astrophysics, image analysis, taxonomy, etc. Hence, many techniques have been developed to tackle this problem. In recent years, techniques inspired by neural networks have also been added to the repertory[1][2][5][6][4]. Kohonen's [5] self-organizing maps have been used to partition the feature space into distinct regions. In the multi-layered architecture[6] the neural net can be trained to partition the feature space by repeated showings of a sample data set to the network. Clustering can also be formulated more precisely in terms of the Hopfield model of neural nets[1][2].

In this paper, we formulate the clustering problem in terms of the mean field theory (MFT) neural nets[4] and investigate its performance. An unsupervised neural network[3] is also used to evaluate the number of clusters.

METHODS AND EXPERIMENTAL RESULTS

By clustering we mean partitioning a set of N patterns into K (given or not given) clusters in such a way that those in a given cluster are more similar to each other than the rest. As is often done, when K is given, we let our criterion for best solution of the problem be the minimum square-error. That is representing the patterns by d-dimensional vector $\{r_i | i = 1, \cdots, N\}$, the best solution is the one minimizing $X^2 = \sum_{i=1}^{N} (r_i^{(p)} - R_p)^2$ with respect to $\{R_p | p = 1, \cdots, K\}$. Here cluster p contains the subset of the vectors $\{r_i^{(p)}\}$ and its centroid is given by $R_p = \sum_{i=1}^{N_p} r_i^{(p)}$, where N_p is the number of patterns in the cluster.

It can be easily shown that

Theorem 1 Clustering problem is independent of translational, scaling and rotational transformation.

1. Clustering with Given K

MFT networks are very suitable for solving optimization problems and the clustering problem with given K can be cast in terms of an energy function of MFT networks. Thus, mean field annealing can be used to obtain the optimal or a very good solution. Since one pattern can belong to only one cluster, multi-state neurons with K states are used in our scheme.

According to the above criterion, the cost term in the energy function can be written as

$$\sum_{p=1}^{K} \sum_{i=1}^{N} R_{ip} V_{ip}^2 \tag{1}$$

where $R_{ip} = (r_i - R_p)^2$ and $R_p = \sum_{i=1}^{N} V_{ip} r_i$. The constraint part of the energy function can be expressed as

$$\sum_{i=1}^{N} \sum_{p=1}^{K} \sum_{q=1}^{K} V_{ip} V_{iq} \tag{2}$$

This term turns out important for the mean field dynamics. It drives every neuron to have only one component on (close to 1). In the case of multi-state neuron, it can be shown that this term is equivalent to

$$- \sum_{i=1}^{N} V_i^2 \tag{3}$$

It is simpler to implement (3) by analog VLSI. The energy function can then be written as

$$E = \frac{A}{2} \sum_{p=1}^{K} \sum_{i=1}^{N} R_{ip} V_{ip}^2 - \frac{B}{2} \sum_{i=1}^{N} V_i^2 \tag{4}$$

MFT equation can be written as

$$V_i = F_K(U_i) \ \textit{where} \ F_K^a(U_i) = \frac{e^{U_{ia}}}{\Sigma_b e^{U_{ib}}} \tag{5}$$

$$U_{ia} = \frac{\partial E}{\partial V_{ia}} \frac{1}{T} - - (AR_{ia} - B) \frac{V_{ia}}{T} \tag{6}$$

In MFT networks, (random) sequential updating is preferred. As the network evloves, R_{ia}'s become generally smaller and the cost term becomes less effective in driving the network toward good solutions. So parameter A is adjusted during the evolution such that $AR_{avg} = CN$ (a constant), where $R_{avg} = \frac{1}{KN} \sum_{i,s} R_{is}$. Because the weights are not fixed, when T decreases to zero, the newwork may not converge to a good solution and sometimes may oscillate.

$$Let\ MV_i = max V_{ip},\ p_i\ satisfy\ V_{ip_i} = MV_i,$$
$$MD_i = \min_{q(q \neq p_i)} (MV_i - V_{iq})\ and\ s = \{j \mid MD_j > 0\}.$$

Theorem 2 At a fixed T, if the parameters can be adjusted, during the evolution, to satisfy the following conditions where b is a constant greater that one, the above MFT network converges to a stable solution.

$$B \geqslant \max_{i \in s}\{\max_{q \neq p_i}[(A(R_{ip_i}MV_i - R_{iq}V_{iq}) + T\ ln \frac{bMV_i}{V_{iq}})/(MV_i - V_{iq})]\} \quad (7)$$

and

$$A(R_{ip_i}MV_i - R_{iq}V_{iq}) + T\ ln \frac{bMV_i}{V_{iq}} \leqslant 0, \quad for\ i \notin s,\ q \neq p_i. \quad (8)$$

Its proof is unimportant and is left out in this paper

The above theorem only ensures convergence. It does not ensure goodness of the final solution. So it has little practical value. In order to obtain a good solution, we adopt another MFT network as a preprocessor. This prepocessor is similar to optimal graph partition network. All pattern vectors can be considered as vertices in a graph. Between each pair of vertices r_i and r_j, there is an edge with weight $(r_i - r_j)^2$. Our objective is to partition the vertices into several subsets $vs_l(l = 1, \cdots, K)$ and minimize the value of $F = \sum_{l=1}^{K} \sum_{i,j \in vs_l} (r_i - r_j)^2$. The energy function can be written as

$$E = \frac{C}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij} V_i \cdot V_j - \frac{D}{2} \sum_{i=1}^{N} V_i^2 \quad (9)$$

where $T_{ij} = (r_i - r_j)^2$. MFT equations can be expressed as

$$V_i = F_K(U_i) \quad (10)$$

$$U_i = -\frac{\partial E}{\partial V_i}\frac{1}{T} = -[\sum_j (CT_{ij} - D\delta_{ij})V_j]\frac{1}{T} \quad (11)$$

where $\delta_{ij} = 1$ when $i = j$
$= 0$ otherwise.

We can see that

Theorem 3 To the above two objective functions X^2 and F, if the set of patterns are equally partitioned in both optimal solutions, X^2 and F are equivalent.

In the preprocessor, all weights are fixed. It can be easily driven to get a good solution. At high temperatures, the updating follows equation (10) and (11) and the patterns are coarsely partitioned, while at low temperatures, the updating is switched to follow equation (5) and (6) and the partition is refined. This mechanism can effectively prevent the network from oscillation and get the optimal or at least a very good solution at last.

Except the parameters CN, B, C and D, there are several others; the dimensionality d of the patterns, the initial temperature T_0, the final temperature T_1, the number of updating epochs AT and the switching time ST. In experiments, we find the coefficient B plays a significant role. If all other parameters are fixed, there exists an interval $[B_0, B_1]$ for B. When $B < B_0$, the network tends to oscillate. When $B > B_1$, the network always get stuck in local minima. Furthermore, there exists $B^* \in [B_0, B_1]$ which can drive the network to get the optimal solution and X^2 can be considered as a monotonous function of B in $[B_0, B^*]$ and $[B^*, B_1]$. According to this fact, we can dertermine the parameters CN, C, D, d, T_0, T_1, AT, ST, B_0 and B_1 in advance and the interval $[B_0, B_1]$ is discretized to points $b_0, b_1, \cdots, b_n$. When input patterns and the number of clusters are given, we make a binary search for the optimal discrete value b^*. The network runs at most three times in each step of the search. Once b^* is found, the optimal or a very good solution is found at the same time. This procedure is called parametric optimization.

In the same experiment which was done in [1], we have determined the following values for the parameters: CN = 1800, C = 30, D = 90, B_0 = 300, B_1 = 2100, T_0 = 50, AT = 50, ST = 25. In all cases, our method needs at most 15 runs of the network and each run needs only 50 epochs. The total time required on sparc station 1 was 135 seconds. While the method introduced in [1] needs an average of 4263 iterations, the total time required on MPP array machine was 163 seconds. The corresponding CPU time on a VAX 8800 was 25066s. Here, we should note that sparc station 1 is a general-purpose computer and MPP is a parallel processor. So our method is very efficient. The method in [1] wishes to meet with an opportunity to get a good solution. So it is based on many trials and no one knows when the opportunity occurs. Because the values of all parameters are fixed during the whole process, it is probable that a good solution will never be obtained with improper parametric values. However, our parametric optimizing procedure makes a more deliberate search. Furthermore, MFT networks can also be implemented easily by VLSI.

The following tables show the values of b^* and X^2 at different noise levels in the experiments when the final solutions are found.

Table 1 Values of b^* at Different Noise Levels

noise ratio / K	0%	10%	25%	50%	75%
2	900	1000	1000	1900	2000
3	500	700	900	1800	1900
4	400	600	700	1700	1800
5	300	400	600	1600	1700
6	300	400	600	1500	1600
7	300	400	600	1400	1400
8	300	400	600	1300	1300

2. Clustering with Unknown K

If the number of clusters K is not given in advance, we wishes to evaluate it with an unsupervised network. Neither Knonen's map nor competitive learning can take on this task. Here, we use a network introduced in [3] where it is used for dimensionality reduction. Consider a neuron with input vector $x = (x_1, \cdots, x_d)$, synaptic weight vector $m(m_1, \cdots, m_d)$, both in R^d, and acitvity $c = x \cdot m$. Let $\theta_m = E[(x \cdot m)^2]$. The loss function can be expressed as

Table 2 Values of X^2 of Final Solutions

K \ noise ratio	0%	10%	25%	50%	75%
2	0.04333	0.05547	0.06649	0.08698	0.09476
3	0.01274	0.01767	0.02715	0.03178	0.03636
4	0.00614	0.01023	0.01456	0.01767	0.02062
5	0.00218	0.00474	0.01024	0.01081	0.01419
6	0.00169	0.00388	0.00607	0.00852	0.00908
7	0.00140	0.00262	0.00410	0.00540	0.00643
8	0.00112	0.00218	0.00300	0.00413	0.00500

$$L_m(x) = -\frac{u}{3}[(x \cdot m)^3 - \theta_m (x \cdot m)^2] \quad (12)$$

The risk (expected value of the loss) is given by

$$R_m = -\frac{u}{3}(E[(x \cdot m)^3] - \theta_m^2) \quad (13)$$

Since the risk is continuously differentiable, its minimization can be achieved via a gradient descent method with respect to m, namely

$$m_i(t+1) = m_i(t) - \frac{\partial}{\partial m_i} R_m, i = 1, \cdots, d \quad (14)$$

In terms of theorem 1, we translate the input patterns to satisfy $E(x \cdot m_0) = 0$ where m_0 is the initial weight vector. Thus the distribution will not be concentrated at only one side of θ_m. $L_m(x)$ has a local maximum at $x \cdot m = \frac{2}{3}\theta_m$ and a local minimum at $x \cdot m = 0$. It is monotonously decreasing in $[\frac{2}{3}\theta_m, +\infty)$. If the patterns make up several clusters, it is of little probability that $x \cdot m$ falls in the vicinity of $\frac{2}{3}\theta_m$ when the unsupervised learning finishes. This unsupervised neuron can find the gaps between clusters. So it can be used for evaluation of K. As in [3], we can use a nonlinear neuron instead of the above linear one in order to address the oversensitivity to outliers.

The following is the algorithm CUK for clustering with unknown K.

A. Determine an interval $[u_0, u_1]$, where $u_0 \in [0, \frac{2}{3}\theta_m]$, $u_1 \in [\frac{2}{3}\theta_m, \theta_m]$. Determine threshold t $(t \geqslant 0)$ and the minimum number of patterns MP in a cluster. Define $L = u_1 - u_0$, $mx = \max(x \cdot m)$ and $mn = \min(x \cdot m)$;

B. Set $s_0 = \{r_i | i = 1, \cdots, N\}$ and push s_0 into stack ST;

C. Set $c = 0$;

```
D. While ST not empty do
   begin
      pop a set $s_i$ from ST;
      if $|s_i| \geqslant MP$ then
      begin
         adjust m by the unsupervised learning rule in (14);
         set $s_m = \{x | u_0 \leqslant x \cdot m \leqslant u_1, x \in s_i\}$;
         $ct_1 = |s_m|$;
         $v_0 = u_1$;
         $v_1 = u_1 + L$;
         $ct_2 = 0$;
         while $v_0 < mx$ do
         begin
            $s_r = \{x | v_0 \leqslant x \cdot m \leqslant v_1, x \in s_i\}$;
            if $|s_r| > ct_2$ then $ct_2 = |s_r|$;
            $v_0 = v_0 + L/2$;
            $v_1 = v_1 + L/2$;
         end
         $v_0 = u_0 - L$;
         $v_1 = u_0$;
         $ct_0 = 0$;
         while $v_1 > mn$ do
         begin
            $s_l = \{x | v_0 \leqslant x \cdot m \leqslant v_1, x \in s_i\}$;
            if $|s_l| > ct_0$ then $ct_0 = |s_l|$;
            $v_0 = v_0 - L/2$;
            $v_1 = v_1 - L/2$;
         end
         if $\frac{ct_0 - ct_1}{|s_i| \cdot L} > t$ and $\frac{ct_2 - ct_1}{|s_i| \cdot L} > t$ then
         begin
            $s_i$ is partitioned into subsets $s_1$ and $s_2$,
            where $s_1 = \{x | x \in s_i, x \cdot m \leqslant \frac{2}{3}\theta_m\}$, $s_2 = s_i - s_1$;
            $s_1$ and $s_2$ are pushed into ST;
         end
         else $c = c + 1$
      end
   end
E. for $K = 2c$ to 2 step $-1$ do
   begin
         run the previous MFT network;
         if the solution partition the set of patterns into K clusters and there is a gap between each pair of clusters, then break;
   end
```

In experiments, we generate clusters with within-cluster Gaussian distributions and run this algorithm 50 times. The percentage of correct evaluation and good clustering is over 80%.

CONCLUSIONS

In this paper, MFT neural networks and unsupervised networks are incorporated to solve the clustering problem. This method is promising. In one of the adopted MFT networks, the weights are changing with time. Although the experimental results are good, there is few theoretical results about convergence of this kind of networks. This topic needs further investigation.

REFERENCES

1. B. K. Parsi, J. A. Gualtieri and etc, Clustering with Neural Networks, Biol. Cybern 63 (1990), 201-208.
2. B. K. Parsi and B. K. Parsi, Simultaneous Fitting of Several Planes to Point Sets Using Nerual Networks, CVGIP 52, (1990), 341-359.
3. N. Intrator, Feature Extraction Using an Unsupervised Neural Network, Neural Computation 4 (1992), 98-107.
4. C. Peterson, Mean Field Theory Neural Networks for Feature Recognition, Content Addressable Memory and Optimization, Connection Sci., Vol. 3(No. 1, 1991), 3-33.

5. Kohonen T, Self-organization and associative memory, 3rd edn. Springer. Berlin Heidelberg New York (1989).
6. B. Widrow and M. A. Lehr, 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Back-propagation, Proc of the IEEE, Vol. 78 (NO. 9, 1990), 1415-1442.
7. S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi, Optimization by Simulated Annealing, SCIENCE, Vol. 220(No. 4598, 1983), 671-680.
8. D. E. Rumelhart and D. Zipser, Feature Discovery by Competitive Learning, Cognitive Sicence 9 (1985), 75-112.
9. B. Müller and J. Reinhardt, Neural Networks, Springer-Verlag Berlin Heidelberg (1990).