Unsupervised Learning and Clustering

Owen Roberts, Zach Busser,
Ganesh Sugunan

Hierarchical Clustering

Instead of flat clusters, we create a hierarchy of clusters.

Important variables:

- $c =$ number of clusters
- $n =$ number of samples
- At level k when $c = n - k + 1$
  - Level 1: n clusters (each sample is its own cluster)
  - Level n: one cluster (everything together)

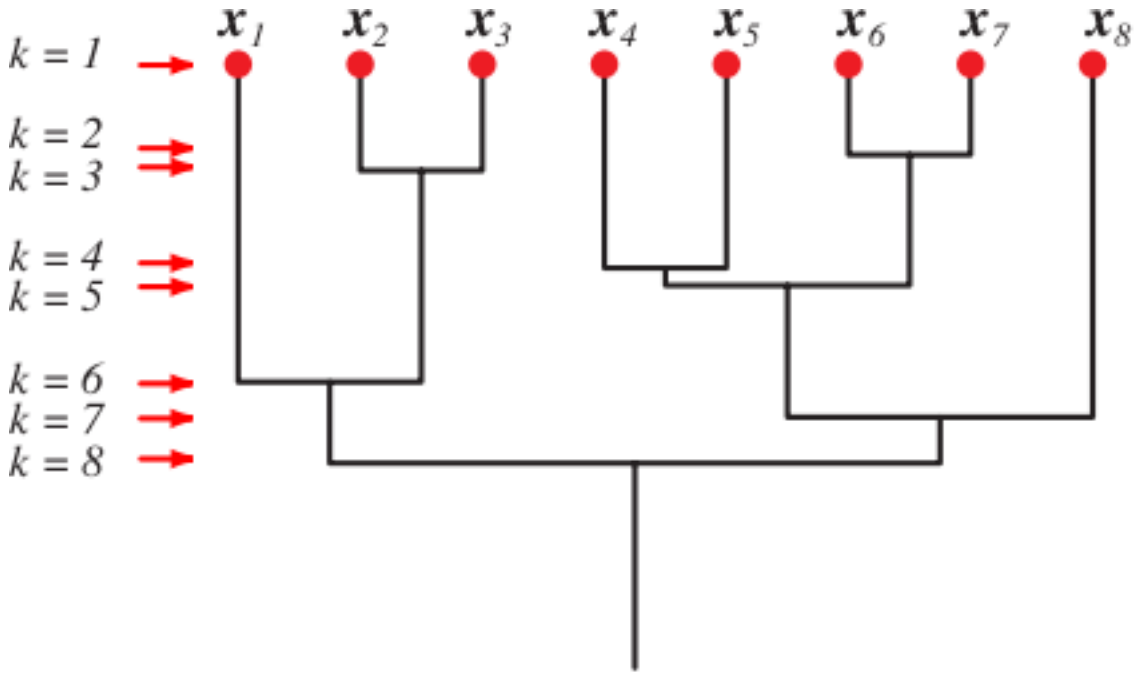
Hierarchical Clustering

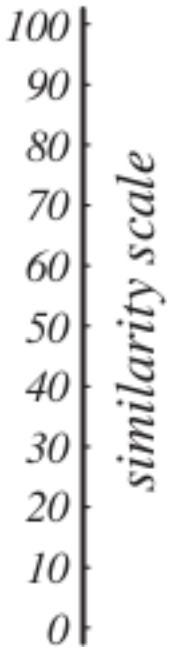
For the hierarchy business to hold, we need:

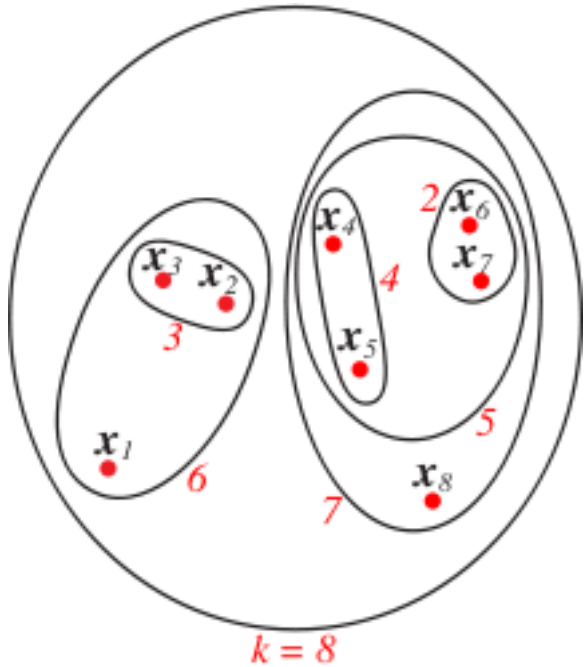
- $\forall \mathbf{x}_i, \mathbf{x}_j$, there is some minimum level k at which $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same cluster.

- For all $l > k$, $\mathbf{x}_i$ and $\mathbf{x}_j$ are still in the same cluster

Hierarchical Representations







$$\{\{x_1, \{x_2, x_3\}\}, \{\{\{x_4, x_5\}, \{x_6, x_7\}\}, x_8\}\}$$

Hierarchical Algorithms

There's two major ways to do hierarchical clustering:

- Agglomerative - bottom up, clump clusters together
- Divisive - top down, split clusters apart

We're going to restrict our attention to agglomerative procedures, as they are usually less computationally complex.

Agglomeration

AGGLOMERATE (c, $\mathcal{D}_i = \{\mathbf{x}_i\}$ for $i = 1..n$)

let $\hat{c} \leftarrow n$

do $\hat{c} \leftarrow \hat{c} - 1$

find nearest clusters $\mathcal{D}_i$ and $\mathcal{D}_j$

merge $\mathcal{D}_i$ and $\mathcal{D}_j$

until $c = \hat{c}$

return c clusters

"Nearest" Clusters

Different ways to find the nearest clusters:

- $d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' \in \mathcal{D}_j} ||\mathbf{x} - \mathbf{x}'||$
- $d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' \in \mathcal{D}_j} ||\mathbf{x} - \mathbf{x}'||$
- $d_{ave}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_j} ||\mathbf{x} - \mathbf{x}'||$
- $d_{mean}(\mathcal{D}_i, \mathcal{D}_j) = ||\mathbf{m}_i - \mathbf{m}_j||$

As for $\mathbf{m}_i$, remember: $\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$ (Section 10.7)

Complexity

Consider trying to agglomerate c clusters with n pattern in d dimensional space using, say, d_{min}.

- Calculate $n(n-1)$ distances: $O(n^2)$
- Each such calculation is an $O(d)$ operation: $O(n^2 d)$
- Do this c times: $O(cn^2 d)$, where typically $n \gg c$

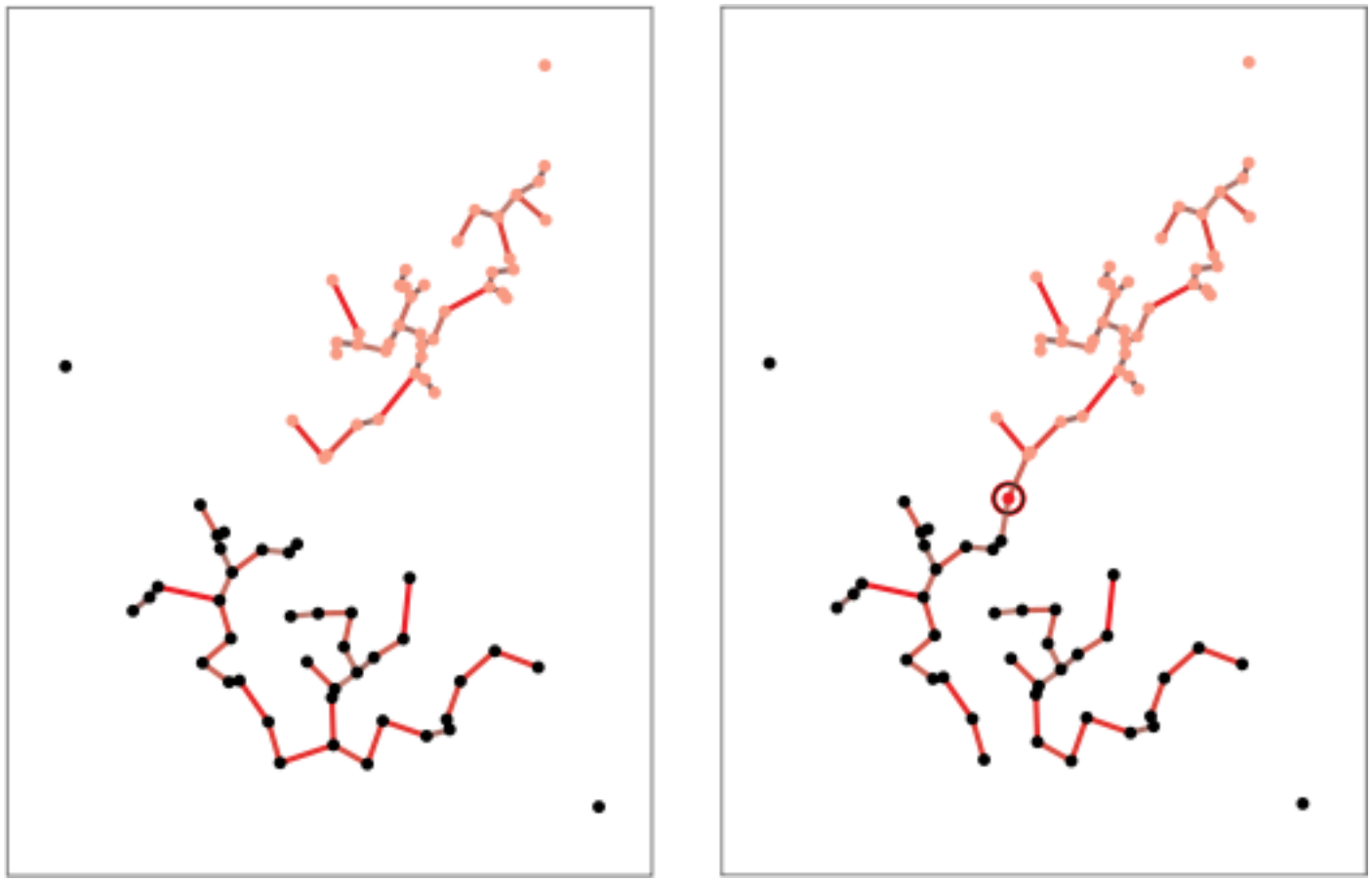
Nearest-Neighbor/Single-Linkage

Nearest-Neighbor: d_{min} is used to find nearest clusters

Single-Linkage: terminate when the smallest d_{min} exceeds some threshold. Think graph theory:

- Each $\mathbf{x}$ is a vertex of the graph
- Edges are a path along the vertices of a cluster
- Merging two clusters adds an edge between the two closest points
- Generates a tree, or a (minimum) spanning tree if we take $c = 1$

Nearest-Neighbor/Single-Linkage



Farthest-Neighbor/Complete-Linkage

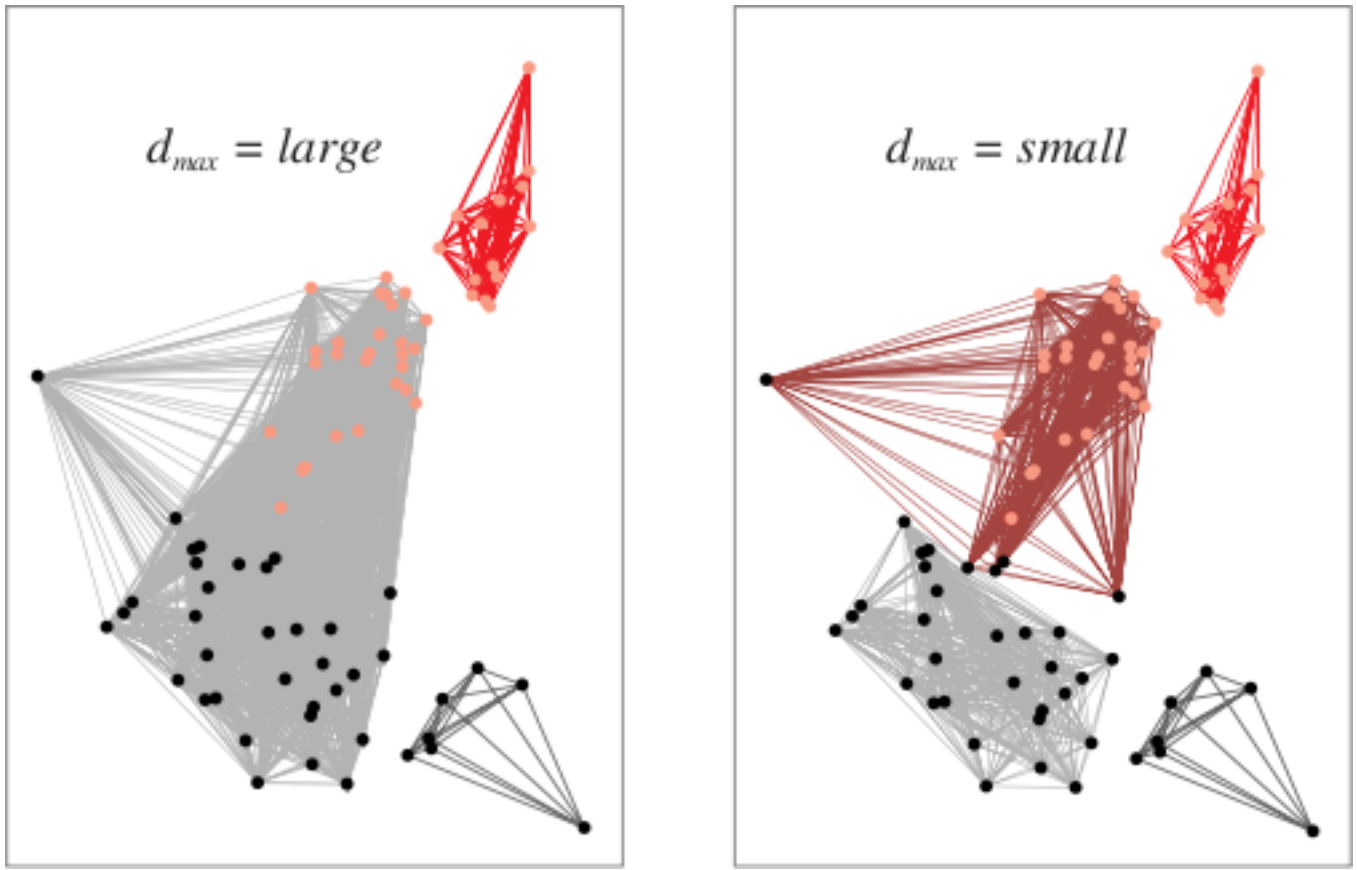
Farthest-Neighbor: d_{max} is used to find nearest clusters

Complete-Linkage: terminate when the smallest d_{max} exceeds some threshold. Again, graph theory:

- All vertices in the same cluster are connected (cluster $\mathcal{D}_i$ forms the subgraph K_{n_i})

- Merging two clusters adds an edge between every vertex in the new cluster

Farthest-Neighbor/Complete-Linkage



Compromises and Criterion Functions

d_{min} and d_{max} are both vulnerable to outliers

Enter averaging: d_{ave} and d_{mean}

d_{mean} is computationally the simplest, but may be hard to define

If we seek to minimize a criterion function at every merger, we have a step-wise optimal clustering.

- Example: d_{max} minimizes the change in diameter of the clusters

Stepwise Optimal Clustering

STEPWISE-OPTIMAL $(c,\ \mathcal{D}_i = \{\mathbf{x}_i\}$ for $i = 1..n)$

let $\widehat{c} \leftarrow n$

do $\widehat{c} \leftarrow \widehat{c} - 1$

find clusters $\mathcal{D}_i$ and $\mathcal{D}_j$ whose merger changes the criterion the least

merge $\mathcal{D}_i$ and $\mathcal{D}_j$

until $c = \widehat{c}$

return c clusters

Criterion Functions

Remember $J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} ||\mathbf{x} - \mathbf{m}_i||^2$?

This is another possible criterion function.

The pair of clusters that minimizes the increase in J_e is:

$$d_e(\mathcal{D}_i, \mathcal{D}_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} ||\mathbf{m}_i - \mathbf{m}_j||$$

Dissimilarity

If we can't come up with a metric, but we can measure dissimilarity as $\delta(\mathbf{x}, \mathbf{x}')$, then we can still use agglomerative clustering so long as δ satisfies a few properties.

If we define dissimilarity between clusters as either:

- $\delta_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' \in \mathcal{D}_j} \delta(\mathbf{x}, \mathbf{x}')$
- $\delta_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' \in \mathcal{D}_j} \delta(\mathbf{x}, \mathbf{x}')$

Then we induce a distance function for our n samples, which preserves ranking across monotonic transformations (think ln).

Generalized Distance Functions

The induced distance function is $d(\mathbf{x}, \mathbf{x}')$ is the lowest level such that $\mathbf{x}$ and $\mathbf{x}'$ are in the same cluster.

There are some conditions we need to satisfy for a generalized distance function $d(\mathbf{x}, \mathbf{x}')$, such as our d function, to hold. $\forall \mathbf{x}, \mathbf{x}', \mathbf{x}''$:

- Nonnegativity: $d(\mathbf{x}, \mathbf{x}') \geq 0$
- Reflexivity: $d(\mathbf{x}, \mathbf{x}') = 0$ iff $\mathbf{x} = \mathbf{x}'$
- Symmetry: $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$
- Triangle Inequality: $d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{x}'') \geq d(\mathbf{x}, \mathbf{x}'')$

Ultrametric

Furthermore, if, like our dissimilarity metric, we satisfy:

$d(\mathbf{x}, \mathbf{x}'') \leq \max(d(\mathbf{x}, \mathbf{x}'), d(\mathbf{x}', \mathbf{x}'')) \ \forall \mathbf{x}'$

Then d is an ultrametric, which is more resistant to local minima problems.

The intuition behind this is that our distances are more strictly ordered.

As we have seen so far, one of the ways to deal with an unknown value is to substitute values for the unknown value and watch how the function behaves on the data set. The function we choose to use is the sum of squared error function, denoted by J_e from here on out. If we assume there are a theoretical number of good clusters, $\hat{c}$, on the set D, then J_e would drop quickly as you increase the number of clusters, c, until $c = \hat{c}$; at that point, the error would still decrease as you gradually seperate each point into it's own cluster, until $J_e = 0$ when $c = n$.

Another way to deal with an unknown value, rather than to watch how a prexisting function behaves, is to generate a fitness function yourself, and apply it to the set. We cannot use the normal functions in this setting, because the feature space has such a high dimensionality that any complex function is too computationally demanding for serious use, so instead we will use a simple criterion function denoted $J(c)$. Our goal is to figure out at what level the reduction in improvement from $J(c)$ to $J(c+1)$ becomes sufficient to decide $c = \hat{c}$. We will decide this by going back to the null hypothesis that there are c well defined clusters present, and calculating the difference between $J(c)$ and $J(c+1)$. Please keep in mind that the whole process from here on out is extremely weakly founded, and that all results may be false.

To begin with, assume that n has a normal population with average μ and variance $\sigma^2 I$; which means that there is only one good cluster. With only one cluster, any improvement in $J(c+1)$ should be considered insignificant. For example when m is the mean of D:

$$J_e(1) = \sum_{x \in D} ||x - m||^2$$

Resulting in a distribution normal to average $nd\sigma^2$, and variance $2nd\sigma^4$. Now, we will subpartition it into two partitions to get the lower $J_e(2)$:

$$J_e(2) = \sum_{i=1}^{2} \sum_{x \in D_i} ||x - m_i||^2$$

Because we assume that $\hat{c} = 1$, we know that while $J_e(2) < J_e(1)$, $J_e(1)$ is the best fit. Without knowing the distribution of $J_e(2)$,it is impossible to determine what exactly the smalles difference between $J_e(2)$ and $J_e(1)$ for meaningful partitioning would be, but we can guess by looking at the bad partition. In our example $J_e(2)$ is normal with the average $n(\frac{d-2}{\pi})\sigma^2$ and variance $2n(\frac{d-8}{\pi^2})\sigma^4$. This is a reduction as you can see by comparing the averages of the two sum of squared error functions, so we should consider this reduction a bottom limit for a significant reduction as a result of increasing the number of partitions.

In order to obtain an optimal partition, we will have to make a few assumptions. We will assume the suboptimal partition from earlier is nearly optimal, that the sample is distributed normally (which we established earlier), and that the best variance is:

$$\hat{\sigma}^2 = \frac{1}{nd} J_e(1) = \frac{1}{nd} \sum_{x \in D} ||x - m||^2$$

our final result for the minimum diffrence in order to be considered significant comes out to:

$$\frac{J_e(2)}{J_e(1)} < 1 - \frac{2}{\pi d} - \alpha\sqrt{\frac{2(\frac{1-8}{\pi^2 d})}{nd}}$$

We determine α from the error function($erf(.)$) at percent significance p(the chance that we had a bad sample) using the following:

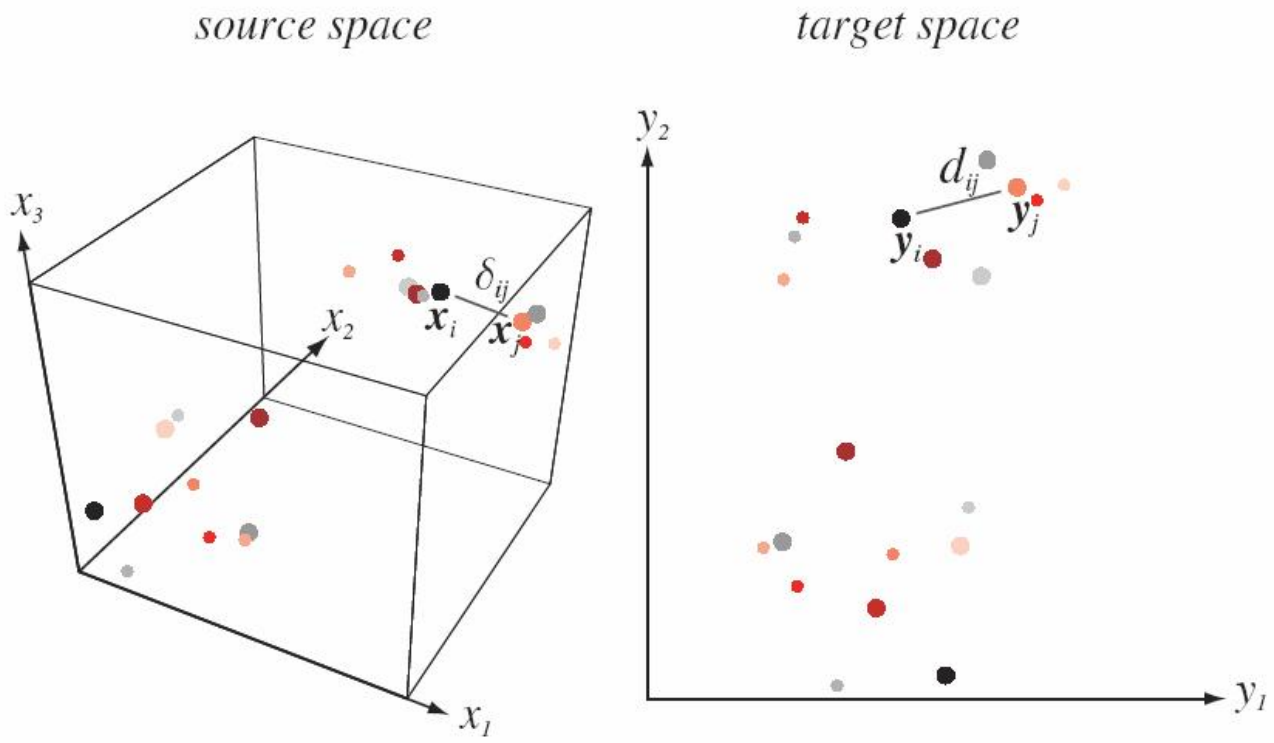
$$p = 100\int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}} du = 50(1 - erf(\frac{\alpha}{\sqrt{2}}))$$

Multidimensional Scaling

There is an inability to visualize the structure of multidimensional data.

It is made worse when similarity and dissimilarity measures lack familiar notion of distance.

The Solution is to represent higher dimensional possibly nonmetric similarities in a lower dimensional space with normal distances.


source space
target space
x_3
x_2
x_1
δ_{ij}
$\mathbf{x}_i$
$\mathbf{x}_j$
y_2
y_1
d_{ij}
$\mathbf{y}_i$
$\mathbf{y}_j$

A Simple Case

let $x_1, ..., x_n$ be the samples

let $y_1, ..., y_n$ be the lower dimensional images

let δ_{ij} be the distance between x_i and x_j

let d_{ij} be the distance between y_i and y_j

Note that is very rare that $\delta_{ij} = d_{ij}$ for all i and j

Deciding on Criterion Function

We use one of these sum of squared error functions

$$J_{ee} = \frac{\sum_{i<j}(d_{ij}-\delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

$$J_{ff} = \sum_{i<j}(\frac{d_{ij}-\delta_{ij}}{\delta_{ij}})^2$$

$$J_{ef} = \frac{1}{\sum_{i<j}\delta_{ij}}\sum\frac{(d_{ij}-\delta ij)^2}{\delta_{ij}}$$

Sum of Squared Errors

J_{ee} emphasizes large errors

J_{ff} emphasizes large fractional errors

J_{ef} is a good compromise and emphasizes large products of error and fractional error

Optimal Configuration

An optimal configuration is one that minimizes the criterion function.

It can be sought by a gradient descent method.

$$\nabla_{y_k} J_{ee} = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{ij}) \frac{y_k - y_j}{d_{kj}}$$

$$\nabla_{y_k} J_{ff} = 2 \sum_{j \neq k} \frac{d_{kj} - \delta_{ij}}{\delta_{kj}^2} \frac{y_k - y_j}{d_{kj}}$$

$$\nabla_{y_k} J_{ef} = \frac{2}{\sum_{i<j} \delta_{ij}} \sum_{i \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}} \frac{y_k - y_j}{d_{kj}}$$

The starting configuration function can be choosen randomly or in a convient way.

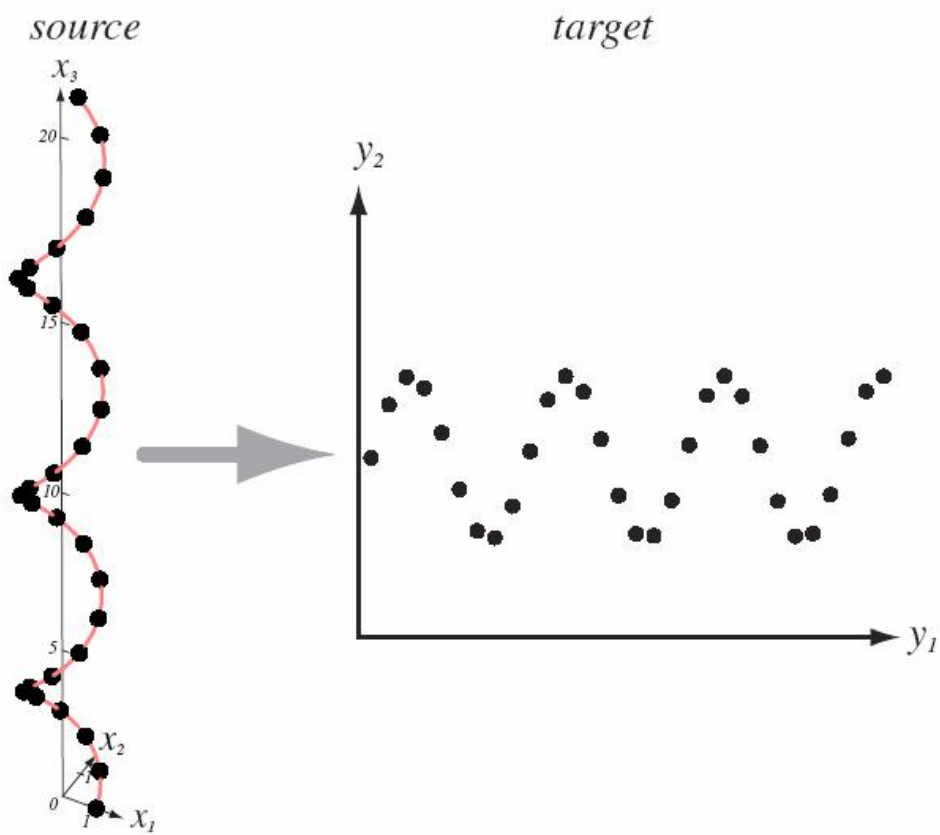

source
target
x_3
20
15
10
5
0
x_2
1
1
x_1
y_2
y_1

Fig 10.27

30 points in a spiral in $\mathbb{R}^3$

J_{ef} criterion function is used

20 iterations of the gradient descent

Monotonicity Constraint

In nonmetric multidimensional scaling problems δ_{ij} are dissimilarities whos numerical values are not as important as their rank order. Ideally, d_{ij} will be the same as δ_{ij} but that is not usually the case.

If you order the dissimilarities

$$\delta_{i_1 j_1} \leq \cdots \leq \delta_{i_m j_m}$$

Then for any m members satisfying the below equation they make the Monotonicity Constraint.

$$\hat{d}_{i_1 j_1} \leq \hat{d}_{i_2 j_2} \leq \cdots \leq \hat{d}_{i_m j_m}$$

$\hat{J}_{mon}$

The degree by which d_{ij} satisfy the Monotonicty Constraint is

$\hat{J}_{mon} = min_{\hat{d}_{ij}} \sum_{i<j} (d_{ij} - \hat{d}_{ij})^2$

This measures the degree that $y_1, ..., y_n$ represent the original dissimilarities.

$\hat{J}_{mon}$ **cont.**

But, this is not optimal. You could have $\hat{J}_{mon}$ vanish at a point.

To side step this we normalize it.

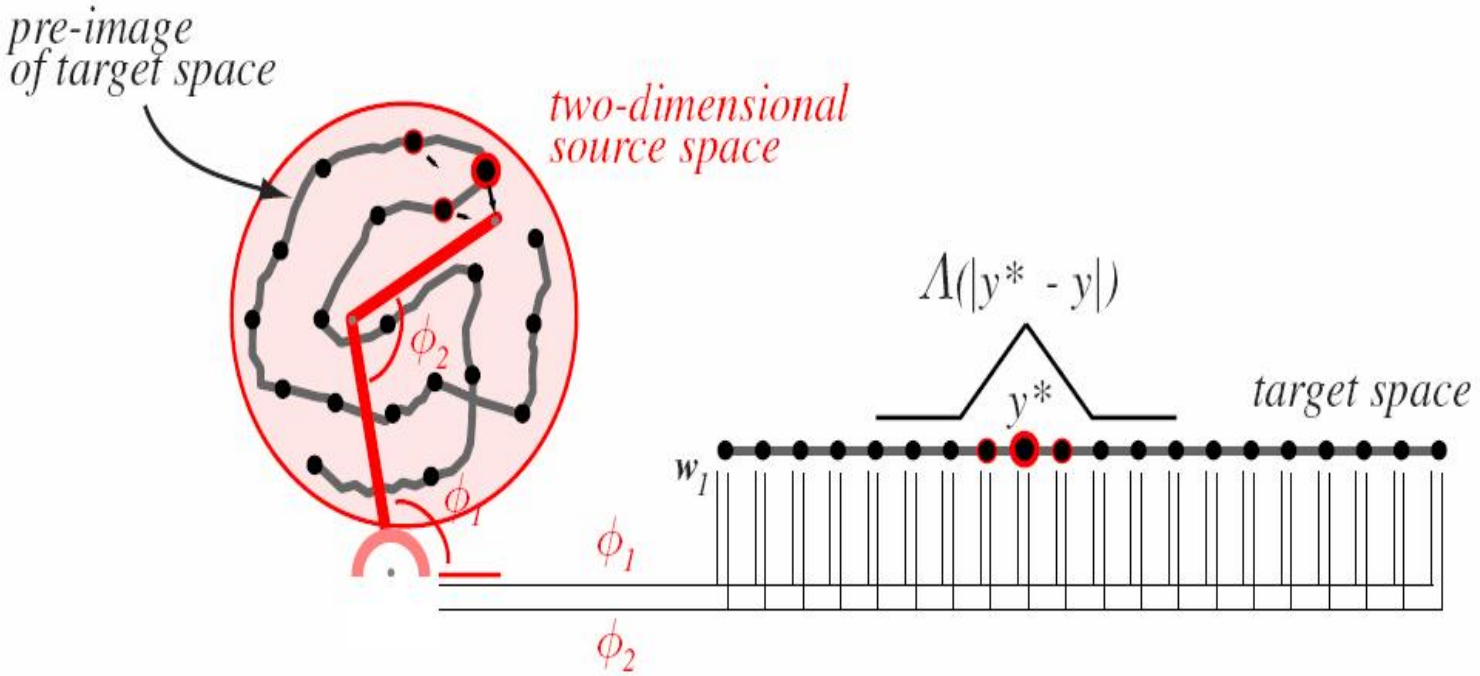
$$J_{mon} = \frac{\hat{J}_{mon}}{\sum_{i<j} d_{ij}^2}$$

Self-Organizing Feature Maps

a.k.a. Kohonen self-organizing feature maps.

They are useful when there is a non-linear mapping in the problem.

Example first!


pre-image
of target space
two-dimensional
source space
ϕ_2
ϕ_1
$\Lambda(|y^* - y|)$
y^*
target space
w_1
ϕ_1
ϕ_2

Self-Organizing Feature Maps cont.

We choose to sense the source space by a two joint arm. Every pair of points (x_1, x_2) in the space has a corresponding pair of angles $(\phi_1, \phi_2) = \phi$.

We use a sequence of ϕ values. We map from ϕ to y where points neighboring in the source space are neighboring in the target space.

Self-Organizing Feature Maps

A two layer neural network is used. ϕ_1, ϕ_2 are the inputs. The outputs are the points on the target line.

When ϕ is presented, then each node in the target space computes the net activation

$$net_k = \phi^t w_k$$

The unit that is most activated y^* and its weights and its neighbors weights are computed by

$$w_{ki}(t+1) = w_{wi}(t) + \eta(t)\Lambda(|y - y^*|)(\phi_i - w_{ki}(t))$$

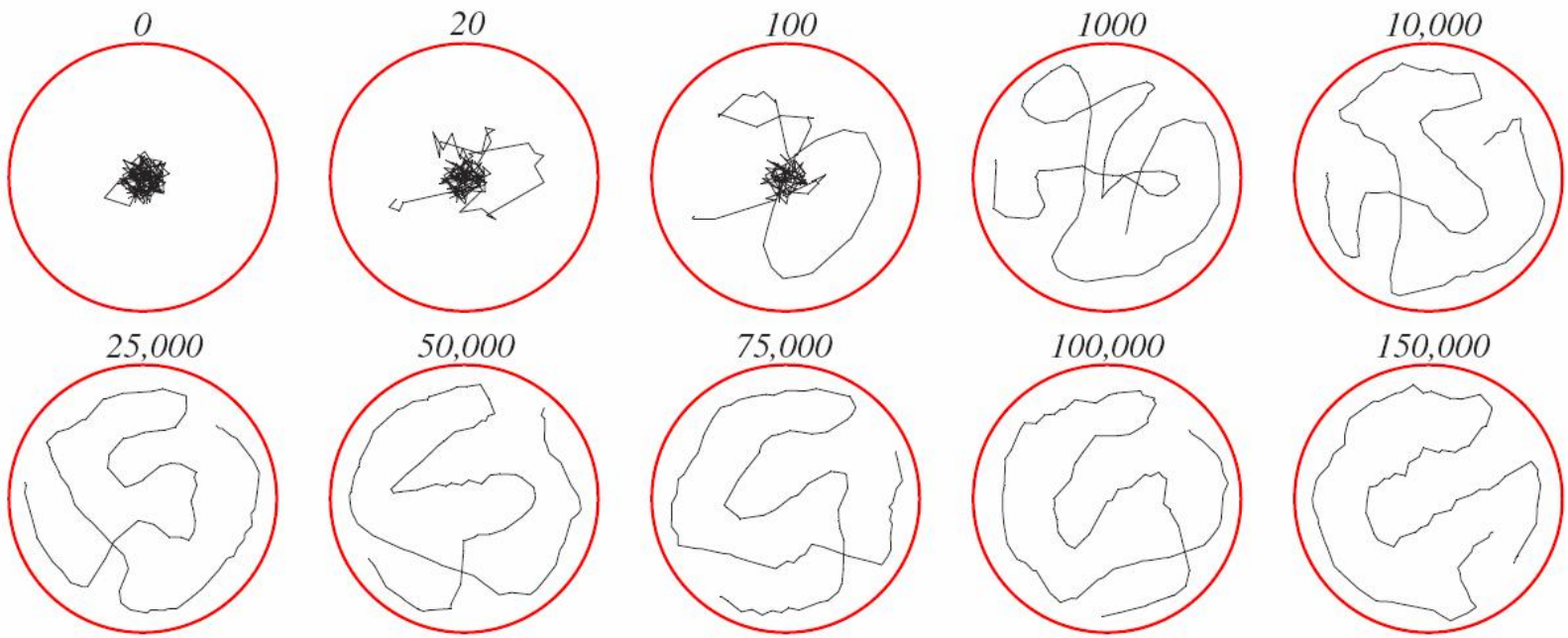
Self-Organizing Feature Maps

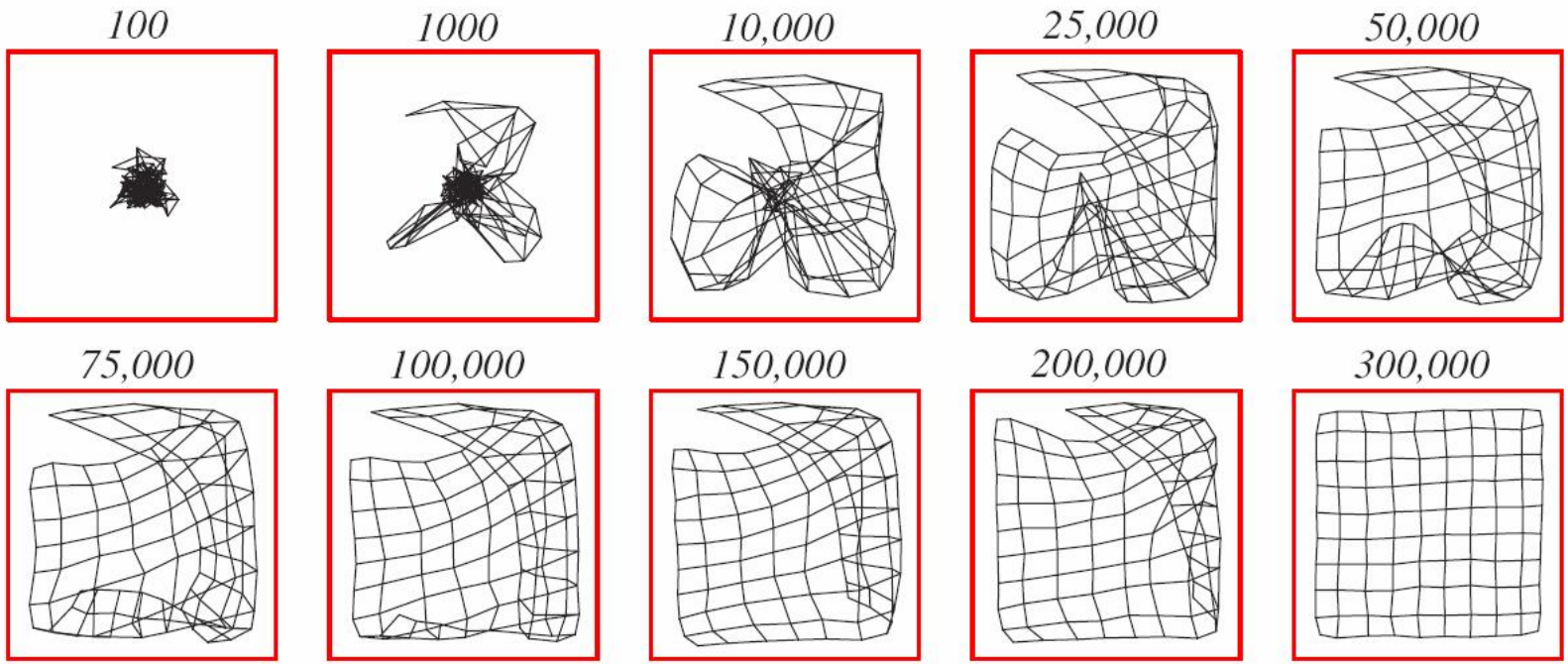
$\eta(t)$ is the learning rate

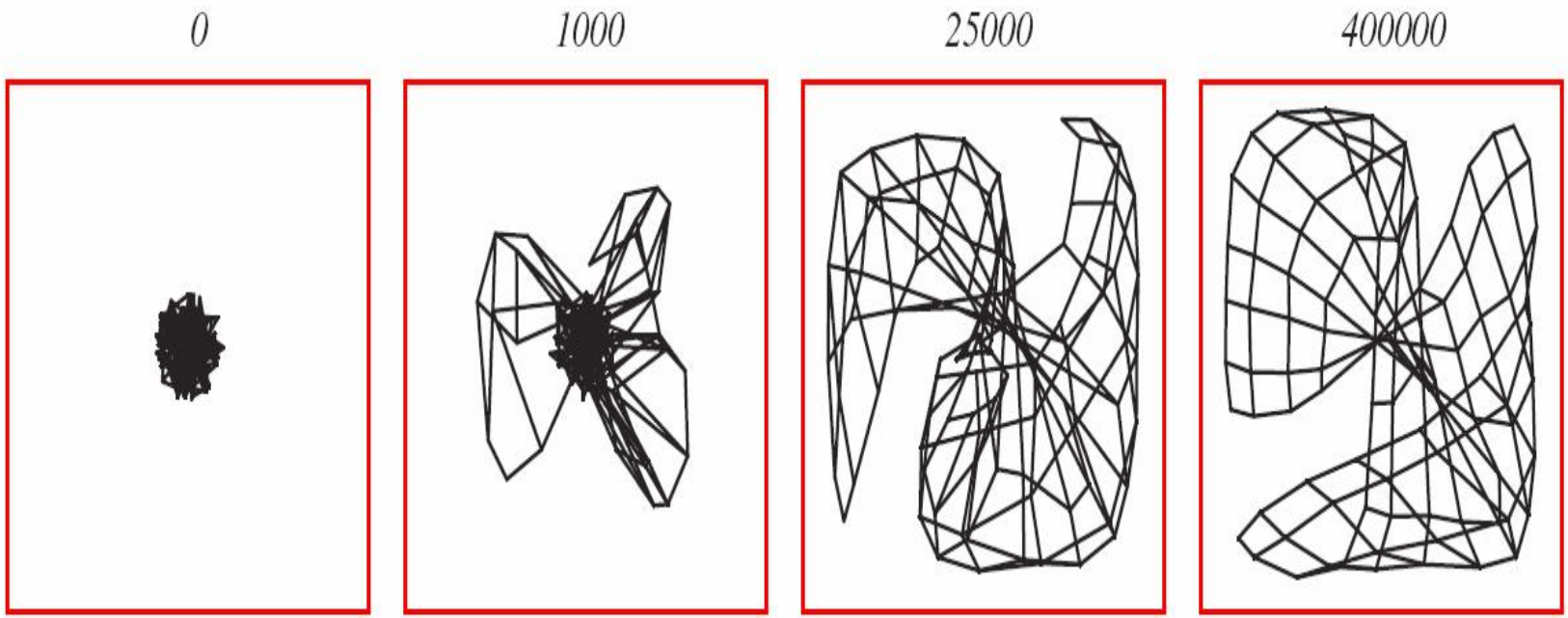
$\Lambda(|y - y^*|)$ is the window function

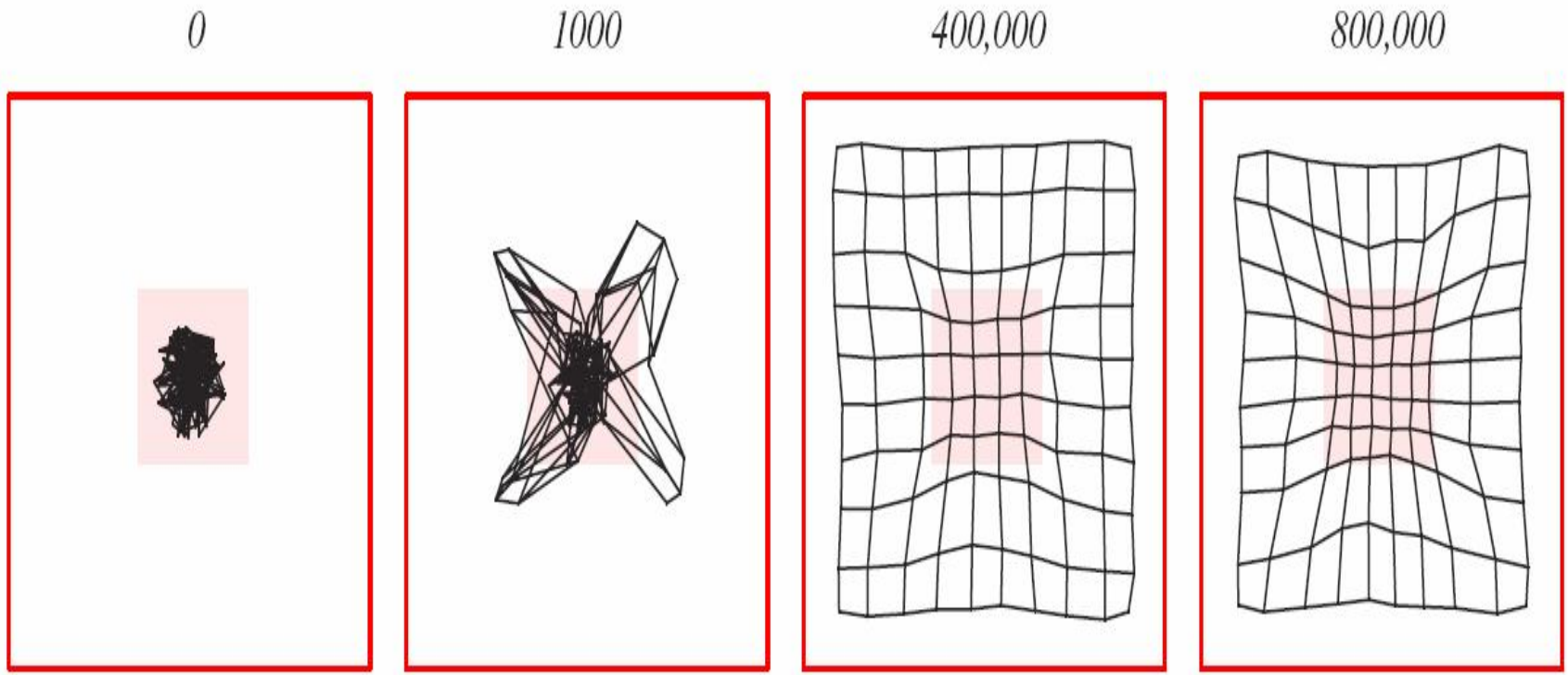
The window function equals one if $y = y^*$. The window function makes sure that neighboring points have weights that are similiar.

$\eta(t)$ decreases with time to assure that the algorithm terminates.


0
20
100
1000
10,000
25,000
50,000
75,000
100,000
150,000


100
1000
10,000
25,000
50,000
75,000
100,000
150,000
200,000
300,000


0
1000
25000
400000


0
1000
400,000
800,000

Clustering and Dimensionality Reduction

Factor Analysis - find a lower dimensional representation that accounts for correlations between features.

By removing or combining highly related features.

Data Matrix - n rows are are the d-dimensional samples.

Ordinary Clustering - group the rows with similar numbers of cluster centers.

Dimensionality Reduction - grouping the columns, with combined features being used to represent the data.

Modification of Hierarchial Clustering

In place of n by n matrix of distances, consider d by d correlation matrix

$$R = [\rho_{ij}]$$

Where ρ_{ij} is the correlation coefficient related to the covariances by

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

With the facts that

$0 \leq \rho_{ij}^2 \leq 1$

where $\rho_{ij} = 0$ for uncorrelated features.

and $\rho_{ij} = 1$ for completely correlated features.

Two features where ρ_{ij}^2 are large are good canidates to be merged and thus reducing the dimensionality by 1.

Hierarchical Dimensionality Reduction

1 **begin** **initialize** d', $D_i \leftarrow \{x_i\}$, I = 1, ..., d

2 d-hat <- d+1

3 **do** d-hat $\leftarrow$ d – 1

4 compute R

5 find most correlated distinct clusters, say D_i and D_j

6 $D_i \leftarrow D_i \cup D_j$

7 delete D_i

8 **until** d-hat = d'

9 **return** d' clusters

10 **end**

References

Duda, Richard, Hart, Peter, and Stork, David. Pattern Classification. New York: John Wily & Sons, 2001.

Munkres, James. Topology. 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 28 December 1999.