Attendance Quiz Code MF22T7N7
Intelligent Data Analysis: Self-Organizing Maps (SOMs)

Martin Russell

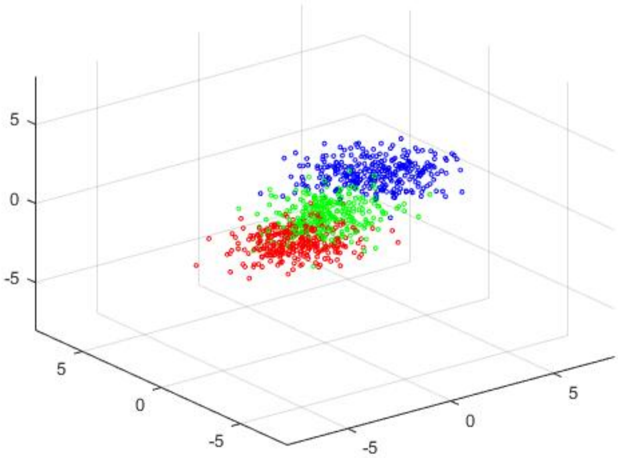
School of Computer Science, University of Birmingham

February 27, 2020

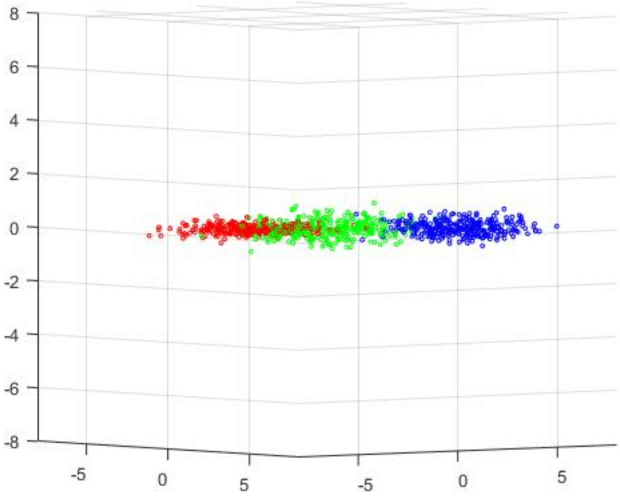
Overview

1. Motivation
   - Linear embedding and PCA
2. Clustering revisited
   - Alternative to k-means
   - Optimality
   - MatLab demonstration
3. Alternatives to k-means clustering
   - 'Online' clustering
4. Self-organizing maps / topographic maps
   - Neighbourhoods
   - Self-Organizing maps
   - 2 dimensional SOMs

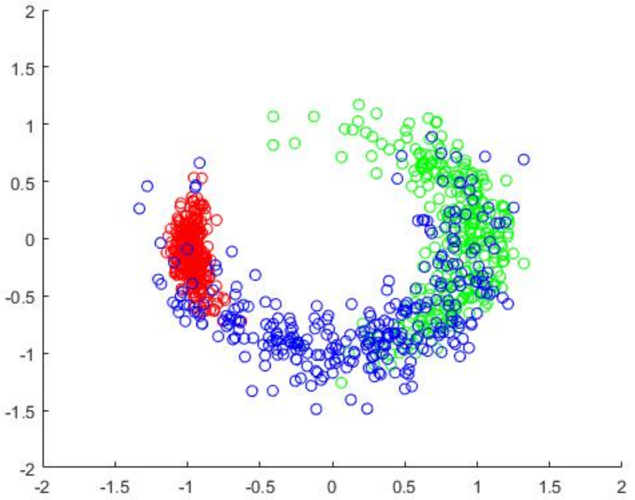
Linear embedding of low-dimensional object



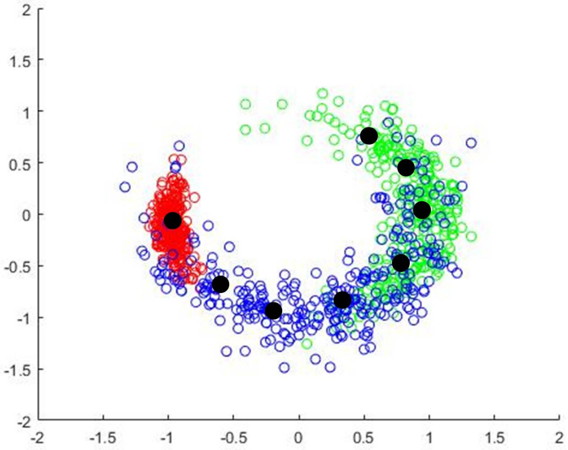
Linear embedding of low-dimensional object



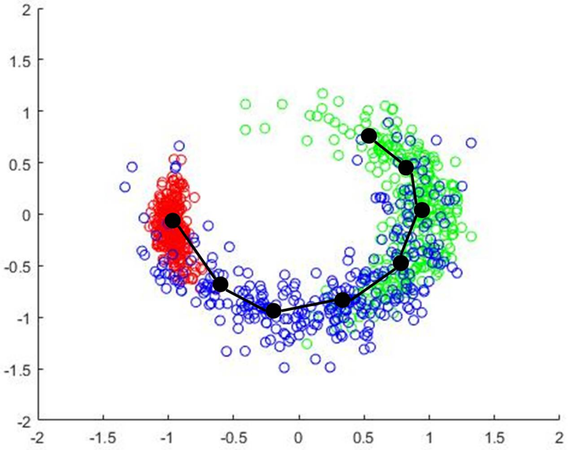
Non-linear embedding of low-dimensional object



Non-linear embedding of low-dimensional object



Non-linear embedding of low-dimensional object



The k-means clustering algorithm

k-means clustering is an **iterative** algorithm

1. Estimate initial centroid values $c_1^0, \cdots, c_K^0$
2. Set $i = 0$
3. For $n = 1, \cdots, N$ and $k = 1, \cdots K$ calculate $d(x_n, c_k^i)$
4. For $k = 1, \cdots, K$
5. Let $X^i(k)$ be the set of x_ns that are closest to c_k^i
6. Define c_k^{i+1} to be the average of the data points in $X^i(k)$

$$c_k^{(i+1)} = \frac{1}{|X^i(k)|} \sum_{x \in X^i(k)} x \tag{1}$$

7. $i = i + 1$. Go back to step 3.

Example

Let

$$x_1 = \begin{bmatrix} 0 \\ -5 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, x_3 = \begin{bmatrix} -2 \\ 3 \end{bmatrix}, x_4 = \begin{bmatrix} -4 \\ 7 \end{bmatrix}, x_5 = \begin{bmatrix} 3 \\ 1 \end{bmatrix},$$

$$x_6 = \begin{bmatrix} 4 \\ -2 \end{bmatrix}, x_7 = \begin{bmatrix} -1 \\ 6 \end{bmatrix}, x_8 = \begin{bmatrix} 5 \\ -6 \end{bmatrix}. \tag{2}$$

and suppose that the initial estimates of two centroids are

$$c_1^0 = \begin{bmatrix} -3 \\ 5 \end{bmatrix}, c_2^0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \tag{3}$$

Find the new values of c_1 and c_2 after one iteration of k-means clustering. Use the “city block” d_1 metric.

Example (continued)

The first step is to calculate the distances. For example

$$\begin{aligned} d_1(x_1, c_1^0) &= d_1(\begin{bmatrix} 0 \\ -5 \end{bmatrix}, \begin{bmatrix} -3 \\ 5 \end{bmatrix}) \\ &= |0-(-3)| + |-5-5| \\ &= 3+10 = 13 \end{aligned} \tag{4}$$

Continue in this way to obtain the matrix of distances between data points and centroids

Example (continued)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
c_1^0	13	7	3	3	10	14	3	19
c_2^0	9	1	5	11	2	6	7	11
c_1^0			1	1			1	
c_2^0	1	1			1	1		1

Table 1: Distances between centroids and data points (rows 2,3) and indicator of closest centroid to each data point (rows 4,5)

- So $X^0(1) = \{x_3, x_4, x_7\}$ and $X^0(2) = \{x_1, x_2, x_5, x_6, x_8\}$, and

$$c_1^1 = \frac{1}{3}(x_3 + x_4 + x_7) = \begin{bmatrix} -2.33 \\ 5.33 \end{bmatrix} \quad (5)$$

$$c_2^1 = \frac{1}{5}(x_1 + x_2 + x_5 + x_6 + x_8) = \begin{bmatrix} 2.6 \\ -2 \end{bmatrix} \quad (6)$$

Optimality

- Is the set of k centroids $\hat{C}$ created by k-means globally optimal? In other words is it true that for any set of k centroids

$$D(C, X) \geq D(\hat{C}, X)? \tag{7}$$

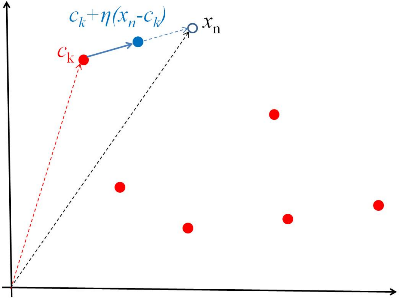
- No, k-means clustering is only guaranteed to find a *local* optimum.
- The solution obtained from k-means clustering depends on the *initial* centroids.

MatLab demonstration

- "Toy" 2-dimensional data set
- $K = 6$ (6 centroids)
- Initial centroids chosen at random in the "box" $-10 \leq x, y \leq 10$
- 20 iterations of k-means clustering
- Repeated 20 times

Alternative to k-means clustering

- For k-means, calculate distances between **all** data points and **all** centroids before centroids are updated
- But centroid locations could be improved after seeing **just one** data point x_n



Alternative to k-means clustering

- **'online' clustering** - update centroids with each sample

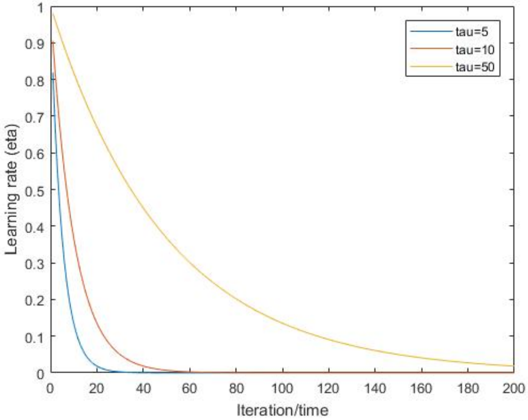
$$c_k^{new} = c_k^{old} + \eta(x_n - c_k^{old}) \tag{8}$$

- $\eta > 0$ is the **learning rate**
  - If η is too small convergence will be too slow
  - If η is too big, algorithm will be unstable
- Start with big η then shrink η as time (number of iterations) increases

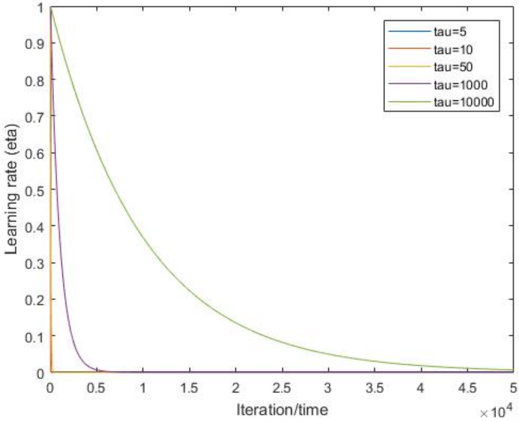
$$\eta(t) = \eta(0) \times e^{\frac{-t}{\tau}} \tag{9}$$

- $\tau > 0$ is the **time scale**. Determines how fast η will decrease

Learning rate

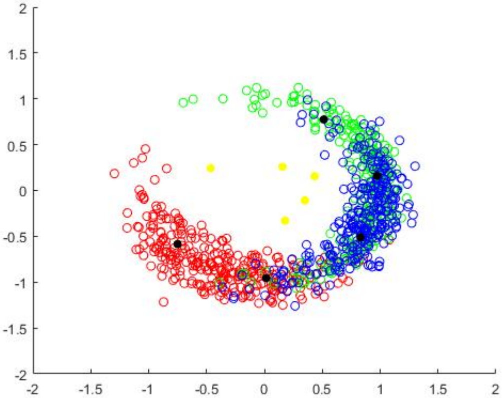


Learning rate - revised



Result of 'online' clustering

- Time-constant $\tau = 10000$



Online clustering algorithm - summary

1. Choose the number of centroids K
2. (Randomly) choose initial codebook $\{c_1, \cdots, c_K\}$
3. Cycle through the data points and for each data point x_n do:
    1. Find the closest centroid $c_{i(n)}$
    2. Move $c_{i(n)}$ closer to x_n:

       $$c_{i(n)}^{new} = c_{i(n)}^{old} + \eta(t)(x_n - c_{i(n)}^{old}) \quad (10)$$

       where $\eta(t) > 0$ is a small **learning rate** which reduces with time

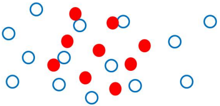
       $$\eta(t) = \eta(0) \times e^{\frac{-t}{\tau}} \quad (11)$$

    3. $\tau > 0$ is the **timescale**

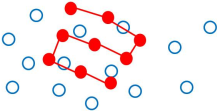
Enhancements to online clustering

- **Batch training** accumulates changes to centroids over (small) subsets of the training set
- **Stochastic** batch training accumulates changes to centroids over (small) randomly-chosen subsets of the training set
- Compare with **gradient descent** and **stochastic gradient descent**, for example in Neural Network training

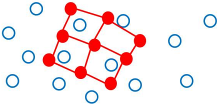
Imposing neighbourhood structure on the centroid set



Conventional centroid set

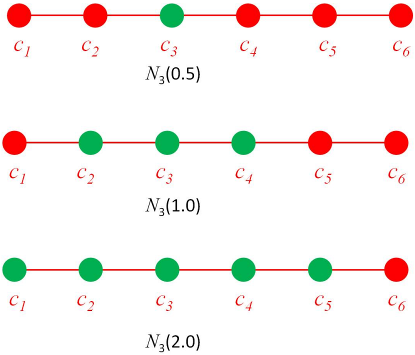


1-dimensional topographic map



2-dimensional topographic map

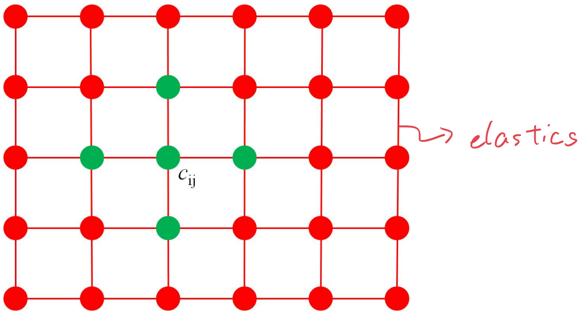
Neighbourhood structure - 1 dimension



preserve
relationships
between indices

$$N_j(d) = \{c_k \mid |k-j| \leq d\} \tag{12}$$

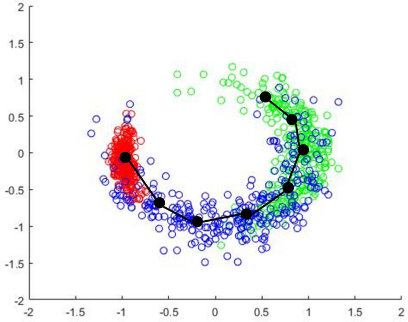
Neighbourhood structure - 2 dimensions



$$N_{ij}(d) = \{c_{kl} \,\Big|\, \left\|\begin{bmatrix} k \\ l \end{bmatrix} - \begin{bmatrix} i \\ j \end{bmatrix}\right\| \leq d\} \tag{13}$$

Constrained clustering - topographic maps

- Discover hidden 1-dimensional structure of high-dimensional data by clustering, but constrian centroids $\{c_1, \cdots, c_K\}$ to lie on a one-dimensional 'elastic'



Online vs SOM / topograhic map (constrained clustering)

- Online clustering:

vector $i(n) \rightarrow x_n$

$$c_{i(n)}^{new} = c_{i(n)}^{old} + \eta(t)(x_n - c_{i(n)}^{old}) \tag{14}$$

- **Constrained** clustering - **topographic map** - **Self-Organizing Map (SOM)**: For **every** centroid c_k

$$c_k^{new} = c_k^{old} + h(i(n), k) \times \eta(t) \times (x_n - c_k^{old}) \tag{15}$$

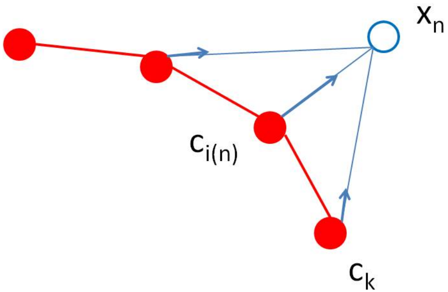
neighbourhood function → amount of movement

- $h(i(n), k)$ indicates how close the k^{th} centroid is to the centroid $c_{i(n)}$ closest to x_n.

index

close ? distant ↕

1 dimensional topographical map

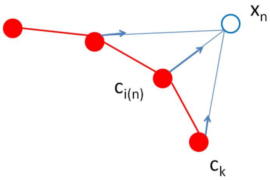


Constrained clustering (continued)

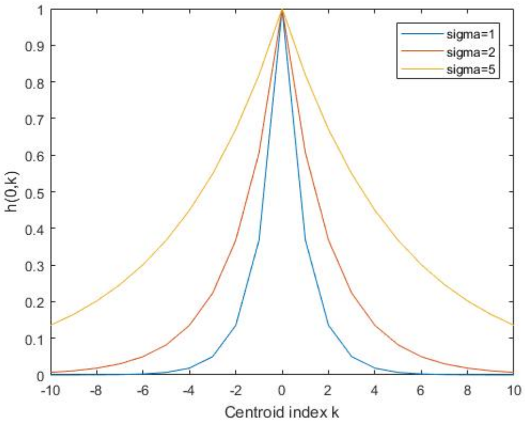
- Want:
  - $h(i(n), i(n)) = 1$
  - $h(i(n), k)$ decreases as c_k becomes further away from $c_{i(n)}$
- For example, choose:

$$h(i(n), k) = e^{\frac{-||i(n)-k||}{\sigma}} \tag{16}$$

- σ is the **neighbourhood width** (strength of the elastic)



Neighbourhood width (sigma)



$6\uparrow$ wider

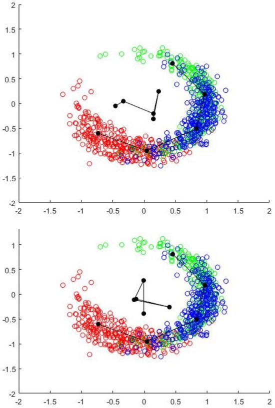
Neighbourhood width

- Initially choose a large value of σ to allow broad cooperation between centroids
- As algorithm proceeds, reduce the value of σ for fine tuning of topographic structure of codebook vectors
- For example, by analogy with the learning rate:

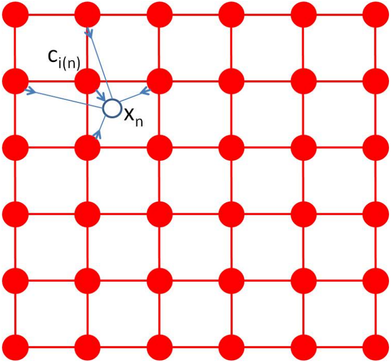
$$\sigma(t) = \sigma(0) \times e^{\frac{-t}{\nu}} \tag{17}$$

  where $\nu > 0$ is the **timescale**
- $\sigma(0)$ is the **initial neighbourhood width**

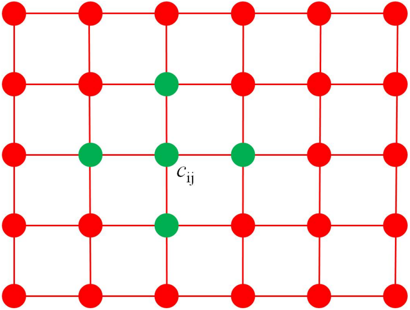
SOM results



2-dimensional SOM



Neighbourhood structure - 2 dimensions



$$N_{ij}(d) = \{c_{kl} \left| \, \left\| \begin{bmatrix} k \\ l \end{bmatrix} - \begin{bmatrix} i \\ j \end{bmatrix} \right\| \leq d \right\} \quad (18)$$

Summary

- Revision of k-means clustering
- The ‘curse of dimensionality’ and Vector Quantization (VQ)
- Alternative to k-means - ‘online clustering’ - the role of learning rate
- Self-organizing Maps (SOMs) = topographic maps - neighbourhood structures
- 1 and 2 dimensional SOMs