WEISFEILER-LEMAN MODELS CONTRA LINK PREDICTION SCHEMES

Algebraic Graph Theory Conference

Fifty years since Weisfeiler-Leman

Plzeň, Česko

Katie Brodhead

Florida A&M University

July 3, 2018

Motivation

In the data mining & machine learning communities:

Effective Link Prediction

1) Has diverse real-world applications
   - Friend recommendation in social networks
   - Product recommendation in e-commerce
   - Knowledge graph completion
   - Finding interactions between proteins
   - Recovering missing reactions in metabolic networks
2) Has been a central challenge for researchers
   - Which method is best for a particular situation?
   - At issue: each scheme is grounded in a particular heuristic.

Brief History

Historically link prediction models have been feature-based, of two types.

I. Topological models. (e.g. Common Network, Katz, Adamic-Adar, PageRank)

- Leverage node similarities, locally or globally.
- Do not perform well when similarity scores do not capture the network formation mechanisms

II. Latent Models. (e.g. Matrix Factorization, Ranking, Stochastic Block Methods)

- Assume that latent groups exist for nodes and that links are determined by group memberships.
- Extract group memberships via the low-rank decomposition of a network adjacency matrix, or via training which fits a probabilistic models.
- Central weakness: understanding how networks are formed, due to models' focus on individual nodes

Weisfeiler-Leman Prediction Methods

Rather than focusing on nodes, WL-Methods focus on links.

More specifically, fix k > 2. Suppose a graph G = (V, E) exists, but only $V = \{v_1, v_2, .., v_n\}$ is known, and only certain elements of E.

Our goal is to accurately predict whether each $e_{ij} = (v_i, v_j)$ lies in E, if unknown, using the information that we have.

To do this, for every e_{ij}, build an e_{ij}-containing graph G_{ij} of size k, outwards from e_{ij}, with actual links (edges) that are known.

The idea now is to "use WL" on each G_{ij} to encode graph-theoretic information.

Weisfeiler-Leman Prediction Methods

Using machine learning, we then train on the graphs.

Based on the graph theoretic-information surrounding each e_{ij} (which is "learned"), edge e_{ij} is predicted to be in or out of graph G.

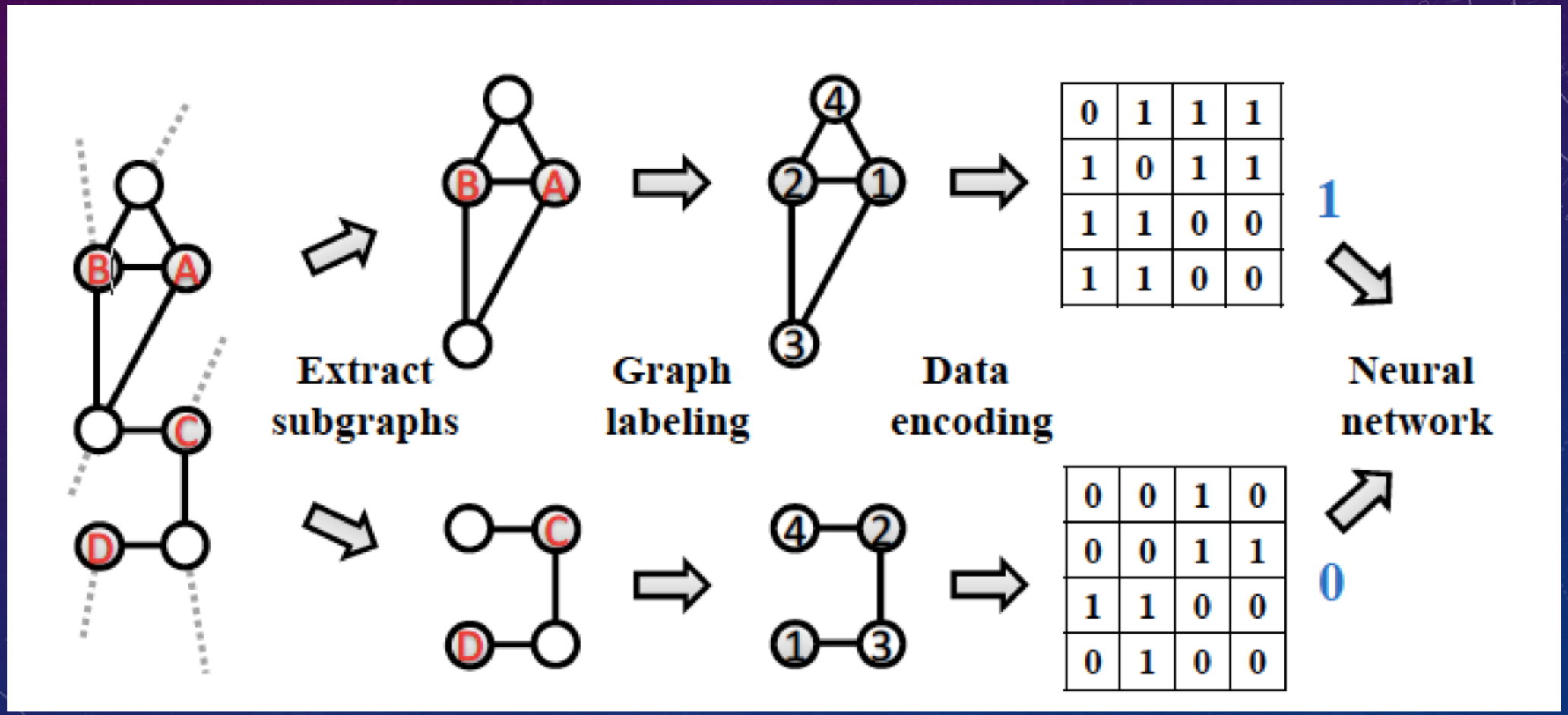
Thus, we can recommend whether two nodes should be "friends" (Facebook)

PROBLEM: After WL-stabilization, the information encoding which pair of vertices is "e_{ij}", may be lost. This makes training useless!!!

To ensure that the e_{ij} remains distinguished after color stabilization, a modified WL-Algorithm needs to be used. It is also much faster than WL.

Weisfeiler-Leman Prediction Methods

Overview of the Learning Approach

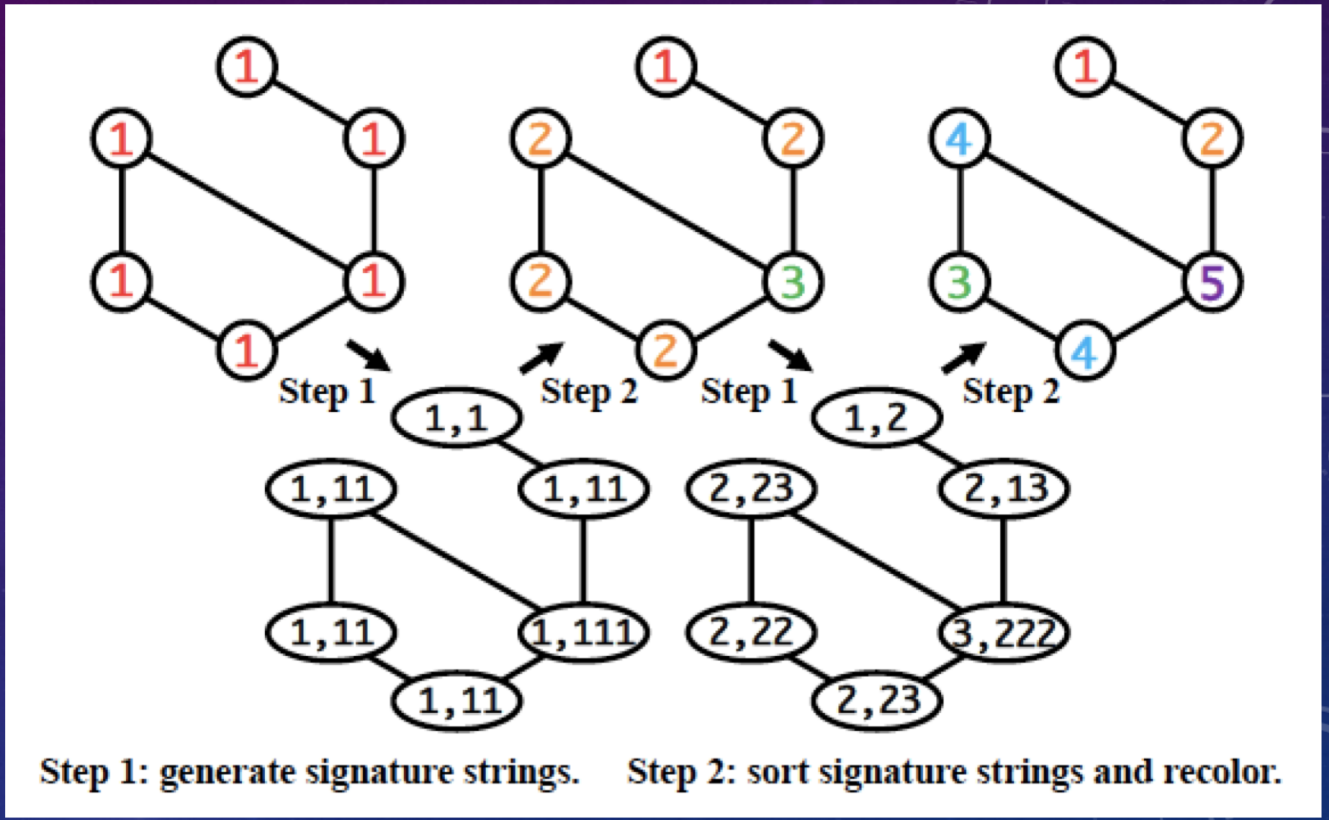


WL Algorithm Review

Basic Idea [WL, 1968]

(1) All vertices are set to the same color “1”

(2) Each vertex then gets a signature string by concatenating its own color and the sorted colors of its immediate neighbors.



Step 1: generate signature strings. Step 2: sort signature strings and recolor.

(3) Vertices are then sorted by the ascending order of their signature strings and assigned new colors 1, 2, 3.

(4) Vertices with the same signature strings get the same color.

One Modified WL-Algorithm: “Palette-WL”

Instead of partitioning vertices using the signature string for each vertex x, vertices are iteratively partitioned using a hash value h(x) for each vertex x.

The hash value function h is a 1-1 mapping from signatures to real values.

This modification preserves information encoding which pair of vertices is “e_{ij}”. The property which holds is called “Perfect Hashing”.

Let Γ(x) be the vertices adjacent to x, and c(x) the color of x from set C.

h has perfect hashing means that: h(x) = h(y) if and only if
c(x) = c(y), and Γ(x) and Γ(y) contain the same colors with the same cardinality

Palette-WL

In this algorithm, c(x) = f(h(x)) where $f: \mathbb{R}^N \rightarrow \mathrm{C}^\mathrm{N}$ is a fixed, order-preserving (smallest real maps to 1, next smallest to 2, etc) function.

We use the following hash function [X] which has perfect hashing.

- The initial hash h(v) for vertex v in G_{ij} is $\sqrt{d(x_i, v)d(v, x_j)}$) where $\mathrm{d}(v_1, v_2)$ is the length of the shortest path from v_1 to v_2.
- In subsequent iterations,

$$h(x) = c(x) + \frac{1}{\left\lceil \sum_{z' \in V_K} \log(\mathcal{P}(c(z'))) \right\rceil} \cdot \sum_{z \in \Gamma(x)} \log(\mathcal{P}(c(z)))$$

P(n) is the nth prime number, V_K is a particular vertex set G_{ij}

The reason for the complex form is that this particular hash function preserves color orderings from iteration to iteration, a key feature of WL.

WL Models – Full Outline

Works via three steps.

1. Extract graphs for every vertex pair

   Generate the K-vertex graphs G_{ij} for every vertex pair $(v_i, \mathrm{v_j})$.
2. Encode graph patterns.

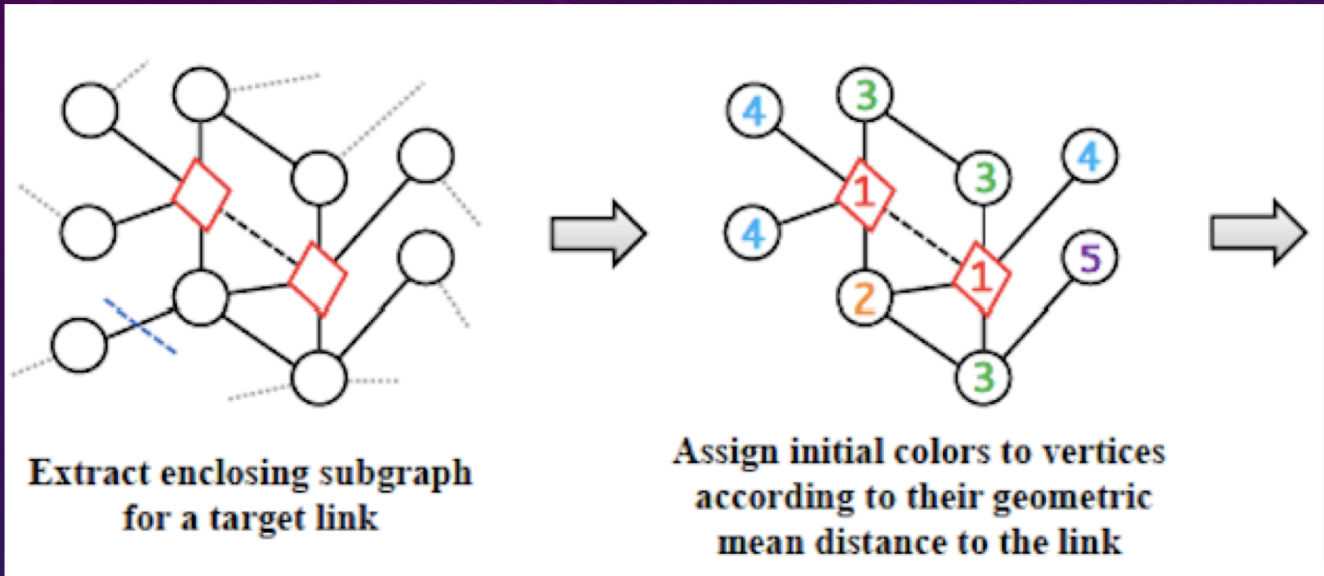
   Code each G_{ij} with an adjacency matrix; vertex ordering via (a modified) WL. [We use Palette-WL.]
3. Network Training.

   Learn nonlinear graph theoretic features for link prediction.

Different WL-Models arise from the training method chosen (e.g. linear regression, neural network), and the (modified)WL-algorithm used.

WL Models – Full Outline

A Depiction of the process when a neural network is used:



WL Models Contra the State-of-the-Art

Palette-WL with Neural Network [2] outperforms

- 9 state-of-the-art link-prediction methods developed by heuristic means
  Katz, PageRank, SimRank, Common Neighbor, Resource Allocation, Jaccard, Preferential Attachment, Adamic-Adar, Resistance Distance

- 3 latent feature models
  Stochastic Model Block, 2 Matrix Factorization Methods

Performance measurements were via Area Under the ROC Curve (AUC).

WL Models Contra the State-of-the-Art

"Area under Curve (AUC)" Results [2]

Data	CN	Jac	AA	RA	PA	Katz	RD	PR
USAir	0.940	0.903	0.950	0.956	0.894	0.931	0.898	0.944
NS	0.938	0.938	0.938	0.938	0.682	0.940	0.582	0.940
PB	0.919	0.873	0.922	0.923	0.901	0.928	0.883	0.935
Yeast	0.891	0.890	0.891	0.892	0.824	0.921	0.880	0.927
C.ele	0.848	0.792	0.864	0.868	0.755	0.864	0.740	0.901
Rank	7.875	10.625	7.500	6.875	12.875	7.125	10.375	5.125

Data	SR	SBM	MF-c	MF-r	$WLLR^{10}$	$WLNM^{10}$
USAir	0.782	0.944	0.918	0.849	0.896	0.958
NS	0.940	0.920	0.636	0.720	0.862	0.984
PB	0.773	0.938	0.930	0.943	0.827	0.933
Yeast	0.914	0.914	0.831	0.881	0.854	0.956
C.ele	0.760	0.867	0.832	0.844	0.803	0.859
Rank	11.000	5.625	10.500	9.500	10.125	2.500

The five datasets used above are USAir, NS, PB, Yeast, and C.ele. USAir is a network of US airlines. NS is a collaboration network of researchers who publish papers on network science. PB is a network of US political blogs. Yeast is a protein-protein interaction network in yeast. C.ele is a neural network of C. elegans.

WL Models Contra the State-of-the-Art

Further Results [1, B. 2018]

We did further testing against 22 additional embedding-based link prediction techniques arising from common neighbor-, path-, and random walk-based schemes.

Data Set	WLLR	Top Score	Models with Top Score	Top Score Class	Top Score in Our Paper
USAir	0.930	0.9540	RA	Common neighbor	
NS	0.865	0.9690	LHNII B ∈{3,4,5}	Path-based	
PB	0.838	0.9367	LRW3	Random-walk	
Yeast	0.860	0.8990	AA, RA, LNBAA	Common neighbor	
C.ele	0.804	0.9197	LRW3	Random-walk	

WL Models Contra the State-of-the-Art

Our focus was on training with linear regression. The reason for this is that we seek to implement WL models which do not have to rely on heavy computational neural networks. Although, we due pursue this line as well.

While neural network results fair better, our eventual goal is to use recent work which, via hashing, allows heavy deep learning capabilities in small operating systems, such as those on smartphones. Aspects of this are already in use (e.g. Siri for natural language processing; 200 MB in iPhone)

We believe that variations of WL Models, along with tools from algorithmic randomness may be employed to make our goal successful.

Thank you.

References

[1] K. Brodhead, "Link Prediction Schemes Contra Weisfeiler-Leman Models", Int. Journal of Adv. Computer Science and Applications 9, no 6 (2018) 16-24.

[2] Y. Chen and M. Zhang, Weisfeiler-Lehman Neural Machine for Link Prediction, 23rd ACM SIGKDD Int. Conf. on KDD Mining, New York, NY, USA, pp. 575-583, 2017.