Spectral Graph Theory

Social and Technological Networks

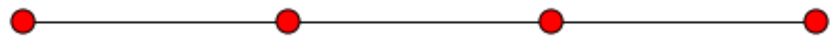
Rik Sarkar

University of Edinburgh, 2016.

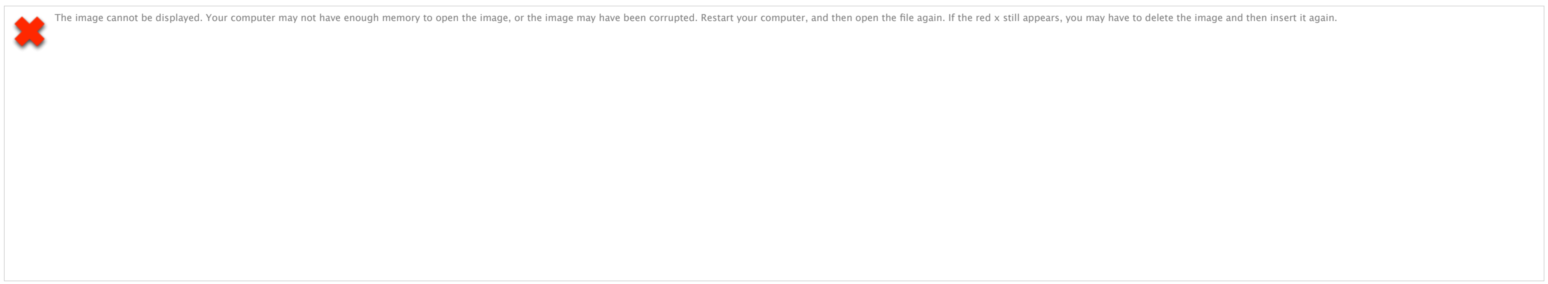
Spectral methods

- Understanding a graph using eigen values and eigen vectors of the matrix
- We saw:
- Ranks of web pages: components of 1st eigen vector of suitable matrix
- Pagerank or HITS are algorithms designed to compute the eigen vector
- Today: other ways spectral methods help in network analysis

Laplacian



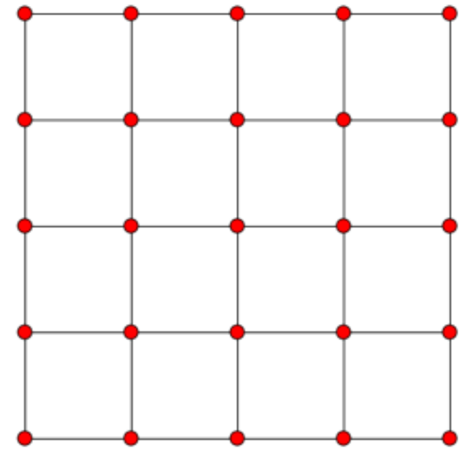
- L = D – A   [D is the diagonal matrix of degrees]

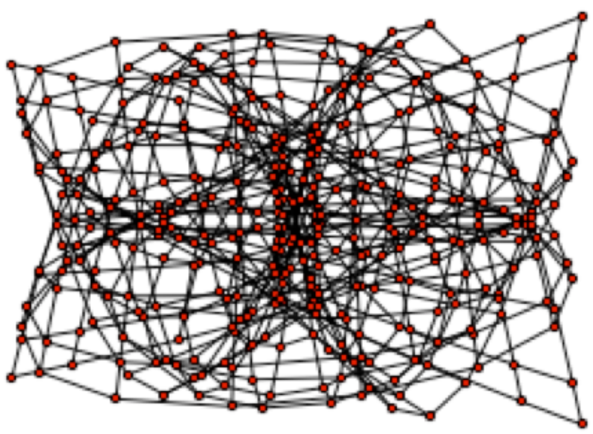


- An eigen vector has one value for each node
- We are interested in properties of these values

Application 1: Drawing a graph

- Problem: Computer does not know what a graph is supposed to look like
- A graph is a jumble of edges
- Consider a grid graph:
- We want it drawn *nicely*



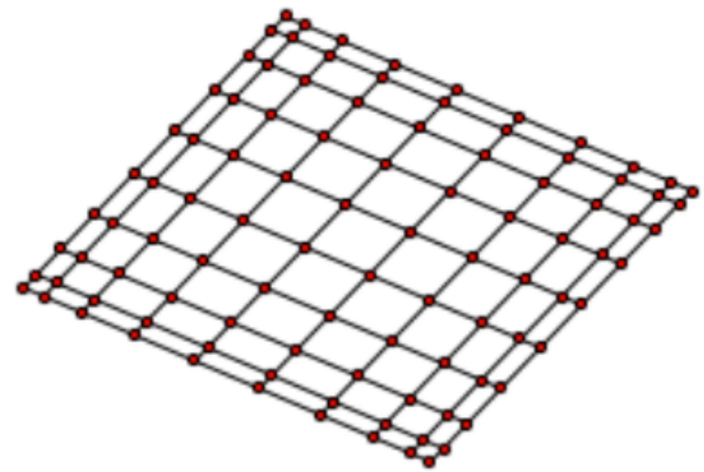


Graph embedding

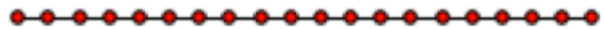
- Find positions for vertices of a graph in low dimension (compared to n)
- Common objective: Preserve some properties of the graph e.g. approximate distances between vertices
  - Useful in visualization
  - Finding approximate distances
- Using eigen vectors
  - One eigen vector gives x values of nodes
  - Other gives y-values of nodes … etc

Draw with v[1] and v[2]

- Suppose v[0], v[1], v[2]… are eigen vectors
  - Sorted by increasing eigen values
- Plot graph using X=v[1], Y=v[2]
- Produces the grid



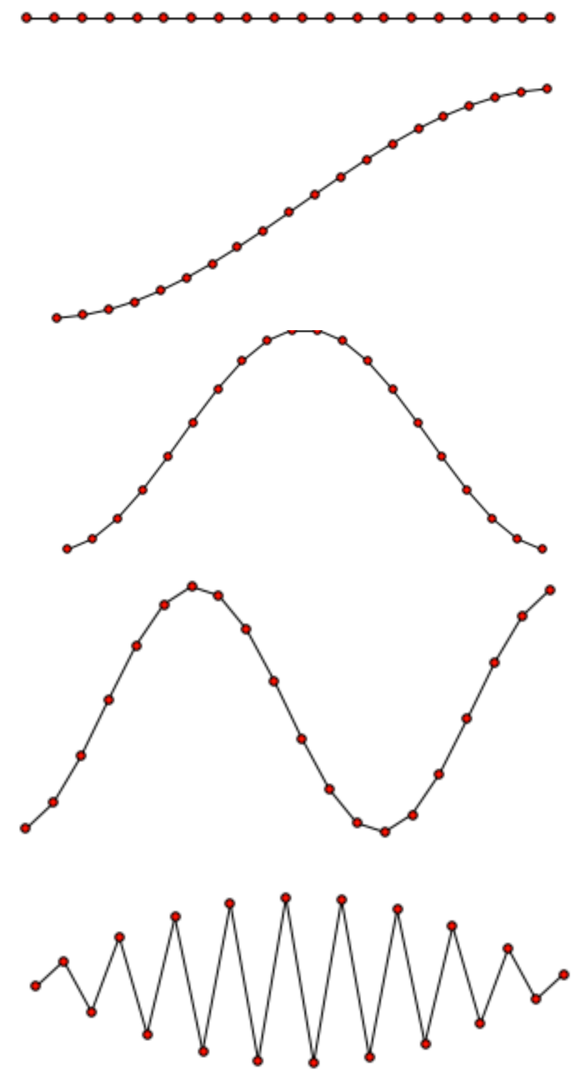
Intuitions: the 1-D case



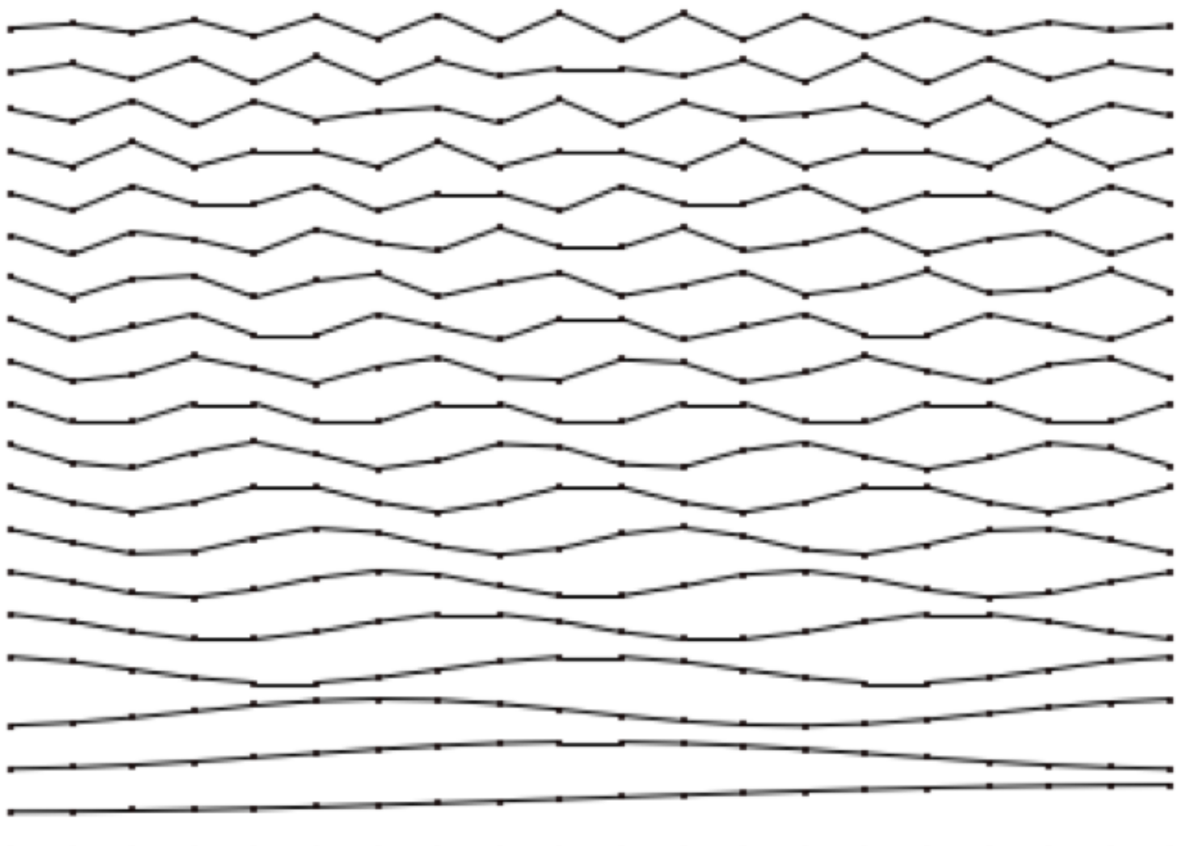
- Suppose we take the jth eigen vector of a chain
- What would that look like?
- We are going to plot the chain along x-axis
- The y axis will have the value of the node in the jth eigen vector
- We want to see how these rise and fall

Observations

- j = 0
- j=1
- j=2
- j =3
- j = 19

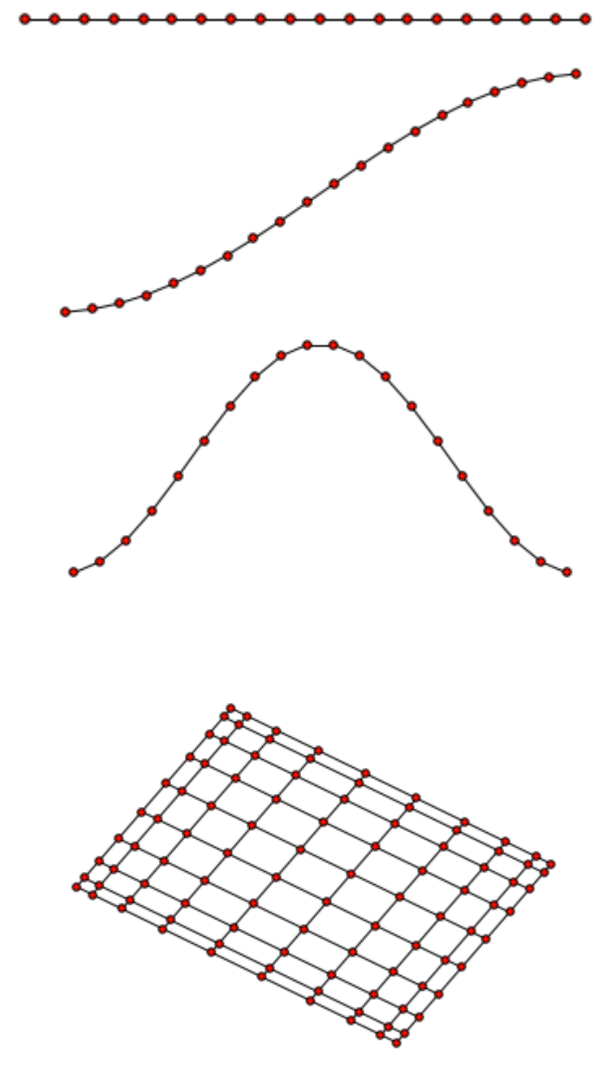


For All j

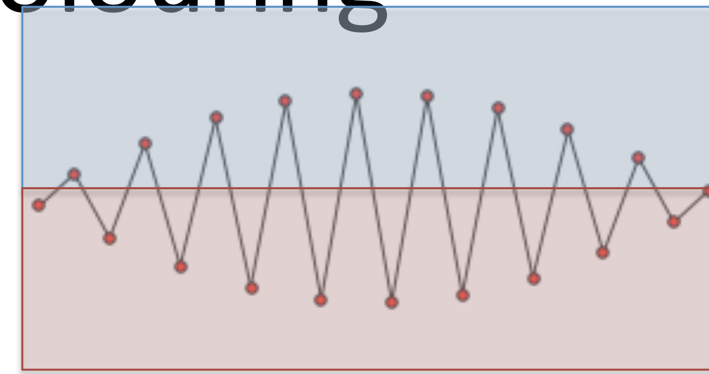


Observations

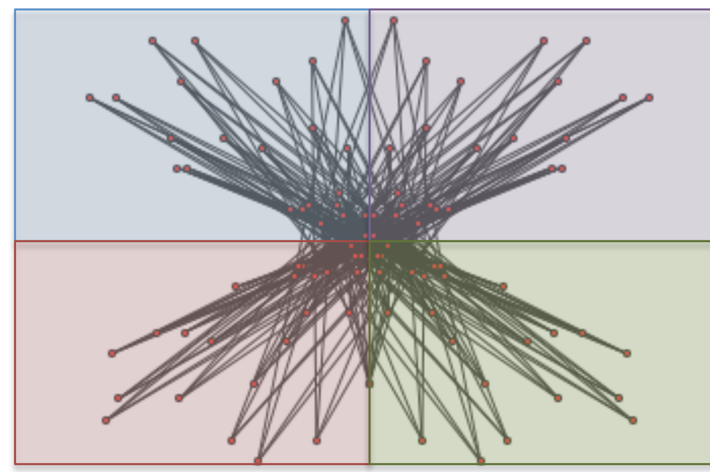
- In Dim 1 grid:
  - v[1] is monotone
  - v[2] is not monotone
- In dim 2 grid:
  - both v[1] and v[2] are monotone in suitable directions
- For low values of j:
  - Nearby nodes have similar values
    - Useful for embedding



Application 2: Colouring

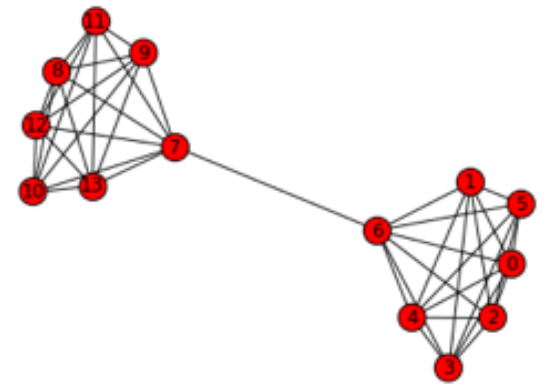


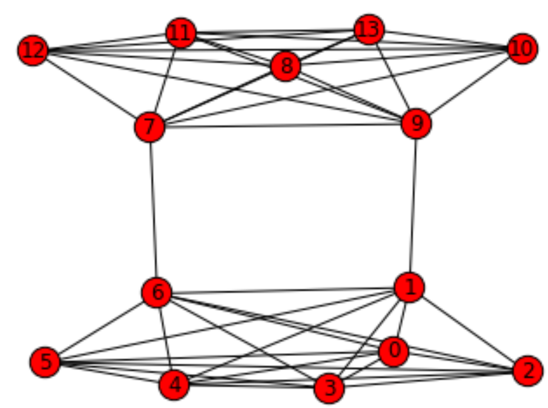
- Colouring: Assign colours to vertices, such that neighboring vertices do not have same colour
  - E.g. Assignment of radio channels to wireless nodes. Good colouring reduces interference
- Idea: High eigen vectors give *dissimilar* values to nearby nodes
- Use for colouring!



Application 3: Cuts/segmentation/ clustering

- Find the smallest 'cut'
- A small set of edges whose removal disconnects the graph
- Clustering, community detection…



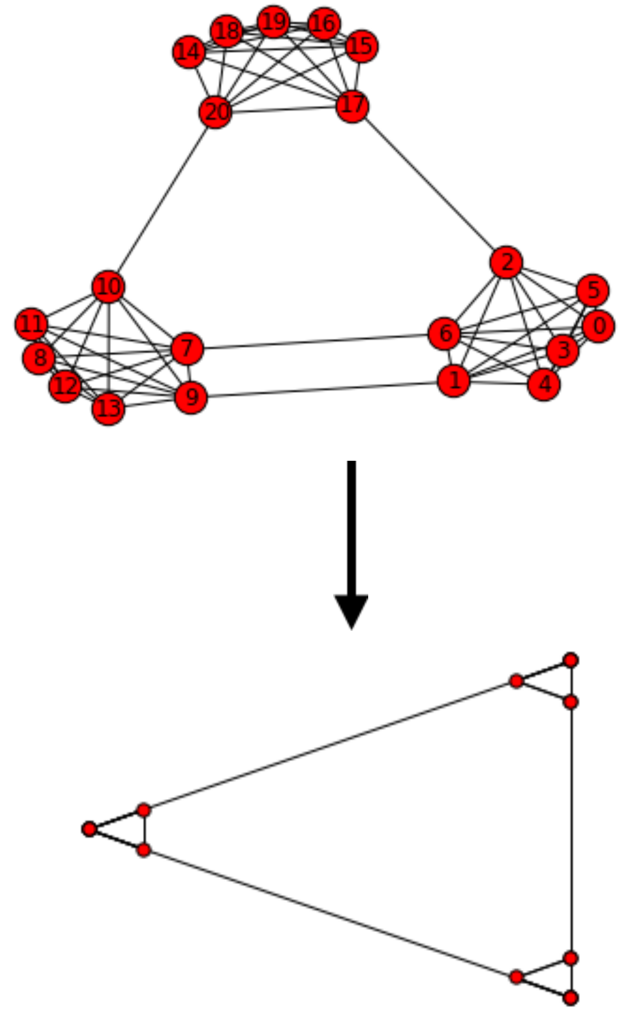


Clustering/community detection

- v[1] tends to stretch the narrow connections: discriminates different communities

Clustering: community detection

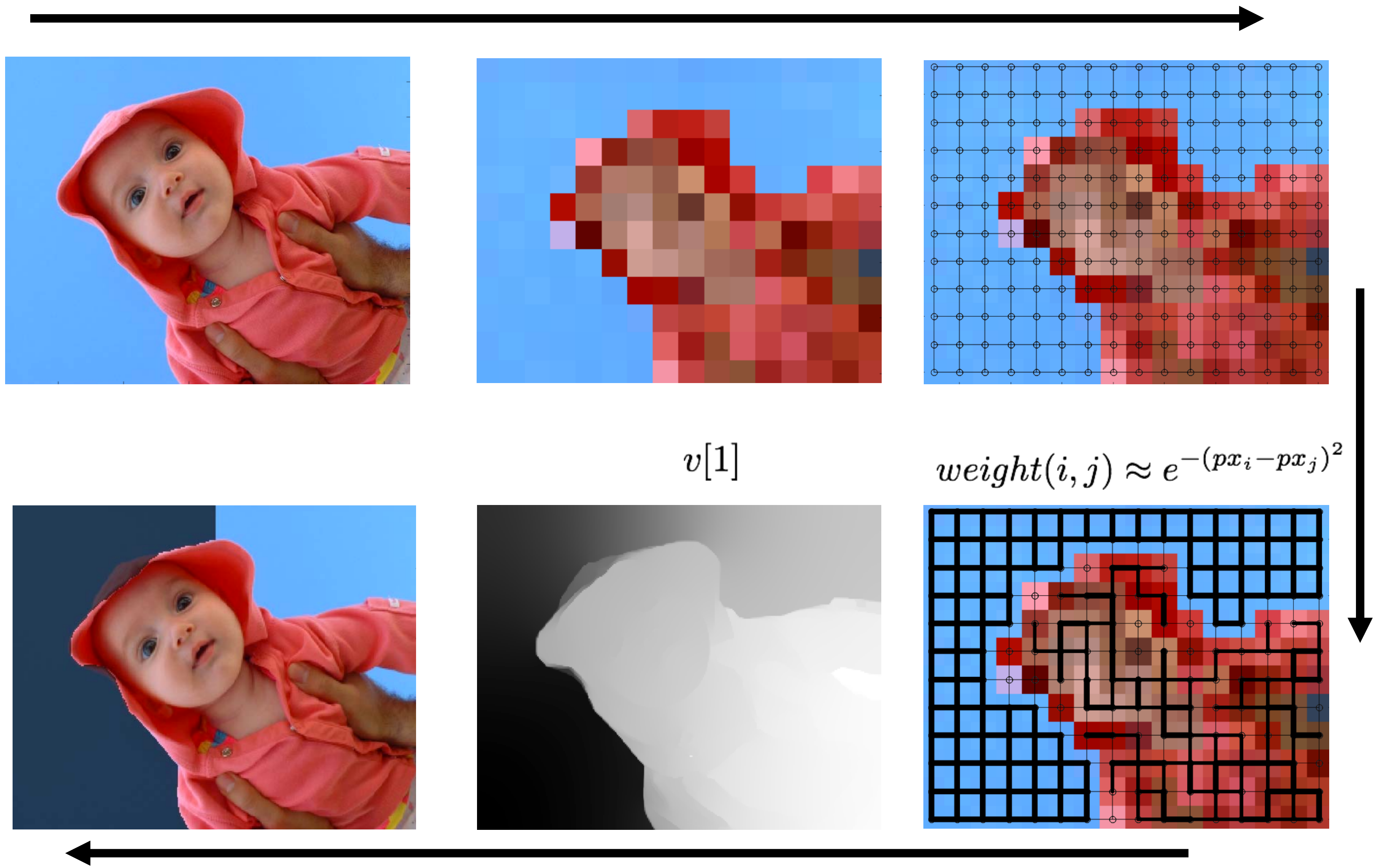
- More communities
- Need higher dimensions



- Warning: it does not always work so cleanly
- In this case, the data is very symmetric

Image segmentation

Shi & malik ’00



Laplacian matrix

- Imagine a small and different quantity of heat at each node (say, in a metal mesh)
- we write a function u: u(i) = heat at i
- This heat will spread through the mesh/graph
- Question: how much heat will each node have after a small amount of time?
- "heat" can be representative of the probability of a random walk being there

Heat diffusion

- Suppose nodes i and j are neighbors
  - How much heat will flow from i to j?

Heat diffusion

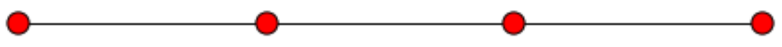
- Suppose nodes i and j are neighbors
- How much heat will flow from i to j?
- Proportional to the gradient:
  - $u(i) - u(j)$
  - this is signed: negative means heat flows into i

Heat diffusion

- If i has neighbors j1, j2….
- Then heat flowing out of i is:

  u(i) - u(j1) + u(i) - u(j2) + u(i) - u(j3) + …

  degree(i)*u(i) - u(j1) - u(j2) - u(j3) - ….
- Hence L = D - A



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

The heat equation

$$\frac{\partial u}{\partial t} = L(u)$$

- The net heat flow out of nodes in a time step
- The change in heat distribution in a small time step
  - The rate of change of heat distribution

The smooth heat equation

- The smooth Laplacian:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}.$$

- The smooth heat equation:

$$\Delta f = \frac{\partial f}{\partial t}$$

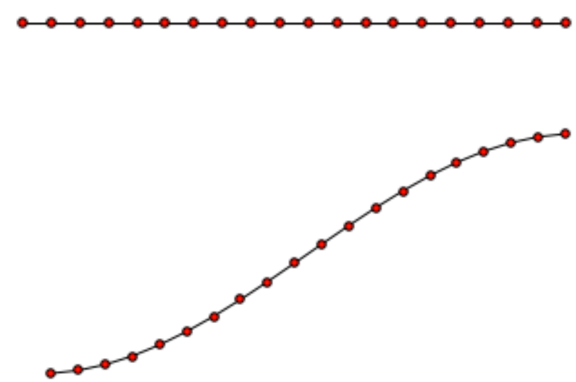
Heat flow

- Will eventually converge of v[0] : the zeroth eigen vector, with eigen value $\lambda_0 = 0$
- v[0] is a constant: no more flow!

v[0] = const

Laplacian

- Changed implied by L on any input vector can be represented by sum of action of its eigen vectors (we saw this last time for MM^T)
- v[0] is the slowest component of the change
  - With multiplier $\lambda_0=0$
- v[1] is slowest non-zero component
  - with multiplier λ_1



Spectral gap

- $\lambda_1 - \lambda_0$

- Determines the overall speed of change
- If the slowest component v[1] changes fast
  - Then overall the values must be changing fast
  - Fast diffusion
- If the slowest component is slow
  - Convergence will be slow
- Examples:
  - Expanders have large spectral gaps
  - Grids and dumbbells have small gaps ~ 1/n

Application 4: isomorphism testing

- Eigen values different implies graphs are different
- Though not necessarily the other way

Spectral methods

- Wide applicability inside and outside networks
- Related to many fundamental concepts
  - PCA
  - SVD
- Random walks, diffusion, heat equation…
- Results are good many times, but not always
- Relatively hard to give provable properties
- Inefficient: eig. computation costly on large matrix
- (Somewhat) efficient methods exist for more restricted problems
  - e.g. when we want only a few smallest/largest eigen vectors