Lecture 15

Edges and segmentation
From Pixels to Perception: Mid-level operations of Segmentation and Grouping
Figure / Ground
Finding groups of pixels that go together (parts, objects, textures, holes)

Predicted scene categories:
forest - broadleaf (0.498), swimming hole (0.402), bayou (0.062)

Predicted scene categories:
forest - broadleaf (0.979)
Figure / Ground

A “simple” segmentation problem
It can get a lot harder

Segmentation is a global process

What are the occluded numbers?
Segmentation is a global process

What are the occluded numbers?

Occlusion is an important cue in grouping.
... but not too global
Groupings by Invisible Completions

A

B

C

D

* Images from Steve Lehar’s Gestalt papers
Emergence

http://en.wikipedia.org/wiki/Gestalt_psychology
Perceptual organization

“...the processes by which the bits and pieces of visual information that are available in the retinal image are structured into the larger units of perceived objects and their interrelations”

Stephen E. Palmer, Vision Science, 1999
Gestalt principles

There are hundreds of different grouping laws
Not grouped

Proximity

Similarity

Similarity

Common Fate

Common Region
Parallelism

Symmetry

Continuity

Closure

Familiar configuration
I. Edges
What is an edge?

Depth discontinuity
Material change
Texture boundary

Paint

surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity
Finding edges: Computing derivatives

- **Image**
- **Intensity function (along horizontal scanline)**
- **First derivative**

Edges correspond to extrema of derivative.
Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
1: Filter Image with derivatives of Gaussian
2D edge detection filters

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

\[ h_x(x, y) = \frac{\partial h(x, y)}{\partial x} = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

\[ h_y(x, y) = \frac{\partial h(x, y)}{\partial x} = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]
Gaussian filters

\[ G_\sigma = \frac{1}{2\pi\sigma^2}e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

Convolution with self is another Gaussian

\[
\begin{align*}
\begin{array}{c}
\sigma = 1 \text{ pixel} \\
\sigma = 5 \text{ pixels} \\
\sigma = 10 \text{ pixels} \\
\sigma = 30 \text{ pixels}
\end{array}
\end{align*}
\]

- Convolving two times with Gaussian kernel of width \( \sigma = \text{convolving once with kernel of width } \sigma \sqrt{2} \)
1: Filter Image with derivatives of Gaussian
2D edge detection filters

1 pixel 3 pixels 7 pixels

Smoothing filters with different scales
The Sobel Operator: A common approximation of derivative of Gaussian

- Common approximation of derivative of Gaussian

$$s_x = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term is needed to get the right gradient value
Canny edge detector

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

```matlab
edge(image,'canny')
```
2: Gradient: Find edge strength (magnitude) and direction (angle) of gradient

\[ h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

\[ h_y(x,y) = \frac{\partial h(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

Magnitude: \( h_x(x,y)^2 + h_y(x,y)^2 \) \quad \text{Edge strength}

Angle: \( \arctan\left(\frac{h_y(x,y)}{h_x(x,y)}\right) \) \quad \text{Edge normal}
**Image Gradient**: gradient points in the direction of most rapid increase in intensity

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]

\[ \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]

Can think of it as the slope of a 3D surface

Gradient at a single point \((x,y)\) is a vector:
- **Direction** is the direction of maximum slope:
  \[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
- **Length** is the magnitude (steepness) of the slope
  \[ ||\nabla f|| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Issues:

1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trails; how do we identify the significant points?
3) How do we link the relevant points up into curves?
4) Noise.

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.
Canny edge detector

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
**Goal**: mark points along the curve where the magnitude is biggest.

**How**? looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues:
- at which point is the maximum
- where is the next one?

Forsyth, 2002
Non maximum suppression: check if pixel is local maximum along gradient direction

At $q$, we have a maximum (1) if the value is larger than those at both $p$ and at $r$. Interpolate between $p$ and $r$ to get these values.

Predicting the next edge point: Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either $r$ or $s$).
Examples: Non-Maximum Suppression

Original image

Gradient magnitude

But some edges are broken

Non-maxima Suppressed
(remaining pixels are the local Maximum)
Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
Closing edge gaps

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use hysteresis
- use a high threshold to start edge curves and a low threshold to continue them.
Example: Canny Edge Detection

Original image

Strong edges only

Strong + connected weak edges

gap is gone

Weak edges

courtesy of G. Loy
Example: Canny Edge Detection
Example: Canny Edge Detection
Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues

David R. Martin, Member, IEEE, Charless C. Fowlkes, and Jitendra Malik, Member, IEEE

Abstract—The goal of this work is to accurately detect and localize boundaries in natural scenes using local image measurements. We formulate features that respond to characteristic changes in brightness, color, and texture associated with natural boundaries. In order to combine the information from these features in an optimal way, we train a classifier using human labeled images as ground truth. The output of this classifier provides the posterior probability of a boundary at each image location and orientation. We present precision-recall curves showing that the resulting detector significantly outperforms existing approaches. Our two main results are 1) that cue combination can be performed adequately with a simple linear model and 2) that a proper, explicit treatment of texture is required to detect boundaries in natural images.
Fig. 3. Two Decades of Boundary Detection. The performance of our boundary detector compared to classical boundary detection methods and to the human subjects’ performance. A precision-recall curve is shown for each of five boundary detectors: 1) Gaussian derivative (GD), 2) Gaussian derivative with hysteresis thresholding (GD+H), the Canny detector, 3) A detector based on the second moment matrix (2MM), 4) our gray-scale detector that combines brightness and texture (BG+TG), and 5) our color detector that combines brightness, color, and texture (BG+CG+TG). Each detector is represented by its precision-recall curve, which measures the trade off between accuracy and noise as the detector’s threshold varies. Shown in the caption is each curve’s F-measure, valued from zero to one. The F-measure is a summary statistic for a precision-recall curve. The points marked by a “+” on the plot show the precision and recall of each ground truth human segmentation when compared to the other humans. The median F-measure for the human subjects is 0.80. The solid curve shows the F=0.80 curve, representing the frontier of human performance for this task.
Slides credit: Jitendra Malik
DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection

Submitted on 2 Dec 2014

Gedas Bertasius
University of Pennsylvania
gberta@seas.upenn.edu

Jianbo Shi
University of Pennsylvania
jshi@seas.upenn.edu

Lorenzo Torresani
Dartmouth College
lt@dartmouth.edu

Sample patches on canny edges

https://arxiv.org/abs/1412.1123
DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection

Submitted on 2 Dec 2014

Gedas Bertasius
University of Pennsylvania
gberta@seas.upenn.edu

Jianbo Shi
University of Pennsylvania
jshi@seas.upenn.edu

Lorenzo Torresani
Dartmouth College
lt@dartmouth.edu

https://arxiv.org/abs/1412.1123
Holistically-Nested Edge Detection

Saining Xie  
Dept. of CSE and Dept. of CogSci  
University of California, San Diego  
9500 Gilman Drive, La Jolla, CA 92093  
s9xie@eng.ucsd.edu

Zhuowen Tu  
Dept. of CogSci and Dept. of CSE  
University of California, San Diego  
9500 Gilman Drive, La Jolla, CA 92093  
ztu@ucsd.edu

(a) original image  
(b) ground truth  
(c) HED: output

(d) HED: side output 2  
(e) HED: side output 3  
(f) HED: side output 4

(g) Canny: $\sigma = 2$  
(h) Canny: $\sigma = 4$  
(i) Canny: $\sigma = 8$
CASENet: Deep Category-Aware Semantic Edge Detection

Zhiding Yu*
Carnegie Mellon University
yzhiding@andrew.cmu.edu

Chen Feng*  Ming-Yu Liu†  Srikumar Ramalingam†
Mitsubishi Electric Research Laboratories (MERL)
cfeng@merl.com, mingyul@nvidia.com, srikumar@cs.utah.edu

(a) Input image  (b) Ground truth

(c) CASENet output

II. Segmentation
II.1 Bottom-up segmentation

• Group together similar-looking pixels
  – “Bottom-up” process
  – Unsupervised

• Bottom-up segmentation
  – Clustering
  – Mean shift
  – Graph-based

“superpixels”
Issues

• How do we decide that two pixels are likely to belong to the same region?

• How many regions are there?
Method 1: Clustering

- Cluster similar pixels (features) together

Source: K. Grauman
Segmentation as clustering

• Cluster together (pixels, tokens, etc.) that belong together…

• Agglomerative clustering
  – attach closest to cluster it is closest to
  – repeat

• Divisive clustering
  – split cluster along best boundary
  – repeat

• Dendrograms
  – yield a picture of output as clustering process continues

Chapter – Forsyth & Ponce
A simple segmentation algorithm

- Each pixel is described by a vector
  \[ z = [r, g, b] \text{ or } [Y, u, v], \ldots \]

- Run a clustering algorithm (e.g. k-means) using some distance between pixels:
  \[ D(\text{pixel}_i, \text{pixel}_j) = \| z_i - z_j \|^2 \]
Dendrogram

Data set

Dendrogram obtained by agglomerative clustering
A Dendrogram Shows How the Clusters are Merged Hierarchically

Decompose data objects into several levels of nested partitioning (tree of clusters), called a dendrogram.

A clustering of the data objects is obtained by cutting the dendrogram at the desired level. Then each connected component forms a cluster.
**K-Means Clustering**

- Given $k$, the *k-means* algorithm consists of four steps:
  
  - Select initial centroids at random.
  - Assign each object to the cluster with the nearest centroid.
  - Compute each centroid as the mean of the objects assigned to it.
  - Repeat previous 2 steps until no change.
• K-means (k=5) clustering based on intensity (middle) or color (right) is essentially vector quantization of the image attributes
  – Clusters don’t have to be spatially coherent

See pdf chapter 14

each pixel is replaced with the mean value of its cluster
K-means using color alone (k=11 clusters) 
Showing 4 of the segments, (not necessarily connected) 
Some are good, some meaningless
Including spatial relationships

Augment data to be clustered with spatial coordinates.

\[ z = \begin{pmatrix} Y \\ u \\ v \\ x \\ y \end{pmatrix} \]

- color coordinates (or r,g,b)
- spatial coordinates

- Cluster similar pixels (features) together

\[
\begin{array}{c}
\text{R}=0 \\
\text{G}=200 \\
\text{B}=20 \\
\text{X}=30 \\
\text{Y}=20
\end{array}
\]

\[
\begin{array}{c}
\text{R}=15 \\
\text{G}=189 \\
\text{B}=2 \\
\text{X}=20 \\
\text{Y}=400
\end{array}
\]

\[
\begin{array}{c}
\text{R}=3 \\
\text{G}=12 \\
\text{B}=2 \\
\text{X}=100 \\
\text{Y}=200
\end{array}
\]
• Clustering based on \((r,g,b,x,y)\) values enforces more spatial coherence

K-means using colour and position, 20 segments

Still misses goal of perceptually pleasing or useful segmentation
No measure of texture

Hard to pick \(K\)...
K-Means for segmentation

• Pros
  – Very simple method
  – Converges to a local minimum of the error function

• Cons
  – Memory-intensive
  – Need to pick K
  – Sensitive to initialization
  – Sensitive to outliers
Method 2: Mean shift clustering

- An advanced and versatile technique for clustering-based segmentation

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

The mean shift algorithm seeks *modes* or local maxima of density in the feature space.
Mean Shift Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the "mode" or point of highest density of a data distribution:

Two issues:
(1) Kernel to interpolate density based on sample positions.
(2) Gradient ascent to mode.
Mean shift clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode
Mean Shift Segmentation

1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same “peak” or mode.
5. The data these merged windows traversed are clustered together.

Pixels in L*u* space

Corresponding trajectories with peaks marked as red dots
Apply mean shift jointly in the image (left col.) and range (right col.) domains.

1. Window in image domain

2. Intensities of pixels within image domain window
   - From 0 to 1

3. Window in range domain

4. Center of mass of pixels within both image and range domain windows

5. Window in image domain

6. Center of mass of pixels within both image and range domain windows

7. Window in range domain
Mean Shift color & spatial Segmentation

Results:
Mean shift pros and cons

• Pros
  – Clusters are places where data points tend to be close together
  – Just a single parameter (window size)
  – Finds variable number of modes
  – Robust to outliers

• Cons
  – Output depends on window size
  – Computationally expensive
  – Does not scale well with dimension of feature space

Slide credit: S. Lazebnik
Method 3: Graph-Theoretic Image Segmentation

Build a **weighted graph** $G=(V,E)$ from image

- $V$: image pixels
- $E$: connections between pairs of nearby pixels

$W_{ij}$: probability that $i$ & $j$ belong to the same region

Segmentation = graph partition
Segmentation by graph cut

- Fully connected graph (node for every pixel $i,j$)
- Edge/link between every pair of pixels: $p,q$
- Each edge is weighted by the *affinity* or similarity of the two nodes:
  - cost $c_{pq}$ for each link: $c_{pq}$ measures similarity (or affinity)
  - similarity is *inversely proportional* to difference in color and position
Segmentation by graph cut

- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (similarity or affinity)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: S. Seitz
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph

2b- Build a similarity/affinity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
We represent each pixel by a feature vector $\mathbf{x}$, and define a distance function appropriate for this feature representation (e.g. Euclidean distance). Features can be brightness value, color—RGB, L*$u*$v; texton histogram, etc—and calculate distances between vectors (e.g. Euclidean distance).
Computing distance

• We represent each pixel by a feature vector \( \mathbf{x} \), and define a distance function appropriate for this feature representation.

• Then we can convert the distance between two feature vectors into an affinity/similarity measure with the help of a generalized Gaussian kernel:

\[
\exp\left( -\frac{1}{2\sigma^2} \text{dist}(\mathbf{x}_i, \mathbf{x}_j)^2 \right)
\]
Affinity between pixels

Similarities among pixel descriptors

\[ W_{ij} = \exp\left(-\| z_i - z_j \|^2 / \sigma^2 \right) \]

\( \sigma = \text{Scale factor…} \)

it will hunt us later
Affinity between pixels

Similarities among pixel descriptors

\[ W_{ij} = \exp\left(-\frac{|| z_i - z_j ||^2}{\sigma^2}\right) \]

Interleaving edges

\[ W_{ij} = 1 - \max_{Pb} \]

Line between i and j

With Pb = probability of boundary

\( \sigma = \) Scale factor… it will hunt us later
Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph
2b- Build a similarity/affinity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
2a- What is a graph?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a graph?

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
2a- What is a weighted graph?

**Affinity Matrix** represents the weighted links

![Diagram of a weighted graph with labeled points and edges with weights.]

Diagonal: each point with itself is 1
Strong links/edges
Weak links/edges
No links/edges connected

\[
W = \begin{bmatrix}
1 & .1 & .3 & 0 & 0 \\
0 & 1 & .4 & 0 & .2 \\
.3 & .4 & 1 & .6 & .7 \\
0 & 0 & .6 & 1 & 1 \\
0 & .2 & .7 & 1 & 1 \\
\end{bmatrix}
\]

\(W_{ij}:\) probability that \(i\) & \(j\) belong to the same region

\(i, j\) are the pixels in the image

See Forsyth-Ponce chapter
Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

E.g. Gaussian kernel similarity function

$$W_{ij} = e^{-\frac{(x_i - x_j)^2}{2\sigma^2}}$$

Controls size of neighborhood

See Forsyth-Ponce chapter
We can do segmentation by finding the minimum cut in a graph.

A weighted graph

Weight matrix associated with the graph (larger values are lighter)

A cut of the graph: two tightly linked components. This cut decomposes the graph’s matrix into two main blocks on the diagonal.

We can do segmentation by finding the *minimum cut* in a graph.
Graph terminology

- **Similarity matrix**: $W = [w_{i,j}]$
  
  $$w_{i,j} = e^{-\frac{||X(i) - X(j)||^2}{2\sigma_X^2}}$$
Scale affects affinity

\[ W_{ij} = \exp(-||z_i - z_j||^2 / \sigma^2) \]

- **Small \( \sigma \): group only nearby points
- **Large \( \sigma \): group far-away points

Dataset of 4 groups
of 10 points drawn
from a normal
distribution with four
different means

See Forsyth-Ponce chapter
Affinity matrix of a natural image

Similarity of image pixels to selected pixel
Brighter means more similar
Graph terminology

- Degree of node:

\[ d_i = \sum_j w_{i,j} \]
Graph terminology

• Volume of set:

\[ \text{vol}(A) = \sum_{i \in A} d_i, A \subseteq V \]
Graph terminology

Cuts in a graph:

$$\text{cut}(A, \overline{A}) = \sum_{i \in A, j \in \overline{A}} w_{i,j}$$
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i,X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph

2b- Build a similarity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
Spectral Clustering

Data

Similarities

Affinity Matrix

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University
Spectral clustering: Using Eigenvalues of the matrix

- **spectral clustering** uses the eigenvalues of the similarity/affinity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions.

\[ w(i,j) \rightarrow \text{distance node } i \text{ to node } j \]
An ideal case

Affinity Matrix

$$W = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}$$

What are the eigenvectors of this matrix?
An ideal case

Affinity Matrix

\[ W = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 \\
\end{bmatrix} \]

Eigenvectors:

\[ \lambda = 4 \quad \lambda = 2 \]

On Spectral Clustering: Analysis and an algorithm. Andrew Y. Ng, Michael I. Jordan, Yair Weiss, NIPS 2001
An ideal case

But we do not know the ordering, so $W$ with have some random permutation:
An ideal case

But we do not know the ordering, so $W$ with have some random permutation:

$$W = \begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 \\
\end{bmatrix}$$

Affinity Matrix

Eigenvectors:

$$\lambda = 4 \quad \lambda = 2$$

On Spectral Clustering: Analysis and an algorithm. Andrew Y. Ng, Michael I. Jordan, Yair Weiss, NIPS 2001
What are eigenvectors?

**Eigenvectors** represent the dimensions of data

**Eigenvalues** are the length of eigenvectors

In a case of two variables, Eigenvectors are the two lines drawn in the scatterplot.

No relationship between variables:

\[
\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}} = 1
\]

A linear relationship between variables:

\[
\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}} = \infty
\]
Eigenvectors example

1\textsuperscript{st} Eigenvector is the all ones vector \textbf{1} (if graph is connected)

- 2\textsuperscript{nd} Eigenvector thresholded at 0 separates first two clusters from last two
- \textit{k}-means clustering of the 4 eigenvectors identifies all 4 clusters
Spectral Clustering pipeline

\[ w_{i,j} = e^{-\frac{|x_i - x_j|^2}{\sigma^2}} \]

Data are projected into a lower-dimensional space (spectral/eigenvector domain) where they are easily separable.

Given number \( k \) of clusters, compute the first \( k \) eigenvectors, \( V_1, \ldots, V_k \) of the affinity matrix \( M \). Build the matrix \( V \) with the eigenvectors as columns. Interpret the rows of \( V \) as new data points \( Z_i \). Cluster the points \( Z_i \) with the k-means algorithms.

Dimensionality reduction \( n \times n \rightarrow n \times k \)
Eigenvectors and blocks

• Block weight matrices have block eigenvectors:

\[
\begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
\lambda_1 = 2 \\
\lambda_2 = 2 \\
\lambda_3 = 0 \\
\lambda_4 = 0 \\
\end{pmatrix}
\]

• Near-block matrices have near-block eigenvectors:

\[
\begin{pmatrix}
1 & 1 & .2 & 0 \\
1 & 1 & 0 & -.2 \\
.2 & 0 & 1 & 1 \\
0 & -.2 & 1 & 1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
\lambda_1 = 2.02 \\
\lambda_2 = 2.02 \\
\lambda_3 = -0.02 \\
\lambda_4 = -0.02 \\
\end{pmatrix}
\]

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University
Spectral Space

Can put items into blocks by eigenvectors:

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>.2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>.2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Clusters clear regardless of row ordering:

<table>
<thead>
<tr>
<th>1</th>
<th>.2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>1</td>
</tr>
</tbody>
</table>
Example eigenvector

The eigenvector corresponding to the largest eigenvalue of the affinity matrix. Most values are small, but some, corresponding to the elements of the main cluster, are large.

The 3 next eigenvectors corresponding to the next 3 largest eigenvalues of the affinity matrix. Most values are small but for (disjoint) sets of elements the values are large. This follows from the block structure of the affinity matrix.

See Forsyth Ponce, Chapter 14 given.
Graph-based Image Segmentation

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph
2b- Build a similarity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
Clustering – How many groups are there?

Out of the various possible partitions, which is the correct one?
Clustering – 5 groups

Optimal?
Clustering – 5 groups

Looks optimal
What does the Affinity Matrix Look Like?
The Eigenvectors and the Clusters

Step-Function like behavior preferred!

Makes Clustering Easier.
The Eigenvectors and the Clusters

1st Eigenvector

2nd Eigenvector

3rd Eigenvector

4th Eigenvector

5th Eigenvector
The Eigenvectors and the Clusters

Eigenvector #1
Eigenvector #2
Eigenvector #3
Eigenvector #4
Eigenvector #5

#1
#2
#3
#4
#5
Clustering – Example 2

Dense Square Cluster

Sparse Square Cluster

Sparse Circle Cluster
The Affinity Matrix
The eigenvectors correspond the 2\textsuperscript{nd} smallest to the 9\textsuperscript{th} smallest eigenvalues.
Issue: Number of Clusters?

$k = 3$

$k = 4$

$k = 6$

Issue: choice of kernel, for Gaussian kernels, choice of $\sigma$

$\sigma = 3$

$\sigma = 13$

$\sigma = 25$
Graph-based Image Segmentation

Goal: Given data points X₁, …, Xₙ and similarities w(Xᵢ,Xⱼ), partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of data

2a- Build a similarity graph

2b- Build a similarity matrix

3- Calculate eigenvectors

4- Cut the graph: apply threshold to eigenvectors
Graph cut

Cuts in a graph:

\[ \text{cut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{i,j} \]

- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
Partition a graph with minimum cut

\[ \text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v) \]

- **Cut**: sum of the weight of the cut edges:
- **Minimum cut**: is the cut of minimum weight
Drawbacks of Minimum Cut

- Weight of cut is directly proportional to the number of edges in the cut.

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Normalized Cut is a better measure ..

• We normalize by the total volume of connections

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

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

where \( assoc(A, V) = \sum_{u \in A, t \in V} w(u, t) \)
Normalized Cut As Generalized Eigenvalue problem

\[ N \text{cut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)} \]

After simplification, Shi and Malik derive

\[ N \text{cut}(A, B) = \left(1 + x\right)^T (D - W) (1 + x) + \left(1 - x\right)^T (D - W) (1 - x) \]

\[ = \frac{(1 + x)^T (D - W) (1 + x)}{k 1^T D 1} + \frac{(1 - x)^T (D - W) (1 - x)}{(1 - k) 1^T D 1} \]

\[ = ... \]

For detailed derivation: http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf
Normalized cuts

Minimize:

\[ Ncut(A, B) = \frac{y^T (D - W) y}{y^T D y} \]

with \( y_i \in \{1, -b\}, y^T D 1 = 0 \).

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

- Instead, solve the generalized eigenvalue problem

\[ (D - W) y = \lambda D y \]

- They show that the 2\(^{nd}\) smallest eigenvector solution \( y \) is a good real-valued approx to the original normalized cuts problem. Then you 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

Many different methods...

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of **data**

2- Build a **similarity** graph

3- Calculate **eigenvectors**

4- Apply **threshold** to largest eigenvectors

---

1- Get vectors of **data**

2- Build normalized cost matrix

3- Get **eigenvectors** with smallest eigenvalues

4- Apply **threshold**

Shi & Malik

... etc
Global optimization

• In this formulation, the segmentation becomes a global process.
• Decisions about what is a boundary are not local (as in Canny edge detector)
Graph-based Image Segmentation

\[ \text{Intensity, Color, Edges, Texture} \]

Affinity matrix \((W)\)

\[ Ncut(A, B) = \text{cut}(A, B) \left( \frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right) \]

\[ (D - W)X = \lambda DX \]

\[ X_A(i) = \begin{cases} 1 & \text{if } i \in A \\ 0 & \text{if } i \notin A \end{cases} \]

Discretization/Thresholding

Slide from Timothee Cour
The Eigenvectors

Eigenvector #7
Normalized cut
Normalized cut
Fully Convolutional Networks for Semantic Segmentation

Jonathan Long*  Evan Shelhamer*  Trevor Darrell
UC Berkeley

Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.