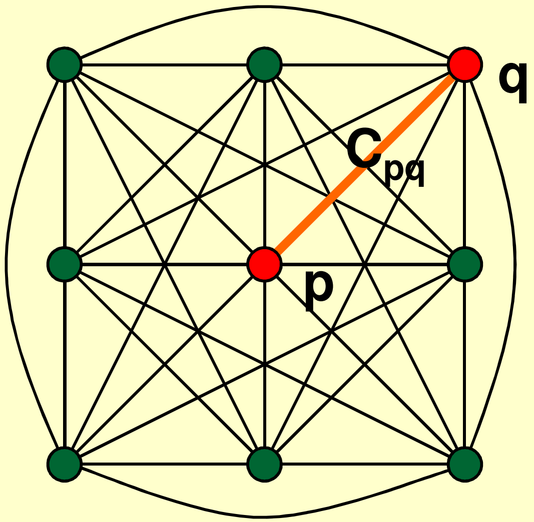
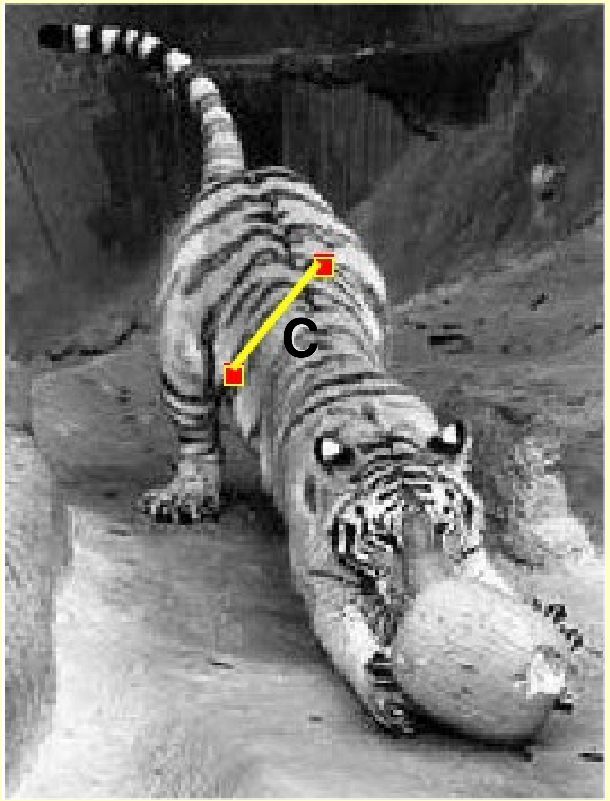
Region based segmentation

Normalized cuts

Normalized cuts

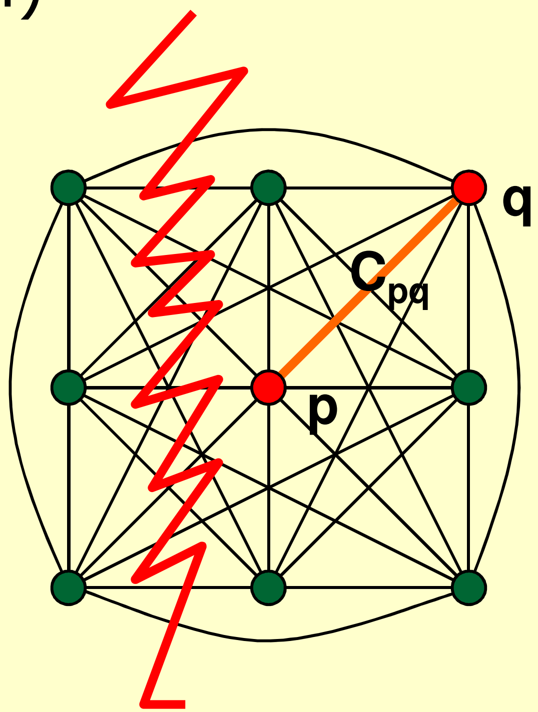
- A different approach to image segmentation develops on the use of graphs
  - The graph is fully connected
  - Nodes represent image pixels or connected patches in the image
  - Edges are labelled with a measure of the similarity between the two nodes they link

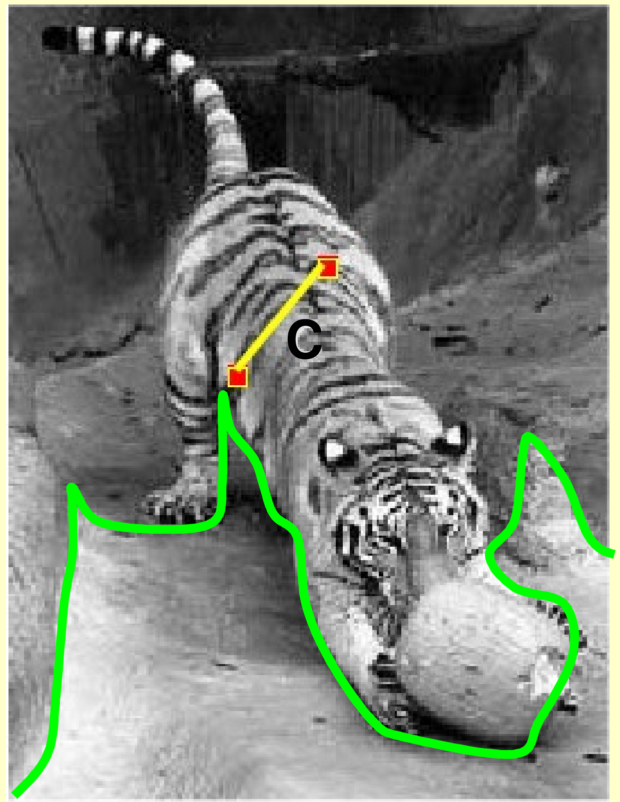




Normalized cuts

- It can be observed that by cutting (removing) some edges the graph can be partitioned into disjoint segments
- Each segment maps into one (or more) sets of pixels in the image (not necessarily one connected region)





Normalized cuts

- Segmentation can be regarded as the problem of identifying a set of **good cuts** that partition the graph into segments such that
  - Similar nodes belong to the same segment
  - Dissimilar nodes belong to different segments

We need to measure the quality of a cut

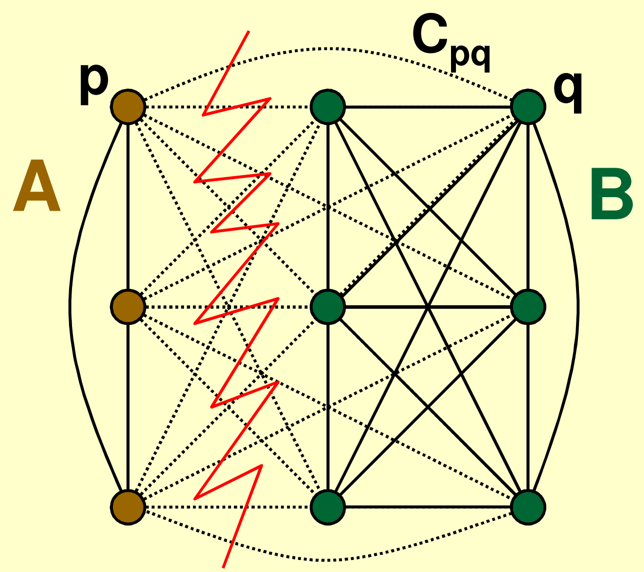
Normalized cuts

- Once a measure of cut quality is available, segmentation can be accomplished by recursively partitioning the graph using the best cut at each iteration step

Normalized cuts

- One cut removes one or more edges from the graph
- Since each edge is associated with a weight (representing the similarity of linked nodes), **a good cut should remove edges with low weight values**
- Thus, a cut that splits the graph into two segments A and B can be associated with a cost measured as:

$$cut(A, B) = \sum_{p \in A,\, q \in B} c_{pq}$$



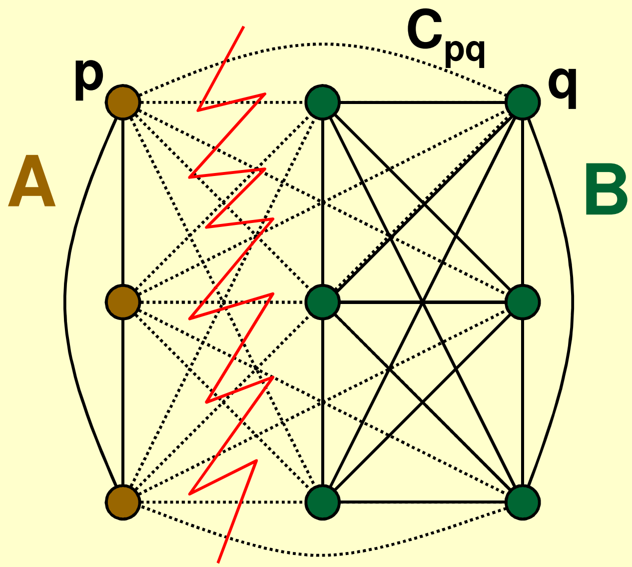
- The best cut would be the cut with minimum cost

Normalized cuts

- However, such an approach priviledges cuts that remove a few edges
  - The lower the number of edges removed the lower cost(A,B)

$$cut(A,B) = \sum_{p \in A,\, q \in B} c_{pq}$$

- In contrast these are not always the best cuts
- In order to adequately estimate the quality of a cut, value of cut(A,B) should be normalized with respect to
  - the sum of all weights associated with edges that have at least one node in A
  - the sum of all weights associated with edges that have at least one node in B

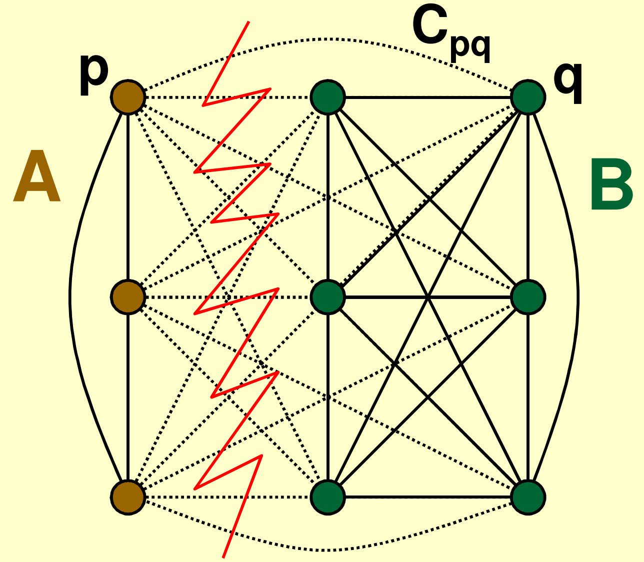


Normalized cuts

- Accordingly, the best cut is the cut that minimizes the normalized cost:

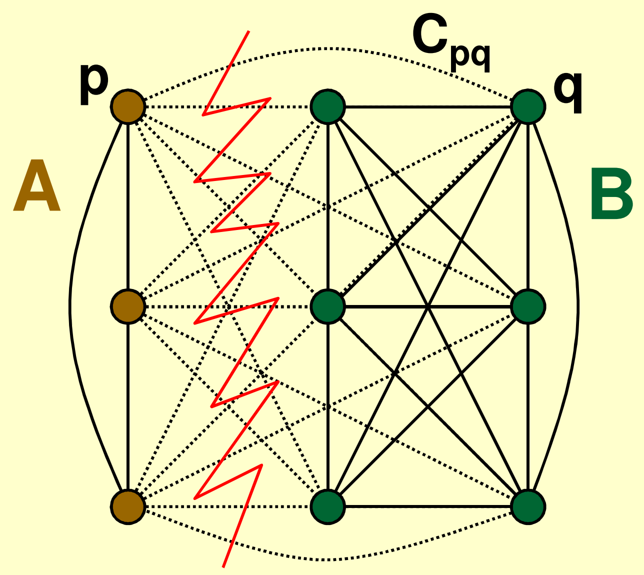
$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

- being:
  - *cut(A,B)* the sum of weights that link one node of A to one node of B
  - *assoc(A,V)* the sum of weights that link one node of A to a generic node of the graph
  - *assoc(B,V)* the sum of weights that link one node of B to a generic node of the graph



Normalized cuts

- It should be observed that the measure cut(A,B) is always less than assoc(A,V) and assoc(B,V)
  - Edges that are removed are just a fraction of the edges that have one end in one segment (either A or B)
- The value assoc(A,V)-cut(A,B) measures the sum of weights of edges linking two nodes of A
- Therefore, the ratio cut(A,B)/assoc(A,V) is low when nodes of A a highly similar with each other and not similar to nodes not in A
  - The same applies to cut(A,B)/assoc(B,V)

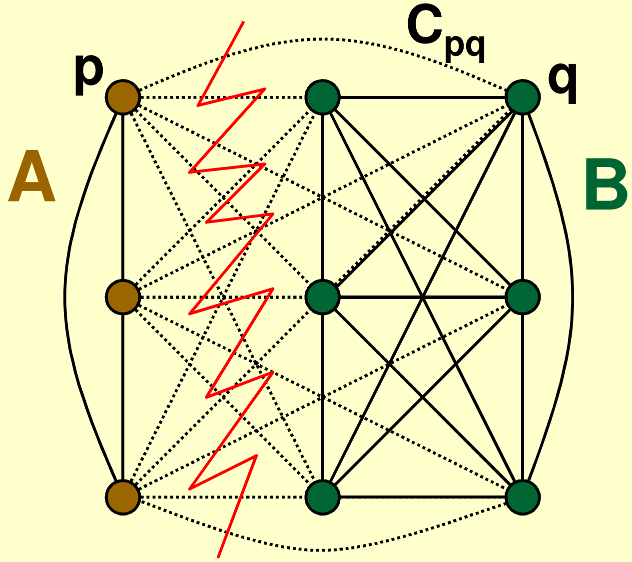


Normalized cuts

- The best cut is the one that minimizes the normalized cost: the **normalized cut**
- It has the property of partitioning the graph into two segments A and B such that
  - The sum of weights of edges between the two segments is low

Problems:

1) How do we compute the weights associated with graph edges?
2) How can we find the best cut efficiently?



Normalized cuts

- In the literature, values of weights associated with graph edges are called **affinity measures**

- The particular form of affinity measures depends on the problem at hand. The following can be considered:
  - Affinity by distance
  - Affinity by intensity
  - Affinity by color
  - Affinity by texture

- In any case, the affinity measure should be large for similar vertices and small for dissimilar ones

Normalized cuts

- Affinity by distance

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

- where σ_d is a parameter that should be large if quite distant points should be grouped and small if only nearby points should be grouped.
- **x** and **y** are vectors identifying the position of two pixels in the image

Normalized cuts

- Affinity by brightness

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

- the larger σ_I the larger is the difference of brightness values of points in the same group
- I(**x**) is the intensity of pixel **x**.

Normalized cuts

- Affinity by color

$$aff(x, y) = \exp\left(-\left(\frac{dist\left(c(x), c(y)\right)^2}{2\sigma_C^2}\right)\right)$$

- the larger σ_C the larger is the allowed color difference between points in the same group
- **c**(**x**) is the color of pixel **x**.

Normalized cuts

- Affinity by texture

$$aff(x, y) = \exp\left(-\left(\frac{\left(f(x) - f(y)\right)^t \left(f(x) - f(y)\right)}{2\sigma_T^2} \right) \right)$$

- the larger σ_T the larger is the allowed difference in terms of texture between points in the same group
- **f**(**x**) is the feature vector representing image texture over a reasonably sized neighborhood of pixel **x**.

Normalized cuts

- Thus, values of affinity measures can be used to build an **affinity matrix** $A=\{a_{ij}\}$
  - a_{ij} representing the similarity between the i-th and j-th pixels (or image patches)
- How can we find the minimum normalized cut?
  - Exploration of all possible solutions (cuts) is computationally untractable
- However…

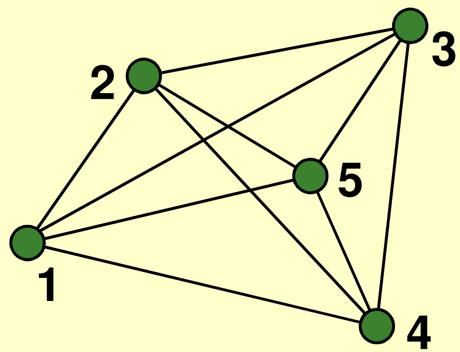
Normalized cuts

- In addition to the **affinity matrix** $A=\{a_{ij}\}$
  - a_{ij} representing the similarity between the i-th and j-th pixels (or image patches)
- Some more variables should be considered:
- A **component vector** ζ
  - This vector represents the cut: it has one element for each graph node. Value of the i-th element is 1 if the i-th node belongs to the first segment; it is –b if the i-th node belongs to the second segment
- The **degree matrix** $D=\{d_{ii}\}$.
  - This is a diagonal matrix, each element d_{ii} being the sum of all weights of edges that start on the i-th node

$$d_{ii} = \sum_j a_{ij}$$

Normalized cuts

- For instance, the following values of variable $\zeta=(1,1,-b,-b,-b)$ partition the graph into two segments A and B as shown below:



Normalized cuts

- It can be shown that the cut (identified by ζ) that minimizes the normalized cost also minimizes the following functional:

$$\frac{\zeta^T (D - \mathrm{A})\zeta}{\zeta^T D \zeta}$$

- The solution ζ should be a vector of discrete elements each one with values in {-b,1}
- This functional is a discrete version of the **Rayleigh quotient**

Normalized cuts

- The problem is tractable if the solution ζ is allowed to take on real values
  - Later on, real values of ζ will be discretized to $\{-b,1\}$ by using a threshold
- Therefore, the issues are:
  - Find the solution with real values
  - Choose the best threshold value

Normalized cuts

- Minimization of the cost function for a vector ζ of real values is equivalent to the solution of the following equation:

$$(D - \mathrm{A})\zeta = \lambda D \zeta$$

- It can be shown that the smallest eigenvalue of *(D-A)* is zero
- Therefore, as a solution to the above equation, the eigenvector corresponding to the second, smallest eigenvalue is chosen

Normalized cuts

- To choose the best threshold value, the following approach can be used:
  - Let n be the number of graph vertices (that is the number of elements of ζ)
  - Let ncut(v) be the cost of performing the cut using v as threshold value
  - The worst case condition corresponds to the case where the elements of ζ have n distinct values. In this case there are at most (n-1) non trivial threshold values (and n-1 ncut(v) values, accordingly)
  - Value of the threshold is set so as to yield the minimum among these n-1 ncut(v) values

Graph based segmentation

- More info about approaches to images segmentation using graphs can be found at:

  http://www.cis.upenn.edu/~jshi/GraphTutorial/