Clustering

Classical Partitioning Methods

➢ The k-means Method

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

Figure 7.2 The *k*-means partitioning algorithm.

Classical Partitioning Methods

➢ The k-means Method

Suppose that the data mining task is to cluster the following eight points (with (x, y) representing location) into three clusters:

$$A_1(2,10), A_2(2,5), A_3(8,4), B_1(5,8), B_2(7,5), B_3(6,4), C_1(1,2), C_2(4,9).$$

The distance function is Euclidean distance. Suppose initially we assign A_1, B_1, and C_1 as the center of each cluster, respectively. Use the *k-means* algorithm to show *only*

(a) The three cluster centers after the first round execution

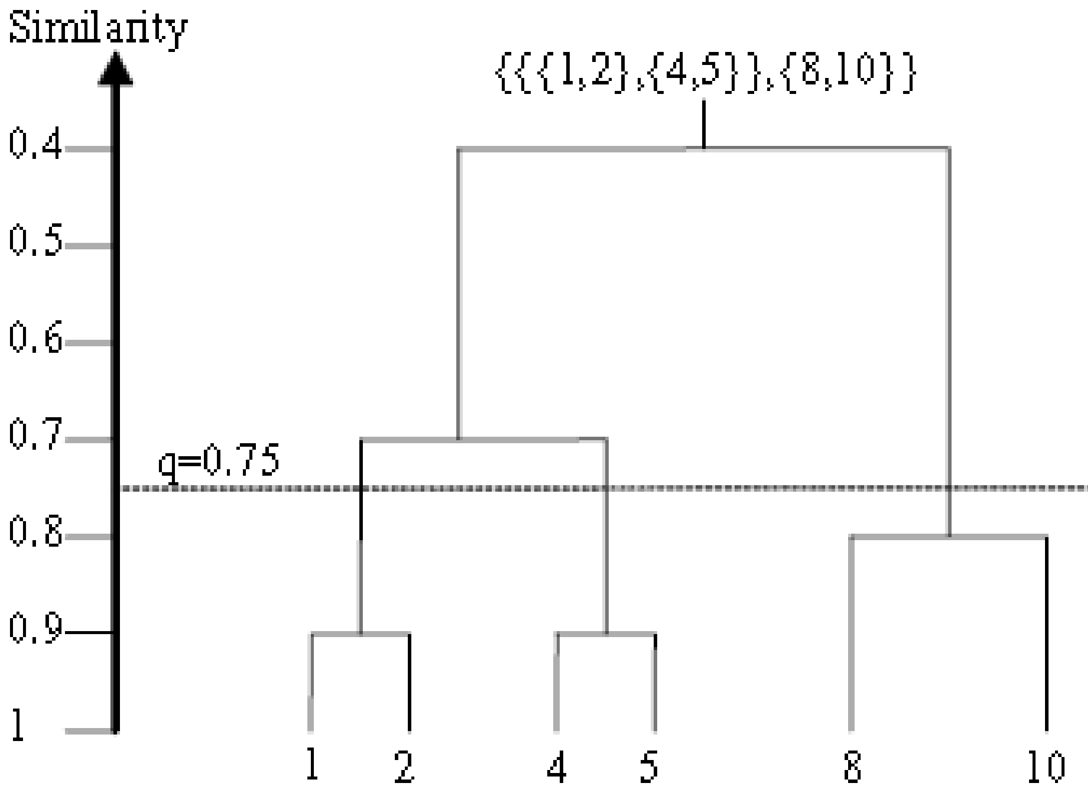
(b) The final three clusters

Classical Partitioning Methods

- The k-medoids Method

Hierarchical Methods

- Hierarchical Agglomerative Clustering



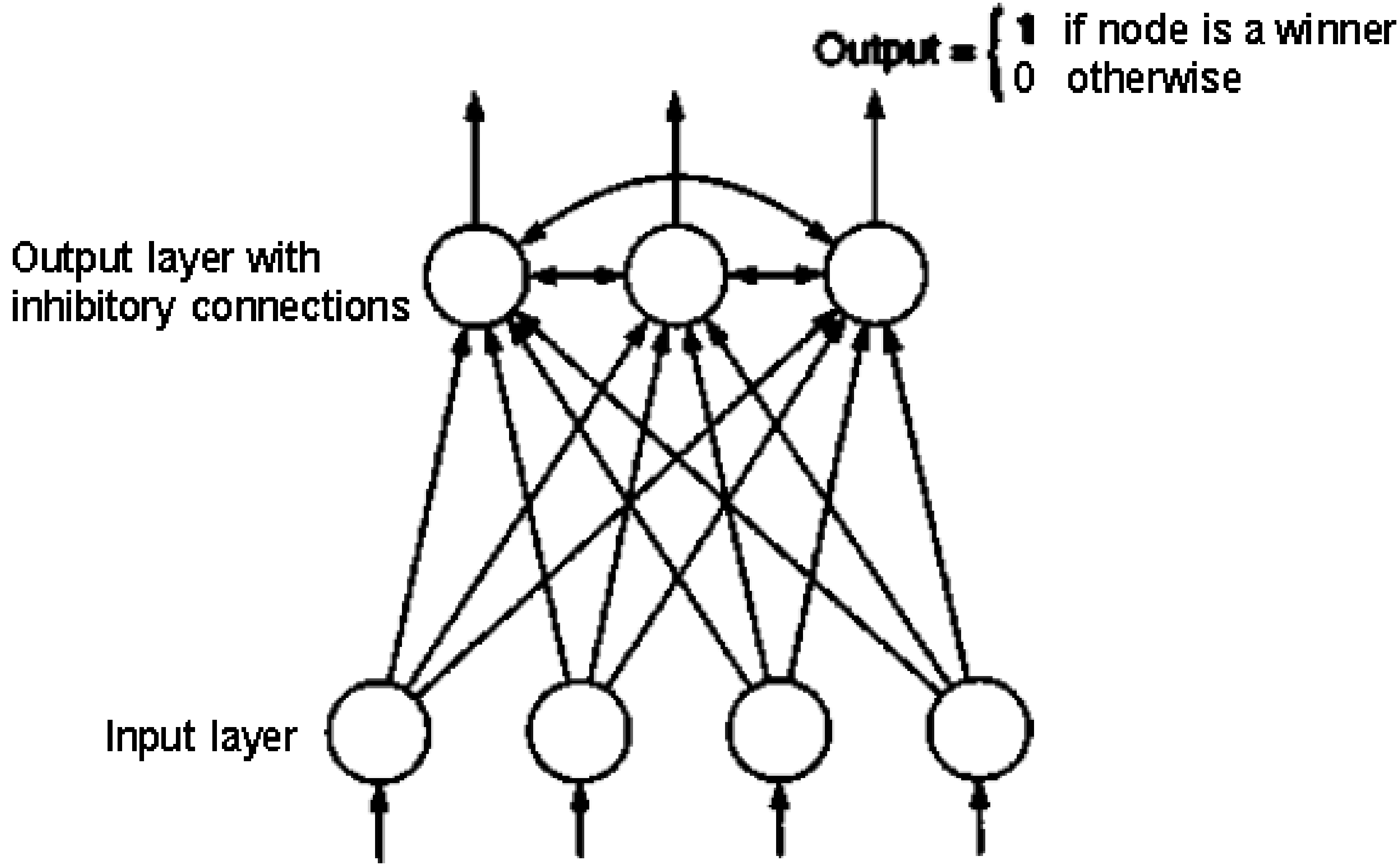
Hierarchical Methods

- Hierarchical Agglomerative Clustering

1. $G \leftarrow \{\{d\}|d \in S\}$ (initialize G with singleton clusters, each containing a document from S).
2. If $|G| \leq k$, then exit (stop if the desired number of clusters is reached).
3. Find $S_i, S_j \in G$ such that $(i, j) = \arg\max_{(i,j)} \operatorname{sim}(S_i, S_j)$ (find the two closest clusters).
4. If $\operatorname{sim}(S_i, S_j) < q$, exit (stop if the similarity of the closest clusters is less than q).
5. Remove S_i and S_j from G.
6. $G = G \cup \{S_i, S_j\}$ (merge S_i and S_j, and add the new cluster to the hierarchy).
7. Go to step 2.

Unsupervised Learning NN

- Simple Competitive Learning Neural Network



- w_{new} vector = w_{old} vector + η(i/p vector – w_{old} weight vector)

Unsupervised Learning

- Simple Competitive Learning Neural Network
  - Samples: (1.1 1.7 1.8), (0 0 0), (0 0.5 1.5), (1 0 0), (0.5 0.5 0.5), (1 1 1) (Three dimensional inputs)
  - Neurons: A, B, C (in output layer)
  - W_A:0.2 0.7 0.3, W_B:0.1 0.1 0.9, W_C:1 1 1 (Randomly)
  - $\eta = 0.5$

I	Winner	I	Winner
1	C : 1.05, 1.35, 1.4	7	C: 1.05, 1.45, 1.5
2	A: 0.1, 0.35, 0.15	8	A: 0.25, 0.2, 0.15
3	B: 0.05, 0.3, 1.2	9	B: 0, 0.4, 1.35
4	A: 0.55, 0.2, 0.1	10	A: 0.6, 0.1, 0.1
5	A: 0.5, 0.35, 0.3	11	A: 0.55, 0.3, 0.3
6	C: 1, 1.2, 1.2	12	C: 1, 1.2, 1.25

- w_{new} vector = w_{old} vector + η(i/p vector – w_{old} weight vector)

Unsupervised Learning NN

- Simple Competitive Learning Neural Network

- Observations
  - Node a becomes repeatedly activated by the samples i2, i4 and i5, node B by i3 alone and node C by i1 and i6. The centroid of i2, i4 and i5 is (0.5, 0.2, 0.2), and convergence of the weight vector for node A towards this location is indicated by the progression.

  - The high value of a learning rate is causing substantial modification of a node position with every sample presentation. This is one reason for unsmooth convergence.

  - The network is sensitive to the choice of the exact distance function.

Unsupervised Learning NN

- Simple Competitive Learning Neural Network
- Observations
  - The initial value of the node positions also plays some role in determining which node is activated by which samples.

  - The result of network computations also depends on the sequence in which samples are presented to the network, especially when the learning rate is not very small.

Unsupervised Learning NN

➢ Self Organizing Maps

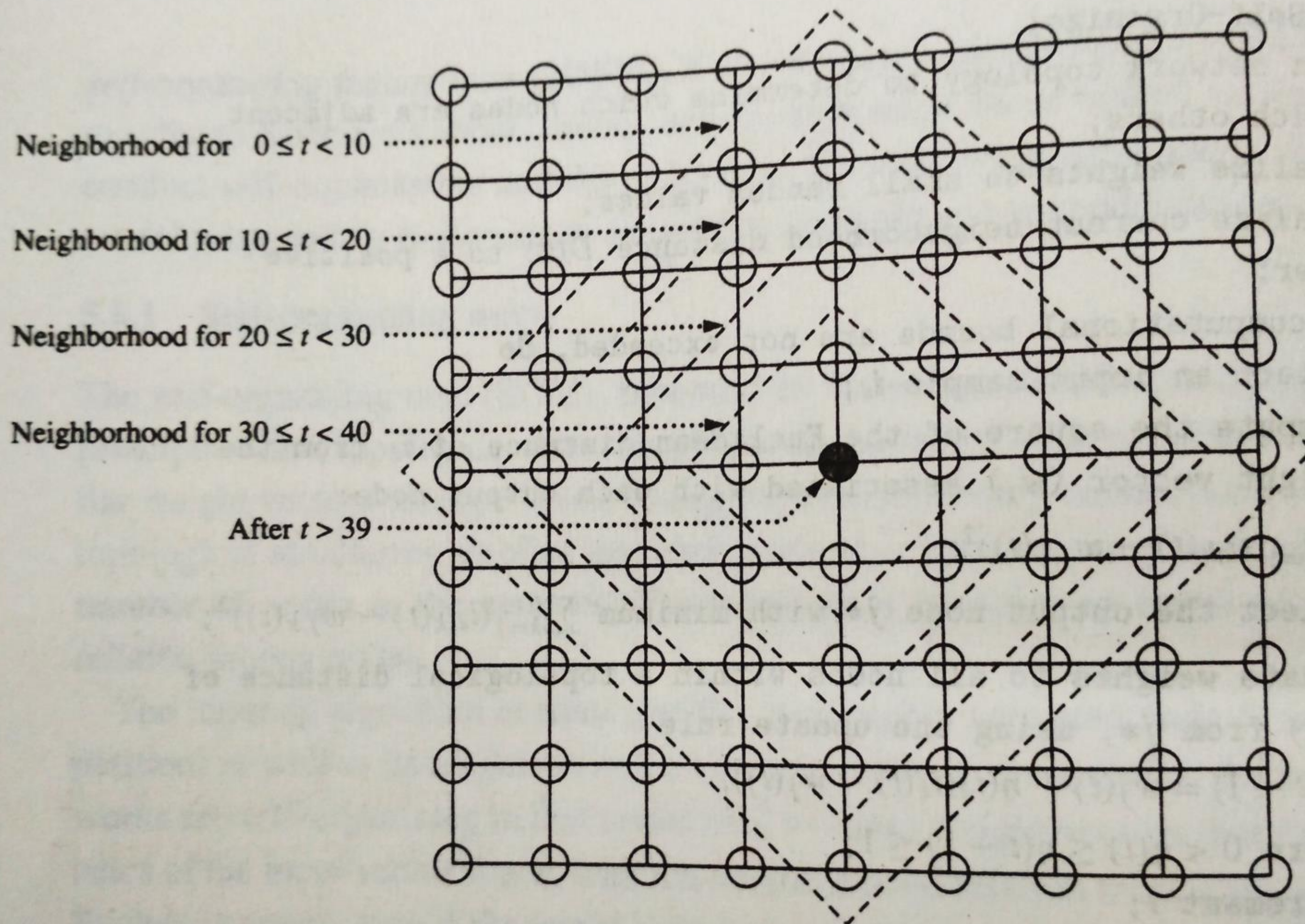
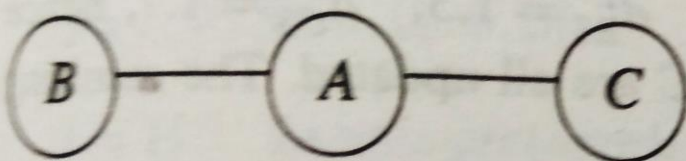


Figure 5.17
Time-varying neighborhoods of a node in an SOM network with grid topology: $D(t) = 4$ for $0 \le t < 10$, $D(t) = 3$ for $10 \le t < 20$, $D(t) = 2$ for $20 \le t < 30$, $D(t) = 1$ for $30 \le t < 40$, and $D(t) = 0$ for $t \ge 40$.

Unsupervised Learning NN

- Self Organizing Maps



Note that this topology is independent of the precise weight vectors chosen initially for the nodes. The training set $T = \{i_1 = (1.1, 1.7, 1.8),\ i_2 = (0, 0, 0), i_3 = (0, 0.5, 1.5),\ i_4 = (1, 0, 0),\ i_5 = (0.5, 0.5, 0.5),\ i_6 = (1, 1, 1)\}$. The initial weight vectors are given by

$$W(0) = \begin{array}{l} w_A: \\ w_B: \\ w_C: \end{array} \begin{pmatrix} 0.2 & 0.7 & 0.3 \\ 0.1 & 0.1 & 0.9 \\ 1 & 1 & 1 \end{pmatrix}.$$

Let $D(t) = 1$ for one initial round of training set presentations (until $t = 6$), and $D(t) = 0$ thereafter. As in the LVQ example, let $\eta(t) = 0.5$ until $t = 6$, then $\eta(t) = 0.25$ until $t = 12$, and $\eta(t) = 0.1$ thereafter.

Unsupervised Learning NN

- Self Organizing Maps

$t = 1$: Sample presented: $i_1 = (1.1, 1.7, 1.8)$.

Squared Euclidean distance between A and i_1: $d^2_{A1} = (1.1 - 0.2)^2 + (1.7 - 0.7)^2 + (1.8 - 0.3)^2 = 4.1$. Similarly, $d^2_{B1} = 4.4$ and $d^2_{C1} = 1.1$.

C is the "winner" since $d^2_{C1} < d^2_{A1}$ and $d^2_{C1} < d^2_{B1}$.

Since $D(t) = 1$ at this stage, the weights of both C and its neighbor A are updated according to equation 5.2. For example,

$$w_{A,1}(1) = w_{A,1}(0) + \eta(1) \cdot (x_{1,1} - w_{A,1}(0)) = 0.2 + (0.5)(1.1 - 0.2) = 0.65$$

The resulting weight matrix is

$$W(1) = \begin{pmatrix} w_A: & 0.65 & 1.2 & 1.05 \\ w_B: & 0.1 & 0.1 & 0.9 \\ w_C: & 1.05 & 1.35 & 1.4 \end{pmatrix}.$$

Note that the weights attached to B are not modified since B falls outside the neighborhood of the winner node C.

Unsupervised Learning NN

➢ Self Organizing Maps

$t = 2$: Sample presented: $i_2 = (0, 0, 0)$. $d^2_{A2} = 3$, $d^2_{B2} = 0.8$, $d^2_{C2} = 4.9$, hence B is the winner. The weights of both B and its neighbor A are updated. The resulting weight matrix is

$$W(2) = \begin{pmatrix} w_A: & 0.325 & 0.6 & 0.525 \\ w_B: & 0.05 & 0.05 & 0.45 \\ w_C: & 1.05 & 1.35 & 1.4 \end{pmatrix}.$$

$t = 3$: Sample presented: $i_3 = (0, 0.5, 1.5)$. $d^2_{A3} = 1.1$, $d^2_{B3} = 1.3$, $d^2_{C3} = 1.7$, hence A is the winner. The weights of A and its neighbors B, C are all updated. The resulting weight matrix is

$$W(3) = \begin{pmatrix} w_A: & 0.16 & 0.55 & 1.01 \\ w_B: & 0.025 & 0.275 & 0.975 \\ w_C: & 0.525 & 0.925 & 1.45 \end{pmatrix}.$$

Unsupervised Learning NN

- Self Organizing Maps

$t = 4$: Sample presented: $\boldsymbol{i}_4 = (1, 0, 0)$. $d_{A4}^2 = 2$, $d_{B4}^2 = 1.9$, $d_{C4}^2 = 3.2$, hence B is the winner; both B and A are updated.

$$W(4) = \begin{array}{l} \boldsymbol{w}_A: \\ \boldsymbol{w}_B: \\ \boldsymbol{w}_C: \end{array} \begin{pmatrix} 0.58 & 0.275 & 0.51 \\ 0.51 & 0.14 & 0.49 \\ 0.525 & 0.925 & 1.45 \end{pmatrix}.$$

$t = 5$: Sample presented: $\boldsymbol{i}_5$. A is the winner. All nodes are updated.

$$W(5) = \begin{array}{l} \boldsymbol{w}_A: \\ \boldsymbol{w}_B: \\ \boldsymbol{w}_C: \end{array} \begin{pmatrix} 0.54 & 0.39 & 0.50 \\ 0.51 & 0.32 & 0.49 \\ 0.51 & 0.71 & 0.975 \end{pmatrix}.$$

$t = 6$: Sample presented: $\boldsymbol{i}_6$. C is the winner; both $\boldsymbol{w}_C$ and $\boldsymbol{w}_A$ are updated.

$$W(6) = \begin{array}{l} \boldsymbol{w}_A: \\ \boldsymbol{w}_B: \\ \boldsymbol{w}_C: \end{array} \begin{pmatrix} 0.77 & 0.69 & 0.75 \\ 0.51 & 0.32 & 0.49 \\ 0.76 & 0.86 & 0.99 \end{pmatrix}.$$

Unsupervised Learning NN

- Self Organizing Maps

$t = 7$: η is now reduced to 0.25, and the neighborhood relation shrinks, so that only the winner node is updated henceforth. Sample presented: i_1. C is the winner; only w_C is updated.

w_C: (0.84 1.07 1.19).

$t = 8$: Sample presented: i_2. B is winner and w_B is updated.

w_B: (0.38 0.24 0.37).

$t = 9$: Sample presented: i_3. C is winner and w_C is updated.

w_C: (0.63 0.93 1.27).

$t = 10$: Sample presented: i_4. B is winner and w_B is updated.

w_B: (0.53 0.18 0.28).

$t = 11$: Sample presented: i_5. B is winner and w_B is updated.

w_B: (0.53 0.26 0.33).

$t = 12$: Sample presented: i_6. Weights of the winner node A are updated.

w_A: (0.83 0.77 0.81).

Unsupervised Learning NN

- Self Organizing Maps

$t = 13$: Now η is further reduced to 0.1.

Sample presented: i_1. Weights of the winner node C are updated.

w_C : (0.68 1.00 1.32).

$t = 14$: Sample presented: i_2. Weights of the winner node B are updated.

w_B : (0.47 0.23 0.30).

$t = 15$: Sample presented: i_3. Winner is C and w_C is updated. At this stage, the weight matrix is given by

$$W(15) = \begin{pmatrix} w_A: & 0.83 & 0.77 & 0.81 \\ w_B: & 0.47 & 0.23 & 0.30 \\ w_C: & 0.61 & 0.95 & 1.34 \end{pmatrix}.$$

Unsupervised Learning NN

- Self Organizing Maps

At the beginning of the training process, the Euclidean distances between various nodes were given by

$$|\mathbf{w}_A - \mathbf{w}_B| = 0.85,\ |\mathbf{w}_B - \mathbf{w}_C| = 1.28,\ |\mathbf{w}_A - \mathbf{w}_C| = 1.22.$$

The training process increases the relative distance between non-adjacent nodes (B, C), while the weight vector associated with A remains roughly in between B and C, with

$$|\mathbf{w}_A - \mathbf{w}_B| = 1.28,\ |\mathbf{w}_B - \mathbf{w}_C| = 1.75,\ |\mathbf{w}_A - \mathbf{w}_C| = 0.80.$$

The above example illustrates the extent to which nodes can move during the learning process, and how samples switch allegiance between nodes, especially in the early phases of computation. Computation continues in this manner until the network stabilizes, i.e., the same nodes continue to be the winners for the same input patterns, with the possible exception of input patterns equidistant from two weight vectors.

References

- Data mining: concepts and techniques, J. Han, and M. Kamber. Morgan Kaufmann, (2006)

- Elements of Artificial Neural Networks, Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka. MIT Press, (1997)

- Matlab Neural Network Tollbox Documentation

Disclaimer

- These slides are not original and have been prepared from various sources for teaching purpose.