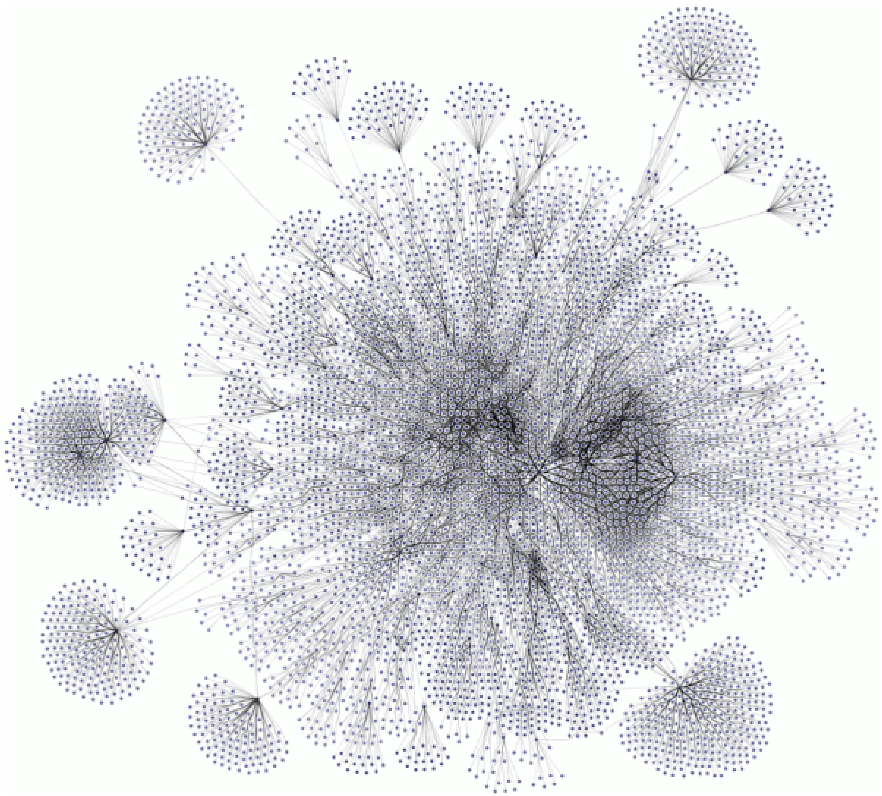
Semi-supervised Classification from Discriminative Random Walks

Jérôme Callut – UCL
Kevin Françoisse – UCL
Marco Saerens – UCL
Pierre Dupont – UCL

Université catholique de Louvain
Machine learning group

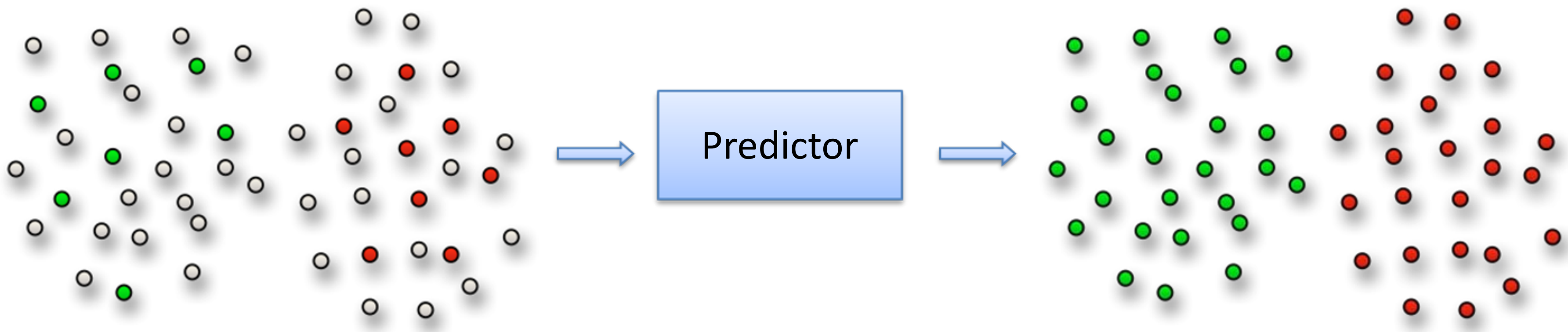
Overview

- Semi-supervised classification (in graphs)
- *D*-Walk approach presentation
- Experimental results
- Conclusion
- Extensions



Semi-Supervised learning

- Typical case : Having a labeled set of data (xi, yi) and a set of unlabeled data (xi*), predict the label of unlabeled data.



Semi-Supervised learning

- Why semi-supervised learning ?

  Supervised data is expensive : manual text categorization, finding 3D protein structure, recorded behavior of individuals in social network, …

  Unlabelled data is easier to obtain : text, primary structure protein, individual link, …

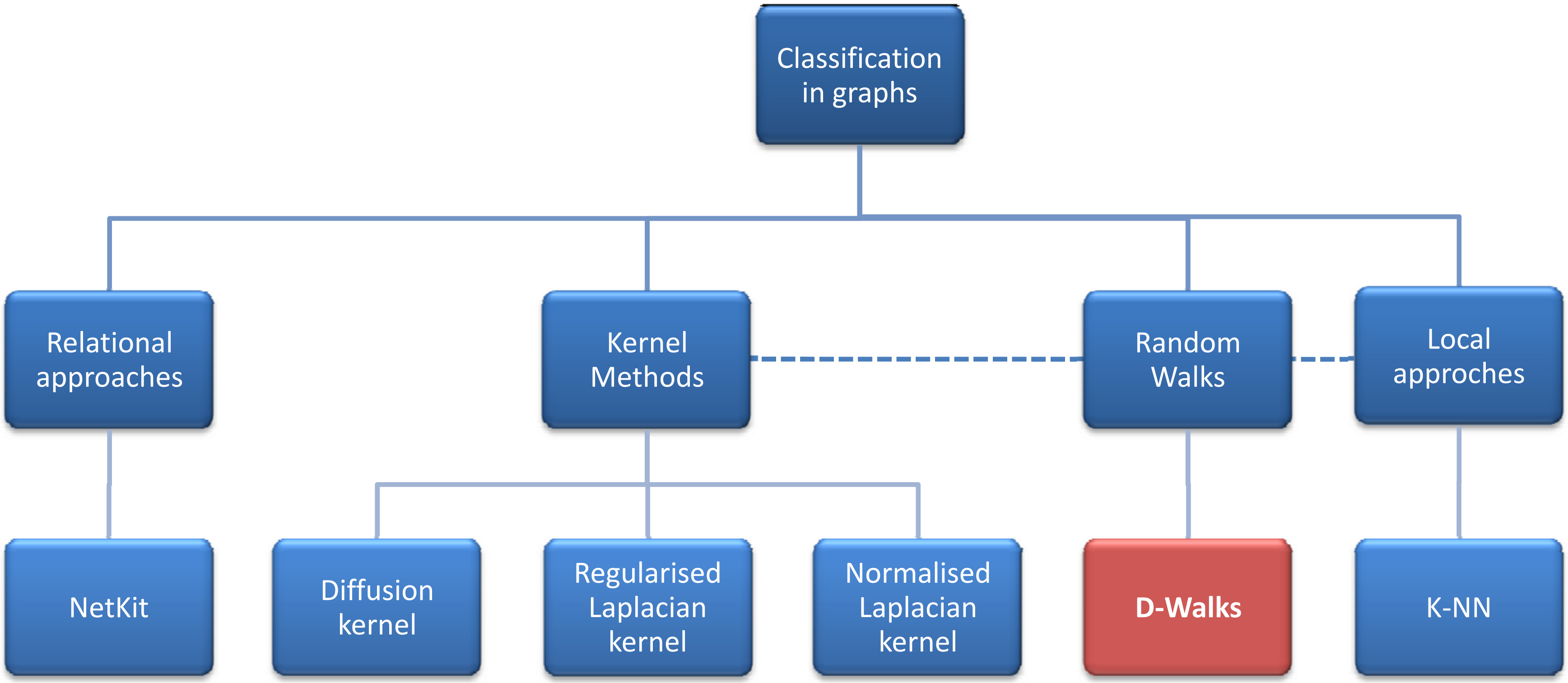
- How does unlabelled data help ?

  They give link and feature information to the labeled data

Classification in graphs

- Objective: Given a graph with some nodes being labeled, we want to predict/classify the missing node labels
- *Real-world applications :*
  - Linked document categorization
  - Classification of individuals and social behavior in social network
  - Protein function prediction
  - Semi-supervised classification from a neighboring graph in a feature space
- *The method should be able to:*
  - handle very large graphs
  - handle a wide variety of graphs
    - *directed or not*
    - *connected or not*
    - *with positive edge weight*
  - give good predictive results
  - be very fast

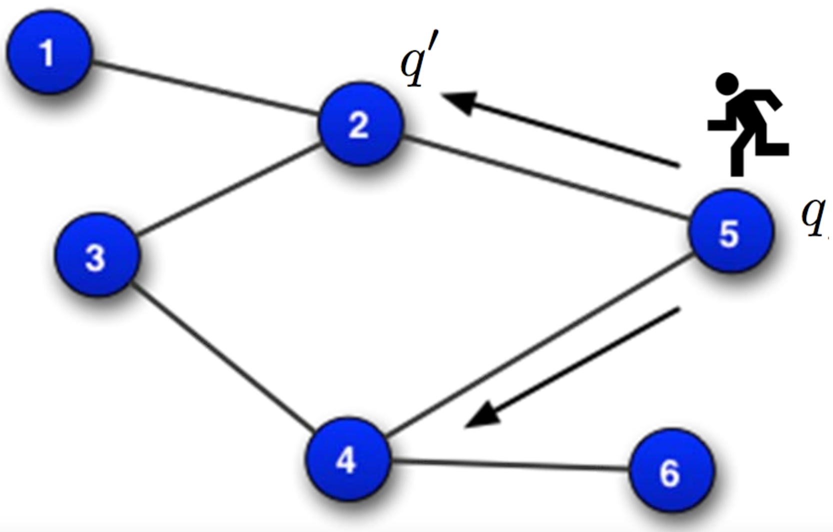
Some related approaches



Random walk preliminaries

- We define a random walk model on the graph which can be modeled by a discrete-time Markov chain :
  - Each node is associated to a state of the Markov chain
  - The random variable X_t represents the state of the Markov model at time step t
  - The random walk is defined by the transition probability matrix

$$P[X_{t+1} = q' \mid X_t = q] = p_{qq'} \triangleq \frac{a_{qq'}}{\sum_{q' \in \mathcal{N}} a_{qq'}}$$

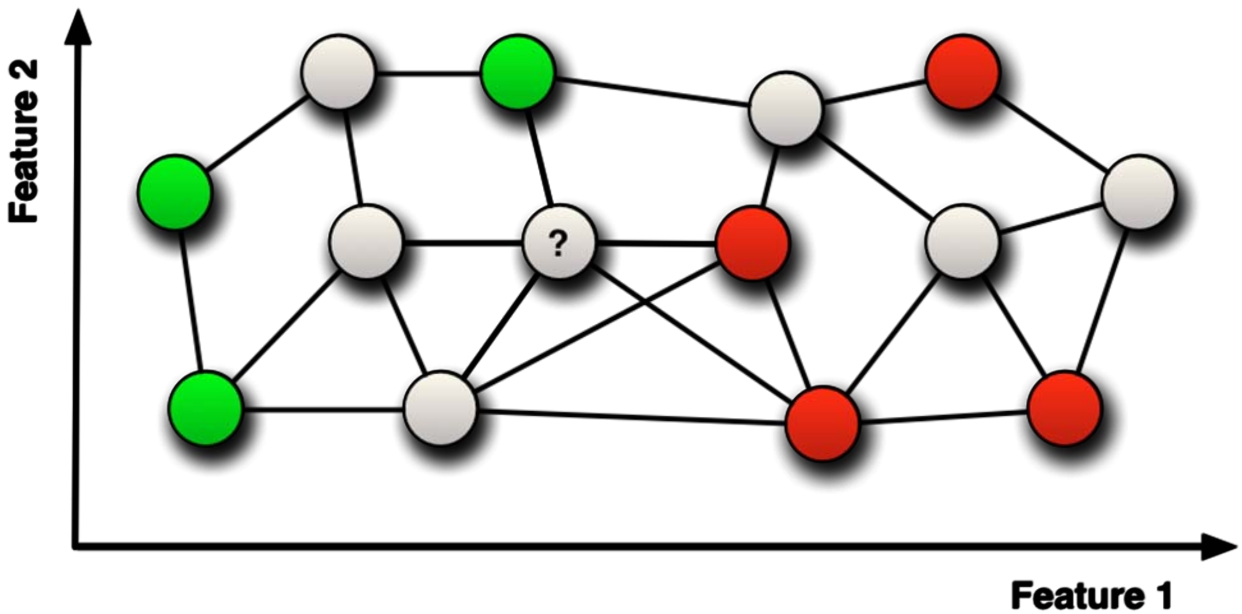


where A is the adjacency matrix possibly weighted

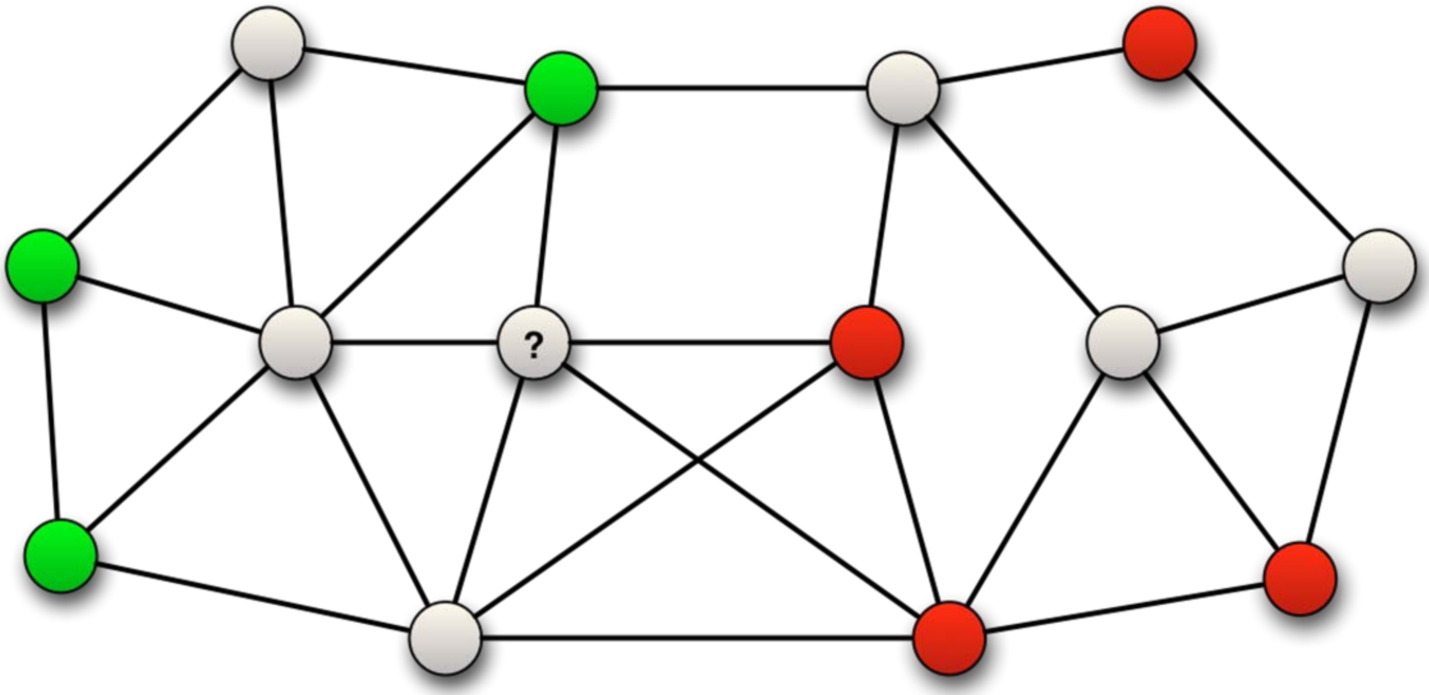
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Classic data to graph

How can a vectorial dataset be seen as a graph ?

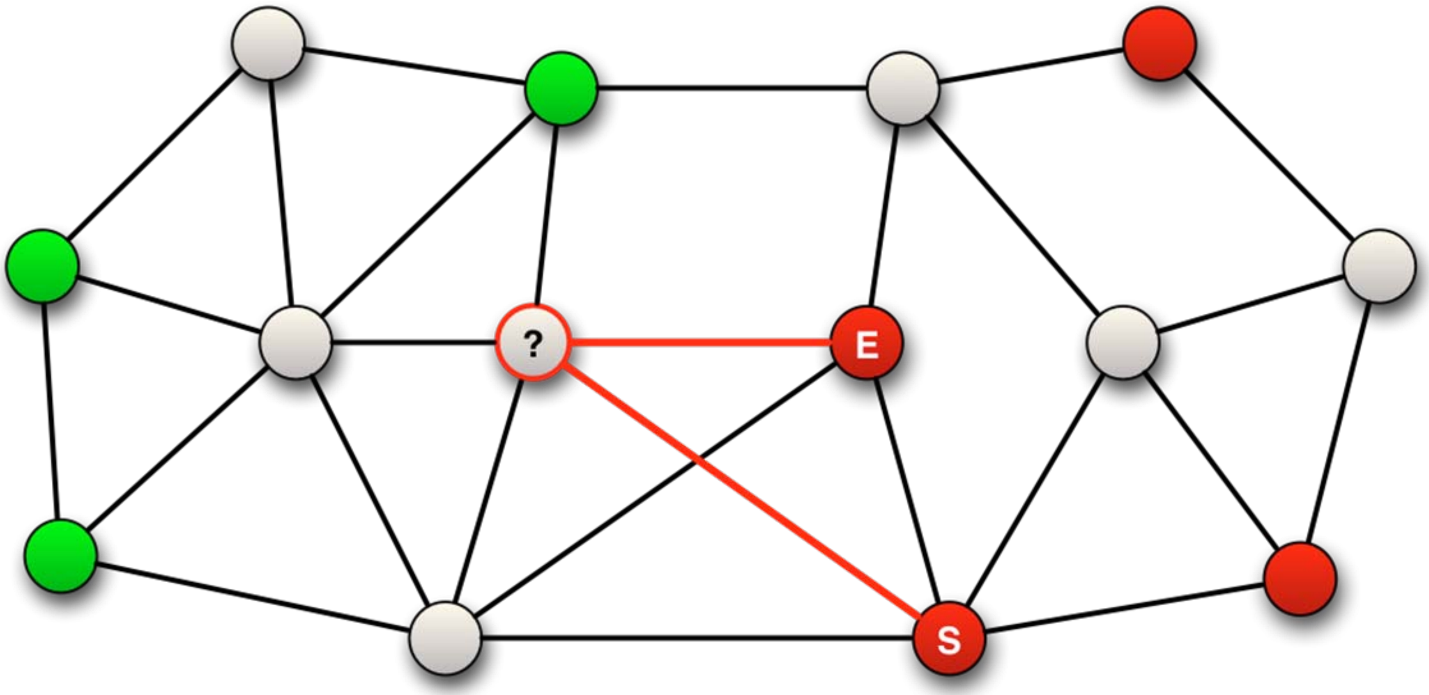


General Idea



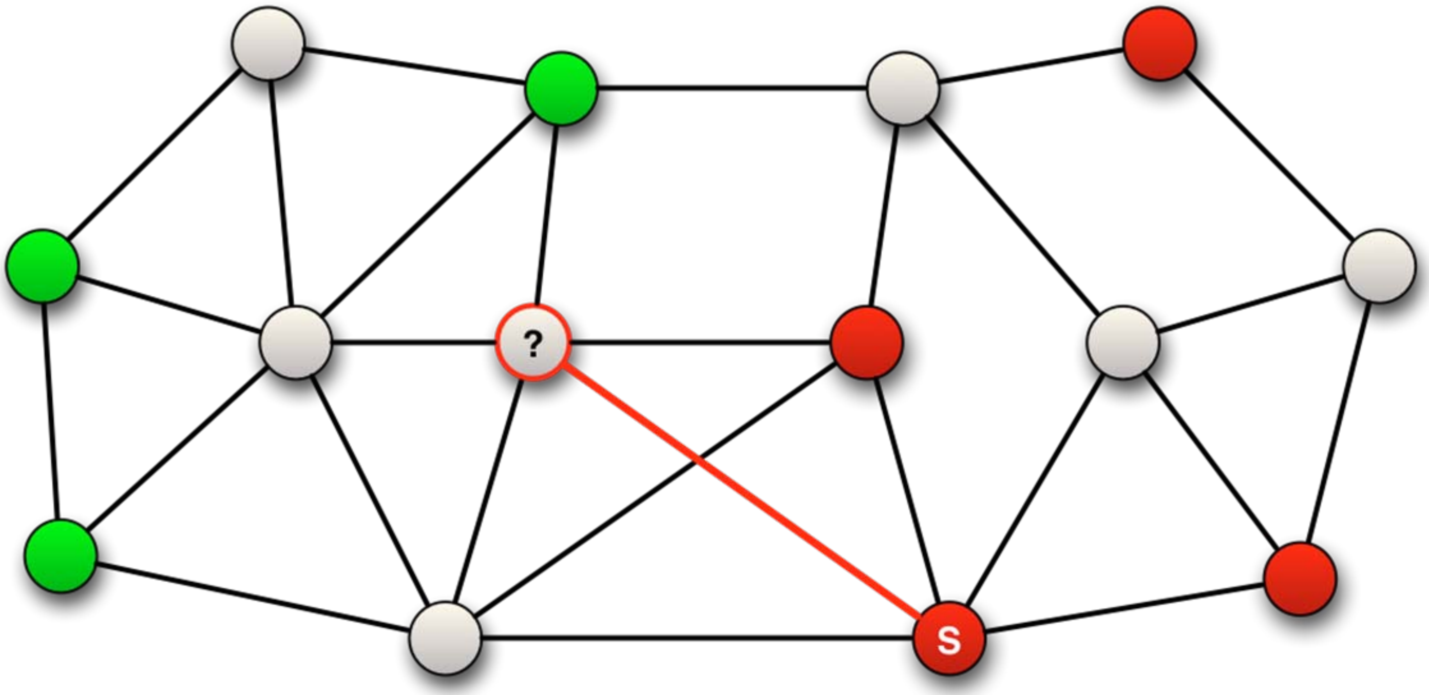
- Consider the classification of the node marked with « ? »

General Idea



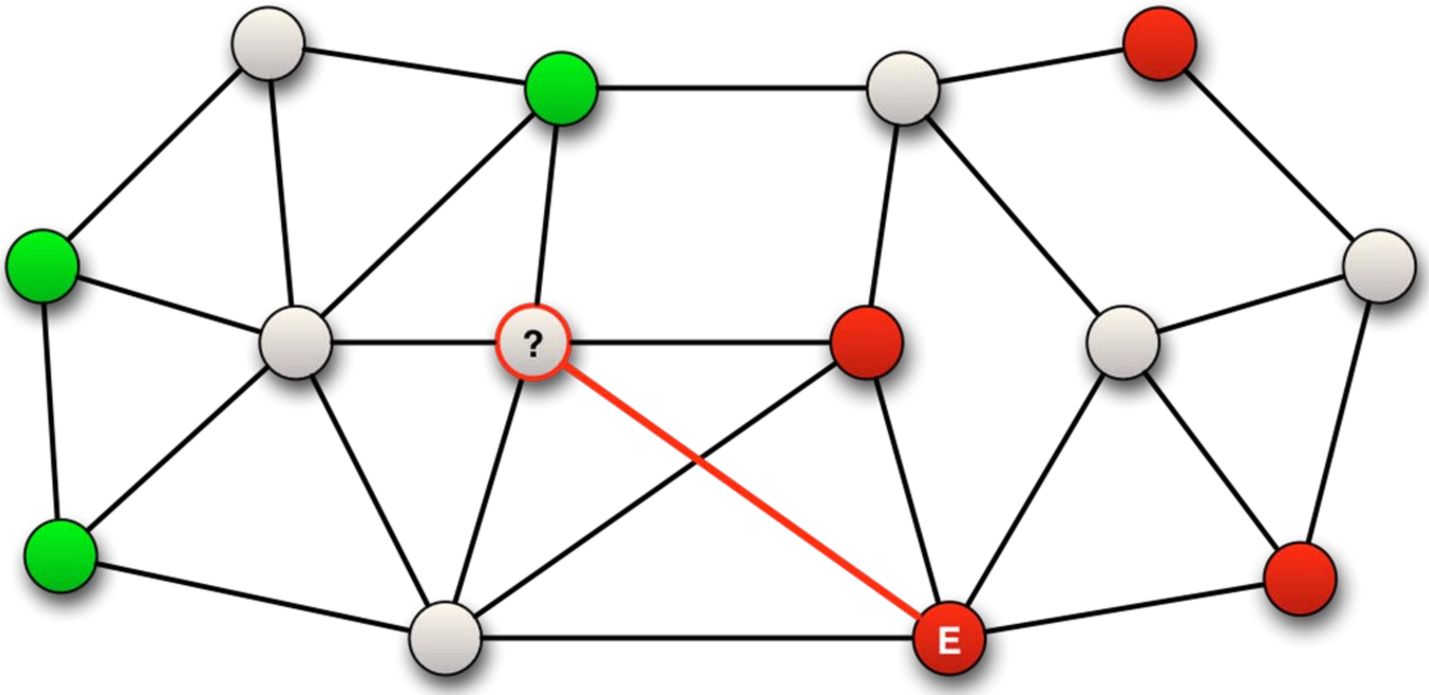
- A *D-walk* always starts and ends in nodes belonging to the same class

General Idea



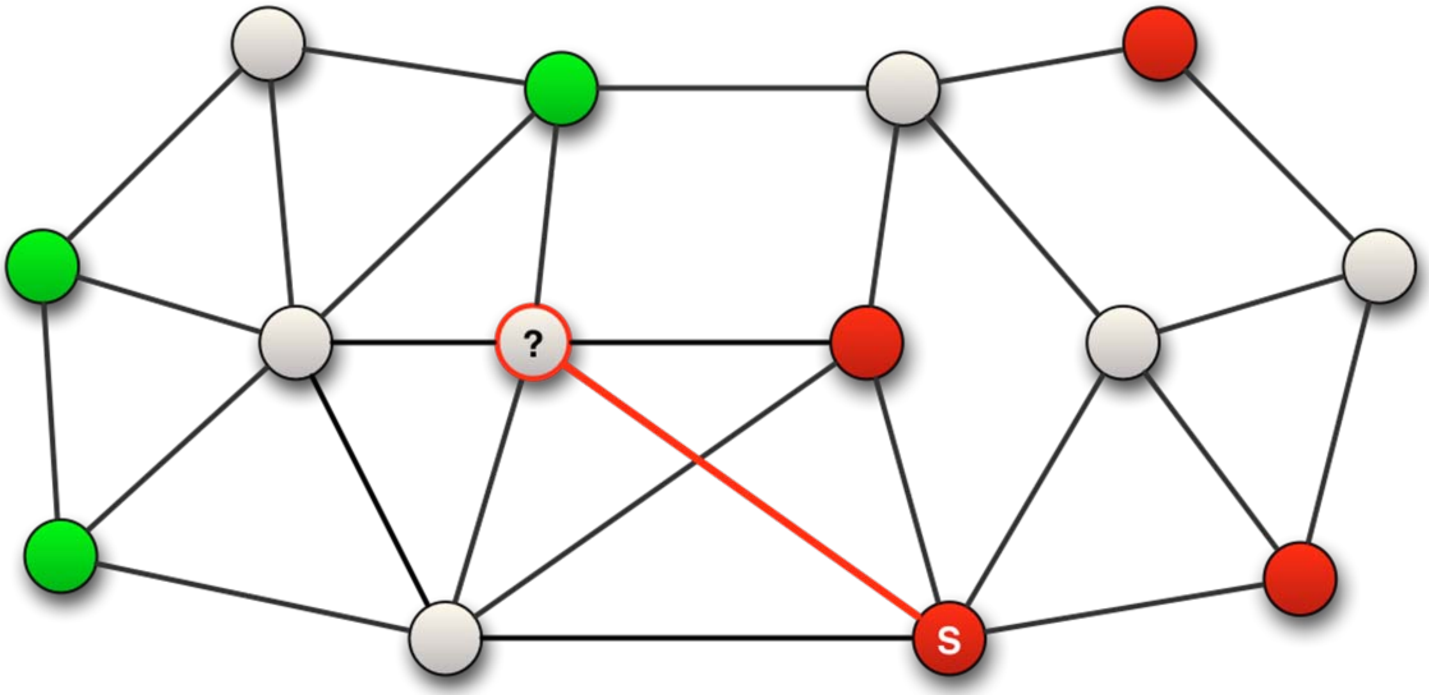
- A D-walks can start from a node an come back to the same node

General Idea



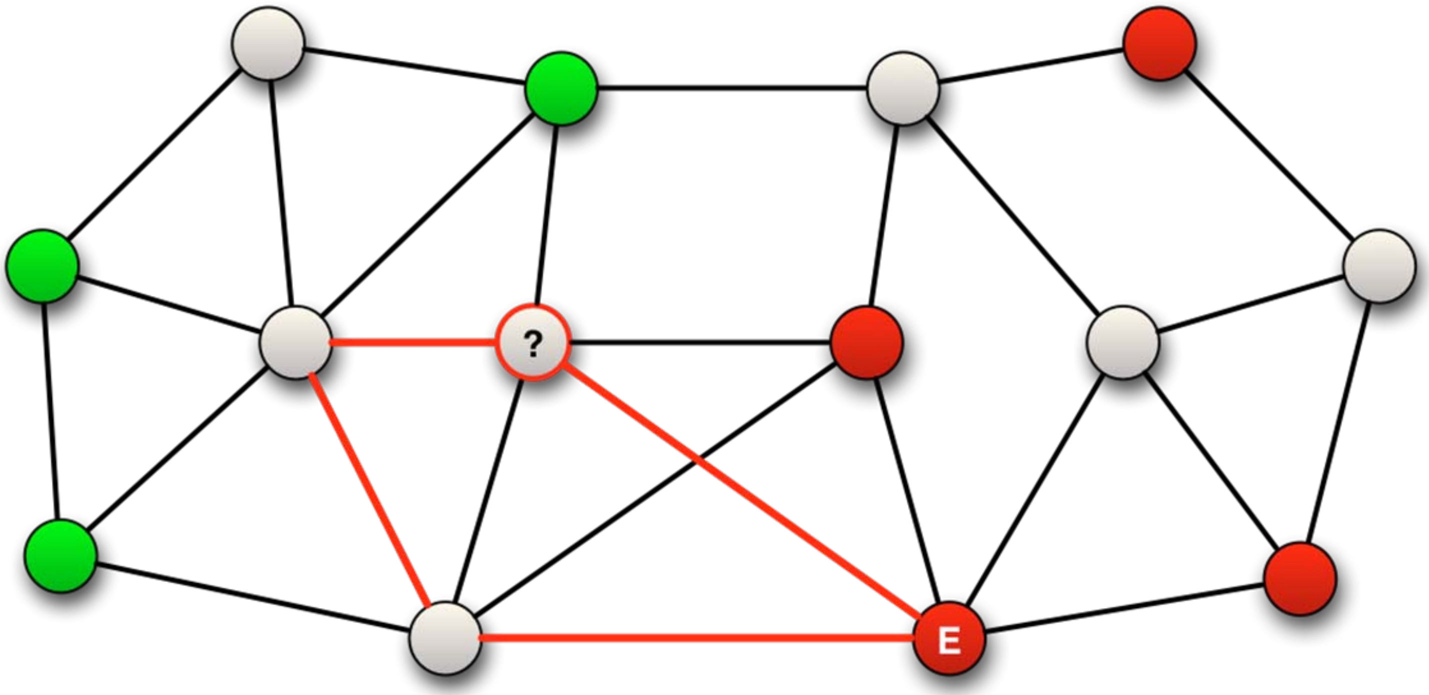
- A D-walks can start from a node an come back to the same node

General Idea



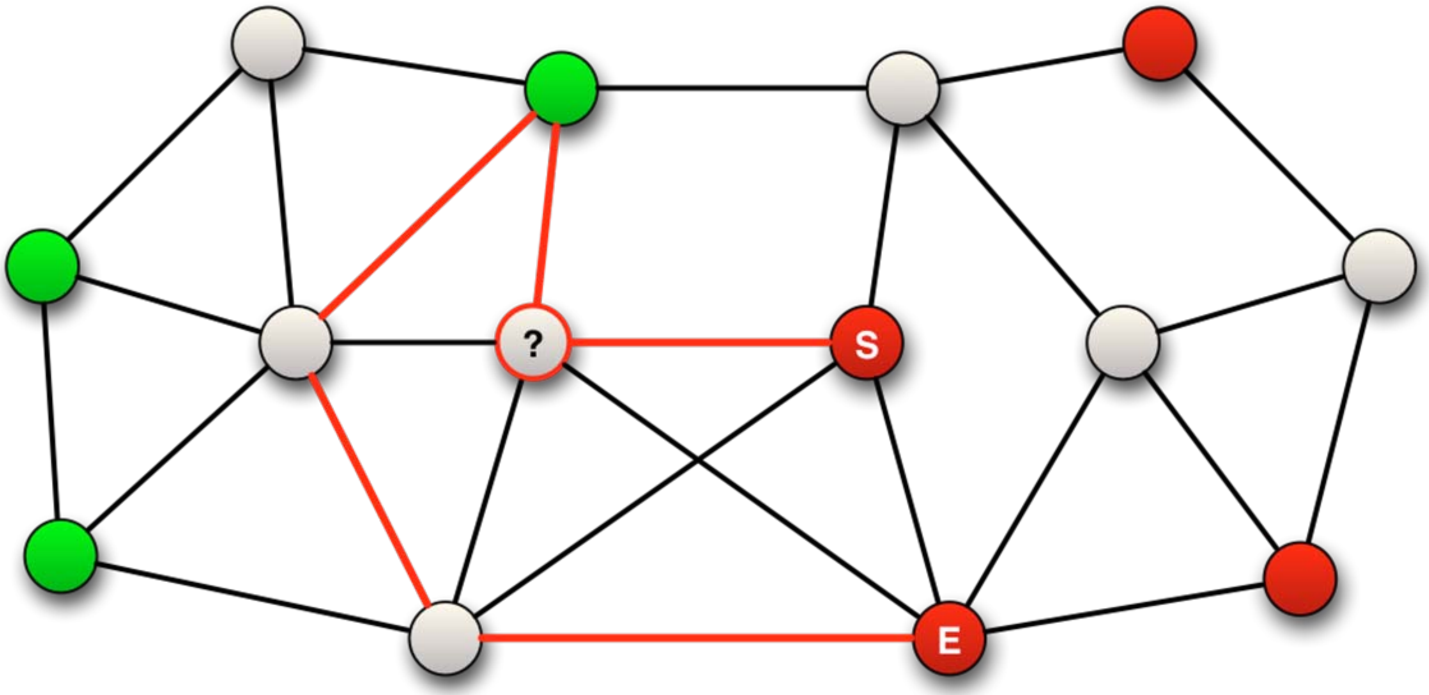
- A D-walks can cross unlabeled nodes

General Idea



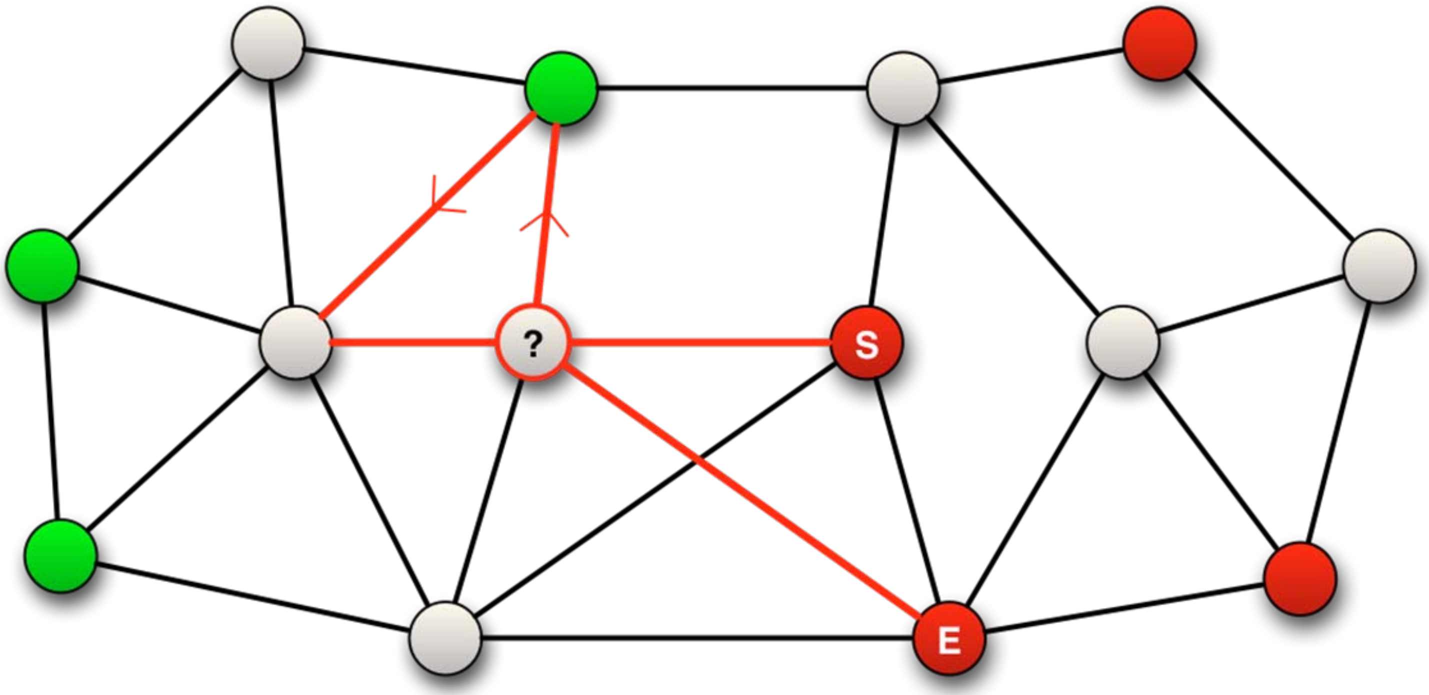
- A D-walk can cross unlabeled nodes

General Idea



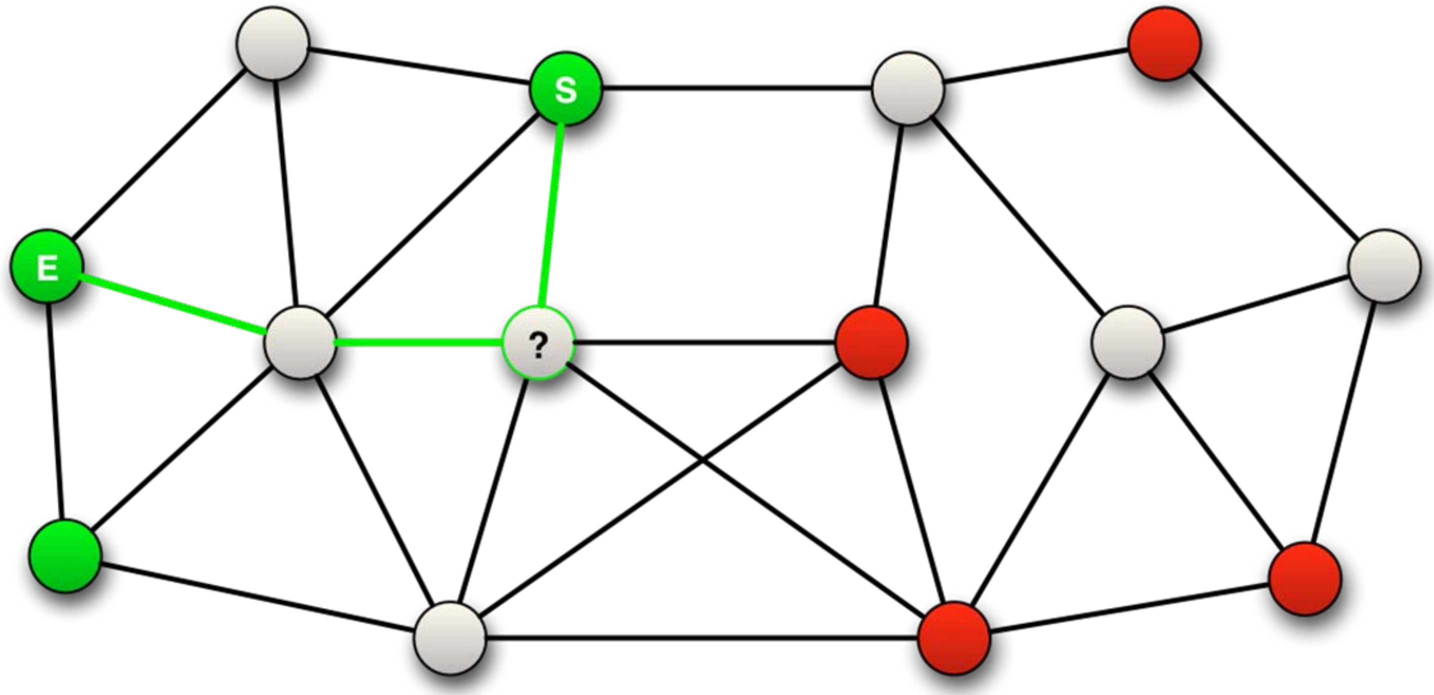
- A D-walks can cross nodes from other classes

General Idea



- A D-walks can pass several time on the same node

General Idea



- Example of a GREEN D-walks

D-walks definitions

- Based on the defined Markov chain model, we defined :

A D-walk $\mathcal{D}^y$ is a random walk starting in a labeled node and ending when any node having the same label is reached for the first time

A D-walk betweenness $B(q, y)$ is the expected passage time on an unlabeled node for each class

Computing the D-Walks betweenness

- D-Walk betweenness :

$$B(q,y) \triangleq \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}^y\right]$$

where pt the number of *passage* in node q

- Passage times on a node :

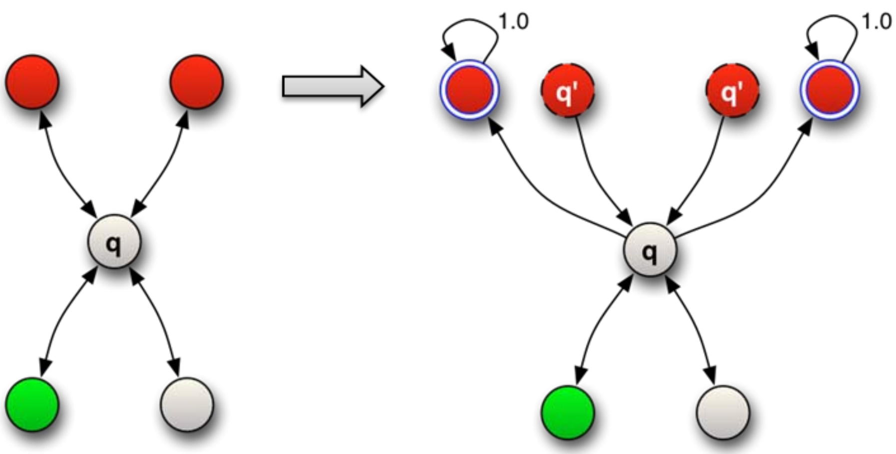
$$\mathrm{pt}(q) = \sum_{t=1}^{\infty} \mathbb{I}\{X_t = q\}$$

Practical computation relies on absorbing Markov chain techniques

Computing the D-Walks betweenness

- Passage times on a node :

$$\text{pt}(q) = \sum_{t=1}^{\infty} \mathbb{I}\{X_t = q\}$$



Suppose P is the transition probability matrix

Interest class nodes are replicated as starting nodes and absorbing nodes

$$^{y}P = \left(\begin{array}{c|c} ^{y}P_T & R \\ \hline 0 & I \end{array} \right)$$

$$\mathbb{E}\left[\text{pt}(q) \mid y\right] = \frac{1}{n_y} \sum_{q' \in \mathcal{L}_y} \left[I + \underbrace{^{y}P_T}_{\text{1st step}} + \underbrace{^{y}P_T^2}_{\text{2d step}} + \underbrace{^{y}P_T^3}_{\text{3th step}} + \ldots\right]_{q'q}$$

$$= \frac{1}{n_y} \sum_{q' \in \mathcal{L}_y} \left(I - {}^{y}P_T\right)^{-1}_{q'q}$$

Because of the matrix inversion, the complexity is $\mathcal{O}(n^3)$

Computing the D-Walks betweenness

- The bounded D-walks approach constrains to perform walks up to a prescribed length

$$B_L(q, y) \triangleq \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}^y_{\leq L}\right]$$

where $\mathcal{D}^y_{\leq L}$ refers to all bounded D-walks up to a given length L

It has three major benefits:

1. Better classification results
2. Betweenness measure can be computed very efficiently
3. Unbounded-betweenness can be approximated by considering large but finite L

⇒ Efficient betweenness computation can be achieved using forward and backward variables (similar to those used in the Baum-Welch algorithm for HMM)

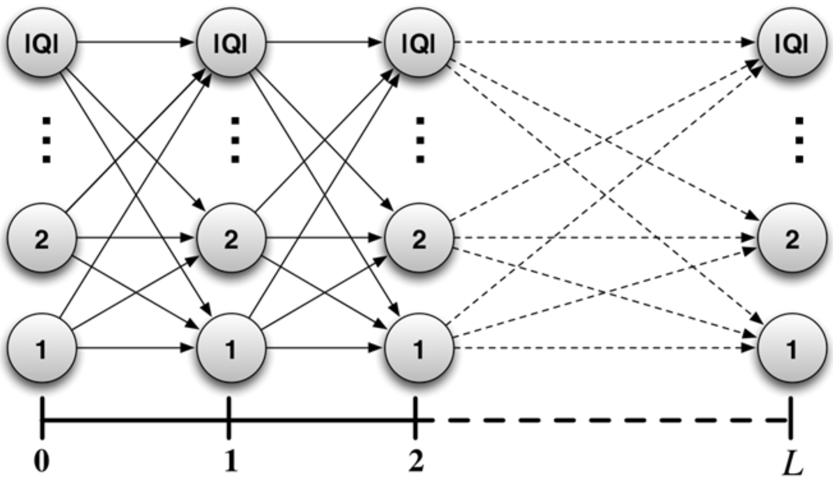
Computing the D-Walks betweenness

Forward

(case $t = 1$) $\alpha^y(q, 1) = \sum_{q' \in \mathcal{L}_y} \frac{1}{n_y} p_{q'q}$

(case $t \geq 2$) $\alpha^y(q, t) = \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \alpha^y(q', t-1)\, p_{q'q}$

Compute the probability to reach state q after t steps without passing through node in class y



and **Backward** recurrence !

(case $t = 1$) $\beta^y(q, 1) = \sum_{q' \in \mathcal{L}_y} p_{qq'}$

(case $t \geq 2$) $\beta^y(q, t) = \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \beta^y(q', t-1)\, p_{qq'}$

Compute the probability that state q is reached by the process t steps before reaching any node labeled y for the first time

Computing the D-Walks betweenness

To compute $B_L(q, y)$

1. Compute the mean passage time in a node q during $\mathcal{D}_l^y$ walks

$$\mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_l^y\right] = \sum_{t=1}^{l-1} P[X_t = q \mid \mathcal{D}_l^y] = \sum_{t=1}^{l-1} \frac{P[X_t = q \wedge \mathcal{D}_l^y]}{P[\mathcal{D}_l^y]}$$

$$= \frac{\sum_{t=1}^{l-1} \alpha^y(q,t)\beta^y(q,l-t)}{\sum_{q' \in \mathcal{L}_y} \alpha^y(q',l)}$$

Recurrences help to compute the mean passage time in a node q for a certain D-walk

Probability to start in any node of class y, to reach node q at time t and to complete the walk l-t steps later

Probability to perform a D-walk

Computing the D-Walks betweenness

2. Finally, the **betweenness measure based on walk up to length *L*** is obtained as an expectation of the betweenness for all length $1 \leq l \leq L$

$$B_L(q, y) = \sum_{l=1}^{L} \frac{P[\mathcal{D}_l^y]}{Z} \, \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_l^y\right]$$

where Z is normalization constant

Computing the D-Walks betweenness

- Node are classifiedusing a maximum a posteriori decision rule from the betweenness of each class :

$$P[q \mid y] \triangleq \frac{B_L(q,y)}{\sum_{y' \in \mathcal{Y}} B_L(q,y')}$$

$$\Downarrow$$

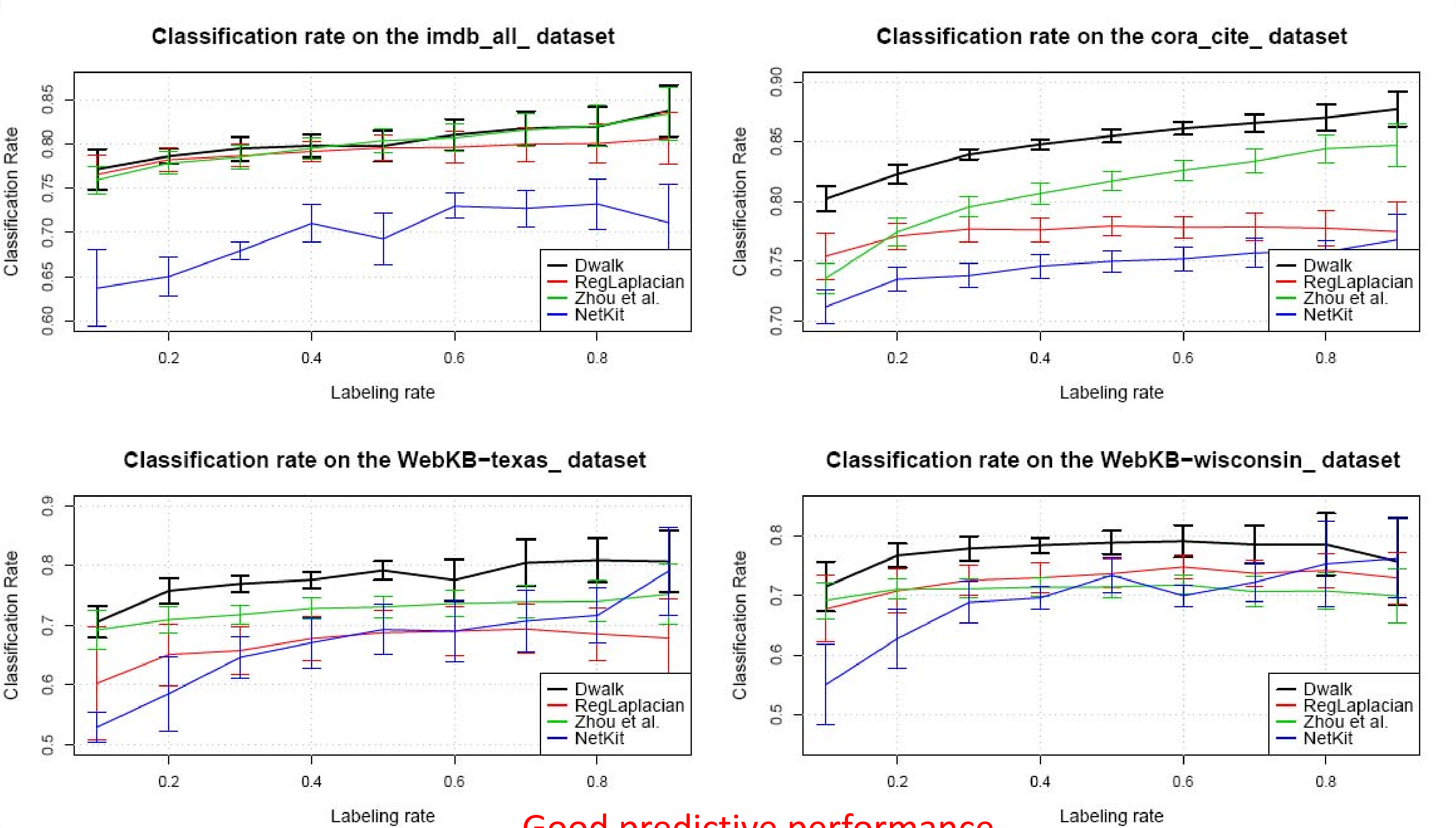
$$\hat{y}_q = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, P[q \mid y]\; P[y]$$

Experiments

- Datasets characteristics:

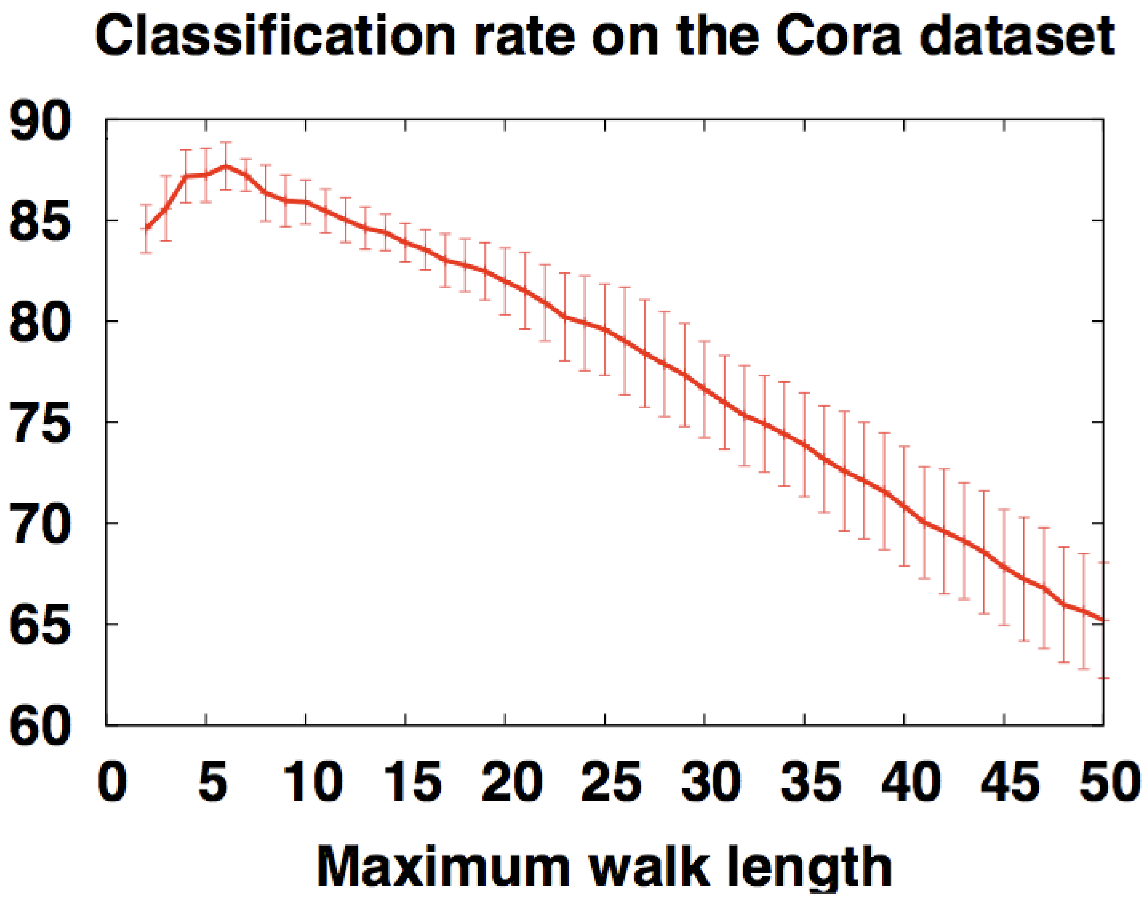
	IMDB	Cora	WebKB-Texas	WebKB-Wisconsin
Nb Classes	2	7	6	6
Nb Nodes	1169	3583	334	348
Majority class acuracy	51.07%	29.7%	48.8%	44.5%
Nb Edges	40564	22516	32988	33250
Mean degree	36.02	6.28	98.77	95.55
Max degree	181	311	215	229
Min degree	1	1	1	1

Experiments



Good predictive performance

Experiments



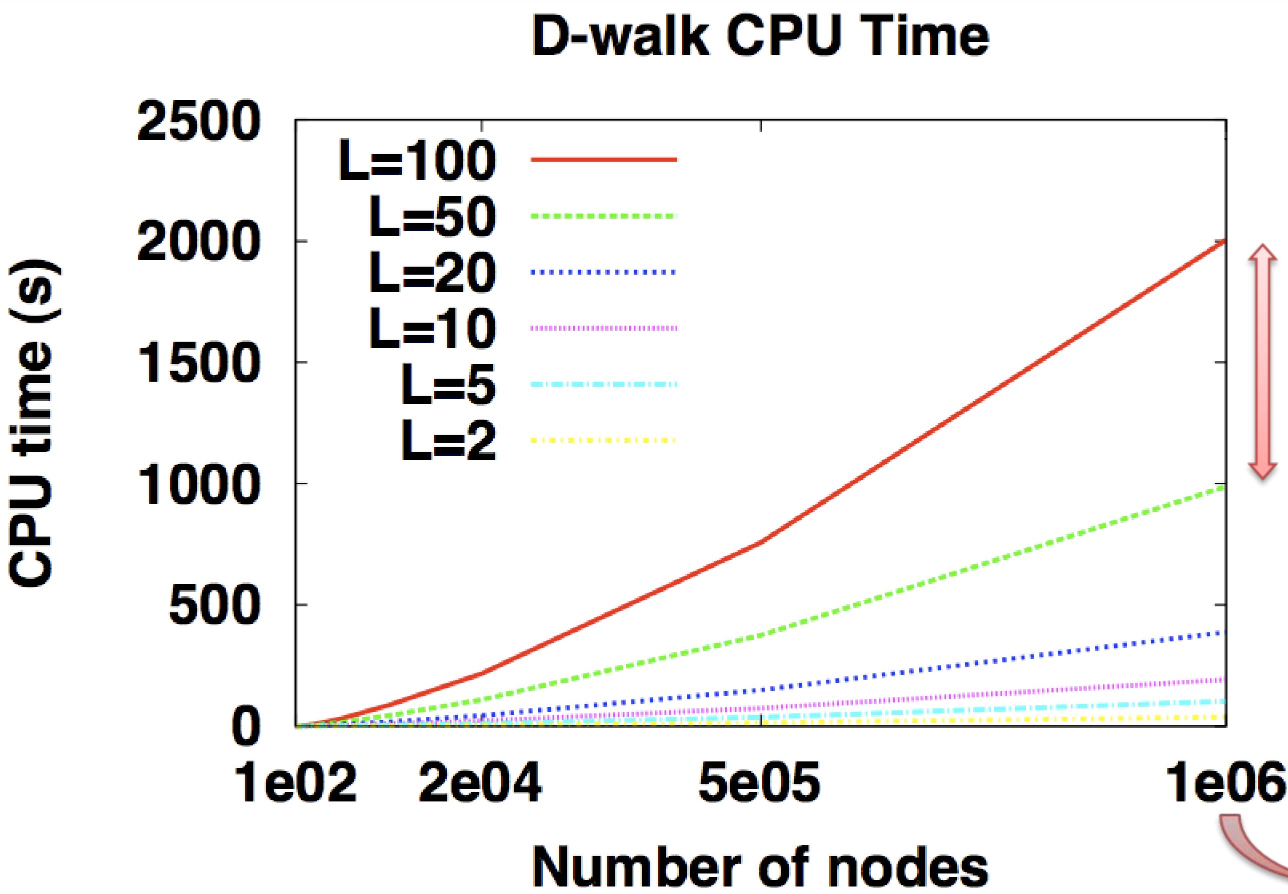
L is tuned by cross-validation

Best walk length for the CORA dataset is **6** !

Higher than 6 the results are worse

Low optimal walk length

Experiments



Linear time computation

$$\Theta(mL)$$

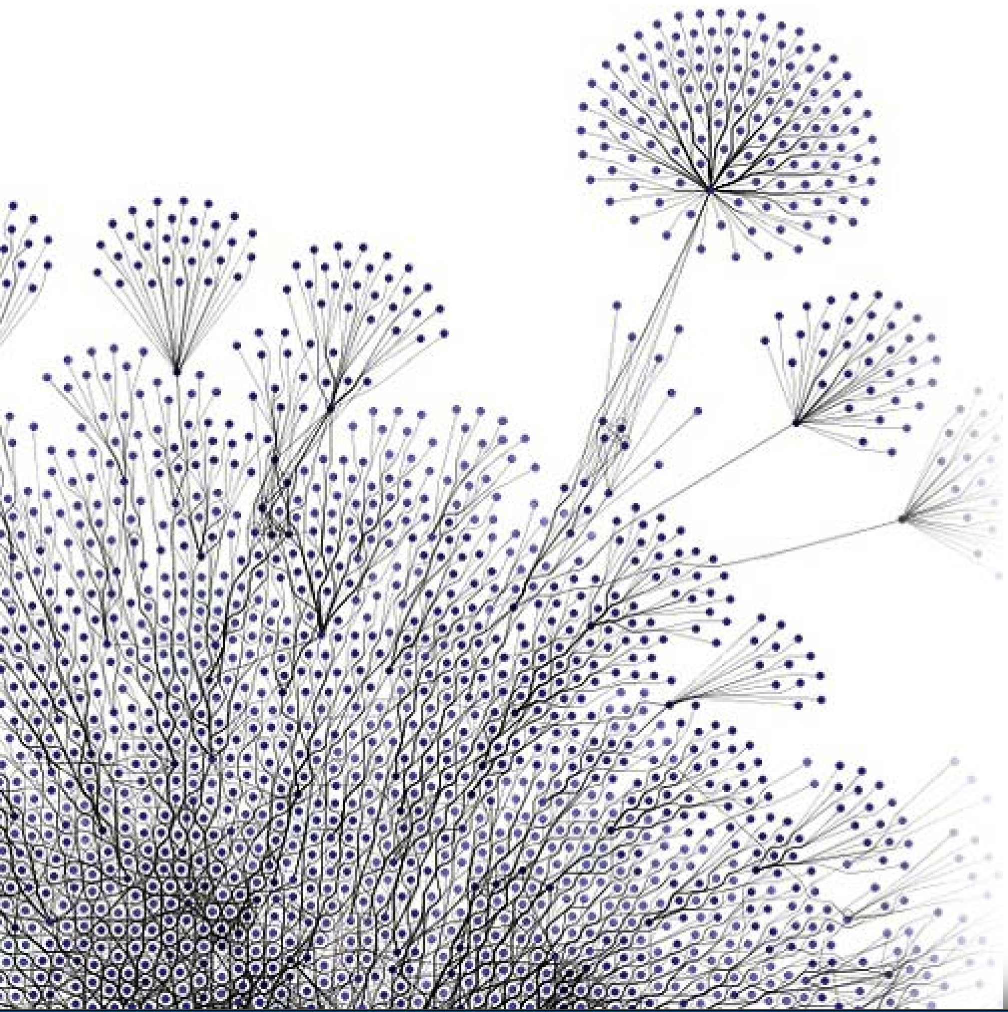
where m is the number of edges and L the maximum walk length considered

Large scale

Conclusion

- D-walks define a node betweenness measure for each class
- Unlabeled nodes are classified under the class for which the betweenness is the biggest.
- Bounding the walk length
  - Provides better classification results, outperforming kernel based approaches
  - Allows to algorithm to be very fast (linear time computation)
- Possibility to deal with very large graphs

Possible extensions



- Node features incorporation like text and numerical attributes to improve classification
- Derive a kernel from D-Walks to be used in kernel methods like SVM
- Define a collaborative recommendation system based on bounded random walks