Normalized Cuts and Image Segmentation

Based on a paper by Jianbo Shi and Jitendra Malik

Linear Algebra and its Applications

Gauranga Kumar Baishya (MDS202325)
Hiba AP (MDS202326)
Esha Bhattacharyya (MDS202324)

February 11, 2025

Outline

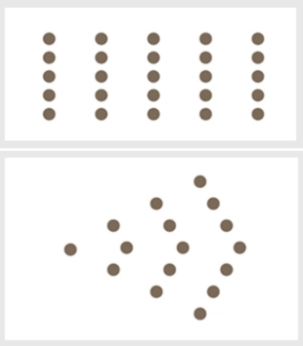
Perceptual grouping in vision

Perceptual grouping in vision

Laws of Organization in Perceptual Forms
Max Wertheimer (1923)

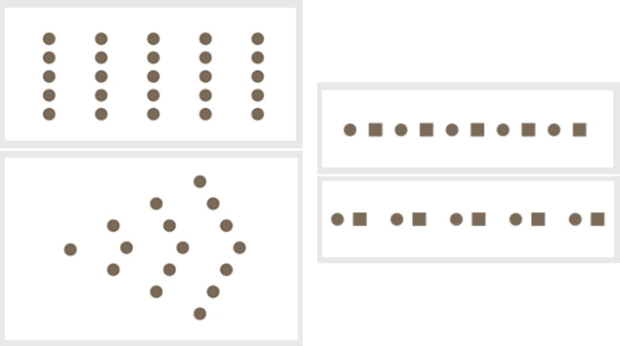
Key Factors

Key Factors

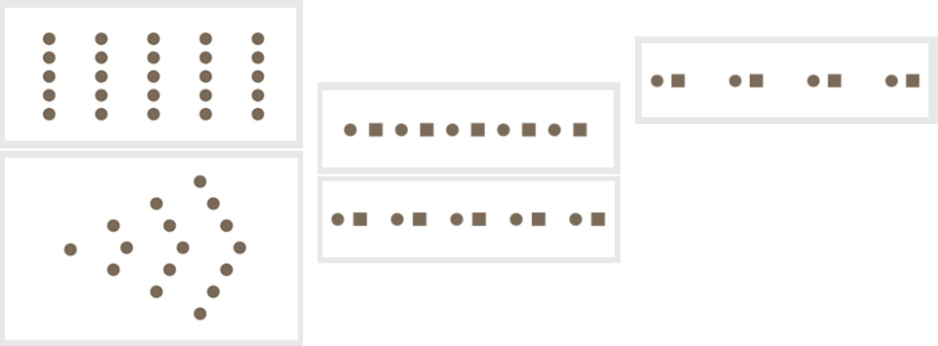


Key Factors

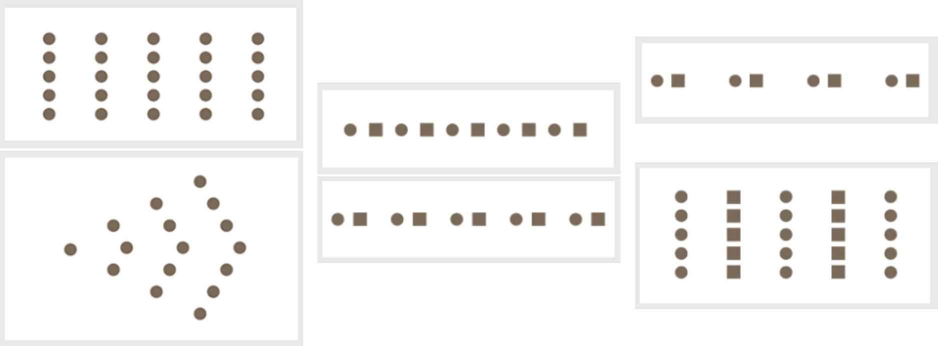
Key Factors



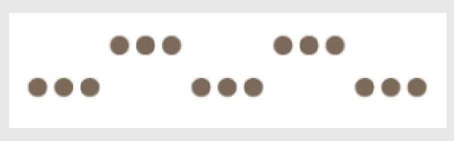
Key Factors



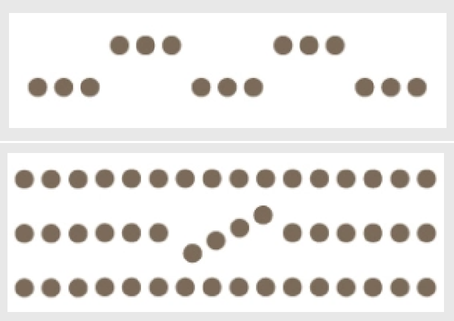
Key Factors



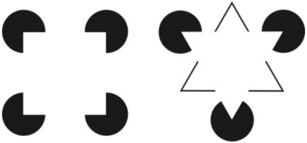
Continued..



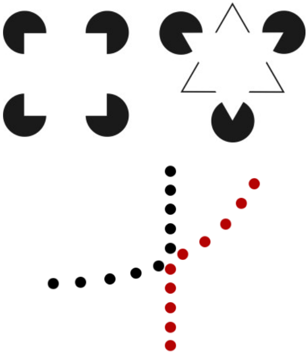
Continued..



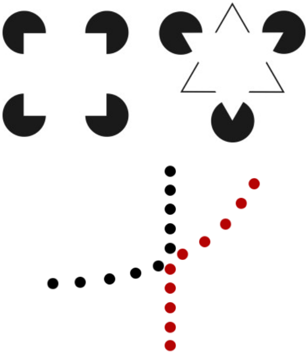
Continued..



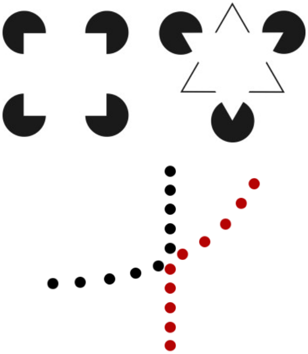
Continued..

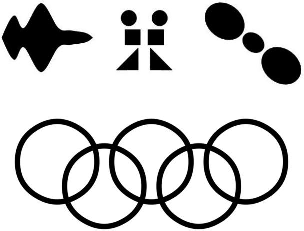


Continued..

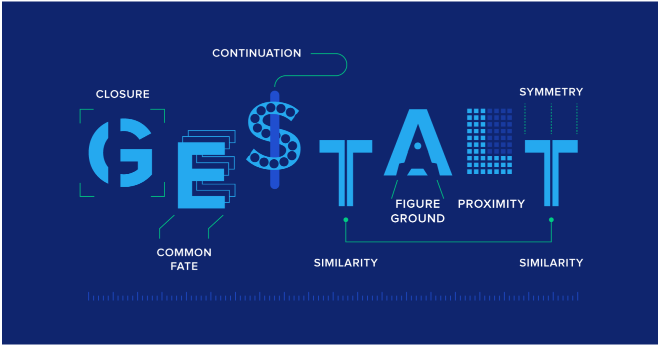


Continued..





Gestalt's Law



Selecting the best partition

Two aspects to consider here -

- First aspect
  - Single correct answer may not exist.
  - Depends on prior world knowledge - Important to specify
  - Could be high level, mid level or low level
  - Low level - brightness, color, texture etc.
    High/mid level - Symmetry, object models etc.
- Second aspect
  - Hierarchy of partitioning
  - Tree structure corresponding to hierarchical partitioning better than a single flat partition

Low level knowledge - Hierarchical partitions.

High level knowledge - Confirm the segments

Graph Theoretic Formulation

$$G = (V, E)$$

- Weighted undirected graph

Graph Theoretic Formulation

$$G = (V, E)$$

- Weighted undirected graph
- V is set of points in feature space

Graph Theoretic Formulation

$$G = (V, E)$$

- Weighted undirected graph
- V is set of points in feature space
- E is set of edges each formed between every node pair

Graph Theoretic Formulation

$$G = (V, E)$$

- Weighted undirected graph
- V is set of points in feature space
- E is set of edges each formed between every node pair

Goal: Partition V into disjoint $V_1, V_2, ..., V_m$ where nodes in a particular V_i are highly similar and across V_i and V_j are highly dissimilar

Graph Theoretic Formulation

$$G = (V, E)$$

- Weighted undirected graph
- V is set of points in feature space
- E is set of edges each formed between every node pair

Goal: Partition V into disjoint $V_1, V_2, ..., V_m$ where nodes in a particular V_i are highly similar and across V_i and V_j are highly dissimilar

1. **What is the precise criterion for a good partition?**

Graph Theoretic Formulation

$$G = (V, E)$$

- Weighted undirected graph
- V is set of points in feature space
- E is set of edges each formed between every node pair

Goal: Partition V into disjoint $V_1, V_2, ..., V_m$ where nodes in a particular V_i are highly similar and across V_i and V_j are highly dissimilar

1. **What is the precise criterion for a good partition?**
2. **How can such a partition be computed efficiently?**

Grouping as Graph Partitioning

Remove edges connecting the nominated partitions (A and B)

Grouping as Graph Partitioning

Remove edges connecting the nominated partitions (A and B)
End Gain: $A, B \subset V, A \cup B = V, A \cap B = \phi$

Grouping as Graph Partitioning

Remove edges connecting the nominated partitions (A and B)
End Gain: $A, B \subset V, A \cup B = V, A \cap B = \phi$
Degree of dissimilarity (assuming edge weight $\propto 1/$distance between nodes)

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, b) \tag{1}$$

Grouping as Graph Partitioning

Remove edges connecting the nominated partitions (A and B)
End Gain: $A, B \subset V, A \cup B = V, A \cap B = \phi$
Degree of dissimilarity (assuming edge weight $\propto 1/\text{distance}$ between nodes)

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, b) \tag{1}$$

Optimal bipartitioning minimizes the cut value

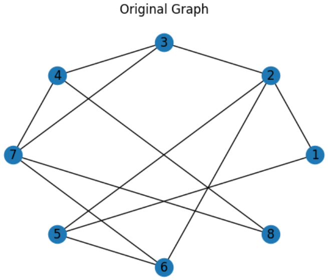
Grouping as Graph Partitioning

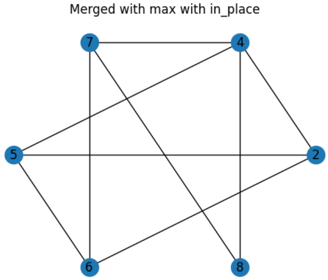
Remove edges connecting the nominated partitions (A and B)
End Gain: $A, B \subset V, A \cup B = V, A \cap B = \phi$
Degree of dissimilarity (assuming edge weight $\propto 1/$distance between nodes)

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, b) \tag{1}$$

Optimal bipartitioning minimizes the cut value
Wu and Leahy proposed a clustering approach based on MinCut

Mincut


Original Graph
3
4
2
7
1
5
8
6


Merged with max with in_place
7
4
5
2
6
8

Mincut

Figure: Original image

Figure: Ncut

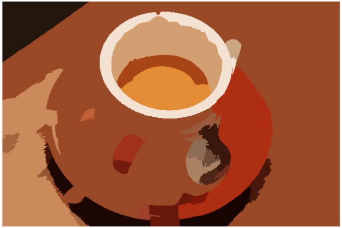
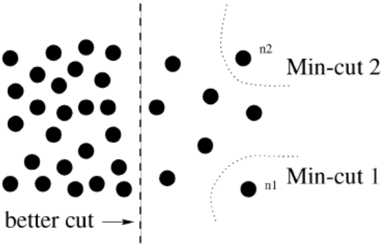


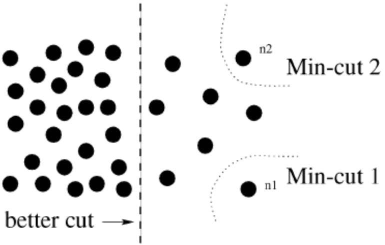
Figure: Min cut

Section

Normalized Cut

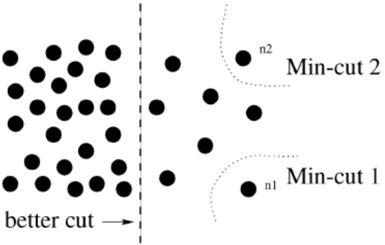


Normalized Cut



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

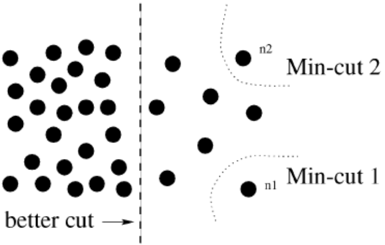
Normalized Cut



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

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

Normalized Cut

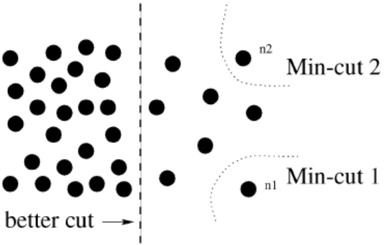


$$NCut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)}$$

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

NCut(A, B)=2-Nassoc(A,B)

Normalized Cut



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

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

NCut(A, B)=2-Nassoc(A,B) $\quad \therefore NCut(A, B) \propto 1/Nassoc(A, B)$ (2)

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$
- D is a NxN diagonal matrix where $D_{ii} = d_i$

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$
- D is a NxN diagonal matrix where $D_{ii} = d_i$
- W is a NxN symmetric matrix with $W(i,j) = w_{ij}$

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$
- D is a NxN diagonal matrix where $D_{ii} = d_i$
- W is a NxN symmetric matrix with $W(i,j) = w_{ij}$
- $k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$ and $b = \frac{k}{(1-k)}$

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = | V |$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$
- D is a NxN diagonal matrix where $D_{ii} = d_i$
- W is a NxN symmetric matrix with $W(i,j) = w_{ij}$
- $k = \dfrac{\sum_{x_i > 0} d_i}{\sum_i d_i}$ and $b = \dfrac{k}{(1-k)}$

$$4NCut(A,B) = \frac{[(\mathbf{1}+x) - b(\mathbf{1}-x)]^T (D-W)[(\mathbf{1}+x) - b(\mathbf{1}-x)]}{b\mathbf{1}^T D\mathbf{1}}$$

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$
- D is a NxN diagonal matrix where $D_{ii} = d_i$
- W is a NxN symmetric matrix with $W(i,j) = w_{ij}$
- $k = \dfrac{\sum_{x_i>0} d_i}{\sum_i d_i}$ and $b = \dfrac{k}{(1-k)}$

$$4NCut(A,B) = \frac{[(\mathbf{1}+x) - b(\mathbf{1}-x)]^T (D-W)[(\mathbf{1}+x) - b(\mathbf{1}-x)]}{b\mathbf{1}^T D\mathbf{1}}$$

setting $y = \dfrac{(1+x) - b(1-x)}{2}$ gives $y^T Dy = b1^T D1$ and $y^T D1 = 0$

Computing the Optimal Partition

(A,B) is a given partition of graph (V, E)

Setup:

- x is $N = \mid V \mid$ dimensional indicator vector
- $d(i) = \sum_j w(i,j)$
- D is a NxN diagonal matrix where $D_{ii} = d_i$
- W is a NxN symmetric matrix with $W(i,j) = w_{ij}$
- $k = \frac{\sum_{x_i > 0} d_i}{\sum_i d_i}$ and $b = \frac{k}{(1-k)}$

$$4NCut(A,B) = \frac{[(\mathbf{1}+x) - b(\mathbf{1}-x)]^T (D-W)[(\mathbf{1}+x) - b(\mathbf{1}-x)]}{b\mathbf{1}^T D\mathbf{1}}$$

setting $y = \frac{(1+x) - b(1-x)}{2}$ gives $y^T D y = b1^T D1$ and $y^T D1 = 0$

Therefore,

$$min_x Ncut(x) = min_y \frac{y^T (D-W) y}{y^T D y}$$

with $y(i) \in \{1, -b\}$ and $y^T D1 = 0$

Continued..

If y is relaxed on real values, we get

$$y_i = arg.min_{y^T D1=0} \frac{y^T(D-W)y}{y^T Dy} \tag{3}$$

Continued..

If y is relaxed on real values, we get

$$y_i = arg.min_{y^T D1=0} \frac{y^T (D - W) y}{y^T D y} \tag{3}$$

We can convert this to $(z = D^{\frac{1}{2}} y)$:

$$z_i = arg.min_{z^T z_0=0} \frac{z^T D^{-1/2} (D - W) D^{-1/2} z}{z^T z} \tag{4}$$

Continued..

If y is relaxed on real values, we get

$$y_i = arg.min_{y^T D1=0} \frac{y^T(D-W)y}{y^T Dy} \tag{3}$$

We can convert this to ($z = D^{\frac{1}{2}} y$):

$$z_i = arg.min_{z^T z_0=0} \frac{z^T D^{-1/2}(D-W)D^{-1/2} z}{z^T z} \tag{4}$$

Theorem

Let A be a real symmetric matrix. Given x is orthogonal to the $j-1$ smallest eigenvectors $x_1, ..., x_{j-1}$, $\frac{x^T Ax}{x^T x}$ is minimized by the next smallest eigenvector x_j and its minimum value is corresponding λ_j.

Continued..

If y is relaxed on real values, we get

$$y_i = arg.min_{y^T D1=0} \frac{y^T(D-W)y}{y^T Dy} \quad (3)$$

We can convert this to ($z = D^{\frac{1}{2}} y$):

$$z_i = arg.min_{z^T z_0=0} \frac{z^T D^{-1/2}(D-W)D^{-1/2} z}{z^T z} \quad (4)$$

Theorem

Let A be a real symmetric matrix. Given x is orthogonal to the $j-1$ smallest eigenvectors $x_1, ..., x_{j-1}$, $\frac{x^T Ax}{x^T x}$ is minimized by the next smallest eigenvector x_j and its minimum value is corresponding λ_j.

Thus the second smallest eigenvector of the eigensystem gives us the real valued solution to our normalized cut problem

Trailing

The generalized eigensystem can be transformed into a standard eigenvalue problem of $D^{-1/2}(D - W)D^{-1/2}x = \lambda x$

Trailing

The generalized eigensystem can be transformed into a standard eigenvalue problem of $D^{-1/2}(D - W)D^{-1/2}x = \lambda x$
Takes $O(n^3)$ operations ($n = \mid V \mid$).
Impractical when n is pixels in a high dimension image!

Trailing

The generalized eigensystem can be transformed into a standard eigenvalue problem of $D^{-1/2}(D - W)D^{-1/2}x = \lambda x$
Takes $O(n^3)$ operations ($n =| V |$).
Impractical when n is pixels in a high dimension image!

- graphs are locally connected and **resulting eigensystems are very sparse**
- only **top few eigenvectors** are needed for partitioning
- precision requirement for eigenvectors is very low, **except the right sign bit**

Trailing

The generalized eigensystem can be transformed into a standard eigenvalue problem of $D^{-1/2}(D-W)D^{-1/2}x = \lambda x$
Takes $O(n^3)$ operations ($n =| V |$).
Impractical when n is pixels in a high dimension image!

- graphs are locally connected and **resulting eigensystems are very sparse**
- only **top few eigenvectors** are needed for partitioning
- precision requirement for eigenvectors is very low, **except the right sign bit**

Lancoz Method
Running time $O(mn) + O(mM(n))$
m - maximum matrix-vector computations
$M(n)$ - cost of a matrix-vector computation of Ax where
$A = D^{-1/2}(D-W)D^{-1/2}$

Trailing

The generalized eigensystem can be transformed into a standard eigenvalue problem of $D^{-1/2}(D - W)D^{-1/2}x = \lambda x$
Takes $O(n^3)$ operations ($n =| V |$).
Impractical when n is pixels in a high dimension image!

- graphs are locally connected and **resulting eigensystems are very sparse**
- only **top few eigenvectors** are needed for partitioning
- precision requirement for eigenvectors is very low, **except the right sign bit**

Lancoz Method
Running time $O(mn) + O(mM(n))$
m - maximum matrix-vector computations
$M(n)$ - cost of a matrix-vector computation of Ax where
$A = D^{-1/2}(D - W)D^{-1/2}$
W is sparse $\rightarrow$ A is sparse $\rightarrow$ **matrix vector computation is** $O(n)$

Creating and stabilizing the partition

Our eigenvectors take continuous values

Creating and stabilizing the partition

Our eigenvectors take continuous values
Splitting point of partitioning is needed.

Creating and stabilizing the partition

Our eigenvectors take continuous values
Splitting point of partitioning is needed.

1. Zero or median may be possible candidate
2. Check l evenly spaced splitting points and compute the best $NCut$ among them

Approach for stabilizing:

Creating and stabilizing the partition

Our eigenvectors take continuous values
Splitting point of partitioning is needed.

1. Zero or median may be possible candidate
2. Check l evenly spaced splitting points and compute the best $NCut$ among them

Approach for stabilizing:
Ignore eigenvectors having smoothly varying eigenvector values

Section

Implementation

Images : Matrices of pixel values (Channels)

- Range of pixel values: 0 - 255
- Black and white Images - 1 channel
- Colored images - 3 channels(RGB)

Scikit-image : Image processing library in python

- Has algorithms for image segmentation(including Ncut)

Ncut function in scikit-image library -

- Based on the paper we are discussing
- Creates a Region Adjacency Graph (RAG)
- Recursively performs a Normalized Cut on the RAG

Implementation

For our implementation, we use a 3468 x 4624 pixels image.

Figure: Original image

Implementation

We then tried segmentation using different number of segments, the output for which is as follows -

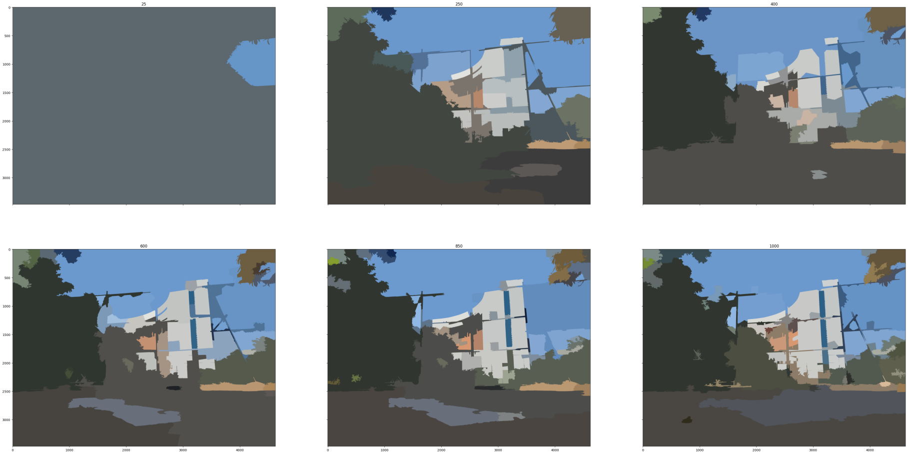


Figure: Segmented images

Implementation

Comparing the original image and a segmented image -

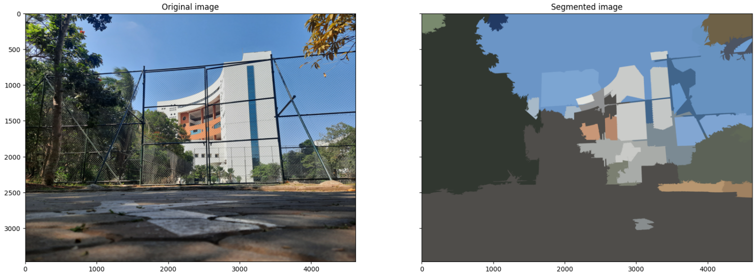


Figure: Comparision

The full code for the image segmentation performed above can be found here - `https://github.com/Swastikamohapatra/Normalized-cut-Image-segmentation`

Section

1. Introduction
2. Normalized Cut
3. Implementation
4. Comparision
5. Conclusion

Comparision

The normalized cut formulation has a certain resemblance to the average cut, as well as the average association formulation. All three of these algorithms can be reduced to solving certain eigenvalue systems.

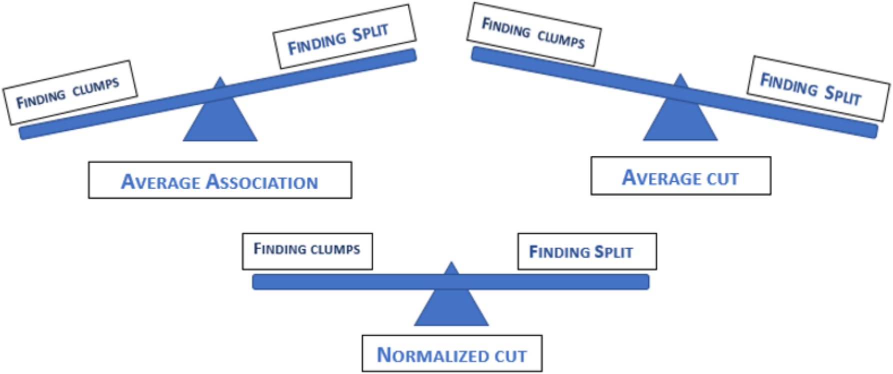


Figure: Comparision

Illustration

To illustrate these above said points, we consider a set of randomly distributed data in 1 Dimension. The 1 Dimensional data is made up of two subsets of points:-

- The first 20 points are randomly distributed from 0 to 0.5.
- The remaining 12 points are randomly distributed from 0.65 to 1

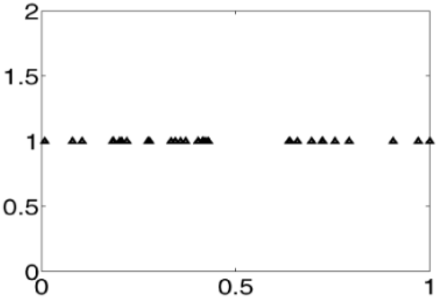


Figure: 1D Data set

Graph

Making a Weighted Graph out of the data set:

- Node = Data Points
- weight = Inversely Proportional to the distance between two Nodes
- Three Monotonically decreasing Weight function with different rate of fall-off:
  - $w(x) = e^{(-d(x)/-0.1)^2}$
  - $w(x) = 1 - d(x)$
  - $w(x) = e^{(-d(x)/-0.2)}$

  Where $d(x)$ is a distance function

First Function

First Function: $w(x) = e^{(-d(x)/-0.1)^2}$

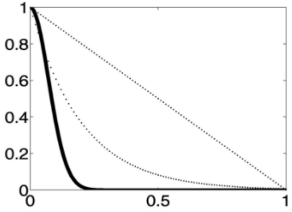


Figure: Weight function

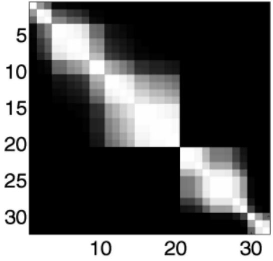
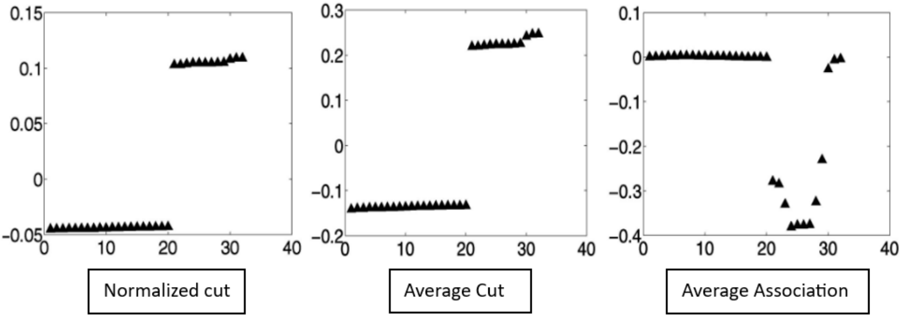


Figure: Graph weight matrix

Note:-

- This function has the fastest decreasing rate among the three.
- Weight is represented by brightness(i.e. higher the weight higher the brightness and vice versa.
- With this weight function, only close-by points are connected.

First Function



Findings/Observation:-

- Using the second extreme eigenvector, both Normalized and Average Cut partitioned the data point into two clusters which is the true situation
- Average Association fails to do the right clustering. It partitioned the data point into isolated small clusters because of its bias towards finding "tight" clusters.

Second Function

Second Function: $w(x) = 1 - d(x)$

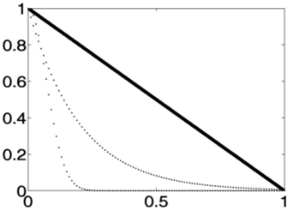


Figure: Weight function

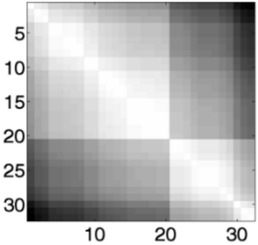
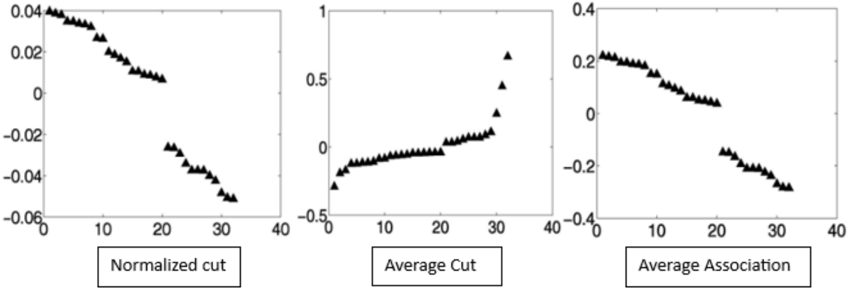


Figure: Graph weight matrix

Note:-

- This function has the slowest decreasing rate among the three.
- With this weight function, most points have non trivial connection to the rest.

Second Function



Findings/Observation:-

- Normalized Cut gives the right partition.
- Average Association also gives the right partition because it easily finds the two the two tight cluster by eliminating few edges with heavy weight across the two clusters.
- Average cut fails because the cluster on the right has less within-group similarity comparision with the cluster on the left. Thus average has trouble deciding on where to cut.

Third Function

Third Function: $w(x) = e^{(-d(x)/-0.2)}$

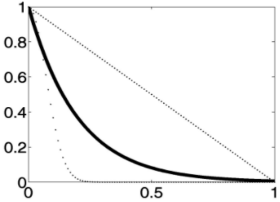


Figure: Weight function

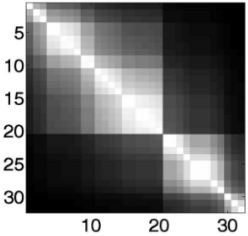
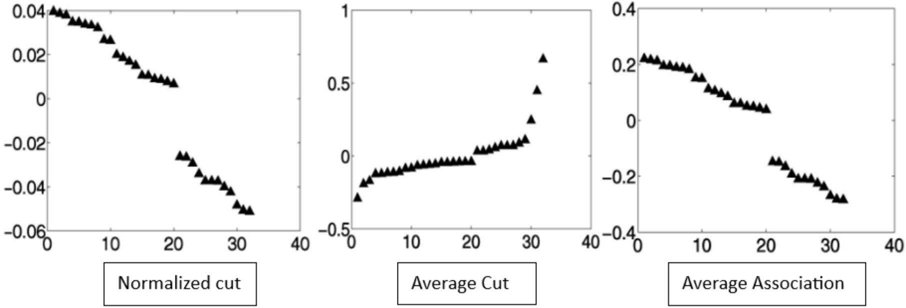


Figure: Graph weight matrix

Note:-

- This function has the moderate decreasing rate among the three.
- with this weight function, the nearby-point connections are balanced against far-away point connections. .

Third Function



Findings/Observation:-

- Normalized cut produces a more clearer solution than the other two because it somehow balance the goal of “Clustering” and “Segmentation”.

Hence, Normalized cut performs well for all these three situation.

Section

Conclusion

- Finally, we got a grouping algorithm that focuses on perceptual grouping and aims to extract the global impression of a picture.
- Minimizing Normalized Cut(unbiased measure of disassociation between subgroups of a graph) leads to directly maximizing the Normalized Association(unbiased measure for total association within the subgroups).
  $Ncut(A, B) = 2 - Nassoc(A, B)$
- Converting the problem of computing the minimum Normalized cut into a problem of solving a generalized eigenvalue system, makes the algorithm more efficient.

Thank You