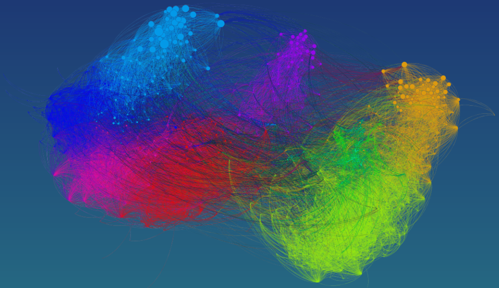
# Graphs in Machine Learning

Michal Valko

*Inria Lille - Nord Europe, France*

TA: Pierre Perrault

Partially based on material by: Ulrike von Luxburg, Gary Miller, Doyle & Schnell, Daniel Spielman



# Previous lecture

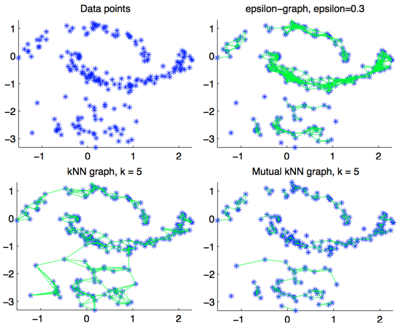
- where do the graphs come from?
  - social, information, utility, and biological networks
  - we create them from the flat data
  - random graph models
- specific applications and concepts
  - maximizing influence on a graph **gossip propagation, submodularity**, proof of the approximation guarantee
  - Google pagerank **random surfer process, steady state vector, sparsity**
  - online semi-supervised learning **label propagation, backbone graph, online learning, combinatorial sparsification, stability analysis**
  - Erdős number project, real-world graphs, **heavy tails, small world** – when did this happen?
- similarity graphs
  - different types
  - construction
  - practical considerations

# This lecture

- **Laplacians** and their properties
- spectral graph theory
- random walks
- recommendation on a bipartite graph
- resistive networks
  - recommendation score as a resistance?
  - Laplacian and resistive networks
  - resistance distance and random walks
- PS: some students have started working on their projects already

# Similarity Graphs: $\varepsilon$ or $k$-NN?

## DEMO IN CLASS




http://www.ml.uni-saarland.de/code/GraphDemo/DemoSpectralClustering.htm

http://www.tml.cs.uni-tuebingen.de/team/luxburg/publications/Luxburg07_tutorial.pdf

# Generic Similarity Functions

Gaussian similarity function/Heat function/RBF:

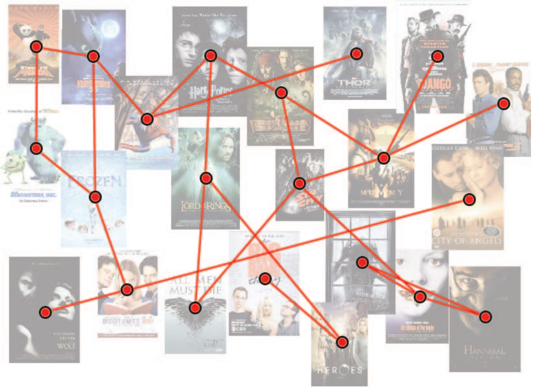
$$s_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

Cosine similarity function:

$$s_{ij} = \cos(\theta) = \left(\frac{\mathbf{x}_i^\mathsf{T}\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)$$

Typical Kernels

# Similarity Graphs




$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

# Sources of Real Networks

- **http://snap.stanford.edu/data/**
- http://www-personal.umich.edu/~mejn/netdata/
- http://proj.ise.bgu.ac.il/sns/datasets.html
- http://www.cise.ufl.edu/research/sparse/matrices/
- http://vlado.fmf.uni-lj.si/pub/networks/data/default.htm

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

# graph Laplacian

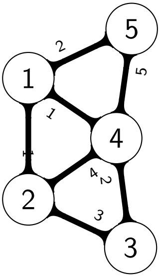
...the only matrix that matters

# Graph Laplacian

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - with a set of **nodes** $\mathcal{V}$ and a set of **edges** $\mathcal{E}$

| | |
|---|---|
| $\mathbf{A}$ | adjacency matrix |
| $\mathbf{W}$ | weight matrix |
| $\mathbf{D}$ | (diagonal) degree matrix |
| $\mathbf{L} = \mathbf{D} - \mathbf{W}$ | graph **Laplacian** matrix |

$$\mathbf{L} = \begin{pmatrix} 4 & -1 & 0 & -1 & -2 \\ -1 & 8 & -3 & -4 & 0 \\ 0 & -3 & 5 & -2 & 0 \\ -1 & -4 & -2 & 12 & -5 \\ -2 & 0 & 0 & -5 & 7 \end{pmatrix}$$




$\mathbf{L}$ is SDD!

# Properties of Graph Laplacian

**Graph function**: a vector $\mathbf{f} \in \mathbb{R}^N$ assigning values to nodes:

$$\mathbf{f} : \mathcal{V}(\mathcal{G}) \to \mathbb{R}.$$

$$\mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j \leq N} w_{i,j}(f_i - f_j)^2 = S_G(\mathbf{f})$$

# Recap: Eigenwerte und Eigenvektoren

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** $\lambda$

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

If $(\lambda_1, \mathbf{v}_1)$ are $(\lambda_2, \mathbf{v}_2)$ **eigenpairs** for symmetric $\mathbf{M}$ with $\lambda_1 \neq \lambda_2$ then $\mathbf{v}_1 \perp \mathbf{v}_2$, i.e., $\mathbf{v}_1^\mathsf{T}\mathbf{v}_2 = 0$.

If $(\lambda, \mathbf{v}_1)$, $(\lambda, \mathbf{v}_2)$ are eigenpairs for $\mathbf{M}$ then $(\lambda, \mathbf{v}_1 + \mathbf{v}_2)$ is as well.

For symmetric $\mathbf{M}$, the **multiplicity** of $\lambda$ is the dimension of the space of eigenvectors corresponding to $\lambda$.

$N \times N$ symmetric matrix has $N$ eigenvalues (w/ multiplicities).

# Eigenvalues, Eigenvectors, and Eigendecomposition

A vector $\mathbf{v}$ is an **eigenvector** of matrix $\mathbf{M}$ of **eigenvalue** $\lambda$

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}.$$

Vectors $\{\mathbf{v}_i\}_i$ form an **orthonormal** basis with $\lambda_1 \leq \lambda_2 \leq \dots \lambda_N$.

$$\forall i \quad \mathbf{M}\mathbf{v}_i = \lambda_i\mathbf{v}_i \qquad \equiv \qquad \mathbf{MQ} = \mathbf{Q\Lambda}$$

$\mathbf{Q}$ has eigenvectors in columns and $\mathbf{\Lambda}$ has eigenvalues on its diagonal.

Right-multiplying $\mathbf{MQ} = \mathbf{Q\Lambda}$ by $\mathbf{Q}^\mathsf{T}$ we get the **eigendecomposition** of $\mathbf{M}$:

$$\mathbf{M} = \mathbf{MQQ}^\mathsf{T} = \mathbf{Q\Lambda Q}^\mathsf{T} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\mathsf{T}$$

# $\mathbf{M} = \mathbf{L}$: Properties of Graph Laplacian

We can assume **non-negative weights**: $w_{ij} \geq 0$.

**L** is symmetric

**L** positive semi-definite $\leftarrow \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j\leq N} w_{i,j}(f_i - f_j)^2$

Recall: If $\mathbf{L}\mathbf{f} = \lambda\mathbf{f}$ then $\lambda$ is an **eigenvalue** (of the Laplacian).

The smallest eigenvalue of **L** is 0. Corresponding eigenvector: $\mathbf{1}_N$.

All eigenvalues are non-negative reals $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.

Self-edges do not change the value of **L**.

# Properties of Graph Laplacian

The multiplicity of eigenvalue 0 of $\mathbf{L}$ equals to the number of connected components. The eigenspace of 0 is spanned by the components' indicators.

Proof: If $(0, \mathbf{f})$ is an eigenpair then $0 = \frac{1}{2}\sum_{i,j\leq N} w_{i,j}(f_i - f_j)^2$. Therefore, $\mathbf{f}$ is constant on each connected component. If there are $k$ components, then $\mathbf{L}$ is $k$-block-diagonal:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{bmatrix}$$

For block-diagonal matrices: the spectrum is the union of the spectra of $\mathbf{L}_i$ (eigenvectors of $\mathbf{L}_i$ padded with zeros elsewhere).

For $\mathbf{L}_i$ $(0, \mathbf{1}_{|V_i|})$ is an eigenpair, hence the claim.

# Smoothness of the Function and Laplacian

- $\mathbf{f} = (f_1, \ldots, f_N)^\mathsf{T}$: graph function
- Let $\mathbf{L} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\mathsf{T}$ be the eigendecomposition of the Laplacian.
  - Diagonal matrix $\boldsymbol{\Lambda}$ whose diagonal entries are eigenvalues of $\mathbf{L}$.
  - Columns of $\mathbf{Q}$ are eigenvectors of $\mathbf{L}$.
  - Columns of $\mathbf{Q}$ form a basis.
- $\boldsymbol{\alpha}$: Unique vector such that $\mathbf{Q}\boldsymbol{\alpha} = \mathbf{f}$ Note: $\mathbf{Q}^\mathsf{T}\mathbf{f} = \boldsymbol{\alpha}$

**Smoothness of a graph function $S_G(\mathbf{f})$**

$$S_G(\mathbf{f}) = \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \mathbf{f}^\mathsf{T}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\mathsf{T}\mathbf{f} = \boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_{\boldsymbol{\Lambda}}^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

**Smoothness and <u>regularization</u>:** Small value of

**(a)** $S_G(\mathbf{f})$ **(b)** $\boldsymbol{\Lambda}$ norm of $\boldsymbol{\alpha}^\star$ **(c)** $\alpha_i^\star$ for large $\lambda_i$

# Smoothness of the Function and Laplacian

$$S_G(\mathbf{f}) = \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \mathbf{f}^\mathsf{T}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\mathsf{T}\mathbf{f} = \boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{\alpha} = \|\boldsymbol{\alpha}\|_{\boldsymbol{\Lambda}}^2 = \sum_{i=1}^{N} \lambda_i \alpha_i^2$$

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

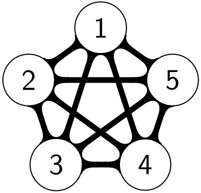
Spectral coordinates of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\mathsf{T}\mathbf{v}_k = \mathbf{e}_k$

$$S_G(\mathbf{v}_k) = \mathbf{v}_k^\mathsf{T}\mathbf{L}\mathbf{v}_k = \mathbf{v}_k^\mathsf{T}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\mathsf{T}\mathbf{v}_k = \mathbf{e}_k^\mathsf{T}\boldsymbol{\Lambda}\mathbf{e}_k = \|\mathbf{e}_k\|_{\boldsymbol{\Lambda}}^2 = \sum_{i=1}^{N} \lambda_i (\mathbf{e}_k)_i^2 = \lambda_k$$

The smoothness of $k$-th eigenvector is the $k$-th eigenvalue.

# Laplacian of the Complete Graph $K_N$

What is the eigenspectrum of $\mathbf{L}_{K_N}$?




$$\mathbf{L}_{K_N} = \begin{pmatrix} N-1 & -1 & -1 & -1 & -1 \\ -1 & N-1 & -1 & -1 & -1 \\ -1 & -1 & N-1 & -1 & -1 \\ -1 & -1 & -1 & N-1 & -1 \\ -1 & -1 & -1 & -1 & N-1 \end{pmatrix}$$

From before: we know that $(0, \mathbf{1}_N)$ is an eigenpair.

If $\mathbf{v} \neq 0_N$ and $\mathbf{v} \perp \mathbf{1}_N \implies \sum_i \mathbf{v}_i = 0$. To get the other eigenvalues, we compute $(\mathbf{L}_{K_N}\mathbf{v})_1$ and divide by $\mathbf{v}_1$ (wlog $\mathbf{v}_1 \neq 0$).

$$(\mathbf{L}_{K_N}\mathbf{v})_1 = (N-1)\mathbf{v}_1 - \sum_{i=2}^{N} \mathbf{v}_i = N\mathbf{v}_1.$$

What are the remaining eigenvalues/vectors?

# Normalized Laplacians

$$\mathbf{L}_{un} = \mathbf{D} - \mathbf{W}$$
$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$
$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

$$\mathbf{f}^{\mathsf{T}}\mathbf{L}_{sym}\mathbf{f} = \frac{1}{2}\sum_{i,j\leq N} w_{i,j}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{D}^{1/2}\mathbf{u})$ is an eigenpair for $\mathbf{L}_{sym}$

# Normalized Laplacians

$\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ are PSD with non-negative real eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$$

$(\lambda, \mathbf{u})$ is an eigenpair for $\mathbf{L}_{rw}$ iff $(\lambda, \mathbf{u})$ solve the generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$.

$(0, \mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{rw}$.

$(0, \mathbf{D}^{1/2}\mathbf{1}_N)$ is an eigenpair for $\mathbf{L}_{sym}$.

Multiplicity of eigenvalue 0 of $\mathbf{L}_{rw}$ or $\mathbf{L}_{sym}$ equals to the number of connected components.

Proof: As for $\mathbf{L}$.

# Laplacian and Random Walks on Undirected Graphs

- stochastic process: vertex-to-vertex jumping
- transition probability $v_i \to v_j$ is $p_{ij} = w_{ij}/d_i$
  - $d_i \stackrel{\text{def}}{=} \sum_j w_{ij}$
- transition matrix $\mathbf{P} = (p_{ij})_{ij} = \mathbf{D}^{-1}\mathbf{W}$ (notice $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$)
- if $G$ is connected and non-bipartite $\to$ unique **stationary distribution** $\pi = (\pi_1, \pi_2, \pi_3, \dots, \pi_N)$ where $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(\mathbf{W}) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$
- $\pi = \frac{\mathbf{1}^\mathsf{T}\mathbf{W}}{\mathrm{vol}(\mathbf{W})}$ verifies $\pi\mathbf{P} = \pi$ as:

$$\pi\mathbf{P} = \frac{\mathbf{1}^\mathsf{T}\mathbf{W}\mathbf{P}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\mathsf{T}\mathbf{D}\mathbf{P}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\mathsf{T}\mathbf{D}\mathbf{D}^{-1}\mathbf{W}}{\mathrm{vol}(\mathbf{W})} = \frac{\mathbf{1}^\mathsf{T}\mathbf{W}}{\mathrm{vol}(\mathbf{W})} = \pi$$

What's the difference from the PageRank™?

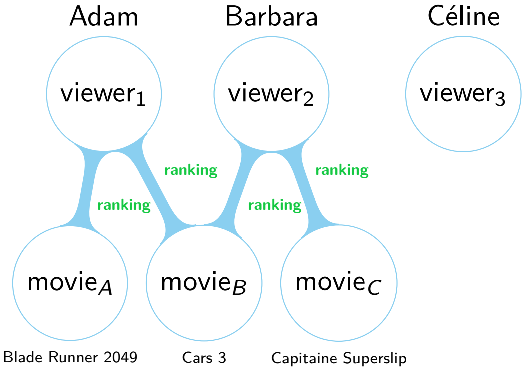
$$\mathrm{score}(v, m)$$

# recommendation on a bipartite graph

...with the graph Laplacian!

# Use of Laplacians: Movie recommendation

How to do movie recommendation on a bipartite graph?




Question: *Do we recommend Capitaine Superslip to Adam?*

Let's compute some $\mathrm{score}(v, m)$!

# Use of Laplacians: Movie recommendation

How to compute the $\text{score}(v, m)$? Using some **graph distance**!

**Idea$_1$: maximally weighted path**

$\text{score}(v, m) = \max_{vPm} \text{weight}(P) = \max_{vPm} \sum_{e \in P} \text{ranking}(e)$

**Idea$_2$: change the path weight**

$\text{score}_2(v, m) = \max_{vPm} \text{weight}_2(P) = \max_{vPm} \min_{e \in P} \text{ranking}(e)$

**Idea$_3$: consider everything**

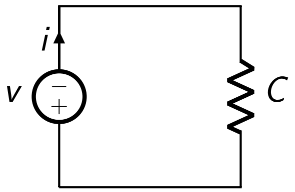
$\text{score}_3(v, m) = \text{max flow from } m \text{ to } v$

# Laplacians and Resistive Networks

How to compute the $\text{score}(v, m)$?

**Idea$_4$: view edges as conductors**

$\text{score}_4(v, m)$ = effective resistance between $m$ and $v$




$C \equiv$ conductance

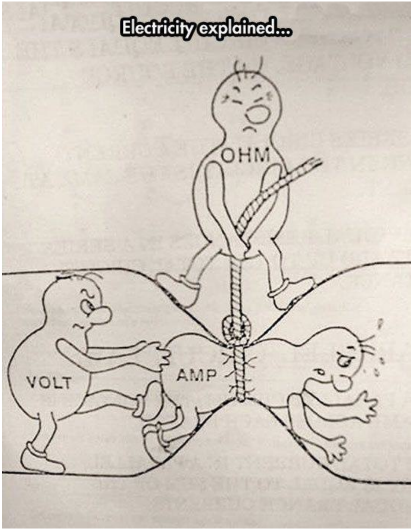
$R \equiv$ resistance

$i \equiv$ current

$V \equiv$ voltage

$$C = \frac{1}{R} \qquad i = CV = \frac{V}{R}$$

# Resistive Networks: Some high-school physics



# Resistive Networks

**resistors in series**

$$R = R_1 + \cdots + R_n \qquad C = \frac{1}{\frac{1}{C_1} + \cdots + \frac{1}{C_N}} \qquad i = \frac{V}{R}$$

**conductors in parallel**

$$C = C_1 + \cdots + C_N \qquad i = VC$$

**Effective Resistance on a graph**

Take two nodes: $a \neq b$. Let $V_{ab}$ be the voltage between them and $i_{ab}$ the current between them. Define $R_{ab} = \frac{V_{ab}}{i_{ab}}$ and $C_{ab} = \frac{1}{R_{ab}}$.

We treat the entire graph as a resistor!

# Resistive Networks: Optional Homework (ungraded)

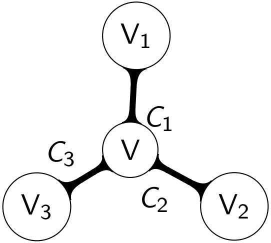
Show that $R_{ab}$ is a metric space.

1. $R_{ab} \geq 0$
2. $R_{ab} = 0$ iff $a = b$
3. $R_{ab} = R_{ba}$
4. $R_{ac} \leq R_{ab} + R_{bc}$

The effective resistance is a distance!

# How to compute effective resistance?

Kirchhoff's Law $\equiv$ flow in $=$ flow out




$V = \frac{C_1}{C} V_1 + \frac{C_2}{C} V_2 + \frac{C_3}{C} V_3$ (convex combination)

residual current $= CV - C_1 V_1 - C_2 V_2 - C_3 V_3$

Kirchhoff says: This is zero! **There is no residual current!**

# Resistors: Where is the link with the Laplacian?

General case of the previous! $d_i = \sum_j c_{ij} =$ sum of conductances

$$\mathbf{L}_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -c_{ij} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

$\mathbf{v}$ = **voltage setting** of the nodes on graph.

$(\mathbf{Lv})_i$ = residual current at $\mathbf{v}_i$ — as we derived

**Use**: setting voltages and getting the current

**Inverting** $\equiv$ injecting current and getting the voltages

The net injected has to be zero $\equiv$ Kirchhoff's Law.

# Resistors and the Laplacian: Finding $R_{ab}$

Let's calculate $R_{1N}$ to get the **movie recommendation score**!

$$\mathbf{L}\begin{pmatrix} 0 \\ v_2 \\ \vdots \\ v_{n-1} \\ 1 \end{pmatrix} = \begin{pmatrix} i \\ 0 \\ \vdots \\ 0 \\ -i \end{pmatrix}$$

$$i = \frac{V}{R} \qquad V = 1 \qquad R = \frac{1}{i}$$

Return $R_{1N} = \dfrac{1}{i}$

Doyle and Snell: Random Walks and Electric Networks

https://math.dartmouth.edu/~doyle/docs/walks/walks.pdf

# Resistors and the Laplacian: Finding $R_{1N}$

$\mathbf{Lv} = (i, 0, \ldots, -i)^\mathsf{T} \equiv$ **boundary valued problem**

For $R_{1N}$

$V_1$ and $V_N$ are the **boundary**

$(v_1, v_2, \ldots, v_N)$ is **harmonic:**

$V_i \in$ **interior** (not boundary)

$V_i$ is a **convex combination of its neighbors**

# Resistors and the Laplacian: Finding $R_{1n}$

From the properties of electric networks (cf. Doyle and Snell) we inherit the useful properties of the Laplacians!

**Example:** Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions (later in the course)

**Maximum Principle**

If $\mathbf{f} = \mathbf{v}$ is harmonic then min and max are on the boundary.

**Uniqueness Principle**

If $\mathbf{f}$ and $\mathbf{g}$ are harmonic with the same boundary then $\mathbf{f} = \mathbf{g}$

# Resistors and the Laplacian: Finding $R_{1N}$

Alternative method to calculate $R_{1N}$:

$$\mathbf{L}\mathbf{v} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \stackrel{\text{def}}{=} \mathbf{i}_{\text{ext}} \quad \text{Return} \quad R_{1N} = v_1 - v_N$$

Why?

**Question:** Does $\mathbf{v}$ exist? $\mathbf{L}$ does not have an inverse :(.
**Not unique:** $\mathbf{1}$ in the nullspace of $\mathbf{L}$ : $\mathbf{L}(\mathbf{v} + c\mathbf{1}) = \mathbf{L}\mathbf{v} + c\mathbf{L}\mathbf{1} = \mathbf{L}\mathbf{v}$
**Moore-Penrose pseudo-inverse** solves LS
**Solution:** Instead of $\mathbf{v} = \mathbf{L}^{-1}\mathbf{i}_{\text{ext}}$ we take $\mathbf{v} = \mathbf{L}^{+}\mathbf{i}_{\text{ext}}$
**We get:** $R_{1N} = v_1 - v_N = \mathbf{i}_{\text{ext}}^{\mathsf{T}}\mathbf{v} = \mathbf{i}_{\text{ext}}^{\mathsf{T}}\mathbf{L}^{+}\mathbf{i}_{\text{ext}}$.
**Notice:** We can reuse $\mathbf{L}^{+}$ to get resistances for any pair of nodes!

# What? A pseudo-inverse?

Eigendecomposition of the Laplacian:

$$\mathbf{L} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\mathsf{T} = \sum_{i=1}^{N} \lambda_i \mathbf{q}_i \mathbf{q}_i^\mathsf{T} = \sum_{i=2}^{N} \lambda_i \mathbf{q}_i \mathbf{q}_i^\mathsf{T}$$

Pseudo-inverse of the Laplacian:

$$\mathbf{L}^{+} = \mathbf{Q}\boldsymbol{\Lambda}^{+}\mathbf{Q}^\mathsf{T} = \sum_{i=2}^{N} \frac{1}{\lambda_i} \mathbf{q}_i \mathbf{q}_i^\mathsf{T}$$

**Moore-Penrose pseudo-inverse** solves a least squares problem:

$$\mathbf{v} = \arg\min_{\mathbf{x}} \left\| \mathbf{L}\mathbf{x} - \mathbf{i}_{\text{ext}} \right\|_2 = \mathbf{L}^{+}\mathbf{i}_{\text{ext}}$$

$$\mathrm{cut}(A, B) = \frac{1}{2}\mathbf{f}^\top \mathbf{L}\mathbf{f}$$

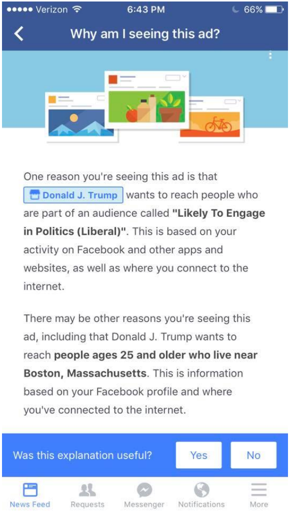
# spectral clustering

...with connectivity beyond compactness

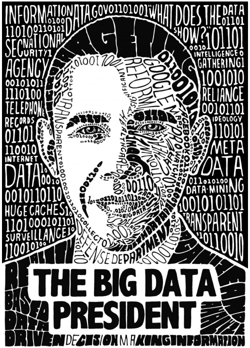
# How to rule the world?

**Let's make France great again!**

# How to rule the world?



# How to rule the world: “AI” is here

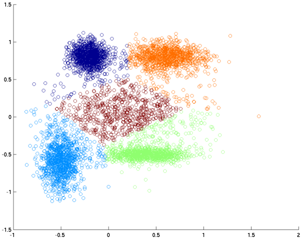


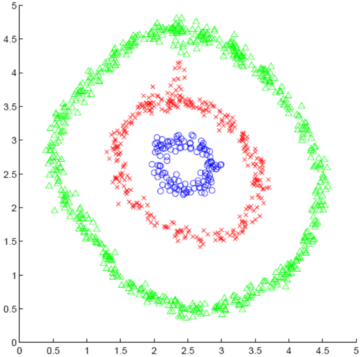

https://www.washingtonpost.com/opinions/obama-the-big-data-president/2013/06/14/1d71fe2e-d391-11e2-b05f-3ea3f0e7bb5a_story.html

https://www.technologyreview.com/s/509026/how-obamas-team-used-big-data-to-rally-voters/

Talk of Rayid Ghaniy: https://www.youtube.com/watch?v=gDM1GuszM_U

# Application of Graphs for ML: Clustering



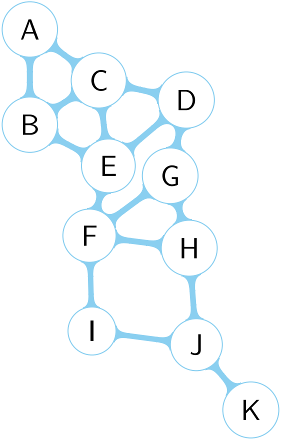
# Application: Clustering - Recap

- What do we know about the **clustering** in general?
  - ill defined problem (different tasks $\rightarrow$ different paradigms)
  - "I know it when I see it"
  - inconsistent (wrt. Kleinberg's axioms)

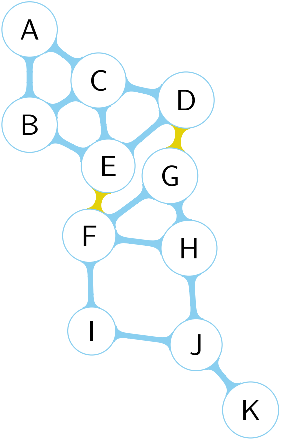
  - number of clusters $k$ need often be known
  - difficult to evaluate

- What do we know about **$k$-means**?
  - "hard" version of EM clustering
  - sensitive to initialization
  - optimizes for **compactness**
  - yet: algorithm-to-go

# Spectral Clustering: Cuts on graphs

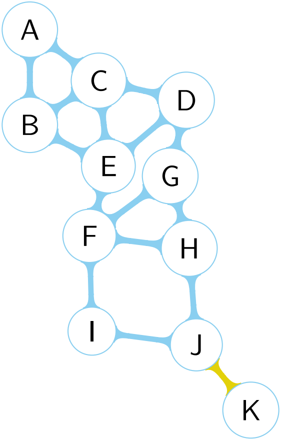


# Spectral Clustering: Cuts on graphs




Defining the cut objective we get the clustering!

# Spectral Clustering: Cuts on graphs

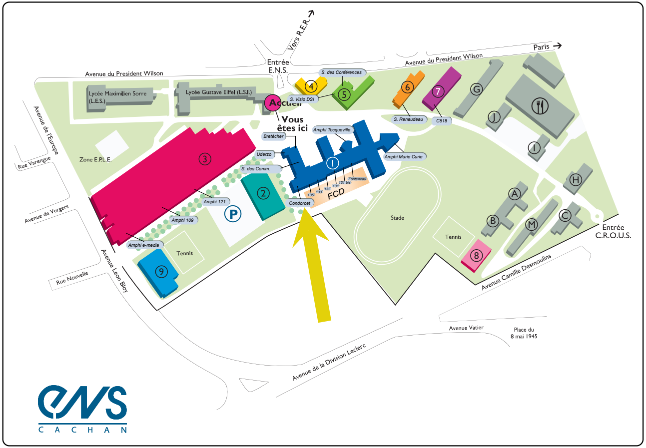



**MinCut**: $\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$

Are we done?

Can be solved efficiently, but maybe not what we want . . . .

# Next class on Wednesday, October 17th at 14:00!



*Michal Valko*

michal.valko@inria.fr

ENS Paris-Saclay, MVA 2018/2019

SequeL team, Inria Lille — Nord Europe

https://team.inria.fr/sequel/