Semi-Supervised Learning on Relational Data

François Théberge
theberge@ieee.org

August 2019

Transductive Learning Framework

We use a **transductive** learning approach:

- no model is explicitly constructed
- learning is “on the data”
- we use a regularization framework on graphs
- regularization captures both:
  - **local** structure: consistency with scarce labeled vertices
  - **global** structure: smoothness over all vertices

Transductive Learning Framework

Let $G = (V, E)$ with $|V| = n$ and $E \subset V \times V$

Let $f : V \to \mathbb{R}$ with $\langle f, g \rangle = \sum_{v \in V} f(v)g(v)$

We seek a function $f^* : V \to \mathbb{R}$ such that

$$f^* = \operatorname*{argmin}_{f \in H(V)} \left(\Omega(f) + \mu ||f - y||^2 \right)$$

$\Omega(f)$ is a functional that depends on G

y encodes prior knowledge (vertex labels)

μ: tradeoff between **smoothness** and **consistency.**

Transductive Learning Framework

$\Omega(f)$ depends on the type of graph:

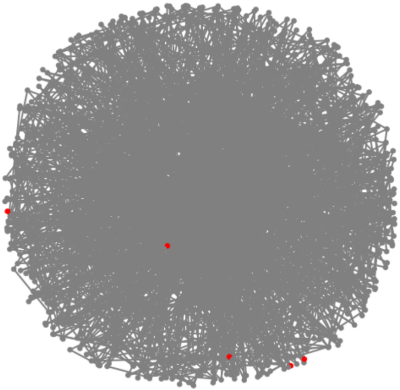
- undirected
- directed
- co-linkage (bipartite-like)

y depends on the problem to solve:

- binary classification: $y \in \{-1, 0, 1\}$
- ranking: $y \in \{0, 1\}$
- unsupervised: $y \in \{0\}$

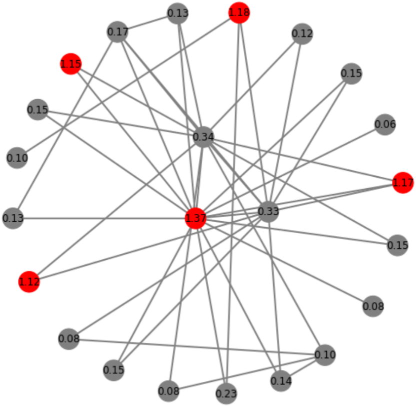
Application - Example

Given a large graph with a few “interesting” entities:



Application - Example

Solve f^* to zoom in on nearby vertices:



Application - Example

- obtain a *ranking* of unknown entities
- can *visualize* key subgraphs
- several applications in cyber-security context
  - anomaly detection
  - malware detection
- graphs and hypergraphs are often too big for direct analysis or visualization

Undirected graph

Let $G = (V, E)$ with W be the matrix of edge weights $w(u, v)$ for all $(u, v) \in E$.

Let D be the diagonal matrix of node degrees: $d(v) = \sum_{u \sim v} w(u, v)$

We first review the (unsupervised) Ncut problem

Ref: Ulrike von Luxburg, *A Tutorial on Spectral Clustering*, Technical Report No. TR-149, Max Planck Inst., Germany, 2006.

Undirected graph

For a partition $V = S \cup S^c$:

$$Ncut(S, S^c) = \frac{Vol\partial S}{VolS} + \frac{Vol\partial S}{VolS^c}$$

where:

$\partial S = \{e \in E; |e \cap S| = |e \cap S^c| = 1\}$
$Vol(S) = \sum_{v \in S} d(v)$
$Vol(\partial S) = \sum_{(u,v) \in \partial S} w(u, v)$

This can be viewed as a random walk with transition probabilities $P = D^{-1}W$:

$$Ncut(S, S^c) = P(S|S^c) + P(S^c|S)$$

.

Undirected graph

The problem can be solved by relaxing over real values

$$f^* = \operatorname*{argmin}_{f \in \mathbb{R}^n} \Omega(f); \quad f \perp D^{1/2} \cdot 1, \ ||f||^2 = vol(V).$$

where $\Omega(f) = \langle f^t, \Delta f \rangle$ and

$$\Delta = I - D^{-1/2} W D^{-1/2}$$

is the (normalized) graph Laplacian.

This is known as *normalized spectral clustering* (section 5.3 in von Luxburg).

Undirected graph

The Laplacian also appears in semi-supervised problems.

If nodes are close (large $w(u, v)$), then they should have similar labels ($f(u) \approx f(v)$) to keep $w(u, v)(f(u) - f(v))^2$ small.

On the entire graph, this amounts to keeping $\Omega(f)$ small.

This can be seen as finding a “smooth” function f that has little variation in dense regions of the graph, but which can vary more in sparse regions.

Undirected graph

Now assume some initial (seed) values y on the vertices.

Define a semi-supervised problem as a trade-off between ”smoothness” with respect to the graph topology, and consistency with respect to y, such as:

$$f^* = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \left(\Omega(f) + \mu ||f - y||^2 \right)$$

Ref: D. Zhou and B. Schölkopf, *A Regularization Framework for Learning from Graph Data*, 2004.

Undirected graph

Let $\Omega(f) = \langle f^t, \Delta f \rangle$, then we can show that:

$$\Omega(f) = \frac{1}{2} \sum_{(u,v) \in E} w(u,v) \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2$$

and if $y \neq 0$, and $\mu > 0$, there exists a closed form solution:

$$f^* = \mu(\Delta + \mu I)^{-1} y = (1 - \alpha)(I - \alpha S)^{-1} y$$

where:

$$\alpha = (1 + \mu)^{-1}$$

$\Delta = I - D^{-1/2} W D^{-1/2}$, the normalized graph Laplacian

$S = I - \Delta$, the smoothness matrix

Undirected graph

f^* can be obtained in various ways:

- iterative method:
  1. start from $f(v) = y$,
  2. iterate $f(v) \leftarrow \alpha(Sf)(v) + (1-\alpha)y, \forall v$.
- it can be written as a *diagonally dominant* linear problem, for which an inversion technique exist with complexity $O(m^{1.31})$, with m the number of non-zero entries
- via a *conjugate gradient* method
- the *map-reduce* framework allows good scalability

Directed graphs

Define the in and out node degrees:
$d_-(v) = \sum_u w(u,v), \quad d_+(v) = \sum_u w(v,u)$.
The natural random walk on V has transition probabilities:

$$p(u,v) = \begin{cases} \frac{w(u,v)}{d^+(u)}, & (u,v) \in E \\ 0, & \text{else.} \end{cases}$$

Let π be the unique stationary distribution, i.e.

$$\pi(v) = \sum_{u \to v} \pi(u) p(u,v).$$

This may require defining a teleporting random walk.

Directed graphs

With the functional:

$$\Omega(f) := \frac{1}{2} \sum_{e=(u,v)} \pi(u) p(u,v) \left(\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}} \right)^2$$

the regularization problem is as before, with

$$\triangle = I - S,\ S = \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^t \Pi^{1/2}}{2}$$

where P is the matrix of transition probabilities and Π is the diagonal matrix of stationary probabilities.

We get as before: $f^* = (1-\alpha)(I - \alpha S)^{-1} y$.

Directed graphs

This is a generalization of the *undirected* case. For an undirected graph, the random walk has stationary probabilities:

$$\pi(v) = \frac{d(v)}{\sum_u d(u)}$$

and we get

$$S = D^{-1/2} W D^{-1/2}$$

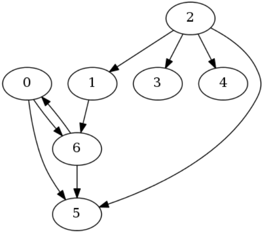
as before.

Hubs&Authorities graphs

Consider two possible roles for a vertex $v \in V$:

- an *authority*, with a high "in-degree", or
- a *hub*, with a high "out-degree".

For directed $e = (u, v)$, u is the "hub" and v the "auhority".



This concept is used in the popular HITS algorithm.

Hubs&Authorities graphs

Define the *authority* proximity of nodes u and v with respect to node h as:

$$C_h(u, v) = \frac{w(h, u)w(h, v)}{d_+(h)}$$

from which we build the smoothness matrix:

$$S_A(u, v) = \sum_{h \in V} \frac{C_h(u, v)}{\sqrt{d_-(u)d_-(v)}}$$

We also define their *hub* proximity with respect to node a as:

$$C_a(u, v) = \frac{w(u, a)w(v, a)}{d_-(a)}$$

with:

$$S_H(u, v) = \sum_{a \in V} \frac{C_a(u, v)}{\sqrt{d_+(u)d_+(v)}}$$

Hubs&Authorities graphs

Let $\Omega_A(f) = \langle f^t, \Delta_A f \rangle$ where $\Delta_A = I - S_A$, we can show that:

$$\Omega_A(f) = \frac{1}{2} \sum_{u,v} \sum_h C_h(u,v) \left(\frac{f(u)}{\sqrt{d_-(u)}} - \frac{f(v)}{\sqrt{d_-(v)}} \right)^2$$

We get a similar expression for $\Omega_H(f)$.

Let:

$$\Delta_\gamma = \gamma \Delta_A + (1-\gamma)\Delta_H$$

We can solve the regularization problem as before:

$$f^* = \underset{f \in H(V)}{\operatorname{argmin}} \left(\Omega_\gamma(f) + \mu ||f - y||^2 \right)$$

For an undirected graph, $\Delta_H = \Delta_A = \Delta_\gamma$.

Hybrid graphs

We can generalize the smoothing functional to:

$$\Omega_{\beta,\gamma}(f) = \beta \cdot \Omega(f) + (1-\beta) \cdot \Omega_\gamma(f)$$

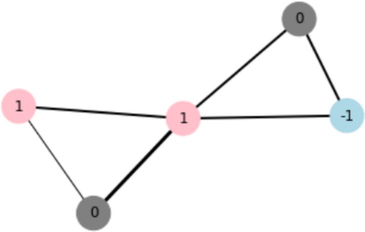
where $\Omega(f)$ is based on a random walk, and $\Omega_\gamma(f)$ is the "hubs&authority" smoothing.

This allows for 3 ways for vertices to be "close":

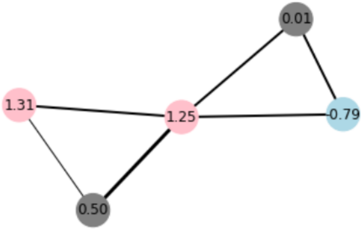
1. short path(s) between them,
2. point to several common vertices, and
3. are pointed to by several common vertices.

2 and 3 are the same for undirected graphs.

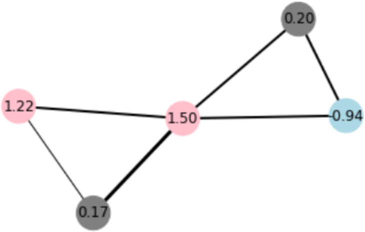
Toy example (undirected)



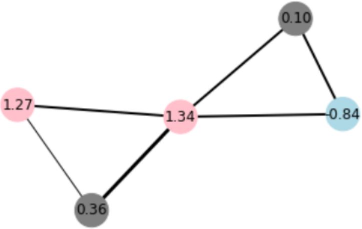
Base



Undirected



Hub/Authority



Hybrid

Hypergraphs

Ref: D. Zhou, J. Huang and B. Schölkopf, *Learning with Hypergraphs: Clustering, Classification and Embedding*, 2007.

For (undirected) hypergraphs, define:

E: set of subsets $e \subset V$
$w(e)$: hyperedge weight
$d(v) = \sum_{e; v \in e} w(e)$
$\delta(e) = |e| \geq 2$, the "hyperedge degree"
$H : |V| \times |E|$ s.t. $h(v, e) = 1$ iff $v \in e$
$W = diag(w(e))$,
$D_v = diag(d(v))$,
$D_e = diag(\delta(e))$.

Hypergraphs

The Ncut problem can be generalized to hypergraphs.

For a partition $V = S \cup S^c$, let:

$\partial S = \{e \in E; e \cap S \neq \emptyset,\ e \cap S^c \neq \emptyset\}$

$Vol S = \sum_{v \in S} d(v)$

$Vol \partial S = \sum_{e \in \partial S} w(e) \frac{|e \cap S| \cdot |e \cap S^c|}{|e|}$

For the last expression, if e is mapped to its 2-section, the numerator is the number of 'edges' that would be cut.

Hypergraphs

The Ncut problem can again be illustrated via a random walk with:

$$p(u, v) = \sum_{e \in E} \frac{w(e)h(u, e)}{d(u)} \frac{h(v, e)}{|e|}$$

with stationary distribution $\pi(v) = \frac{d(v)}{VolV}$.

We get the following results:

$\frac{VolS}{VolV} = \sum_{v \in S} \pi(v)$

$\frac{Vol\partial S}{VolV} = \sum_{u \in S} \sum_{v \in S^c} \pi(u) p(u, v)$.

Hypergraphs

Solving the relaxed problem yields the same form as with graph, but with:

$$\Delta = I - D_v^{-1/2} H^T W D_e^{-1} H D_v^{-1/2}$$

When all $|e| = 2$, we get:

$$\Delta = \frac{1}{2}(I - D_v^{-1/2} W D_v^{-1/2})$$

which is half the graph Lapacian, so Δ can be defined as the *Hypergraph Laplacian*.

Hypergraphs

We define the same semi-supervised problem as with graphs:

$$f^* = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}}(\Omega(f) + \mu||f - y||^2)$$

where:

$$\Omega(f) = \langle f, \triangle f \rangle = \frac{1}{2} \sum_{e \in E} \frac{1}{\delta(e)} \sum_{(u,v) \in e} w(e) \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2$$

The solution to the above problem is given by

$$f^* = (1 - \alpha)(I - \alpha S)^{-1} y, \ \alpha = (1 + \mu)^{-1}, \ S = I - \Delta.$$

Hypergraphs

The random walk described previously amounts to:

- from vertex u, pick an hyperedge e at random for which $u \in e$
- pick a vertex $v \in e$ at random and jump to v.

We can view the above as a graph with a weighted adjacency matrix $\tilde{A} = (a_{ij})$ where:

$$a_{ij} = \sum_{e;(v_i,v_j)\in e} \frac{w(e)}{|e|}, \ a_{ii} = \sum_{e;v_i\in e} \frac{w(e)}{|e|}$$

with row sum

$$a_{i.} = \sum_{e;v_i\in e} w(e) = \sum_{e\in E} w(e)h(e,v_i) = d(v_i).$$

Hypergraphs

If all $|e| = 2$, we get $a_{ii} = \sum_{e; v_i \in e} w(e)/2 = d_i/2$ and for $e = (v_i, v_j)$ we get $a_{ij} = w(e)/2$, therefore

$$\tilde{A} = \frac{1}{2}(D_v + A)$$

where A is the (weighted) adjacency matrix of the graph representation of this hypergraph.

Therefore, the solution to the transductive learning problem will differ if G is seen as a graph or an hypergraph.

Hypergraphs

We define a new random walk as follows:

- from vertex u, pick an hyperedge e at random for which $u \in e$
- pick a vertex $v \in e,\ v \neq u$ at random and jump to v.

We can view the above as a graph with a weighted adjacency matrix $\tilde{A} = (a_{ij})$ where:

$a_{ij} = \sum_{e;(v_i,v_j)\in e} \frac{w(e)}{|e|-1},\ a_{ii} = 0$

with row sum

$a_{i.} = \sum_{e;v_i \in e} w(e) = d(v_i)$.

Hypergraphs

In matrix form: $\tilde{A} = H^T W \tilde{D}_e^{-1} H - D_v$

with $\tilde{D}_e$ the diagonal matrix with entries $\frac{1}{|e|-1}$.

In this case, the *adjusted hypergraph Laplacian* takes the following form:
$\Delta = I - S$ with $S = D_v^{-1/2} \tilde{A} D_v^{-1/2} - I$

If all $|e| = 2$ we get $\tilde{A} = A$ where A is the (weighted) adjacency matrix of the graph representation of this hypergraph.

Hypergraphs

We can generalize to **directed hypergraphs** where:

$$e = e_t \cup e_h \; \forall e \in E, \; |e_t| > 0, |e_h| > 0$$

the tail and head of each hyperedge.

Sending an email to multiple recipients is an example of a directed hyperedge.

Categorical data

Hypergraphs can be used to model *categorical* data.

Example: the “mushroom dataset” (UCI ML repository):

- 22 categorical attributes, 8124 observations from 23 species
- 2 classes: edible or likely not edible
- 1 hyperedge per categorical attribute
  - mushrooms with “cap shape = bell”
  - mushrooms with “cap shape = conical”
  - etc...

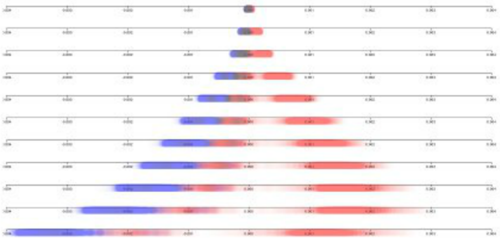
Categorical data

Undergraduate student summer work term:

- code up hypergraph transductive learning in Python
- validate published results over categorical data
  - compare with graph model
- study the impact of tradeoff parameter α
- propose and explore a vertex embedding framework

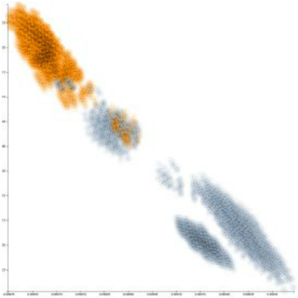
Ranking results

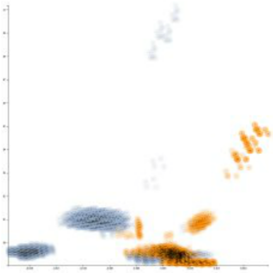
- color w.r.t. class of mushrooms
- parameter α has high impact on the magnitude of the resulting values
- rankings however were mostly the same



Embedding?

- vertex embedding is a hot topic
- use several runs of TL starting from different values
- generate multi-dimensional vertex representation
- similar to random walks





Other References

J. R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, edition 1.25, 1994.

D. Spielman and S. Teng, *Solving sparse, symmetric, diagonally-dominant linear systems in time $o(m^{1.31})$.*, FOCS, pp. 416-427, Cambridge, USA, 2003.

J. Dean and S. Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI 2004.

D. Zhou, B. Schölkopf and T. Hofmann, *Semi-supervised Learning on Directed Graphs*, NIPS 2005.

O. Chapelle, B. Schölkopf and A. Zien eds, *Semi-Supervised Learning*, MIT Press, 2006.

Semi-supervised Learning

Notebook #8