Image segmentation as clustering

D.A. Forsyth,

University of Illinois at Urbana Champaign

Key idea of segmentation

- Break images/videos into large, useful pieces
  - internally coherent pieces
    - same color; same color and texture; etc
  - simplify
  - Identify key objects

Segmentation – master recipe

There is a master recipe for image segmentation. This relies on *clustering*. a procedure that takes individual data items – for example, pixels, image patches – and produces blobs or *clusters* consisting of many similar data items.

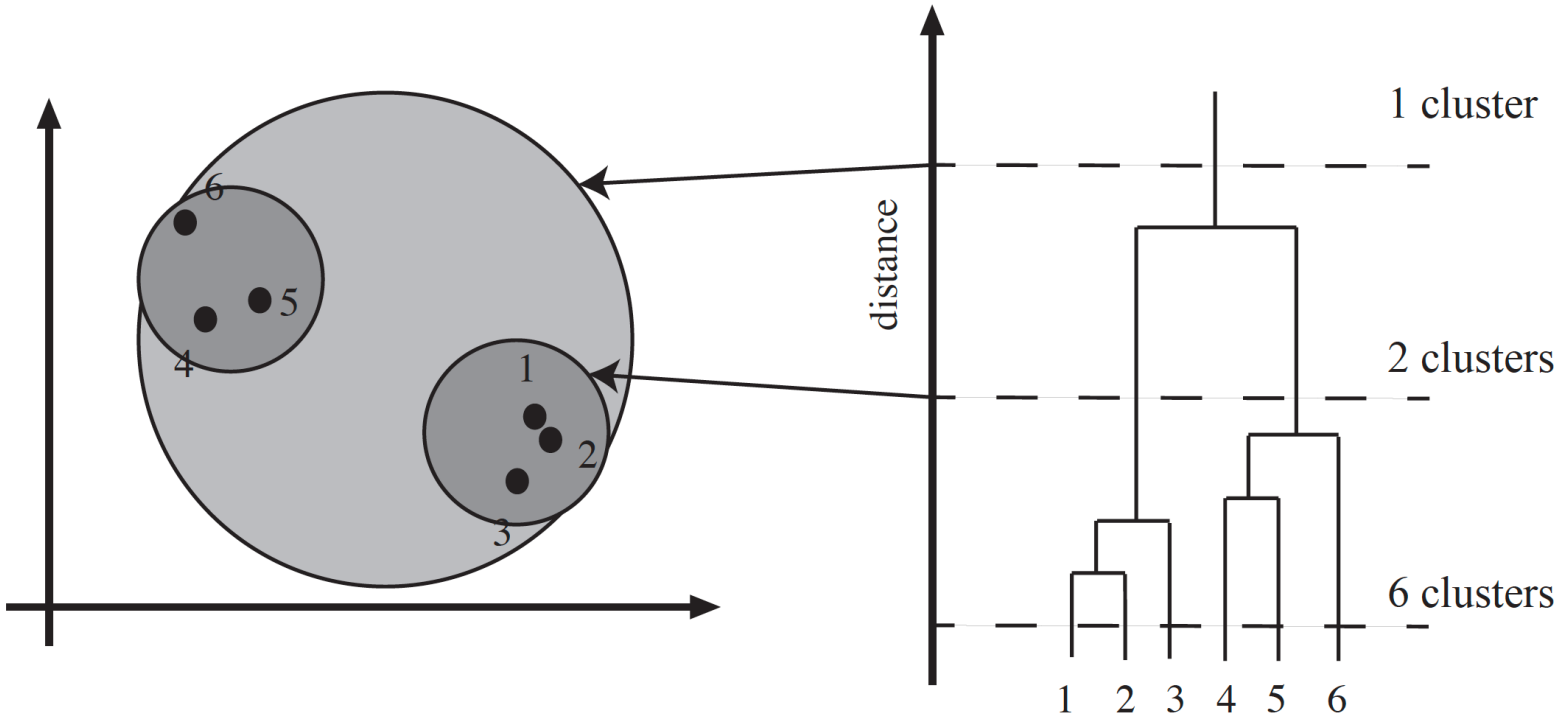
Procedure: 9.1 *Image Segmentation: Master recipe*

Compute a feature vector at each pixel of the image, then cluster the feature vectors. Each segment consists of the pixels whose feature vectors are in the same cluster.

Agglomerative and divisive

Generally, to cluster data items, you must determine (a) how many clusters there are; and (b) which data items belong to which cluster. There are two natural clustering algorithms. In **agglomerative clustering**, you start with each data item being a cluster, and then merge clusters recursively to yield a good clustering. In **divisive clustering**, you start with the entire data set being a cluster, and then split clusters recursively to yield a good clustering. Mostly, divisive clustering isn't much used in vision applications.

Simple clusters and dendrograms



Distances between clusters

- Distances between points are easy
  - but distances between clusters?
- Three standard solutions
  - Distance between the closest pair of points
    - single link clustering
      - can get long thin clusters in feature space
  - Distance between furthest pair of points
    - complete link clustering
      - blobby clusters in feature space
  - Average of distances in clusters
    - average link clustering
      - tends to blobby

Distances

- Cluster on color – feature vector is (r, g, b)
  - but clusters might not even be connected
- Cluster on color and position – (r, g, b, x, y)
  - “blobby” clusters
  - BUT what if r, g, b are in range 0, 1
    - and x, y is in range 0, 1024?
    - scaling problem; fix with elementary fiddling
- More interesting segmentation
  - more complicated feature vectors
  - more serious scaling problems

Mahalanobis

Start with a dataset of N d-dimensional vectors; write $\{\mathbf{x}\}$ for the dataset and $\mathbf{x}_i$ for the i'th item. Write the covariance matrix for this dataset $\mathsf{Covmat}(\{\mathbf{x}\})$.

Definition: 9.1 *The Mahalanobis Distance*

The *Mahalanobis distance* between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is

$$(\mathbf{x}_i - \mathbf{x}_j)^T \mathsf{Covmat}(\{\mathbf{x}\})^{-1} (\mathbf{x}_i - \mathbf{x}_j)$$

To understand this distance, diagonalize the covariance matrix to get

$$\mathcal{U}^T \mathsf{Covmat}(\{\mathbf{x}\})\mathcal{U} = \Lambda.$$

Because $\mathcal{U}$ is a rotation, it has no effect on distances. If you transform the coordinates to obtain $\mathbf{r}_i = \mathcal{U}\mathbf{x}_i$, the covariance for the $\mathbf{r}_i$ is Λ **exercises** . This is diagonal, so the directions are independent. In the Mahalanobis distance, each direction is scaled by its variance **exercises** . This makes sense – if the "blob" of data is spread out more in one direction, large differences in that direction should not count much when you compute the distance. But in directions where the data does not spread out, even small distances are important.

Dimension reduction

- If you use high dimensional features
  - many small eigenvalues in Covmat
    - general experimental phenomenon
- Issue with Mahalanobis distance
  - divide by small number exaggerates effect
  - generalization problems
- Fix
  - ignore variation in these directions
  - equivalently
    - project onto lower dimension
    - compute Mahalanobis distance in that space

Procedure: 9.2 *Computing Mahalanobis Distance with Dimension Reduction*

At training time: Obtain a large, representative sample of the data items you will work with; write $\{\mathbf{x}\}$ for the dataset of N d-dimensional vectors and $\mathbf{x}_i$ for the i'th item. Write the covariance matrix for this dataset $\mathsf{Covmat}(\{\mathbf{x}\})$. Diagonalize this covariance to obtain

$$\mathcal{U}^T \mathsf{Covmat}(\{\mathbf{x}\})\mathcal{U} = \Lambda.$$

Ensure that the values along the diagonal are sorted. Choose $s < d$ for the new dimension of the data. Form the $s \times d$ matrix $\mathcal{P}_s$ consisting of the first s rows of $\mathcal{U}^T$. Form Λ_s consisting of the $s \times s$ upper left block of Λ. Finally, form $\mathcal{D} = \mathcal{P}_s^T \Lambda_s^{-1} \mathcal{P}_s$.

At run time: The dimension reduced Mahalanobis distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ is

$$(\mathbf{u} - \mathbf{v})^T \mathcal{D} (\mathbf{u} - \mathbf{v})$$

Things to think about

9.1. Single link clustering tends to yield extended clusters; why?

9.2. Complete link clustering tends to yield rounded clusters; why?

9.3. Assume the covariance of some features is diagonal. When you compute a distance function, why does it make sense to scale each direction by the standard deviation?

9.4. Recall $\mathsf{Covmat}(\{\mathbf{x}\})$ must be a symmetric matrix. Show you can diagonalize this matrix by finding its eigenvalues and eigenvectors.