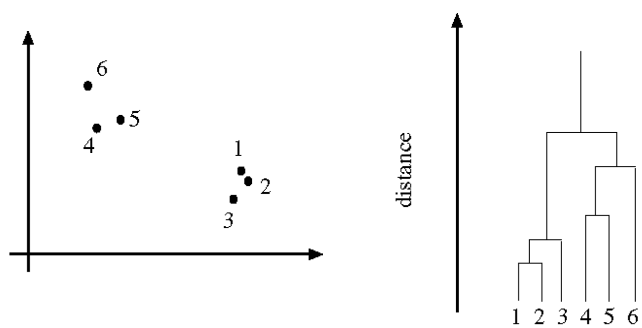
Out of order lecture given by Ranjini due to iPlant travel.

Clustering approaches

- Agglomerative clustering
  - initialize: every item is a cluster
  - attach item that is "closest" to a cluster to that cluster
  - repeat
- Divisive clustering
  - split cluster along best boundary
  - repeat
- Probabilistic clustering
  - Define a probabilistic grouping model

Simple clustering approaches

- Point-Cluster or Cluster-Cluster distance
  - single-link clustering (minimum distance from point to points in clusters or among pairs of points, one from each cluster)
  - complete-link clustering (maximum)
  - group-average clustering (average)
  - (terms are not important, but concepts are worth thinking about)
- Dendrograms
  - classic picture of output as clustering process continues



K-Means

- Choose a fixed number of clusters ("K")
- Choose cluster centers (**means**) and point-cluster allocations (membership) to minimize the error

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

- **x**'s could be any set of features for which we can compute a distance (careful with scaling)

K-Means

- Want to minimize

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

- **Cannot** do this optimization by search, because there are too many possible allocations.
- Standard difficulty which we handle with an iterative process (chicken and egg)

K-Means algorithm (intuition)

- If we know the cluster centers, the best cluster for each point is easy to compute
  - Just compute the distance to each to find the closest
- If we know the best cluster for each point, the cluster centers are also easy to compute
  - Just average the points in each cluster
- Algorithm
  - 1) Guess one of the two.
  - 2) Alternatively re-compute the values for each

K-means flow chart

Choose K

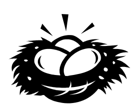
Guess membership

OR

Guess the means

Assume membership is **fixed**. Take averages to get cluster centers (means)

Assume means are **fixed**. Find cluster with closest mean for each point

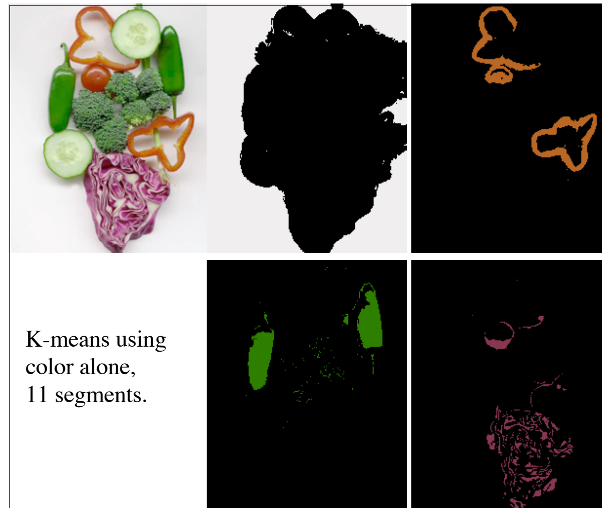


K-means clustering using intensity alone and color alone (Assuming 5 segments, i.e. k=5)

Image

Clusters on color

K-means using color alone, 11 segments



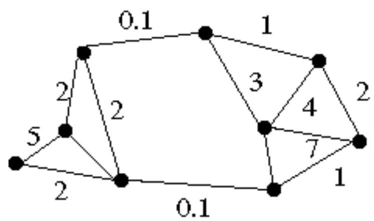
K-means using color alone, 11 segments.

Notes on K-Means

- K-means is “hard” clustering-each point is completely in exactly one cluster
- What you get is a function of starting “guess”
- The error goes down with every iteration
  - This means you get a local minimum
- Unfortunately, the dimension of the space is usually large, and high-dimensional space have lots of local maximum (standard problem!)
  - Dimensionality here is K*dim(**x**)
- Finding the global minimum for a real problem is very optimistic!

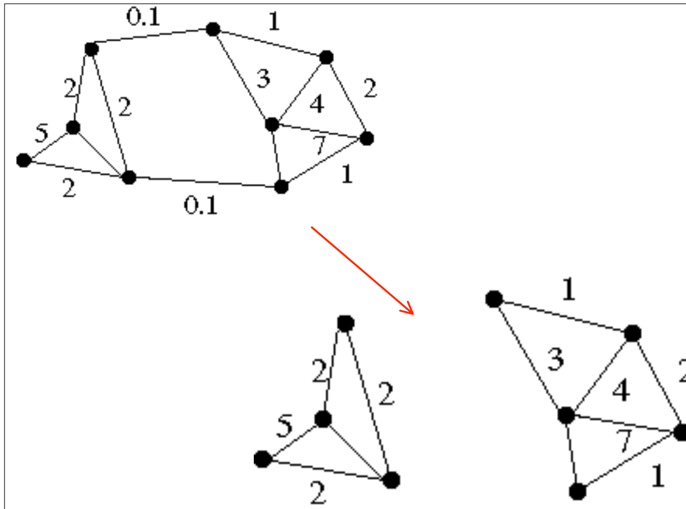
Graph theoretic clustering

- Represent distance between tokens using a weighted graph.
  - affinity matrix
- Cut up this graph to get subgraphs with strong interior links (and weak links between the subgraphs).



Graph for 9 tokens

Image representation of weight matrix



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

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

Texture Descriptor

Eigenvectors and cuts

~~exam~~

- For some cluster*, consider a vector **a** giving the association between each element and that cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- If two elements, i and j, are part of the same cluster, then
  - $\mathbf{a}_i$ and $\mathbf{a}_j$ are both large
  - and the affinity A_{ij} is large
  - thus, $\mathbf{a}_i A_{ij} \mathbf{a}_j$ should be large
- Thus a good cluster is one where $\sum_i \sum_j a_i A_{ij} a_j$ is large.

* In the hand out, **a** is sub-scripted by an "i" for cluster. But this is a distraction. I use "i" for something else below.

Eigenvectors and cuts

~~exam~~

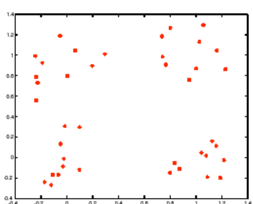
- $\sum_i \sum_j a_i A_{ij} a_j$ should be large for a coherent cluster represented by **a**.
- This suggests maximizing $\mathbf{a}^T A \mathbf{a}$
- But we need the constraint $\mathbf{a}^T \mathbf{a} = 1$ (why?)
  - Arguably it might be more logical to make the sum of the elements of **a** to be one, but the standard (L_2) norm is easier to deal with.

Eigenvectors and cuts

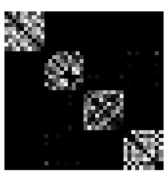
~~exam~~

- We want to maximize $\mathbf{a}^T A \mathbf{a}$ subject to $\mathbf{a}^T \mathbf{a} = 1$
- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue
- This gives the cluster with greatest internal affinity
  - Ideally, most elements of the eigenvalue are near zero, and the others tell us which tokens are in the cluster

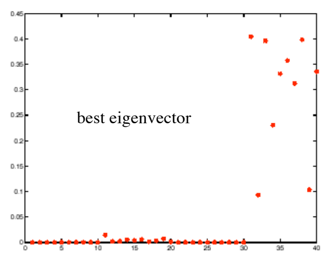
Example eigenvector



points



affinity matrix



(you should know how to interpret these)

Normalized cuts

- Previous criterion evaluates **within** cluster similarity, but does not promote large differences **between** clusters across cluster difference
- N-cuts proposes maximizing the within cluster similarity **compared** to the across cluster difference
- Write graph as V, one cluster as A and the other as B. (V=AUB).
- Maximize

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

- (Solution follows to keep notes self-contained).

Normalized cuts

- Previous criterion evaluates **within** cluster similarity, but does not promote large differences **between** clusters across cluster difference.
- For simplicity, consider the task of splitting the tokens into two groups A and B. The union of the two groups is V.
- N-cuts proposes maximizing the within cluster similarity **compared** to the across cluster difference.
- Define cut(A,B) to be the sum of the weights of the edges that you remove to split up the image.
- Define assoc(A,V) to be the sum of all the weights between elements in A and elements in V.

Normalized cuts

- Two equivalent formulations
- Minimize

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

- Maximize

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

Normalized cuts

Optional

- Let **y** be a vector whose elements are (ideally) 1 if the element is in A, and -b if it's in B.
  - b is theoretically defined for the derivation, but **y** is going to be estimated.
- Write the matrix of the graph as W, and the matrix which has the row sums of W on its diagonal as D. Let **1** be a vector with all ones.
- With some algebra, the criterion becomes $\min_y\left(\frac{y^T(D-W)y}{y^TDy}\right)$
- And we have a constraint $y^TD1=0$
- This is hard to do, because y's values are quantized

Normalized cuts

Optional

- Instead, solve the generalized eigenvalue problem

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

- which gives

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

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

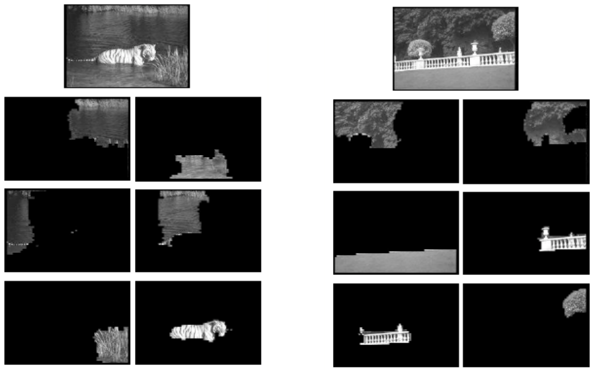


Figure from "Image and video segmentation: the normalised cut framework", by Shi and Malik, copyright IEEE, 1998