Graphs in Machine Learning

Michal Valko

INRIA Lille - Nord Europe, France

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

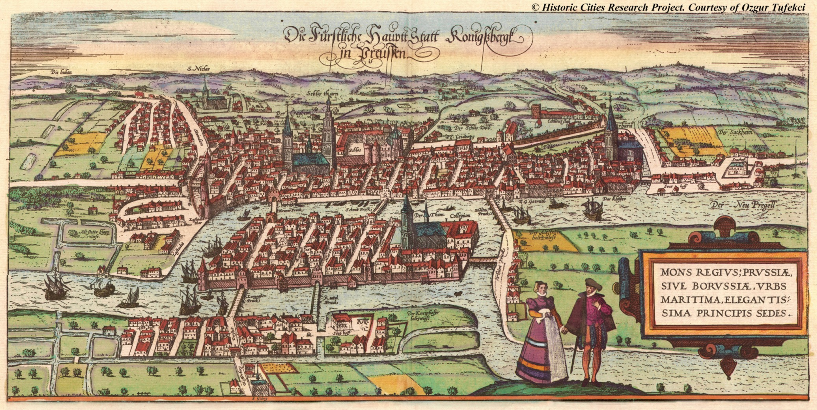
Previous Lecture

- 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**
  - 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 **heavy tails, small world**

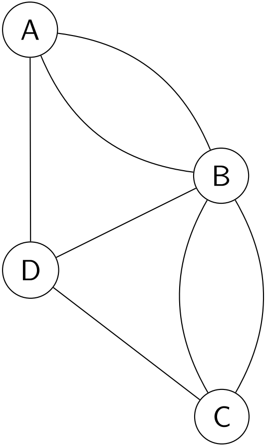
This Lecture

- similarity graphs
  - different types
  - construction
  - practical considerations
- spectral graph theory
- Laplacians and their properties
- random walks
- resistive networks

Graph theory refresher



Graph theory refresher



Graph theory refresher

- 250 years of graph theory
- Seven Bridges of Königsberg (Leonhard Euler, 1735)
- necessary for Eulerian circuit: 0 or 2 nodes of odd degree
- after bombing and rebuilding there are now 5 bridges in Kaliningrad for the nodes with degrees $[2, 2, 3, 3]$
- the original problem is solved but not practical
  `http://people.engr.ncsu.edu/mfms/SevenBridges/`

Similarity Graphs

Input: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n$

- raw data
- flat data
- vectorial data

Similarity Graphs

Similarity graph: $G = (V, E)$ — **(un)weighted**

Task 1: For each pair i, j: define a **similarity function** s_{ij}

Task 2: Decide which edges to include

ε-neighborhood graphs – connect the points with the distances smaller than ε

k-NN neighborhood graphs – take k nearest neighbors

Fully connected graphs - consider everything

This is art (not much theory exists).

http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/publications/Luxburg07_tutorial.pdf

Similarity Graphs: ε-neighborhood graphs

Edges connect the points with the distances smaller than ε.

- distances are roughly on the same scale (ε)
- weights may not bring additional info $\rightarrow$ unweighted
- equivalent to: similarity function is at least ε
- theory [Penrose, 1999]: $\varepsilon = ((\log n)/n)^d$ to guarantee connectivity n nodes, d dimension
- practice: choose ε as the length of the longest edge in the MST - minimum spanning tree
  - Q: What could be the problem with this approach?
  - A: Anomalies can make ε too large.

Similarity Graphs: k-nearest neighbors graphs

Edges connect each node to its k-nearest neighbors.

- asymmetric (or directed graph)
  - option OR: ignore the direction
  - option AND: include if we have both direction (mutual k-NN)
- $k \approx \log n$ - suggested by asymptotics (practice: up to $\sqrt{n}$)
- for mutual k-NN we need to take larger k
- mutual k-NN does not connect regions with different density
- how to chose k?
- why don't we take $k = n - 1$?
  - space and time
  - manifold considerations (preserving local properties)

Similarity Graphs: Fully connected graphs

Edges connect everything.

- choose a "meaningful" similarity function s
- default choice:

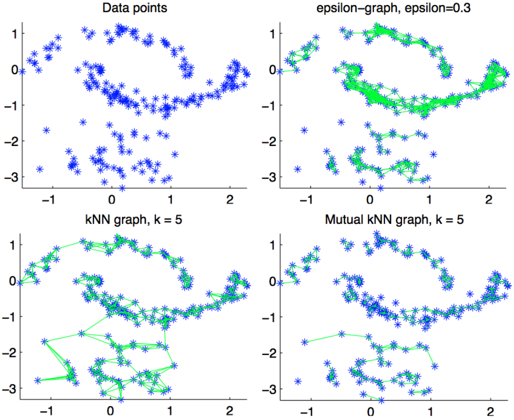
$$s_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- why the exponential decay with the distance?
- σ controls the width of the neighborhoods
  - similar role as ε
  - a practical rule of thumb: 10% of the average empirical std
  - learn σ_i for each feature independently
- metric learning (a whole field of ML)

Similarity Graphs: Important considerations

- *calculate all* s_{ij} *and threshold* has its limits ($n \approx 10000$)
- graph construction step can be a huge bottleneck
- want to go higher? (we often have to)
  - down-sample
  - approximate NN
    - **LSH** - Locally Sensitive Hashing
    - **CoverTrees**
  - sometime we may not need the graph (just the final results)
  - yet another story: when we start with a large graph and want to make it sparse (later in the course)
- these rules have little theoretical underpinning
- similarity is very data-dependent

Similarity Graphs: ε or k-NN?



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

http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/publications/Luxburg07_tutorial.pdf

Generic Similarity Functions

Gaussian similarity function/Heat function/RBF:

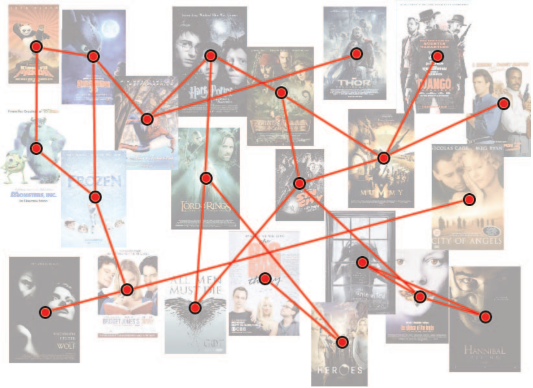
$$s_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

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



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

Eigenwerte und Eigenvektoren

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

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

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., $\mathbf{v}_1^\mathsf{T}\mathbf{v}_2 = 0$.

Proof: $\lambda_1\mathbf{v}_1^\mathsf{T}\mathbf{v}_2 = \mathbf{v}_1^\mathsf{T}\mathbf{M}\mathbf{v}_2 = \mathbf{v}_1^\mathsf{T}\lambda_2\mathbf{v}_2 = \lambda_2\mathbf{v}_1^\mathsf{T}\mathbf{v}_2 \implies \mathbf{v}_1^\mathsf{T}\mathbf{v}_2 = 0$

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

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

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

Eigenvalues and Eigenvectors

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

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

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

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

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

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

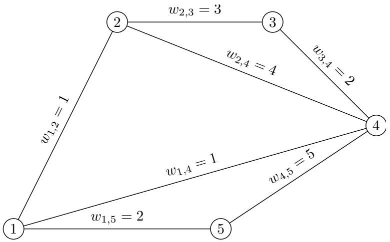
$$\mathbf{M} = \mathbf{M}\mathbf{V}\mathbf{V}^\mathsf{T} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\mathsf{T} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\mathsf{T}$$

Graph Laplacian

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

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

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



Properties of Graph Laplacian

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

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

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

Proof:

$$\mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \mathbf{f}^\mathsf{T}\mathbf{D}\mathbf{f} - \mathbf{f}^\mathsf{T}\mathbf{W}\mathbf{f} = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j\leq n} w_{i,j} f_i f_j$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j\leq n} w_{i,j} f_i f_j + \sum_{j=1}^{n} d_j f_j^2\right) = \frac{1}{2}\sum_{i,j\leq n} w_{i,j}(f_i - f_j)^2$$

Properties of Graph Laplacian

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

$\mathbf{L}$ is symmetric

$\mathbf{L}$ 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 λ is an **eigenvalue**.

The smallest eigenvalue of $\mathbf{L}$ 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 $\mathbf{L}$.

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 the 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: $\mathbf{Q}^\mathsf{T}\mathbf{f} = \boldsymbol{\alpha}$

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

Smoothness and regularization: Small value of

(a) $S_G(f)$ **(b)** $\boldsymbol{\Lambda}$ norm of $\boldsymbol{\alpha}^*$ **(c)** α_i^* for large λ_i

Smoothness of the Function and Laplacian

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

Eigenvectors are graph functions too!

What is the smoothness of an eigenvector?

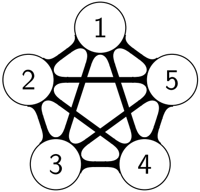
Spectral coordinate of eigenvector $\mathbf{v}_k$: $\mathbf{Q}^\mathsf{T}\mathbf{v}_k = \mathbf{e}_k$

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

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?

Answer: $n-1$ eigenvectors $\perp \mathbf{1}_n$ for eigenvalue n with multiplicity $n-1$.

Question: What changes for weighted complete graphs?

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 $\mathbf{L}$.

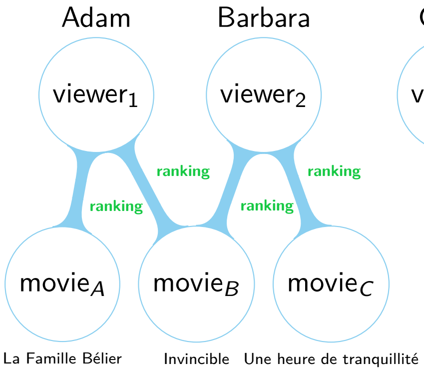
Laplacian and Random Walks on Undirected Graphs

- 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 $\pi_i = d_i/\mathrm{vol}(V)$
  - $\mathrm{vol}(G) = \mathrm{vol}(V) = \mathrm{vol}(W) \stackrel{\text{def}}{=} \sum_i d_i = \sum_{i,j} w_{ij}$
- $\pi = \frac{\mathbf{1}^\mathsf{T}\mathbf{W}}{\mathrm{vol}(\mathbf{W})}$ verifies $\pi\mathbf{P} = \pi$ as:

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

Use of Laplacians: Movie recommendation

Movie recommendation on a bipartite graph



Question: *Do we recommend Une heure de tranquillité to Adam?*

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

Use of Laplacians: Movie recommendation

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

Idea$_1$: maximally weighted path

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

Problem: If there is a weak edge, then the path is not good.

Idea$_2$: change the path weight

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

Problem of 1&2: Additional paths does not improve the score.

Idea$_3$: consider everything

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

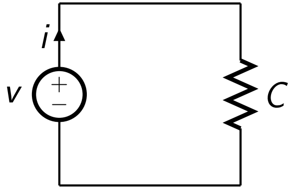
Problem of 3: Shorter paths do not improve the score.

Laplacians and Resistive Networks

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

Idea$_4$: view edges as conductors

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



$C \equiv \mathrm{conductance}$

$R \equiv \mathrm{resistance}$

$i \equiv \mathrm{current}$

$V \equiv \mathrm{voltage}$

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

Michal Valko
michal.valko@inria.fr
sequel.lille.inria.fr