Affinity Segmentation and Clustering

Czech Technical University in Prague
O. Shekhovtsov

- Semi-Supervised Segmentation
  - Energy Minimization Roadmap
  - Dirichlet Energy
  - Label Propagation
  - Random Walker
  - Soft Label Propagation & GCN
- Unsupervised Segmentation / Clustering
  - k-Means
  - Spectral Clustering
  - Normalized Cut

Graphs in Medical Image Processing

Setup

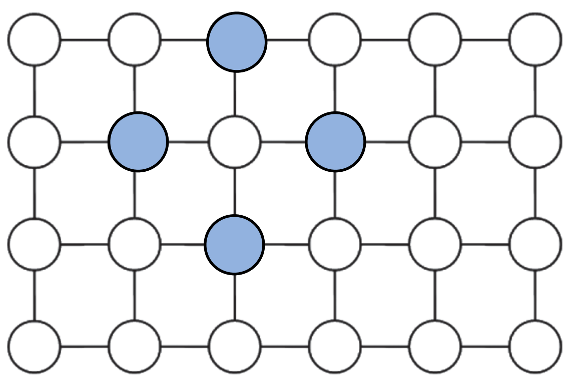
Graph $G = (V, E)$

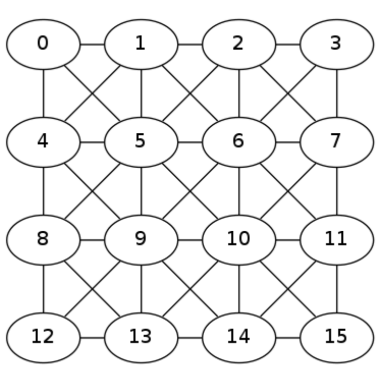
Node features $f_i \in \mathbb{R}^d$, $i \in V$

Affinity weights $A_{ij} \in \mathbb{R}^d$, $(i, j) \in E$

Examples

NN Graph of Pixels



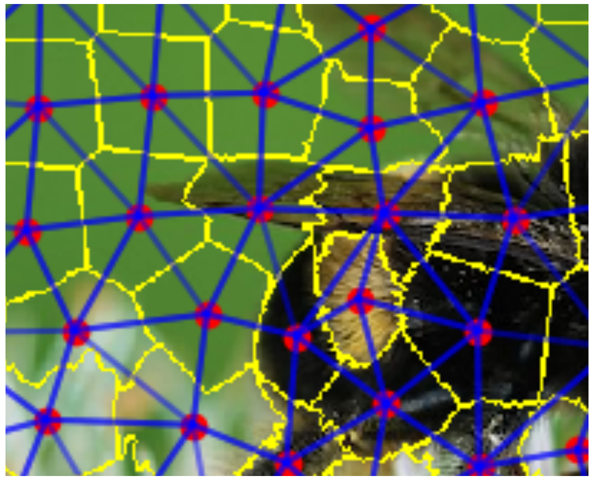


f_i – RGB color

$A_{ij} = e^{-\frac{(f_i - f_j)^2}{\sigma_0^2}} e^{-\frac{(i-j)^2}{\sigma_1^2}}$ – bilateral (color-spatial) affinity

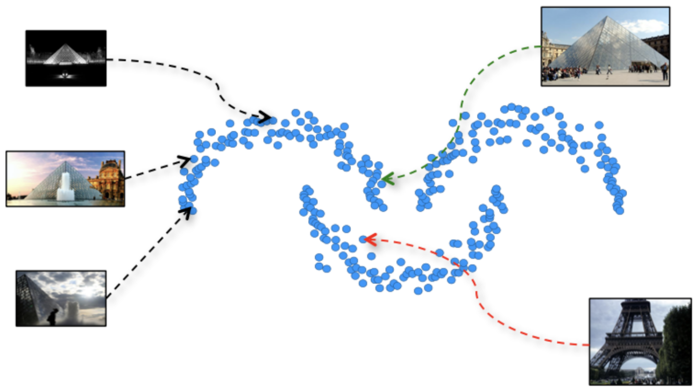
Segmentation, 3D reconstruction

NN Graph of SuperPixels



f_i – e.g. color, texture

Set of Images



f_i – image descriptor

$A_{ij} \propto \mathrm{Sim}(f_i, f_j)$

E – support set

Clustering,
Retrieval, visualization

Energy Minimization Roadmap

- **Energy Minimization Problem**

  Assign labelling $x\colon V \to C$ by

$$\min_x \sum_{ij} A_{ij} \mathcal{V}(x_i, x_j) + \sum_i \mathcal{U}_i(x_i)$$

$\mathcal{V}(x_i, x_j)$ – penalizes different labels

$\mathcal{U}_i(x_i)$ – fidelity to evidence. Special case: partial label assigment

- **Roadmap**:
  - C - ordered, $\mathcal{V}(x_i, x_j)$ convex function of the difference $(x_i - x_j) \Rightarrow$ minimum cut. (polynomial time, very efficient in practice).
  - $\mathcal{V}(x_i, x_j) = [\![x_i \neq x_j]\!] \Rightarrow$ 2 labels — back to previous case. More than 2 labels — Potts model / multiway cut. (NP-hard, approximation algorithms exist).
  - Relaxed formulations: $X\colon V \to \mathbb{R}^C$ – one-hot or soft labels.

Dirichlet Energy on Graphs

Let x be a scalar function on the nodes: $x\colon V \to \mathbb{R}$ (a vector in $\mathbb{R}^V$)

- **Dirichlet energy**: $\mathcal{E}(x) = \frac{1}{2}\sum_{i,j} A_{ij}\|x_i - x_j\|^2$
  - $\mathcal{E}(x)$ is small when nodes with strong A_{ij} have similar values $x_i \approx x_j$
  - Measures the smoothness of x on the graph w.r.t. affinity A
- As quadratic form
  - Denote degree matrix $D = \mathrm{diag}(d_1, \dots, d_n)$, $d_i = \sum_j A_{ij}$

$$\mathcal{E}(x) = \frac{1}{2}\sum_{ij} A_{ij}(x_i^2 + x_j^2 - 2x_i x_j) = x^\mathsf{T} D x - x^\mathsf{T} A x = x^\mathsf{T}(D - A)x = x^\mathsf{T} L x$$

  - $L = D - A$ is the **graph Laplacian** matrix
- Analogy to continuous Laplacian:
  - Let B be the discrete derivative: $(Bx)_e = \sqrt{A_{ij}}(x_i - x_j)$ for edge $e = (i,j)$, i.e., weighted finite differences along each *directed* edge
  - The energy can be written as

$$\mathcal{E}(x) = \sum_e (Bx)_e^2 = \|Bx\|^2 = x^\top B^\top B x$$

  - So $L = B^\top B$, analogous to $\Delta = \nabla \cdot \nabla$ (divergence of derivative)
  - Exercise: verify what B an L are for a 1D chain graph

Label Propagation: Energy

- **Setup**:
  - $V = (L, U)$ – partition into labeled and unlabelled nodes
  - $\underline{X}_L$ – fixed one-hot labels
  - X_U – unknown one-hot labels
- **Energy**
  - Note that $\|X_i - X_j\|^2 = 2$ if $X_i \neq X_j$ and 0 otherwise, so for one-hot labels we can write the Potts energy as.

    $$\mathcal{E}(X) = \frac{1}{2}\sum_{i,j} A_{ij}\|X_i - X_j\|^2 = \sum_k \frac{1}{2}\sum_{i,j} A_{ij}(X_{ik} - X_{jk})^2 = \sum_k \mathcal{E}(X_{:k}),$$

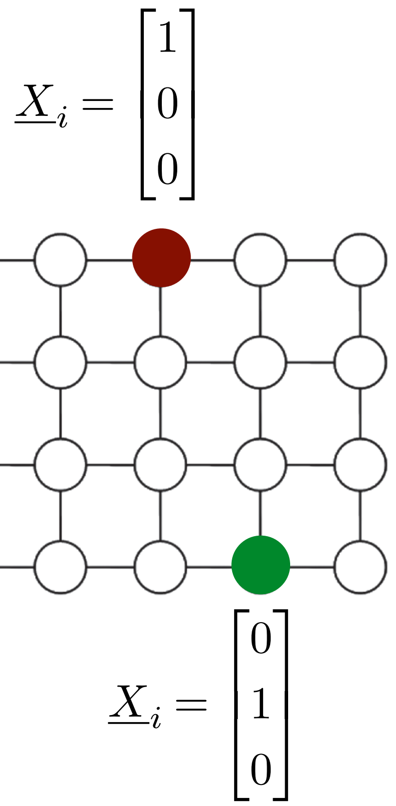
    i.e., sum of Dirichlet energies over each class indicator function.
  - In the matrix form:

    $$\mathcal{E}(X) = \mathrm{Tr}(X^\top L X).$$

- The **Label Propagation Problem**:

$$\min_X \quad \mathcal{E}(X) \quad \text{s.t.} \quad X_i = \underline{X}_i,\ \forall i \in L$$

  - Seeks the most smooth assignment of labels while exactly matching the labeled nodes.
  - Relaxation: allow X_i to be soft labels in $\mathbb{R}^C$.



Label Propagation: Solution

- In matrix form, with nodes reordered so that labeled nodes come first:

$$X = \begin{bmatrix} X_L \\ X_U \end{bmatrix}, \quad L = \begin{bmatrix} L_{LL} & L_{LU} \\ L_{UL} & L_{UU} \end{bmatrix},$$

- The energy is

$$\mathcal{E}(X) = \mathrm{Tr}(X_L^\top L_{LL} X_L + X_L^\top L_{LU} X_U + X_U^\top L_{UL} X_L + X_U^\top L_{UU} X_U)$$

- Differentiate with respect to X_U (X_L is fixed) and set the gradient to zero:

$$\frac{\partial \mathcal{E}}{\partial X_U} = 2L_{UU}X_U + 2L_{UL}X_L = 0 \Longrightarrow \quad \boxed{L_{UU}X_U = -L_{UL}X_L}$$

- Assuming L_{UU} is invertible (each unlabeled node connects to at least one labeled node)
- Closed form solution: $\boxed{X_U^* = -L_{UU}^{-1} L_{UL} X_L}$

Label Propagation: Fixed Point Iteration

◆ **Fixed Point Equation**:

- Substitute $L = D - A$ in the block system:

$$(D_{UU} - A_{UU})X_U = A_{UL}X_L$$

$$\implies \quad X_U = D_{UU}^{-1}A_{UU}X_U + D_{UU}^{-1}A_{UL}X_L$$

- Define the row-stochastic adjacency (random-walk) matrix $P = D^{-1}A$. Then:

$$\boxed{X_U = P_{UU}X_U + P_{UL}X_L}$$

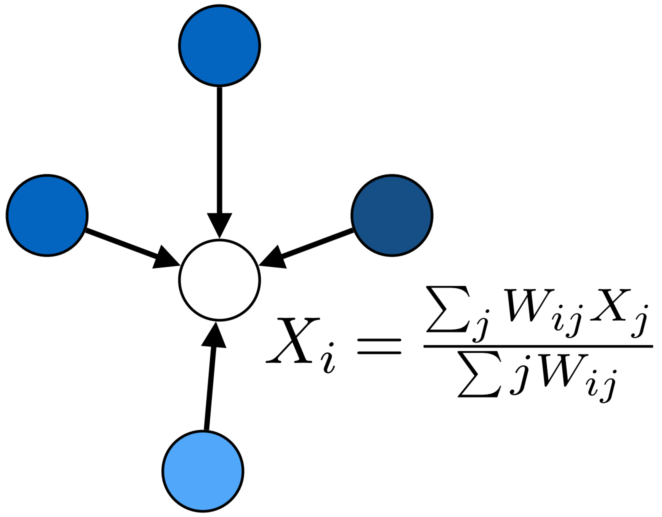
This is a fixed-point equation.

◆ **Label Propagation Algorithm**

Initialization: $X_U^{(0)} = 0$ (or random) Iteration:

$$X_U^{(t+1)} = P_{UU}X_U^{(t)} + P_{UL}X_L, \quad X_L \text{ fixed}$$

$$\boxed{X_i^{(t+1)} = \frac{1}{d_i}\sum_j A_{ij}X_j^{(t)} \quad \forall i \in U}$$



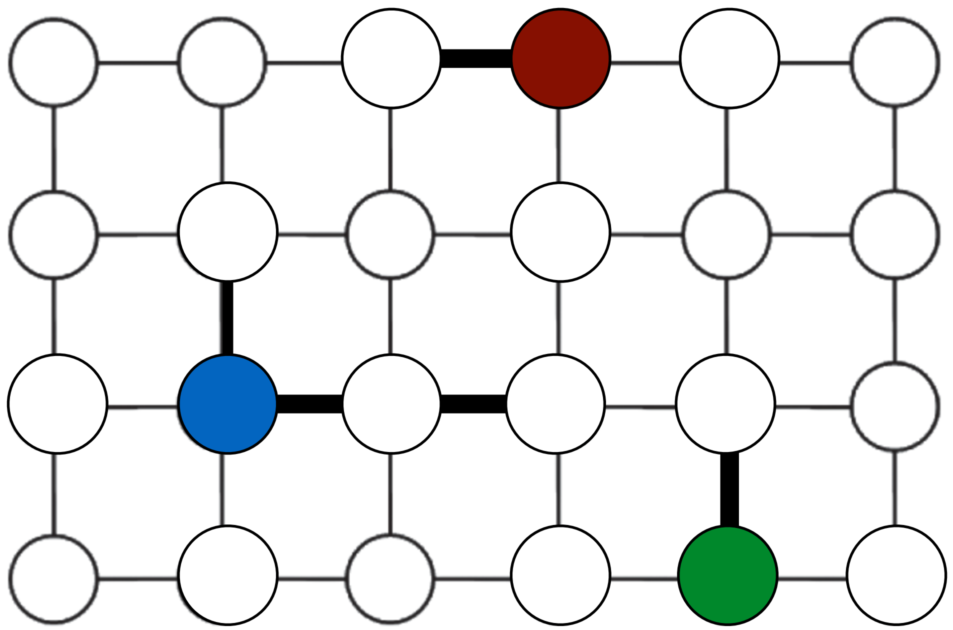
- **Convergence:**

$$X_U^{(t)} \longrightarrow X_U^{*} \text{– the optimal solution}$$

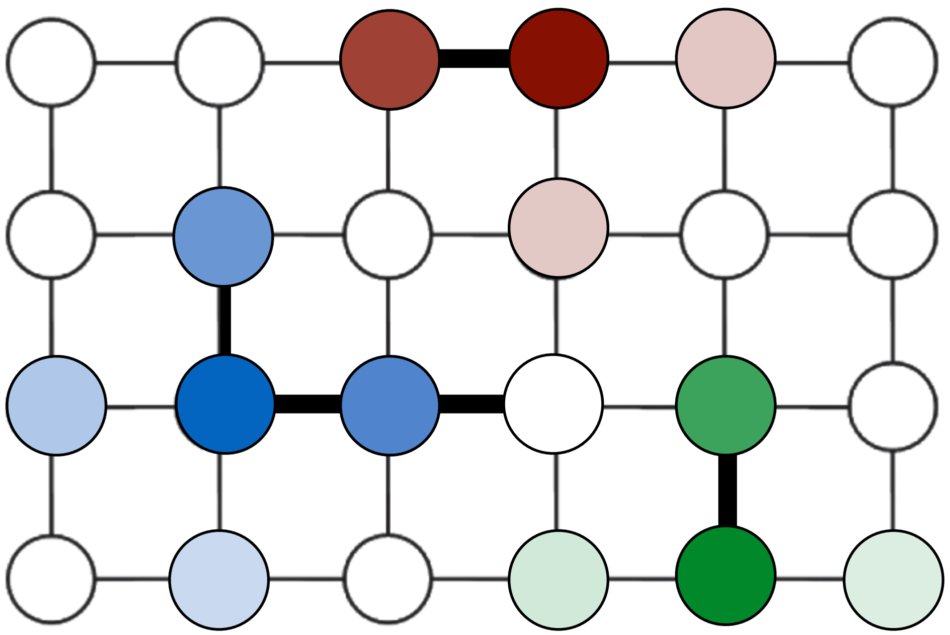
◆ Exercise: $L = D - A$ does not depend on A_{ii}, but the iteration does?

Label Propagation: Example

Initialization



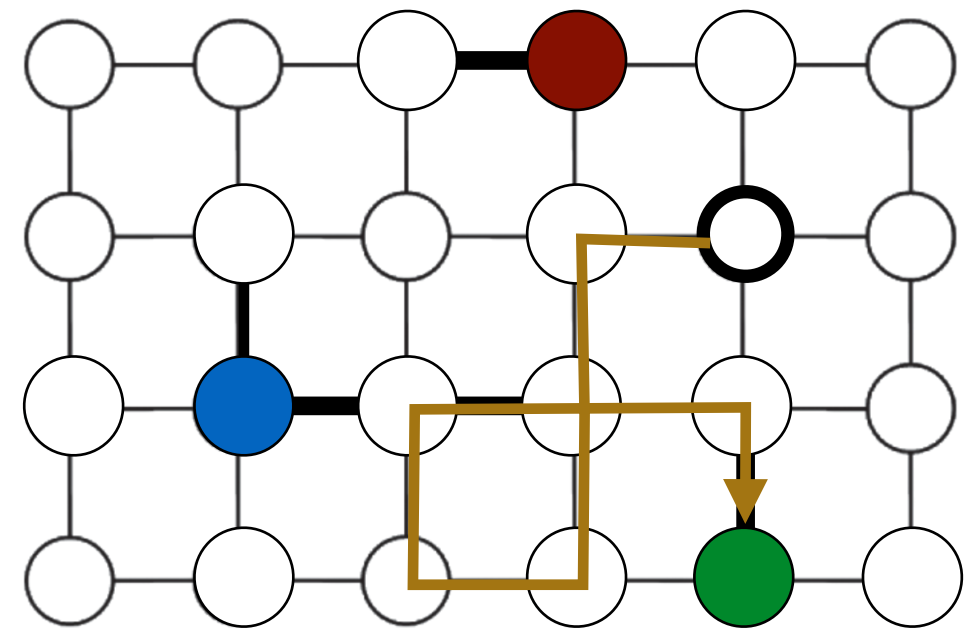
Step 1



- kind of diffusing
- faster along strong edges

Random Walker

- "Algorithm":
  - Start from node i
  - Move to random node j
    with probability proportional to A_{ij}
  - Until hitting a labelled node
  - X_{ik} – probability of hitting a node with label
  - Decide label of i as $\operatorname{argmax}_k X_{ik}$



- Expanding hit probabilities conditioned on the first step:

$$X_{ik} = \sum_{j \in \mathcal{N}(i)} \underbrace{\mathbb{P}[\text{walker steps from } i \text{ to } j]}_{P_{ij}} \cdot \underbrace{\mathbb{P}[\text{first hit label } k \text{ starting from } j]}_{X_{jk}}.$$

Transition probability P_{ij} is proportional to edge weight: $P_{ij} = \frac{A_{ij}}{d_i}$, $d_i = \sum_j A_{ij}$.
Terefore, for each unlabeled node $i \in U$, the first-hit probability satisfies

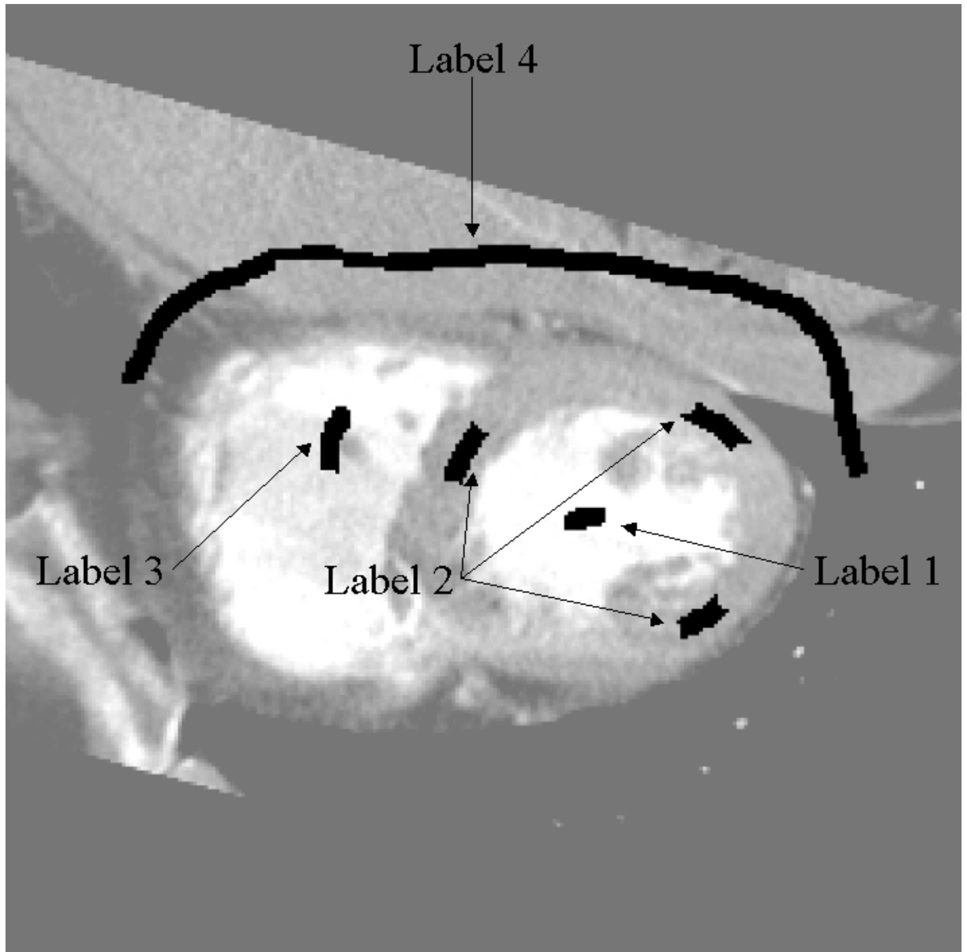
$$\forall i \in U \qquad X_{ik} = \sum_j P_{ij} X_{jk}$$

$$X_U = P_{UU} X_U + P_{UL} X_L$$ — same fixed point equation

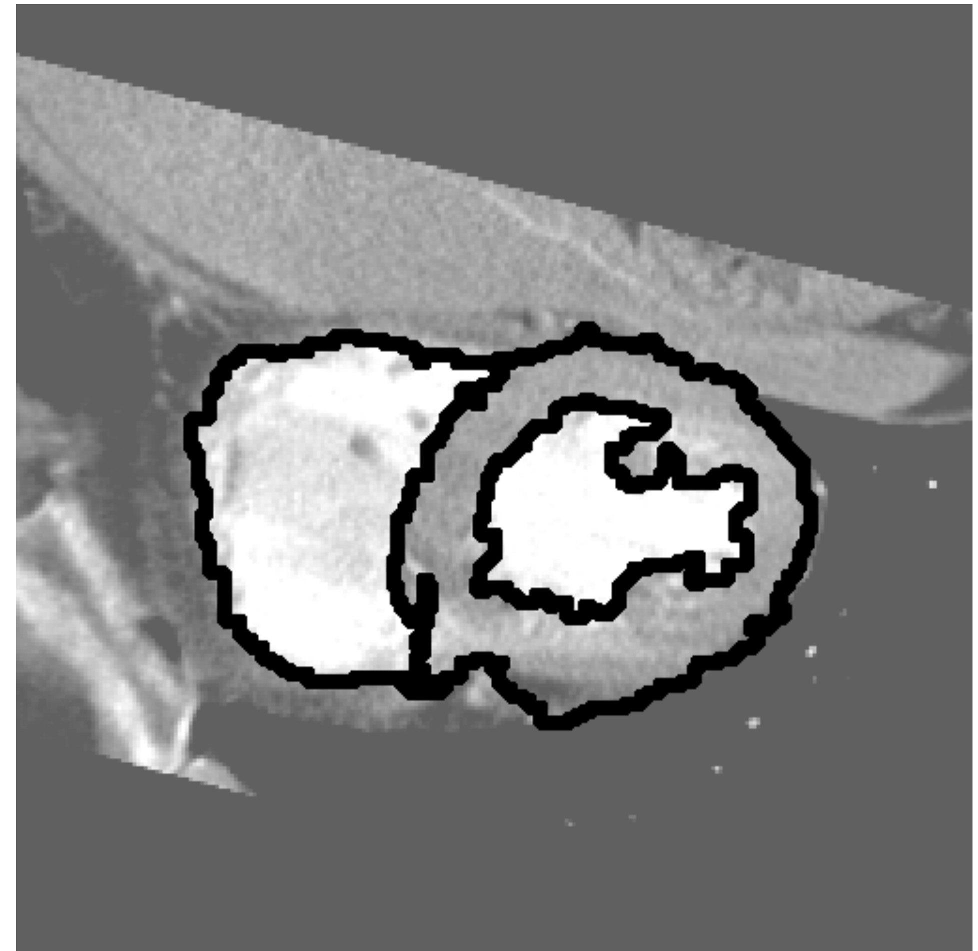
Random Walker

- Conclusions:
  - The fixed-point equation for the first-hit probabilities is equivalent to the label propagation update rule.
  - Thus, the solution X_U minimizes the Laplacian energy with hard label constraints – can solve it by any method
  - The same X_U is the matrix of first-hit probabilities $\Rightarrow$ interpretation of the relaxed labels $X \in \mathbb{R}^{V \times C}$.
  - Not guaranteed to match the optimal discrete segmentation.
  - The reverse process would be a stochastic generative model that starts from seeds (absorbing nodes) and diffuses backward along edges to produce plausible node values. Nowadays we have e.g. “Random Walk Diffusion for Efficient Large-Scale Graph Generation”, for tasks like designing new molecular structures.

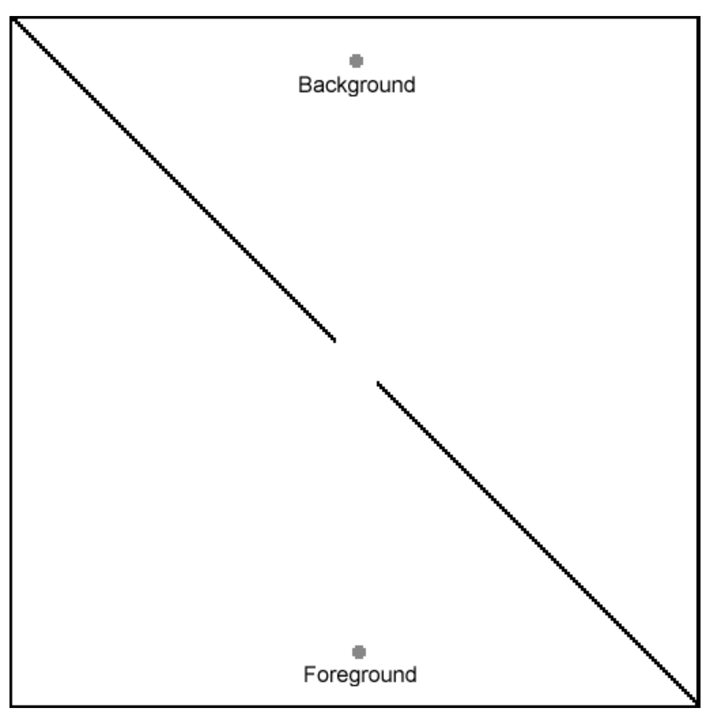
Examples



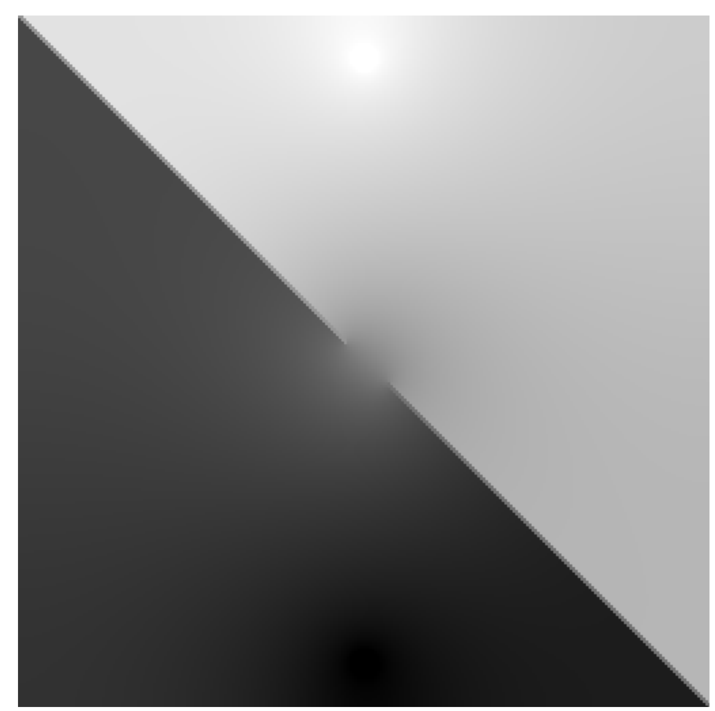
(b) Seeds indicating four objects



(c) Resulting segmentation



(a) Original



(d) Probabilities

Normalized Label Propagation

- Reparameterization
  - Define $Y = D^{1/2}X \quad \implies \quad X = D^{-1/2}Y$,
  - Substituting into the energy we obtain:

$$\mathcal{E}(X) = \mathrm{Tr}\left(X^\top L X\right) = \mathrm{Tr}\left(Y^\top D^{-\frac{1}{2}} L D^{-\frac{1}{2}} Y\right) = \mathrm{Tr}(Y^\top \mathcal{L}_{\mathrm{sym}} Y) =: \mathcal{E}(Y),$$

with the **symmetric normalized Laplacian**

$$\boxed{\mathcal{L}_{\mathrm{sym}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = I - \tilde{A}}, \quad \text{where } \tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

- The fixed-point iteration becomes:

$$Y_U \leftarrow \tilde{A}_{UU} Y_U + \tilde{A}_{UL} Y_L$$

- For $Y_L = D^{\frac{1}{2}} X_L$, it is an equivalent reformulation, $X = D^{-\frac{1}{2}} Y$ are the hitting probabilities
- For generic, Y_L, *i.e.* one-hot labels or features, it is a modified problem formulation
  - Used with generic data and dense graphs where affinities are constructed from feature similarity (node degrees can be very different)
  - The normalization reduces the influence of high-degree nodes and balances propagation

Soft Label Propagation

- Soft label propagation relaxes the hard-clamp constraint and introduces a tradeoff between *smoothness* and *fidelity to initial labels* $\underline{Y}$ (one-hot or zero)
  - Can be applied with unnormalized or normalized formulation
  - We apply it with normalized formulation, to connect to GCN (next)
- **Soft Normalized Label Propagation**
  - Energy minimization formulation:

$$\mathcal{J}(Y) = \alpha \underbrace{\mathrm{Tr}(Y^\top L_{\mathrm{sym}} Y)}_{\text{smoothness energy}} + (1-\alpha)\underbrace{\frac{1}{2}\sum_i \|Y_i - \underline{Y}_i\|^2}_{\text{fidelity to input labels}}, \quad 0 < \alpha < 1$$

  - Closed-form solution:

$$Y^* = (\alpha \mathcal{L}_{\mathrm{sym}} + (1-\alpha)I)^{-1}(1-\alpha)\underline{Y}$$

    where $\mathcal{L}_{\mathrm{sym}} = I - \tilde{A}$ as above
  - Iterative update (normalized soft label propagation):

$$\boxed{Y^{(t+1)} = \alpha \tilde{A} Y^{(t)} + (1-\alpha)\underline{Y}}$$

Graph Convolutional Networks (GCN)

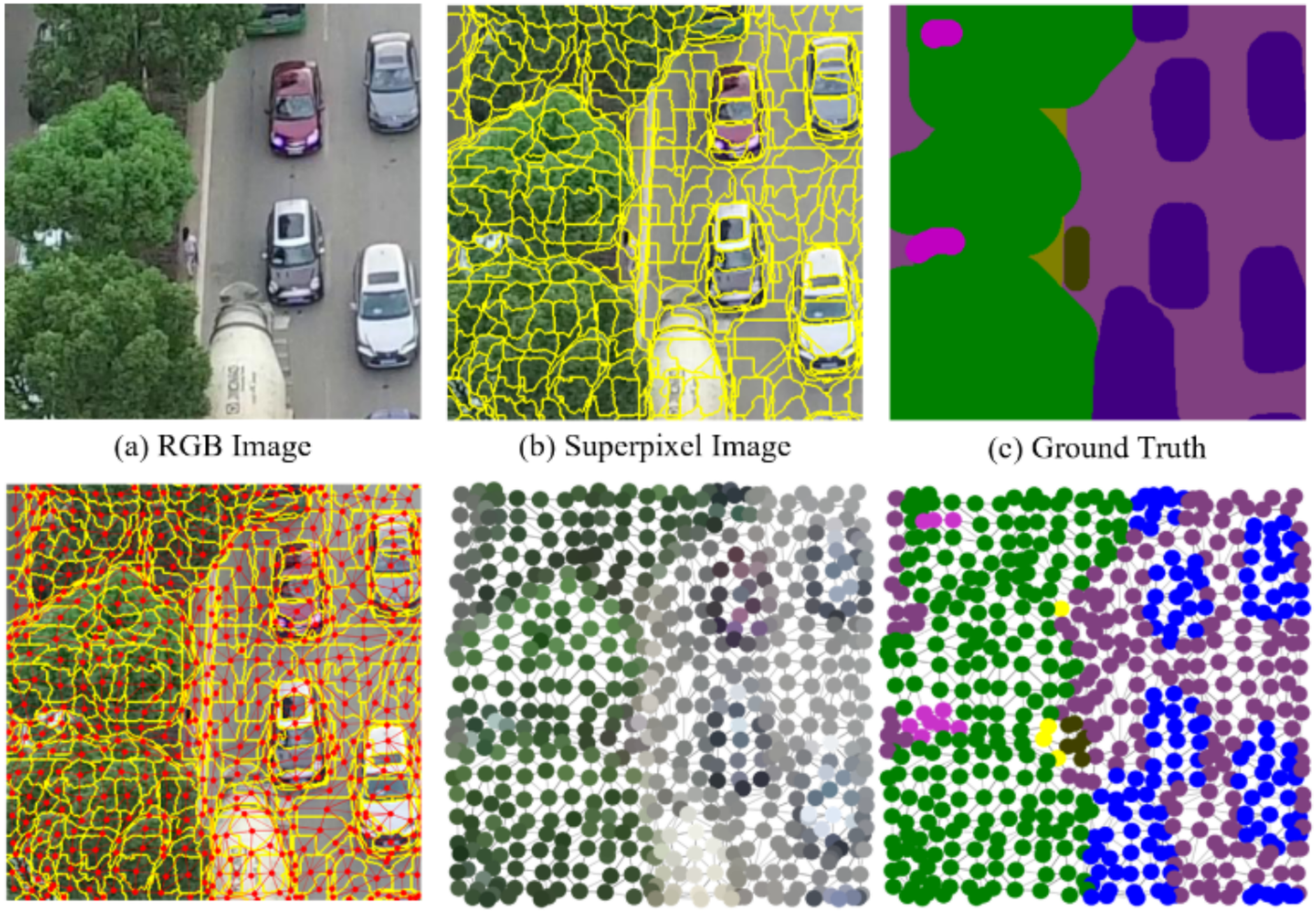
- Disclaimer
  - There are many ways to define convolutions on graphs (*e.g*. spectral w.r.t. to different variants of Laplacian, approximate, *etc*.)
  - This is just to illustrate how the concepts are related.
- Idea
  - One layer of (GCN by Kipf & Welling) can be seen as a single iteration of normalized soft label propagation, but **with learnable weight matrices** and nonlinearities:

$$Y^{(l+1)} = \sigma\left(\underbrace{\tilde{A}}_{\text{neighbour aggregation}} \quad Y^{(l)} \quad \underbrace{W^{(l)}}_{\text{local feature transform}} \right)$$

  - where $Y^{(l)}$ is the node feature matrix at layer l
  - $W^{(l)}$ is a learnable weight matrix
  - $\sigma(\cdot)$ is a nonlinearity (e.g., ReLU)
  - $\tilde{A} = D^{-1/2} A D^{-1/2}$ is the normalized adjacency, as before
- Observations:
  - GCNs use the same graph metric to propagate features across the graph
  - The first layer is initialized with the input: $Y^{(0)} = \underline{Y}$ (features, not the labels)
  - It is trained so that after the last layer we can make decision, *e.g*. $\operatorname{argmax} CY_i$, independently for all nodes i.
  - The initial features are not mixed-in explicitly. Instead they add self-loops in A.

GCN Examples

✦ Superpixel-based Graph Convolutional Network for Semantic Segmentation, Yung et al.



(a) RGB Image (b) Superpixel Image (c) Ground Truth

(d) Graph Generation (e) Superpixel Graph (f) Ground Truth Graph

Unsupervised Segmentation / Clustering

Overview

Unsupervised Segmentation (Clustering) Problem

- Partition the image (set of data) without any seed labels or class affinities
- Euclidean space: $\rightarrow$ k-means clustering
- PSD similarity kernel $K(x, y)$ $\rightarrow$ Kernel k-means clustering
  (local optima, computation cost?)

Spectral Clustering

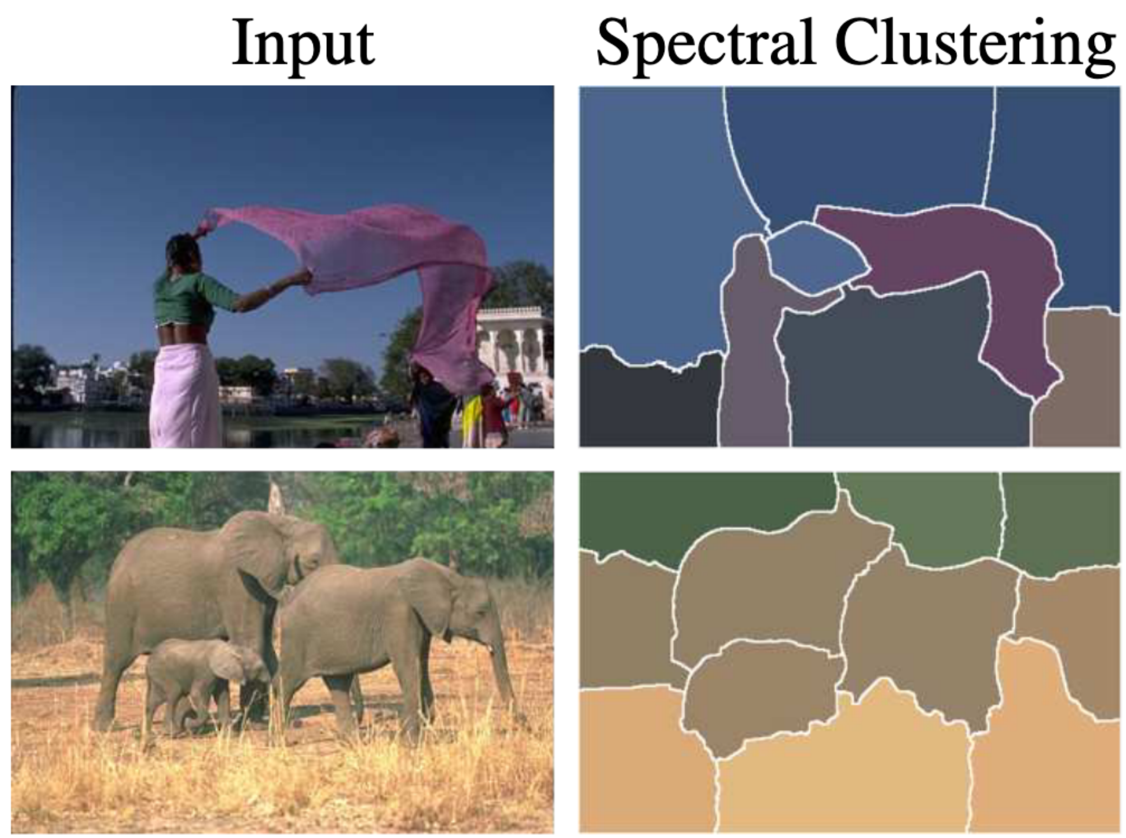
- Graph $G = (V, E)$
- Affinity matrix $A_{ij} \geq 0$ for all ij, symmetric, *need not be PSD*
- Degree matrix $D = \text{diag}(d)$, $d_i = \sum_j A_{ij}$
- Normalized Affinity matrix: $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
- Algorithm:
  1. Compute top-k eigenvectors of $\tilde{A}$, exclude $\mathbf{1}$, $\rightarrow$ matrix U of size $n \times k-1$
     (note: same as smallest k eigenvectors of $L_{\text{sym}} = I - \tilde{A}$)
  2. Each row $U_{i,:}$ gives an embedding of the node i in $\mathbb{R}^{k-1}$
  3. Run standard k-means clustering on rows of U
- Solves the same problem as kernel k-means clustering

Normalized Cut

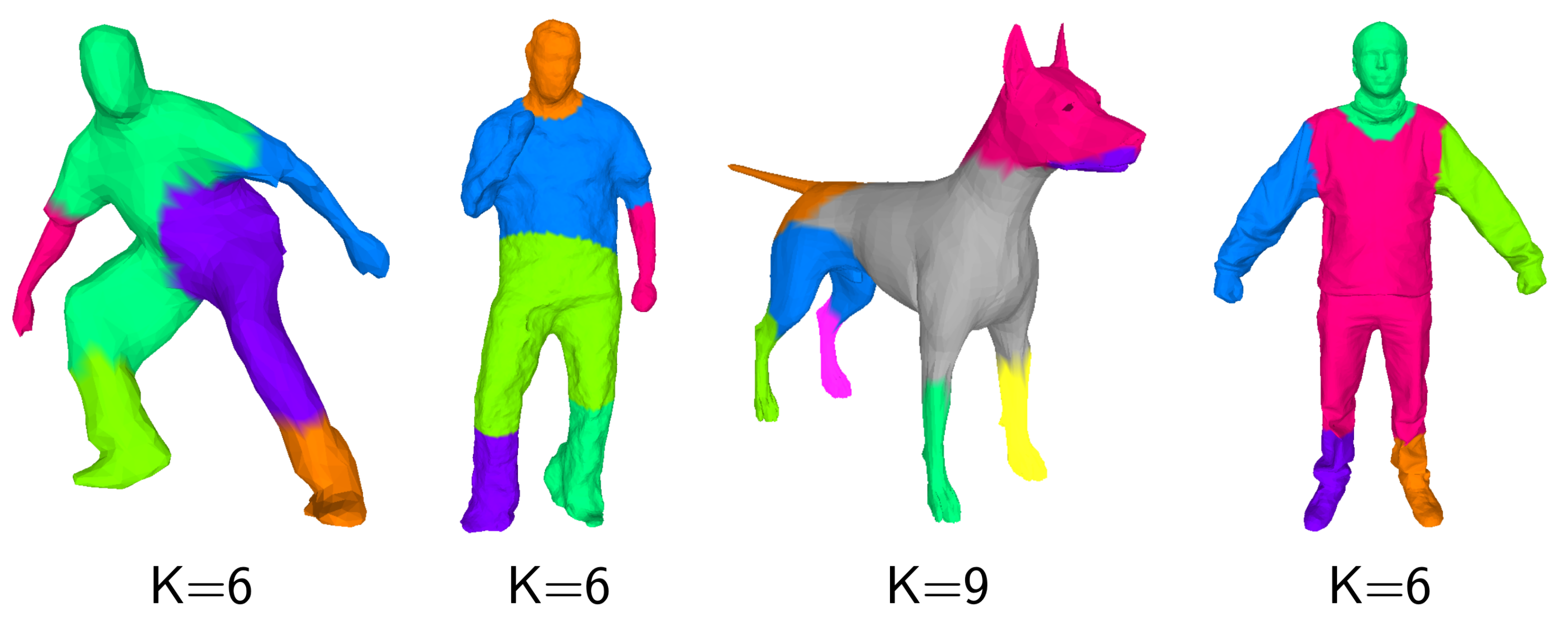
- Somewhat different objective, *same relaxation* $\Rightarrow$ same solution
- 2-way Ncut is special case

Examples

Example 1: unsupervised image segmentation



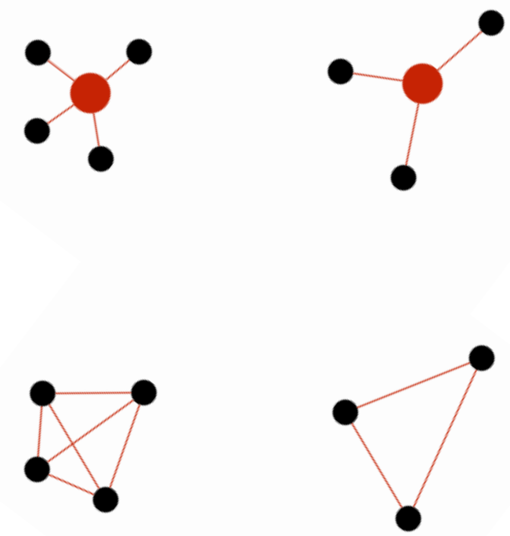
Example 2: unsupervised 3D mesh segmentation



k-Means Clustering Problem

- Let $f = \{f_1, \ldots, f_n\}$ be data points in $\mathbb{R}^d$
- k-**Means clustering problem**: partition the data into k clusters $C_1, \ldots, C_k$ with means μ_j:

$$\min_{C,\mu} \sum_k \sum_{i \in C_k} \|f_i - \mu_k\|^2, \quad \Rightarrow \quad \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} f_i$$



- Equivalent objective substituting μ (exercise):

$$\min_C \sum_k \frac{1}{2|C_k|} \sum_{i,j \in C_k} \|f_i - f_j\|^2$$

- Denoting $K_{ij} = \langle f_i, f_j \rangle$ – kernel matrix,

$$\|f_i - f_j\|^2 = K_{ii} + K_{jj} - 2K_{ij}$$

- Thus the k-means objective becomes:

$$\min_C \sum_k \frac{1}{|C_k|} \sum_{i,j \in C_k} \Big(K_{ii} + K_{jj} - 2K_{ij} \Big) = \boxed{2 \sum_i K_{ii} - \sum_k \frac{2}{|C_k|} \sum_{i,j \in C_k} K_{ij}}$$

Combinatorial problem that needs only the kernel K

Spectral Clustering

k-Means Clustering Problem

$$\max_{C \text{ – partition of } V} \sum_k \frac{1}{|C_k|} \sum_{i,j \in C_k} K_{ij}$$

Rewriting Objective as Trace

- Express the objective using normalized cluster indicator matrix $X \in \mathbb{R}^{n \times k}$:

$$X_{ik} = \begin{cases} \frac{1}{\sqrt{|C_k|}}, & i \in C_k \\ 0, & \text{otherwise} \end{cases}, \quad X^\top X = I. \qquad \text{— combinatorial set } \mathcal{X}$$

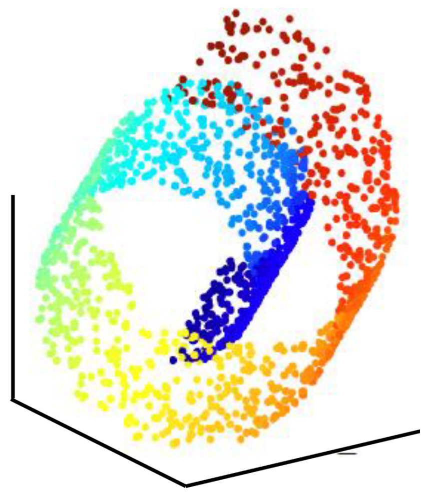
$$\sum_k \frac{1}{|C_k|} \sum_{ij \in C_k} K_{ij} = \sum_k \frac{1}{|C_k|} \sum_{ij} X_{ik} X_{jk} K_{ij} = \mathrm{Tr}(X^\top K X)$$

Relaxation:

$$\max_{X \in \mathbb{R}^{n \times k}} \mathrm{Tr}(X^\top K X) \text{ s.t. } X^\mathsf{T} X = I$$

- **Solution**: X is top-k normalized eigenvectors of K
- For graphs, use $K = \tilde{W}$
- Eigenvectors are the same as those of $\mathcal{L}_{\text{sym}} = I - \tilde{W}$, eigenvalues are in reverse order
- The first eigenvector is always $\mathbf{1}$
- To recover partition, discretize X, by common k-means clustering

Laplacian Eigenvectors

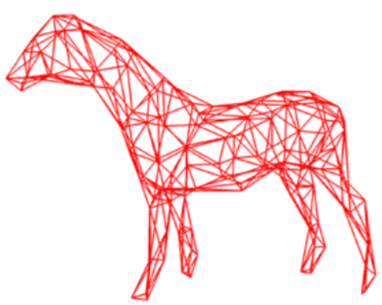


Example 1:

NN graph of data points embedded in 3D

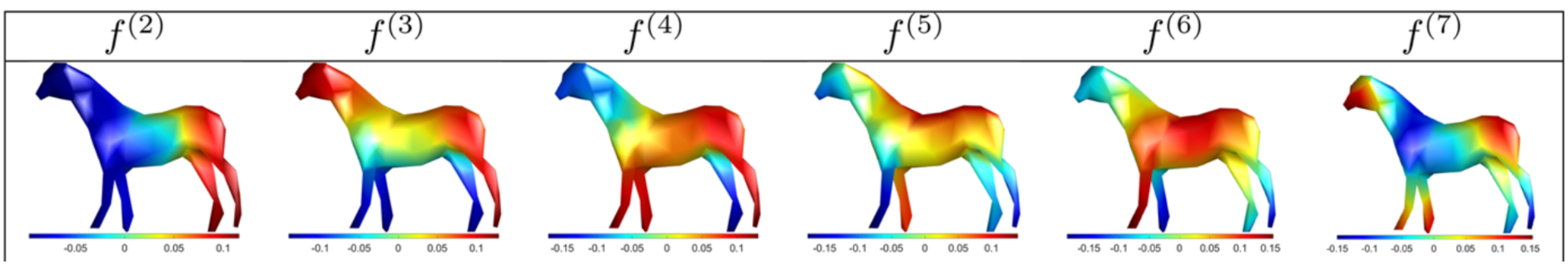
First non-trivial eigenvector, in this example discovers the main ordering direction

Example 2:



horse
$|V| = 152$

Eigenvectors



Can be used as new features, aggregating the shape information and *invariant to isometric transforms*. Useful for (non-rigid) shape matching and positional encoding in Graph NNs.

Multiway Normalized Cut (Ncut)

- **The multiway normalized cut objective**:

$$\text{Ncut}(C_1,\ldots,C_K) = \sum_{k=1}^{K} \frac{\text{cut}(C_k,\bar{C}_k)}{\text{vol}(C_k)}, \quad \text{cut}(C_k,\bar{C}_k) = \sum_{i\in C_k, j\notin C_k} A_{ij}, \quad \text{vol}(C_k) = \sum_{i\in C_k} d_i$$

- Equivalent objective: $\sum_k \frac{1}{\text{vol}(C_k)} \sum_{i,j\in C_k} A_{ij}$, — **similar to k-means**
- Trace Reformulation
  - Introduce the *normalized cluster indicator matrix* $X \in \mathbb{R}^{n\times k}$ with entries:

$$X_{ik} = \begin{cases} \frac{1}{\sqrt{\text{vol}(C_k)}}, & i \in C_k \\ 0, & \text{otherwise} \end{cases}, \quad X^\top D X = I$$

  - Then the multiway Ncut can be written as trace:

$$\text{Ncut}(C_1,\ldots,C_k) = \text{Tr}\left(X^\top A X\right)$$

- **Relaxation**:

$$\max_{X \,:\, X^\top D X = I} \text{Tr}\left(X^\top A X\right) = \boxed{\max_{Y \,:\, Y^\top X = I} \text{Tr}\left(Y^\top \tilde{A} Y\right)}$$

  - **Same relaxation as spectral clustering for $\tilde{A}$**
- Special case $k = 2$: reduces exactly to the 2-way Ncut problem and its relaxation