K-Means, Expectation Maximization and Segmentation

D.A. Forsyth, CS543

K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

- can't do this by search
  - there are too many possible allocations.
- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
  - x could be any set of features for which we can compute a distance (careful about scaling)

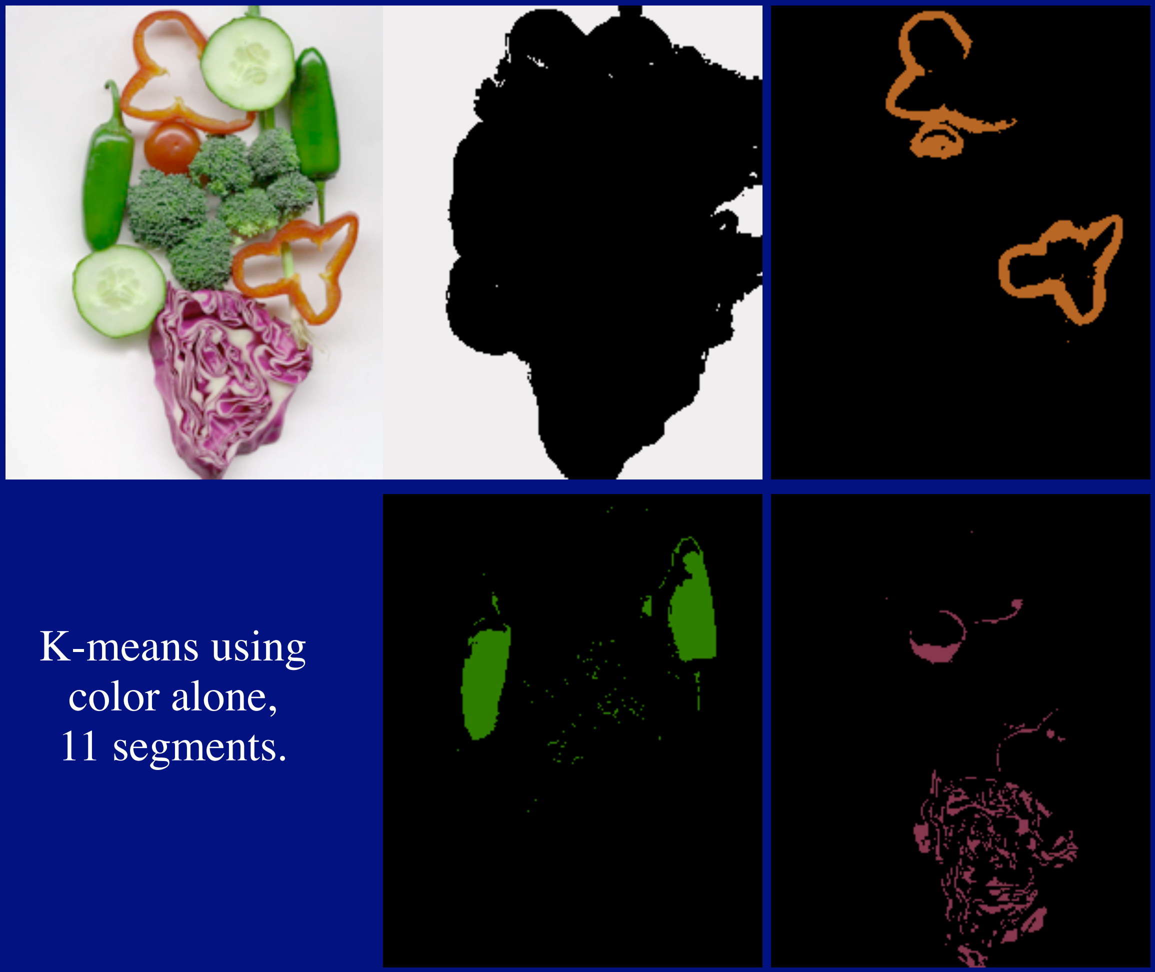
Image | Clusters on intensity | Clusters on color

K-means clustering using intensity alone and color alone

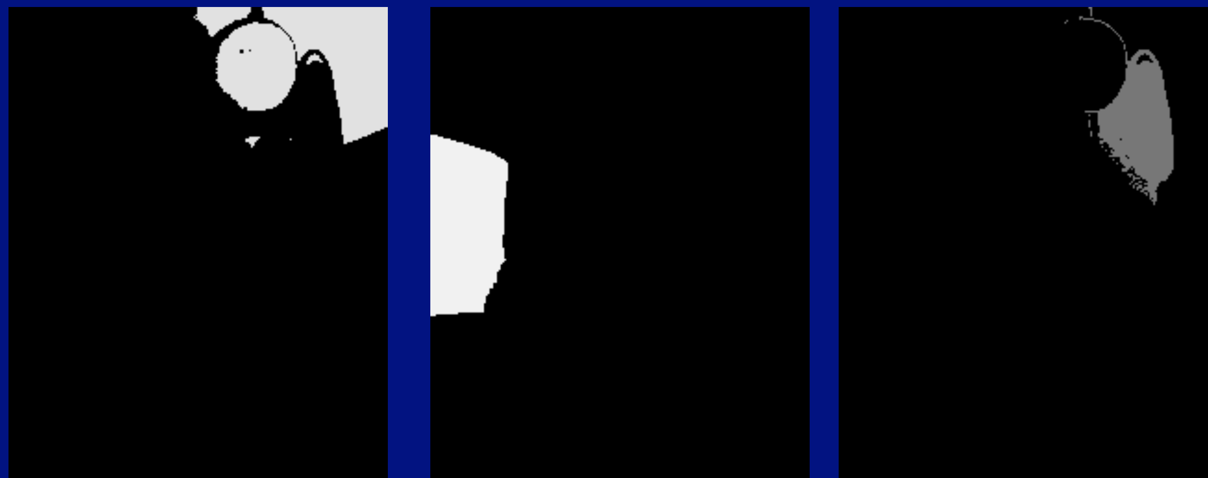
Image

Clusters on color

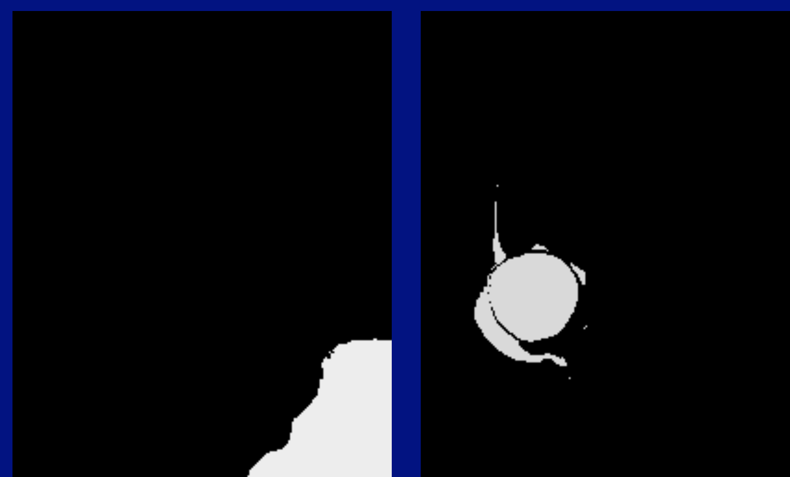
K-means using color alone, 11 segments



K-means using
color alone,
11 segments.



K-means using colour and position, 20 segments



Mixture models and segmentation

- In k-means, we clustered pixels using hard assignments
  - each pixel goes to closest cluster center
  - but this may be a bad idea
    - pixel may help estimate more than one cluster
- We will build a probabilistic mixture model

$$P(\mathbf{x}|\mu_1, \ldots, \mu_k, \pi_1, \ldots, \pi_k, \Sigma) = \sum_i \pi_i P(\mathbf{x}|\mu_i, \Sigma)$$

Mixture model

- Interpretation:
  - obtain pixel by
    - choosing a mixture component
    - given that component, choosing pixel
- Natural to have each mixture component be a Gaussian

$$P(\mathbf{x}|\mu_1, \ldots, \mu_k, \pi_1, \ldots, \pi_k, \Sigma) = \sum_i \pi_i P(\mathbf{x}|\mu_i, \Sigma)$$

Mixture components

- Gaussians
  - are oriented "blobs" in the feature space
  - we will assume covariance is known, and work with mean
  - expression below

$$P(\mathbf{x}|\mu_i, \Sigma) \propto \exp \frac{-(\mathbf{x} - \mu_i)\Sigma^{-1}(\mathbf{x} - \mu_i)}{2}$$

Problem: Learning and IDLL

- We must estimate the mixture weights and means
- Maximising likelihood is very hard
  - in this form, sometimes known as incomplete data log-likelihood

$$L(\theta) = \sum_i \log P(\mathbf{x}_i|\theta) = \sum_i \log \left(\sum_j \pi_j P(\mathbf{x}_i|\mu_j, \Sigma) \right)$$

Complete Data Log-likelihood

- Learning would be easy if we knew which blob each data item came from
  - weights: count what fraction came from which blob
  - means: average data items
- Introduce a set of variables
  - to tell which mixture component a data item came from
  - d_{ij} is 1 if data item i comes from blob j
    - 0 otherwise

$$P(\mathbf{x}_i|\delta_i,\theta) = \prod_j P(\mathbf{x}_i|\mu_j)^{\delta_{ij}}$$

Complete Data Log-likelihood

- Write the probability for x_i, d_ij conditioned on params

$$
\begin{aligned}
P(\mathbf{x}_i, \delta_i | \theta) &= P(\mathbf{x}_i | \delta_i, \theta) P(\delta_i | \theta) \\
&= (\prod_j [P(x_i | \mu_j) \pi_j]^{\delta_{ij}})
\end{aligned}
$$

Complete Data Log-likelihood

- Log likelihood of x, allocation, conditioned on parameters
- Not obviously helpful
  - but notice the useful form - think of d as switches
  -

$$\begin{aligned}\log P(\mathbf{x}, \delta|\theta) &= \sum_i \log P(\mathbf{x}_i, \delta_i|\theta) \\ &= \sum_{ij} \left[\delta_{ij} \left\{\log P(x_i|\mu_j) + \log \pi_j\right\}\right]\end{aligned}$$

Learning and CDLL

- Introduce hidden variables to get complete data log-likelihood
  - d_{ij} is 1 if data item i comes from blob j
    - 0 otherwise
  - Learning would be easy if we knew which blob each data item came from
    - weights: count what fraction came from which blob
    - means: average data items
  - But we don't

$$L_c(\theta, \delta_{ij}) = \sum_{ij} \delta_{ij} \log \left(\pi_j P(\mathbf{x}_i | \mu_j, \Sigma) \right)$$

Working with CDLL

- Notice:
  - with an estimate of the parameters, can estimate blobs data came from
    - this could be a soft estimate
    - we could plug this in, then reestimate the parameters
- Formal procedure:
  - start with estimate
  - form Q function, below (The E-step)
    - maximise in parameters (The M-step)
  - Iterate

$$Q(\theta; \theta^{(n)}) = E_{P(\delta | \mathbf{x}, \theta^{(n)})} \left(L_c(\theta, \delta) \right)$$

The E-step for Gaussian mixtures

- Notice that the expression for the Q function simplifies

$$
\begin{aligned}
E_{P(\delta|\mathbf{x},\theta^{(n)})}\left(L_c(\theta,\delta)\right) &= E_{P(\delta|\mathbf{x},\theta^{(n)})}\left(\sum_{ij}\delta_{ij}\left[\log\pi_j+\log P(\mathbf{x}_i|\mu_j,\Sigma)\right]\right) \\
&= \left(\sum_{ij} P(\delta_{ij}=1|\mathbf{x},\theta^{(n)})\left[\log\pi_j+\log P(\mathbf{x}_i|\mu_j,\Sigma)\right]\right)
\end{aligned}
$$

The E-step for Gaussian mixtures

- We rearrange using probability identities

$$
\begin{aligned}
P(\delta_{ij} = 1 | \mathbf{x}, \theta^{(n)}) &= \frac{P(\mathbf{x}, \delta_{ij} = 1 | \theta^{(n)})}{P(\mathbf{x} | \theta^{(n)})} \\
&= \frac{P(\mathbf{x} | \delta_{ij} = 1, \theta^{(n)}) P(\delta_{ij} = 1 | \theta^{(n)})}{P(\mathbf{x} | \theta^{(n)})} \\
&= \frac{P(\mathbf{x} | \delta_{ij} = 1, \theta^{(n)}) P(\delta_{ij} = 1 | \theta^{(n)})}{P(\mathbf{x} | \delta_{ij} = 1, \theta^{(n)}) P(\delta_{ij} = 1 | \theta^{(n)}) + P(\mathbf{x}, \delta_{ij} = 0 | \theta^{(n)})}
\end{aligned}
$$

The E step for Gaussian mixtures

- And substitute

$$P(\mathbf{x}_i, \delta_{ij} = 0 | \theta^{(n)}) = \sum_{k \neq j} \pi_k \frac{1}{Z} \exp \frac{-(\mathbf{x}_i - \mu_k^{(n)}) \Sigma^{-1} (\mathbf{x}_i - \mu_k^{(n)})}{2}$$

$$P(\mathbf{x}_i | \delta_{ij} = 1, \theta^{(n)}) = \frac{1}{Z} \exp \frac{-(\mathbf{x}_i - \mu_j^{(n)}) \Sigma^{-1} (\mathbf{x}_i - \mu_j^{(n)})}{2}$$

$$P(\delta_{ij} = 1 | \theta^{(n)}) = \pi_j$$

The M step for Gaussian mixtures

- We must maximise

$$\sum_{ij} P(\delta_{ij} = 1|\mathbf{x}, \theta^{(n)}) \left[\log \pi_j + \log P(\mathbf{x}_i|\mu_j, \Sigma)\right] =$$

$$\sum_{ij} P(\delta_{ij} = 1|\mathbf{x}, \theta^{(n)}) \left[\log \pi_j - \frac{-(\mathbf{x}_i - \mu_j)\Sigma^{-1}(\mathbf{x}_i - \mu_j)}{2} - \log Z\right]$$

- in the mixture weights and in the means
  - we can drop log Z

Two ways

- differentiate, set to zero, etc.
- regard the expectations as "soft counts"
  - so mixture weights from soft counts as:

$$\boxed{\pi_j = \frac{\sum_i P(\delta_{ij} = 1|\mathbf{x}_i, \theta^{(n)})}{\sum_{i,j} P(\delta_{ij} = 1|\mathbf{x}_i, \theta^{(n)})}}$$

  - and means from soft counts as:

$$\boxed{\mu_j = \frac{\sum_i \mathbf{x}_i P(\delta_{ij} = 1|\mathbf{x}_i, \theta^{(n)})}{\sum_{i,j} P(\delta_{ij} = 1|\mathbf{x}_i, \theta^{(n)})}}$$

Segmentation with EM

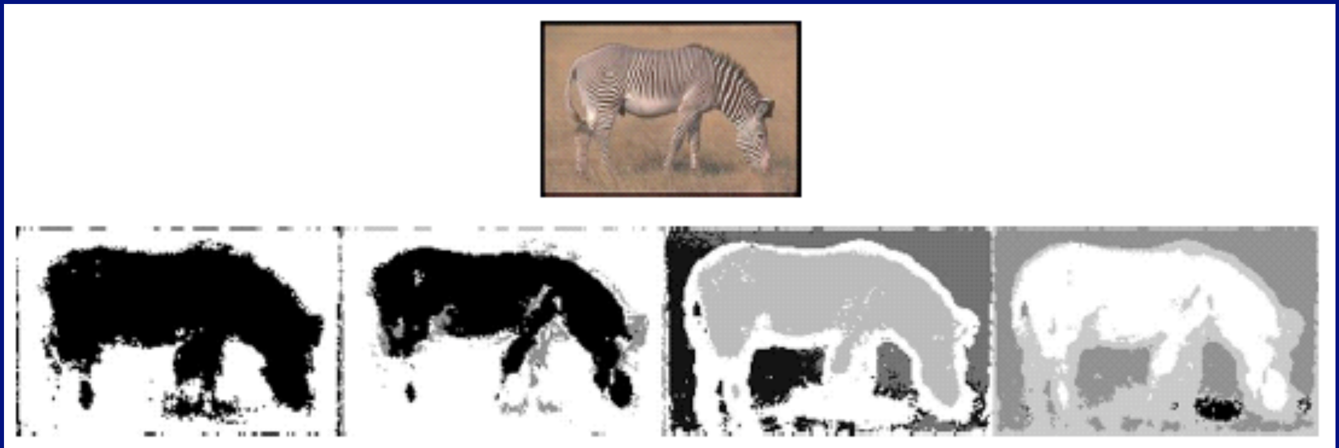
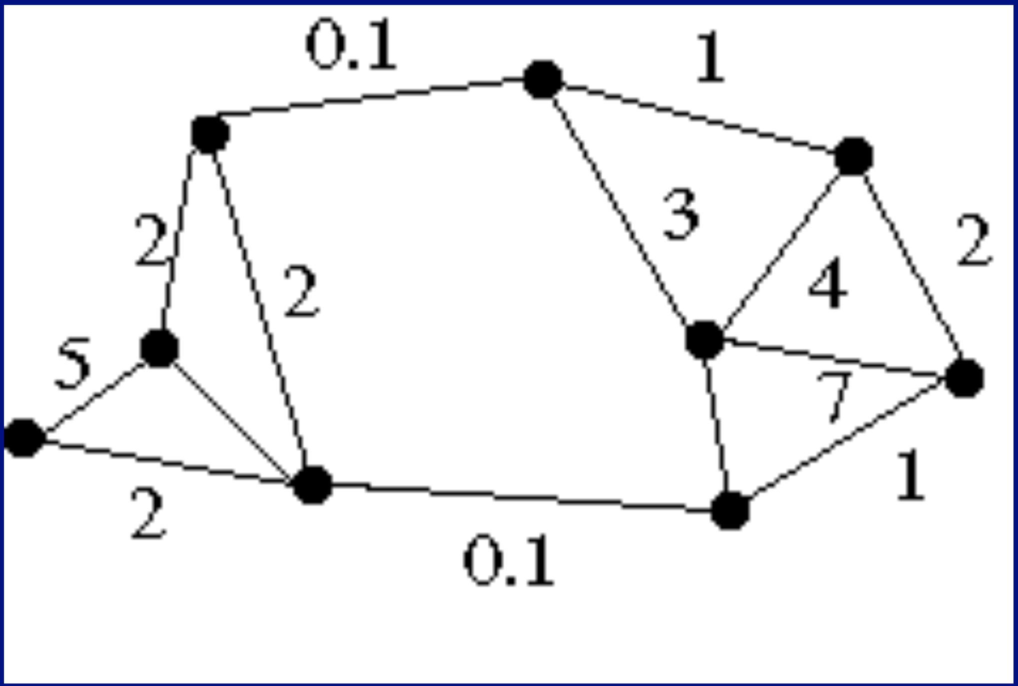
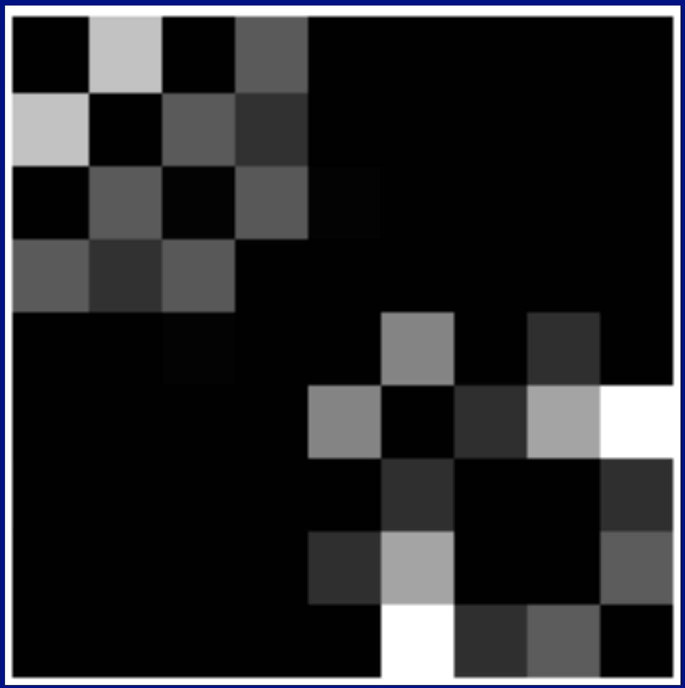


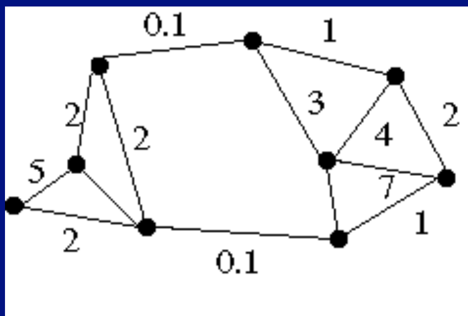
Figure from "Color and Texture Based Image Segmentation Using EM and Its Application to Content Based Image Retrieval",S.J. Belongie et al., Proc. Int. Conf. Computer Vision, 1998, c1998, IEEE

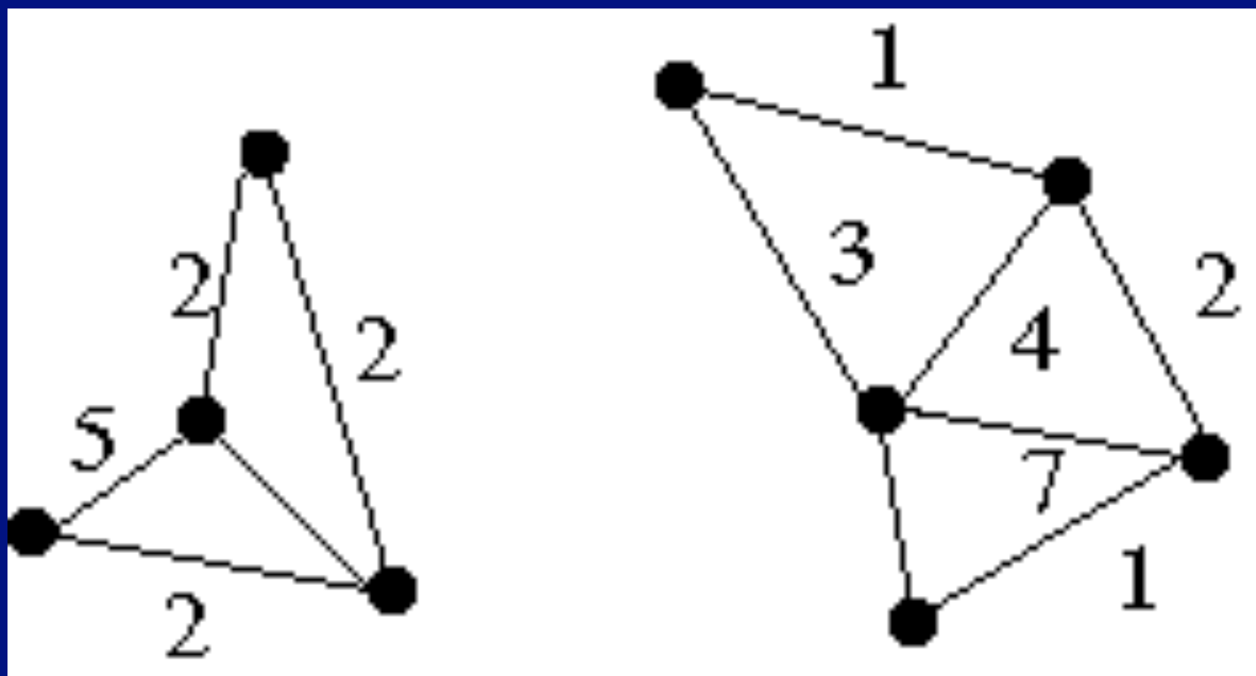
Affinity matrix





Good split





Measuring Affinity

Intensity

$$aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_i^2}\right)\left(\|I(x) - I(y)\|^2\right)\right\}$$

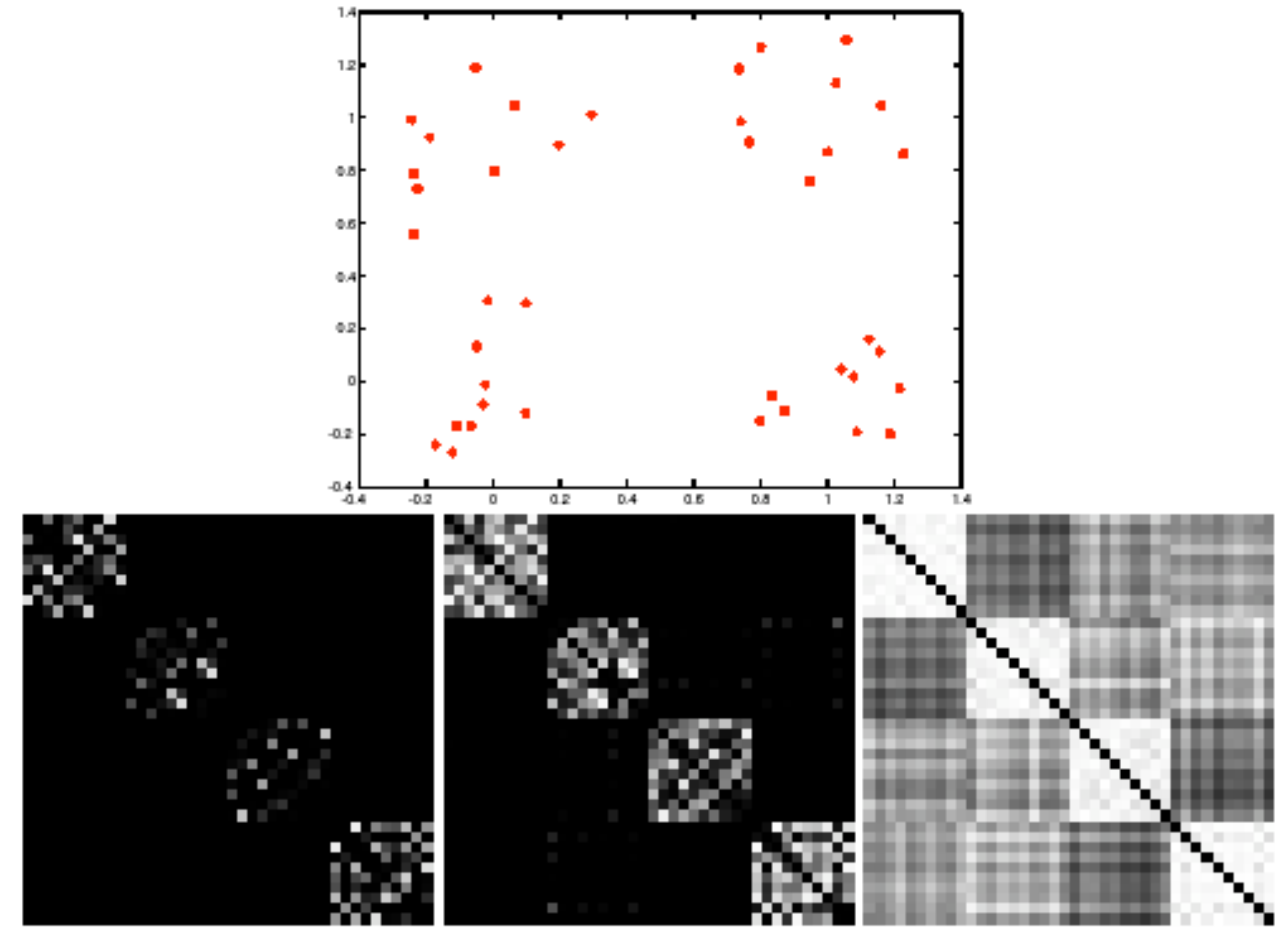
Distance

$$aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)\left(\|x - y\|^2\right)\right\}$$

Texture

$$aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_t^2}\right)\left(\|c(x) - c(y)\|^2\right)\right\}$$

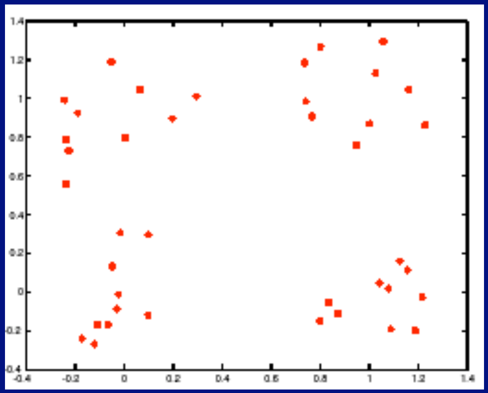
Scale affects affinity



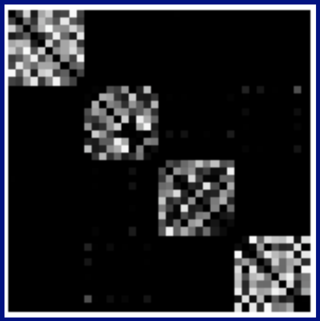
Eigenvectors and cuts

- Simplest idea: we want a vector a giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize $\boxed{a^T A a}$
- But need the constraint $\boxed{a^T a = 1}$
  - This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue

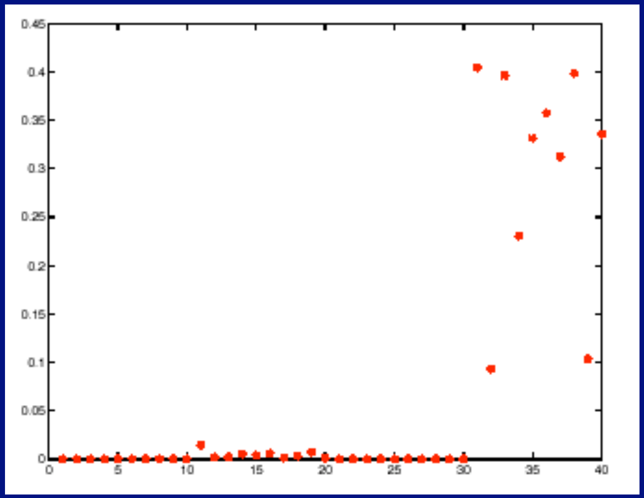
Example eigenvector



points



matrix



eigenvector

More than two segments

- Two options
  - Recursively split each side to get a tree, continuing till the eigenvalues are too small
  - Use the other eigenvectors

Normalized cuts

- Current criterion evaluates within cluster similarity, but not across cluster difference
- Instead, we'd like to maximize the within cluster similarity compared to the across cluster difference
- Write graph as V, one cluster as A and the other as B• Maximize

$$\boxed{\left(\frac{assoc(A,A)}{assoc(A,V)}\right)+\left(\frac{assoc(B,B)}{assoc(B,V)}\right)}$$

  - i.e. construct A, B such that their within cluster similarity is high compared to their association with the rest of the graph

Normalized cuts

- Write a vector y whose elements are 1 if item is in A, -b if it's in B
- Write the matrix of the graph as W, and the matrix which has the row sums of W on its diagonal as D, 1 is the vector with all ones.
- Criterion becomes

$$\min_{y}\left(\frac{y^T(D-W)y}{y^T D y}\right)$$

- and we have a constraint

$$y^T D 1 = 0$$

  - This is hard to do, because y's values are quantized

Normalized cuts

- Instead, solve the generalized eigenvalue problem

$$\max_y \left(y^T (D - W) y \right) \text{ subject to } \left(y^T D y = 1 \right)$$

- which gives

$$(D - W) y = \lambda D y$$

- Now look for a quantization threshold that maximises the criterion --- i.e all components of y above that threshold go to one, all below go to -b

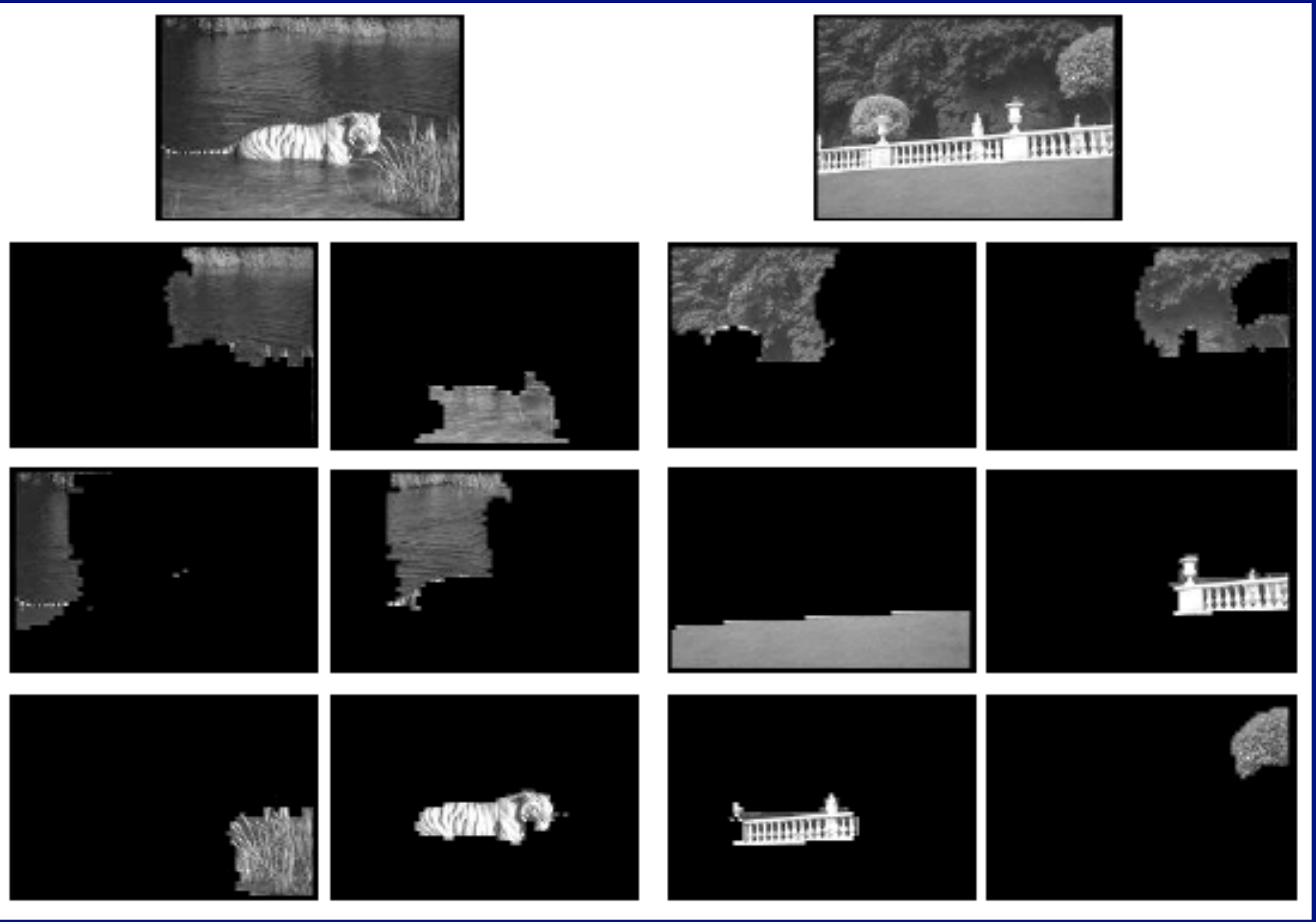


Figure from “Image and video segmentation: the normalised cut framework”,
by Shi and Malik, copyright IEEE, 1998