Machine learning: lecture 14

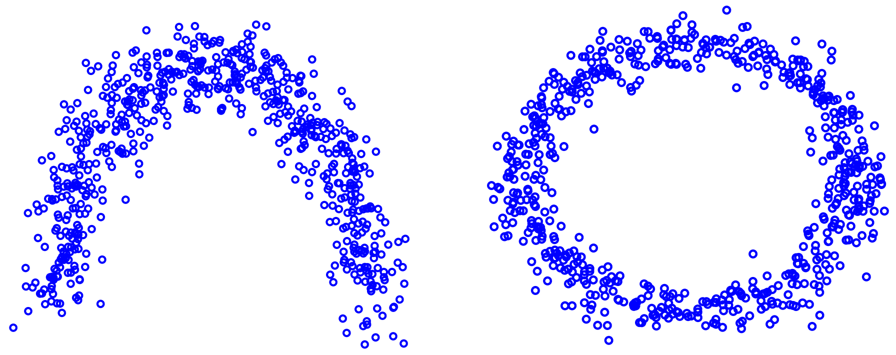
Tommi S. Jaakkola
MIT AI Lab
tommi@ai.mit.edu

Topics

- Non-parametric density estimation
  - Parzen windows
- Clustering
  - mixture models, k-means
  - agglomerative hierarchical clustering
  - Markov random walk and spectral clustering
  - semi-supervised clustering (next lecture)

Beyond parametric density models

- More mixture densities



- We can approximate almost any distribution by including more and more components in the mixture model

$$p(\mathbf{x}|\theta) = \sum_{j=1}^{k} p_j \, p(\mathbf{x}|\mu_j, \Sigma_j)$$

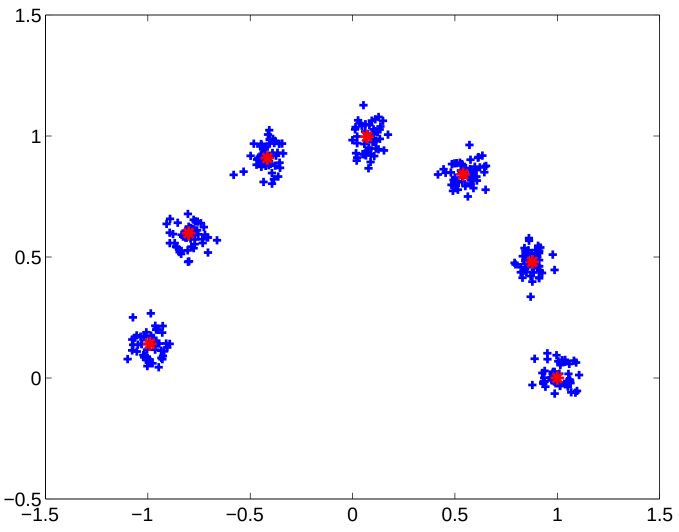
Non-parametric densities

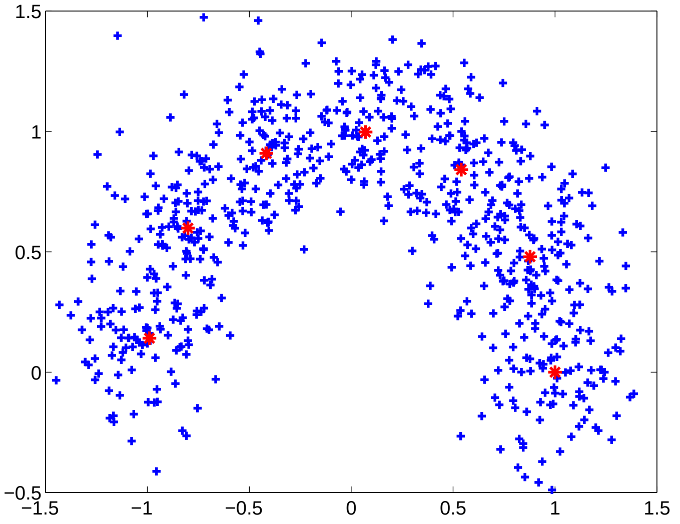
- We can even introduce one mixture component (Gaussian) per training example

$$\hat{p}(\mathbf{x};\sigma^2) = \frac{1}{n}\sum_{i=1}^{n} p(\mathbf{x}|\mathbf{x}_i,\sigma^2 I)$$

where n is the number of examples.

Here the covariances are all equal and spherical; the single parameter σ^2 controls the smoothness of the resulting density estimate





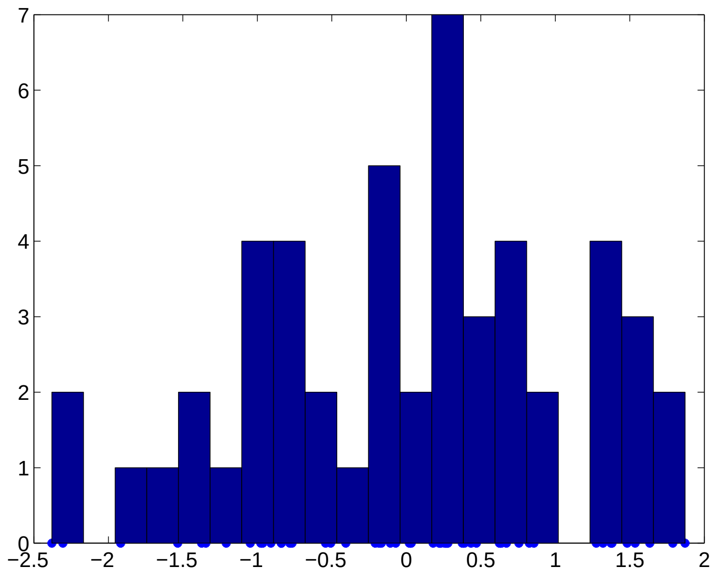
Histograms

- n training points $x_1, \ldots, x_n$

  The real line divided into non-overlapping bins

  $$[\mu_j - h, \mu_j + h)$$

  where $\mu_j = jh$ is the center of the j^{th} bin



The resulting density estimate is

$$\hat{p}_n(x) = \frac{1}{n} \times \frac{\text{\# of } x_i \text{ in the same bin as } x}{\text{width of bin containing } x}$$

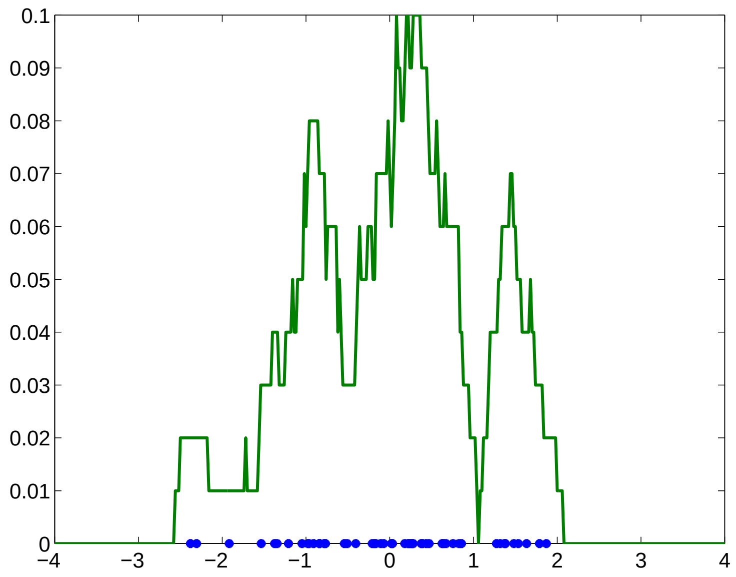
Naive estimator

- Define a window function $w(x)$:

$$w(x) = \begin{cases} \frac{1}{2}, & |x| < 1 \\ 0, & \text{otherwise} \end{cases}$$

By introducing an additional parameter h controlling the window width, we get the following (naive) density estimate

$$\hat{p}_n(x; h) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x - x_i}{h} \right)$$



(here $n = 50$ and $h = 0.02$)

Naive estimator cont'd

- The naive estimator actually converges to the true density provided that both $n \to \infty$ and $h \to 0$ (appropriately).
  For a fixed x

$$\begin{aligned}
\hat{p}_n(x;h) &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x-x_i}{h}\right) \\
&\to \frac{1}{2h} P(x-h < X < x+h) \quad \text{as } n \to \infty \\
&\to p(x) \quad \text{as } h \to 0
\end{aligned}$$

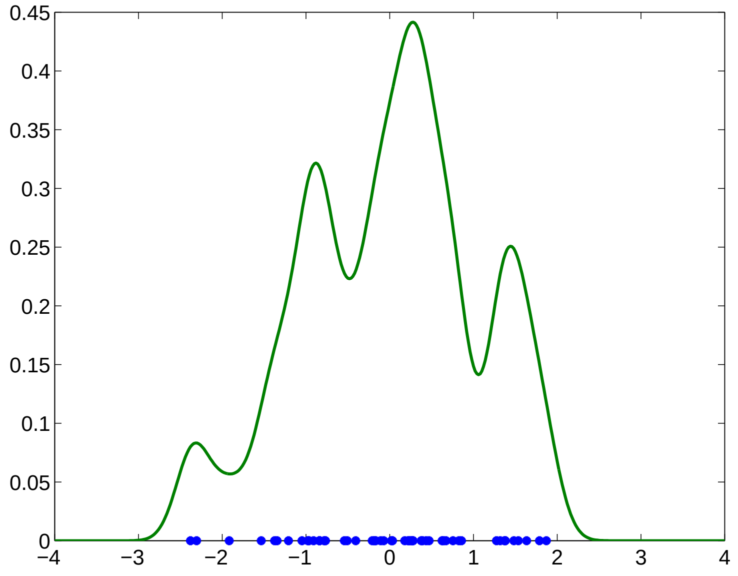
Parzen windows

- Instead of the naive window function, we can put a smooth Gaussian (or other) bump on each training example

$$\hat{p}_n(x;h) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-x_i}{h}\right), \text{ where}$$

$K(z) = \exp(-z^2/2)/\sqrt{2\pi}$ (this is also known as a *kernel function*; very different from SVM kernels).

- As a result we get a smoother estimate ($n = 50$ and $h = 0.02$ as before)

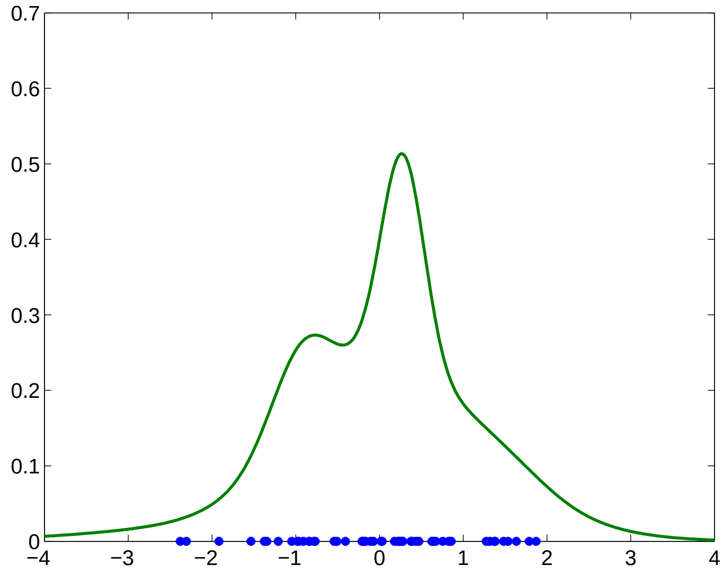


Parzen windows: variable kernel width

- We can also set the kernel width locally

  k-nearest neighbor choice: let d_{ik} be the distance from x_i to its k^{th} nearest neighbor

$$\hat{p}_n(x;k) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{d_{ik}} K\left(\frac{x - x_i}{d_{ik}}\right)$$



- The estimate is smoother where there are only few data points

Parzen windows: optimal kernel width

- We still have to set the kernel width h or the number of nearest neighbors k
- A practical solution: cross-validation

  Let $\hat{p}_{n-i}(x;h)$ be a parzen windows density estimate constructed on the basis of $n-1$ training examples leaving out x_i.

  We select h (or similarly k) that maximizes the leave-one-out log-likelihood

$$CV(h) = \sum_{i=1}^{n} \log \hat{p}_{n-i}(x_i;h)$$

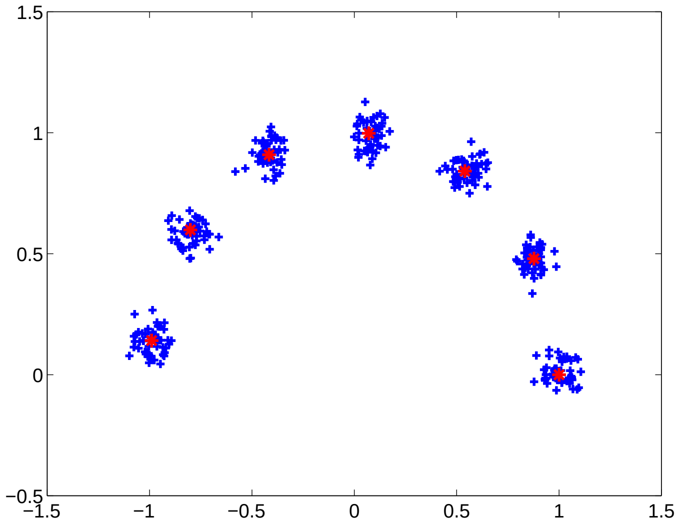
Parzen windows: multi-dimensional case

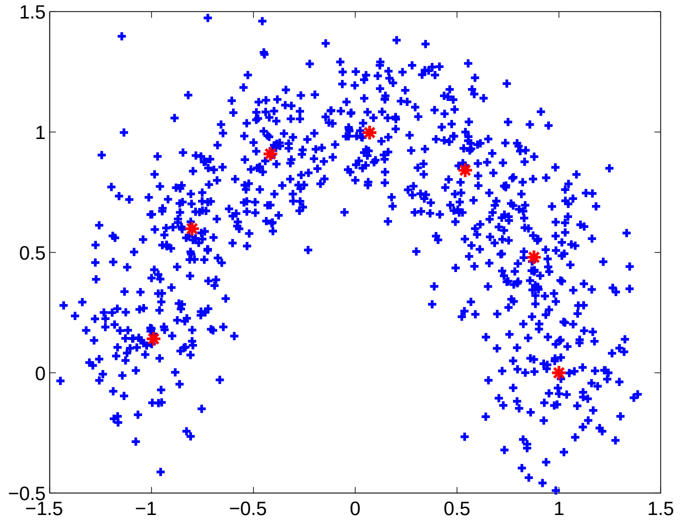
- Multi-dimensional Parzen windows estimate:

$$\hat{p}_{parzen}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} p(\mathbf{x}|\mathbf{x}_i, \sigma^2 I)$$

  where n is the number of examples.

- The covariance matrices are all equal and spherical. The single parameter σ controls the smoothness of the density estimate



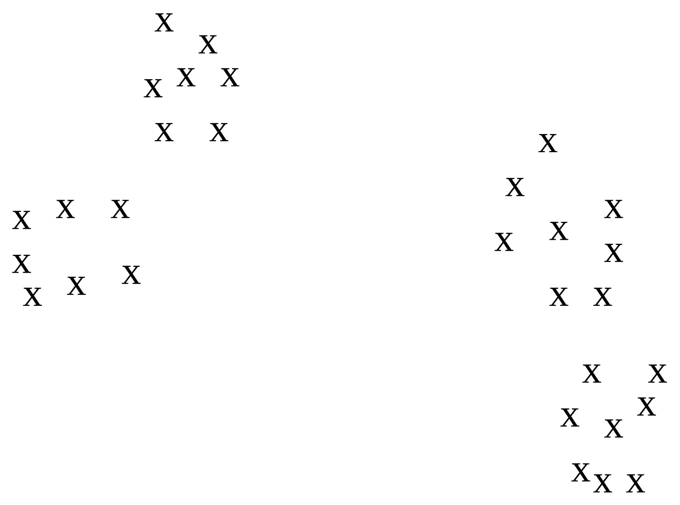


Topics

- Clustering
  - mixture models, k-means
  - agglomerative hierarchical clustering
  - Markov random walk and spectral clustering
  - semi-supervised clustering (next lecture)

Finding structure in the data: clustering

- The definition of "ground truth" often missing
  - the results need to be validated either internally (e.g., consistency) or externally (e.g., whether clusters make sense)



- Clustering relies crucially on the measure of similarity
  - position in "space", input/output relation, dynamics, etc

Basic clustering methods

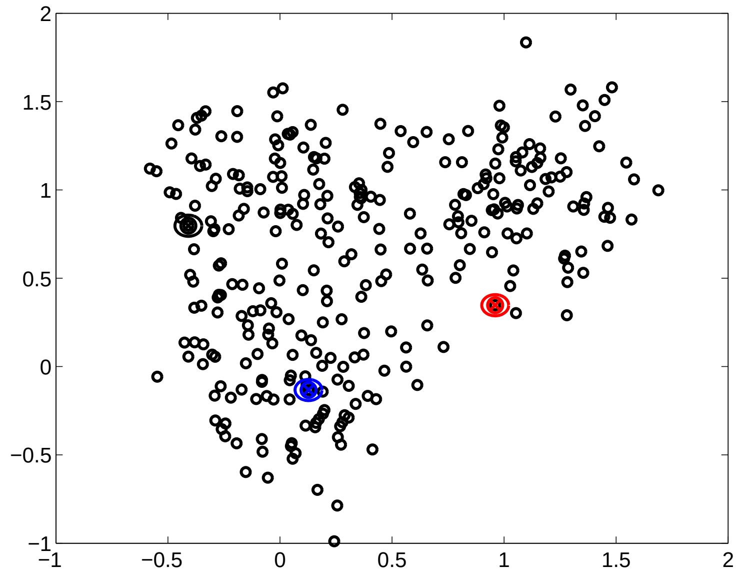
- Flat clustering methods
  - e.g., mixture models, k-means clustering
- Hierarchical clustering methods:
  1. Top-down (splitting)
     - e.g., hierarchical mixture models
  2. Bottom-up (merging)
     - e.g., hierarchical agglomerative clustering
- Other clustering methods
  - spectral clustering
  - semi-supervised clustering, etc

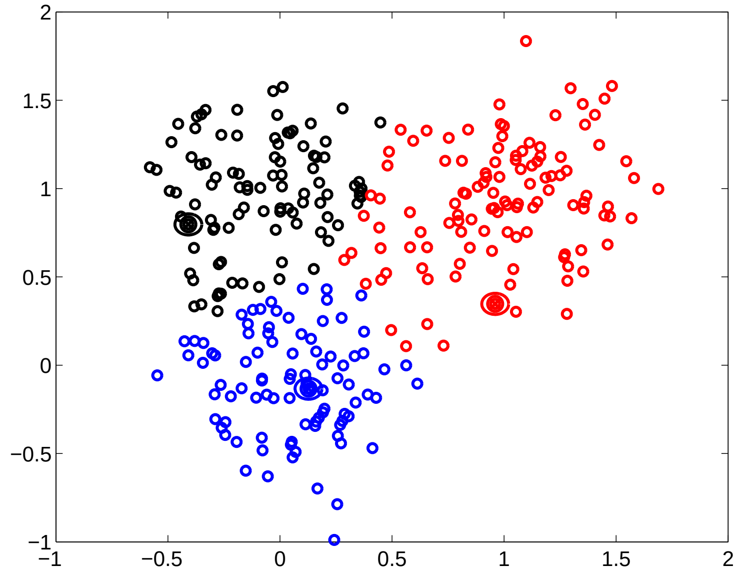
K-means clustering

- The procedure:

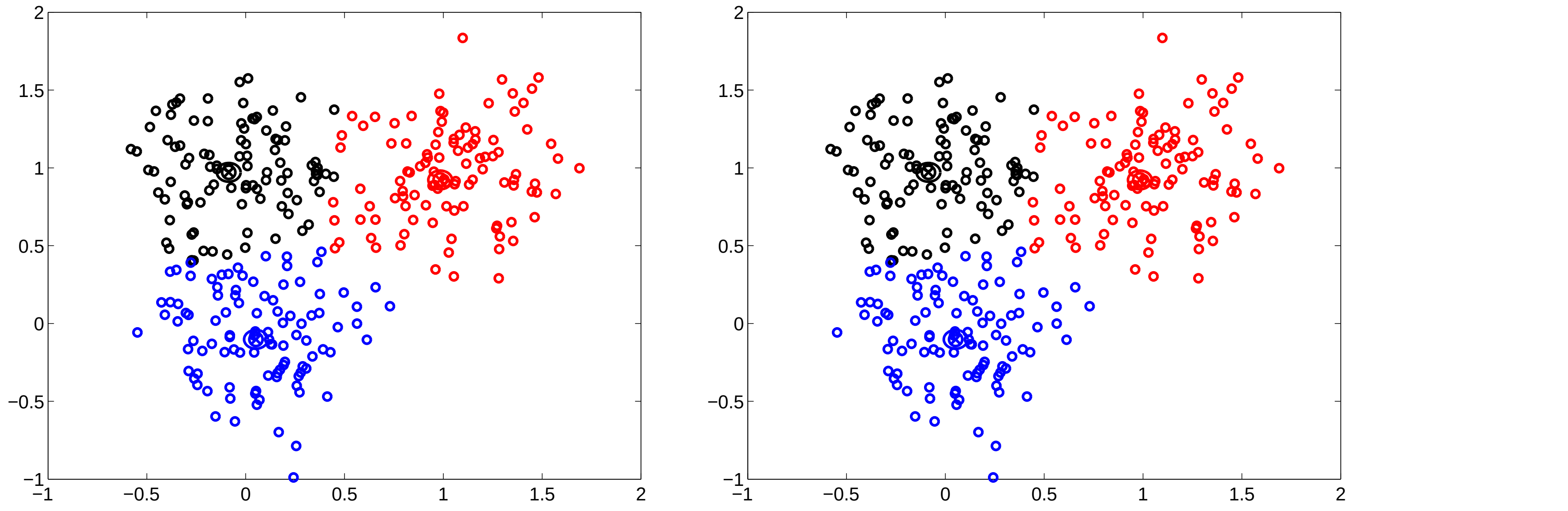
  1. Pick k arbitrary centroids (cluster means)
  2. Assign each example to its "closest" centroid (**E-step**)
  3. Adjust the centroids to be the means of the examples assigned to them (**M-step**)
  4. Goto step 2 (until no change)

- The algorithm is guaranteed to converge in a finite number of iterations





K-means clustering cont'd



- K-means clustering corresponds to a Gaussian mixture model estimation with EM whenever the covariance matrices of the Gaussian components are set to $\Sigma_j = \sigma^2 I$, for all j and some fixed small σ^2

Hierarchical (bottom-up) clustering

- Hierarchical agglomerative clustering: we sequentially merge the pair of "closest" points/clusters
- The procedure
  1. Find two closest points (clusters) and merge them
  2. Proceed until we have a single cluster (all the points)
- Two prerequisites:
  1. distance measure $d(\mathbf{x}_i, \mathbf{x}_j)$ between two points
  2. distance measure between clusters (cluster linkage)

Hierarchical (bottom-up) clustering

- A *linkage* method: we have to be able to measure distances between clusters of examples C_k and C_l

a) Single linkage:

$$d_{kl} = \min_{i \in C_k, j \in C_l} d(\mathbf{x}_i, \mathbf{x}_j)$$

b) Average linkage:

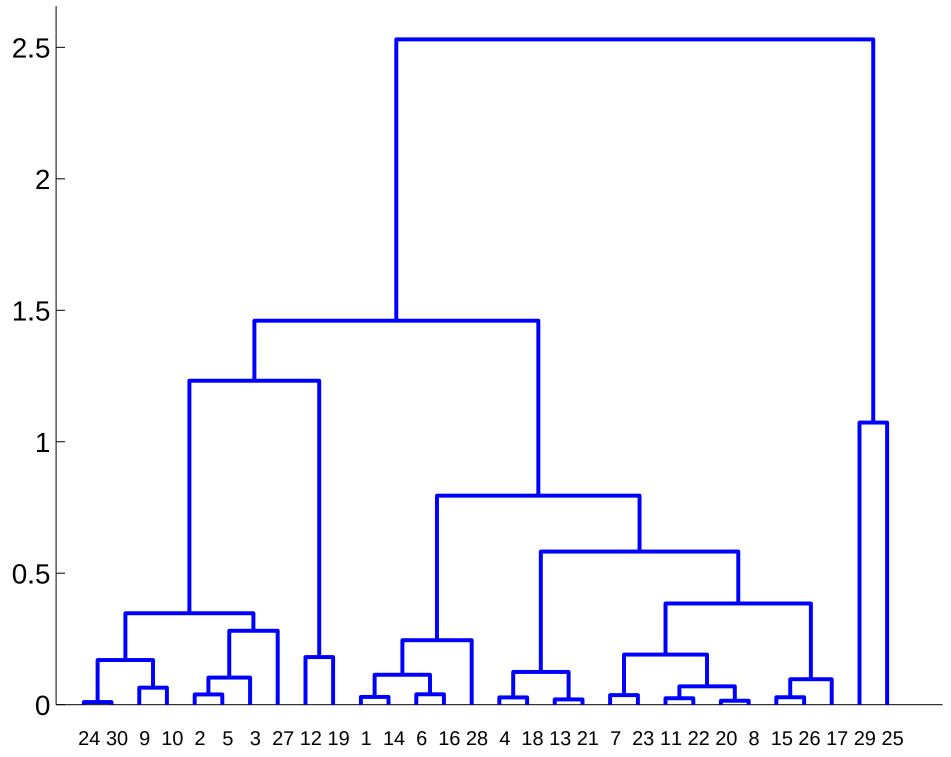
$$d_{kl} = \frac{1}{|C_l|\,|C_k|} \sum_{i \in C_k, j \in C_l} d(\mathbf{x}_i, \mathbf{x}_j)$$

c) Centroid linkage:

$$d_{kl} = d(\bar{\mathbf{x}}_k, \bar{\mathbf{x}}_l), \quad \bar{\mathbf{x}}_l = \frac{1}{|C_l|} \sum_{i \in C_l} \mathbf{x}_i$$

Hierarchical (bottom-up) clustering

- A dendrogram representation of hierarchical clustering



The height of each pair represents the distance between the merged clusters; the specific linear ordering of points is chosen for clarity

Spectral clustering: preliminaries

- Spectral clustering (as described here) relies on a random walk over the points

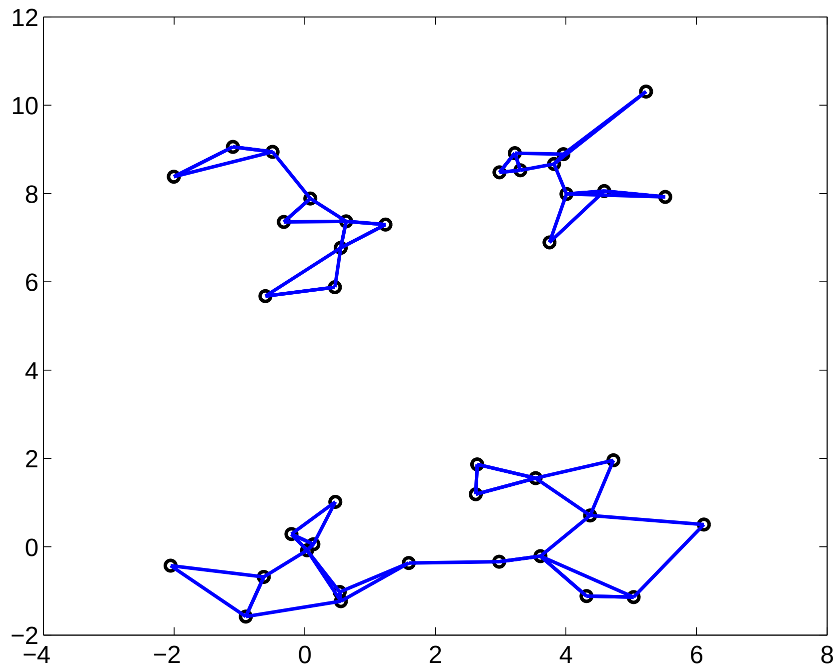
  We find the random walk via the following steps

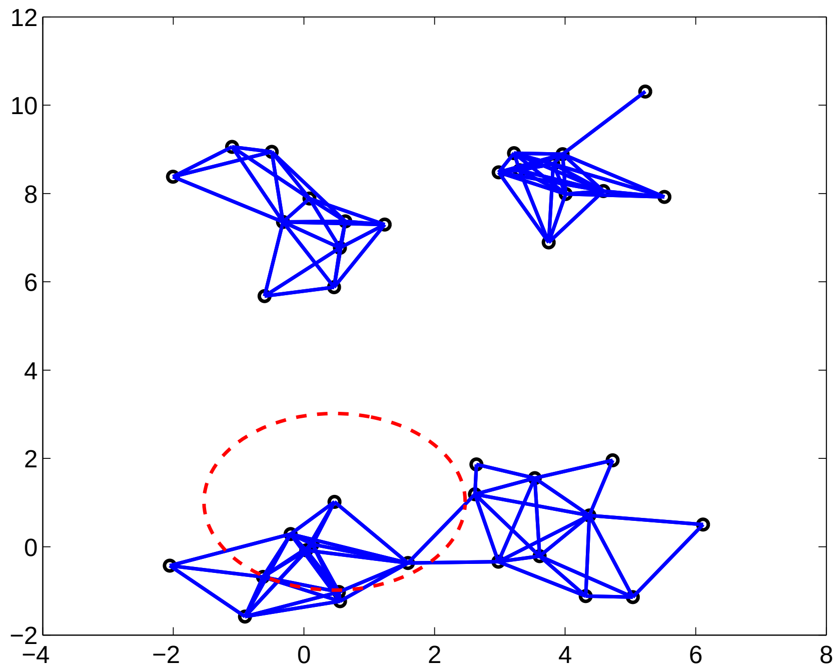
  1. construct a neighborhood graph
  2. assign weights to the edges in the graph
  3. define a transition probability matrix based on the weights

- The points are clustered on the basis of the eigenvectors of the resulting transition probability matrix

Step 1: neighborhood graph

- We can connect each point to its k-nearest neighbors, or connect each point to all neighbors within distance ϵ

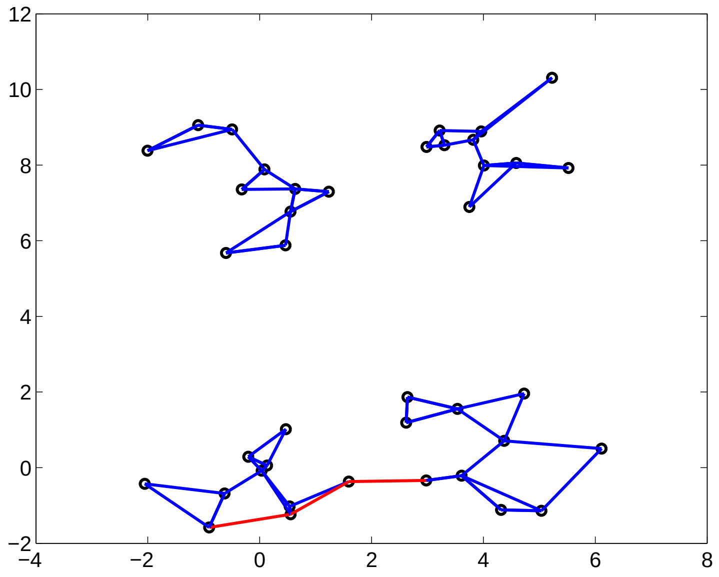




Step 2: edge weights

- We assign symmetric non-negative edge weights W_{ij}:

$$\begin{aligned} W_{ij} &= \exp\{-\beta\|\mathbf{x}_i - \mathbf{x}_j\|\}, \ \text{if } i \text{ and } j \text{ connected} \\ W_{ij} &= 0, \ \text{otherwise} \end{aligned}$$



Note: we do not use a squared distance in the exponent so that a weight for a path is computed analogously to the edge weights

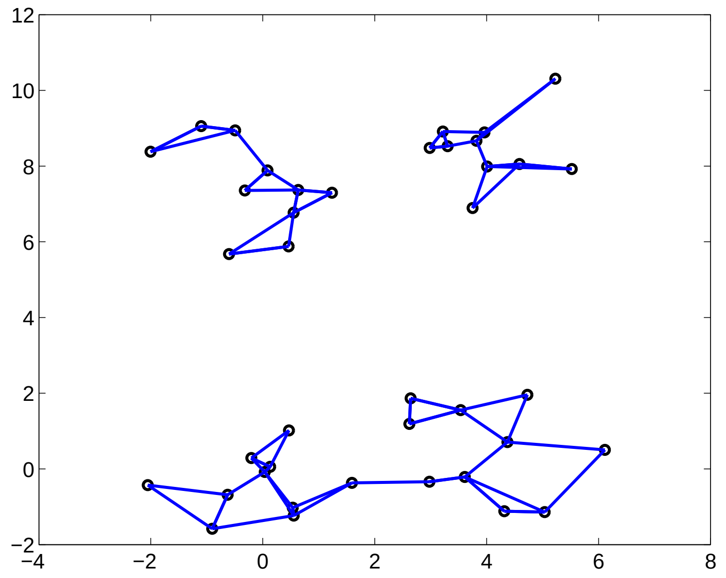
Step 3: transition probability matrix

- Finally, we define a Markov random walk over the neighborhood graph by constructing a transition probability matrix from the edge weights

$$P_{ij} = \frac{W_{ij}}{W_{i\cdot}}, \quad \text{where } W_{i\cdot} = \sum_j W_{ij}$$

and $\sum_j P_{ij} = 1$ for all i.

The random walk proceeds by successively selecting points according to $j \sim P_{ij}$, where i specifies the current location



Random walk: properties

- If we start from i_0, the distribution of points i_t that we end up in after t steps is given by

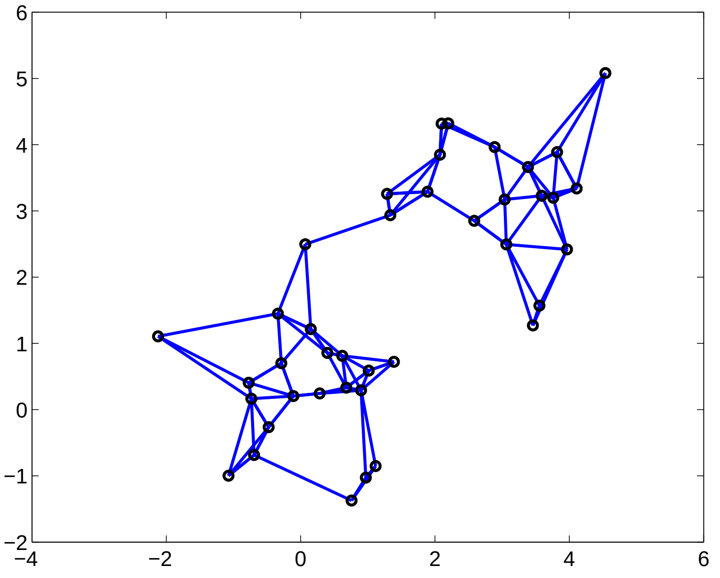
$$\begin{aligned}
&i_1 \sim P_{i_0\, i_1}, \\
&i_2 \sim \sum_{i_1} P_{i_0, i_1} P_{i_1\, i_2} = [P^2]_{i_0\, i_2}, \\
&i_3 \sim \sum_{i_1} \sum_{i_2} P_{i_0, i_1} P_{i_1\, i_2} P_{i_2\, i_3} = [P^3]_{i_0\, i_3}, \\
&\ldots \\
&i_t \sim [P^t]_{i_0\, i_t}
\end{aligned}$$

where $P^t = PP\ldots P$ (t matrix products) and $[\cdot]_{ij}$ denotes the i, j component of the matrix.

Random walk: properties

- The distributions of points we end up in after t steps converge as t increases. If the graph is connected, the resulting distribution is independent of the starting point

  Even for large t, the transition probabilities $[P^t]_{ij}$ have a slightly higher probability of transitioning within "clusters" than across; we want to recover this effect from eigenvalues/vectors



details in the next lecture...