Automatic Feature Learning
In Graphs

CS224W: Analysis of Networks
Jure Leskovec, Stanford University
http://cs224w.stanford.edu
Machine Learning in Networks

Node classification
(Supervised) Machine Learning Lifecycle: This feature, that feature. Every single time!

Raw Data → Structured Data → Learning Algorithm → Model

Feature Engineering

Automatically learn the features

Downstream prediction task
Goal: Efficient task-independent feature learning for machine learning in networks!
Why network embedding?

- We map each node in a network into a low-dimensional space
  - Distributed representation for nodes
  - Similarity between nodes indicates link strength
  - Encode network information and generate node representation

![Diagram](attachment:image.png)

- Anomaly Detection
- Attribute Prediction
- Clustering
- Link Prediction
- ...
Example

- Zachary’s Karate Club network:
Graph representation learning is hard:

- Images are fixed size
  - Convolutions (CNNs)
- Text is linear
  - Sliding window (word2vec)
- Graphs are neither of these!
  - Node numbering is arbitrary (node isomorphism problem)
  - Much more complicated structure
node2vec: Random Walk Based (Unsupervised) Feature Learning

node2vec: Scalable Feature Learning for Networks
Goal: Embed nodes with similar network neighborhoods close in the feature space.

We frame this goal as prediction-task independent maximum likelihood optimization problem.

Key observation: Flexible notion of network neighborhood $N_S(u)$ of node u leads to rich features.

Develop biased 2\textsuperscript{nd} order random walk procedure S to generate network neighborhood $N_S(u)$ of node u.
Intuition: Find embedding of nodes to $d$-dimensions that preserves similarity

Idea: Learn node embedding such that nearby nodes are close together

Given a node $u$, how do we define nearby nodes?

- $N_S(u)$ ... neighbourhood of $u$ obtained by some strategy $S$
**Feature learning as optimization**

- Given $G = (V, E)$,
- Our goal is to learn a mapping $f : u \rightarrow \mathbb{R}^d$.

- Log-likelihood objective:
  $$\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$$
  - where $N_S(u)$ is neighborhood of node $u$.

- Given node $u$, we want to learn feature representations predictive of nodes in its neighborhood $N_S(u)$. 
Feature learning as optimization

\[
\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))
\]

- **Assumption**: Conditional likelihood factorizes over the set of neighbors.

\[
\log \Pr(N_S(u) | f(u)) = \sum_{n_i \in N_S(u)} \log \Pr(f(n_i) | f(u))
\]

- Softmax parametrization:

\[
\Pr(f(n_i) | f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}
\]
Maximize the objective using Stochastic Gradient descent with **negative sampling**.

- Computing the **summation** is expensive
- **Idea**: Just sample a couple of “negative nodes”
- This means at each iteration only embeddings of a few nodes will be updated at a time
- Much faster training of embeddings

\[
\max_f \sum_u \sum_{n \in N_S(u)} \log \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}
\]
Two classic strategies to define a neighborhood $N_S(u)$ of a given node $u$:

- **Local microscopic view:**
  
  $N_{BFS}(u) = \{s_1, s_2, s_3\}$

- **Global macroscopic view:**
  
  $N_{DFS}(u) = \{s_4, s_5, s_6\}$
BFS vs. DFS

BFS: Micro-view of neighbourhood

DFS: Macro-view of neighbourhood
Biased random walk $S$ that given a node $u$ generates neighborhood $N_S(u)$

- Two parameters:
  - Return parameter $p$:
    - Return back to the previous node
  - In-out parameter $q$:
    - Moving outwards (DFS) vs. inwards (BFS)
$N_{S}(u)$: Biased 2$^{nd}$-order random walks explore network neighborhoods:

- **BFS-like:** low value of $p$
- **DFS-like:** low value of $q$

$p, q$ can be learned in a semi-supervised way
node2vec algorithm

1) Compute random walk probs.
2) Simulate $r$ random walks of length $l$ starting from each node $u$
3) Optimize the node2vec objective using Stochastic Gradient Descent

Linear-time complexity.
All 3 steps are individually parallelizable
Experiments: Micro vs. Macro

Interactions of characters in a novel:

\[ p=1, \; q=2 \]

Microscopic view of the network neighbourhood

\[ p=1, \; q=0.5 \]

Macroscopic view of the network neighbourhood
Scalability of node2vec

Scalability on Erdos-Renyi graphs with average degree 10

- Red dots: with SGD
- Blue dots: without SGD

Log_{10} time (in seconds) vs. Log_{10} nodes
Incomplete Network Data (PPI)

Fraction of missing edges

Macro-F₁ score

Fraction of additional edges

Macro-F₁ score
General-purpose feature learning in networks:

- An explicit locality preserving objective for feature learning.
- Biased random walks capture diversity of network patterns.
- Scalable and robust algorithm with excellent empirical performance.
- Future extensions would involve designing random walk strategies entailed to network with specific structure such as heterogeneous networks and signed networks.
OhmNet: Extension to Hierarchical Networks
Let’s generalize node2vec to multilayer networks!
Each network is a layer $G_i = (V_i, E_i)$

Similarities between layers are given in hierarchy $\mathcal{M}$, map $\pi$ encodes parent-child relationships
The Approach

- **Computational framework** that learns features of every node and at every scale based on:
  - Edges *within each layer*
  - Inter-layer relationships between nodes active on different layers
OhmNet

Output: embeddings of nodes in layers as well as internal levels of the hierarchy

Input:

$G_1$, $G_2$, $G_3$, $G_4$, $G_5$, $G_6$, $G_7$, $G_8$, $G_9$, $G_{10}$
**OhmNet**

- **OhmNet**: Given layers $G_i$ and hierarchy $M$, learn node features captured by functions $f_i$.

- Functions $f_i$ embed every node in a $d$-dimensional feature space.

A multi-layer network with four layers and a two-level hierarchy $M$. 

---

28
Given: Layers $\{G_i\}$, hierarchy $\mathcal{M}$

- Layers $\{G_i\}_{i=1..T}$ are in leaves of $\mathcal{M}$

Goal: Learn functions: $f_i: V_i \rightarrow \mathbb{R}^d$
Features in Multi-Layer Network

- Approach has two components:
  - **Per-layer objectives**: Nodes with similar network neighborhoods in each layer are embedded close together.
  - **Hierarchical dependency objectives**: Nodes in nearby layers in hierarchy are encouraged to share similar features.
Intuition: For each layer, find a mapping of nodes to $d$-dimensions that preserves node similarity

Approach: Similarity of nodes $u$ and $v$ is defined based on similarity of their network neighborhoods

Given node $u$ in layer $i$ we define nearby nodes $N_i(u)$ based on random walks starting at node $u$
Per-Layer Objective: node2vec

- Given node $u$ in layer $i$, learn $u$'s representation such that it predicts nearby nodes $N_i(u)$:
  \[
  \omega_i(u) = \log Pr(N_i(u)|f_i(u))
  \]

- Given $T$ layers, maximize:
  \[
  \Omega_i = \sum_{u \in V_i} \omega_i(u), \quad \text{for } i = 1, 2, \ldots, T
  \]

- **Notice:** Nodes in different networks representing the same entity have different features
Interdependent Layers

- So far, we did not consider hierarchy $\mathcal{M}$
- Node representations in different layers are learned independently of each other

How to model dependencies between layers when learning node features?
We use regularization to share information across the hierarchy

We want to enforce similarity between feature representations of networks that are located nearby in the hierarchy
Given node $u$, learn $u$’s representation in layer $i$ to be close to $u$’s representation in parent $\pi(i)$:

$$c_i(u) = \frac{1}{2} \| f_i(u) - f_{\pi(i)}(u) \|_2^2$$

**Multi-scale:** Repeat at every level of $\mathcal{M}$

$$C_i = \sum_{u \in L_i} c_i(u)$$

$L_i$ has all layers appearing in sub-hierarchy rooted at $i$.
Nodes in different layers representing the same entity have the same features in hierarchy ancestors

We learn feature representations at multiple scales:
- features of nodes in the layers
- features of nodes in non-leaves in the hierarchy

This model is more efficient than the fully pairwise model, where dependencies between layers are modeled by pairwise comparisons of nodes across all pairs of layers
**OhmNet: Final Model**

Learning node features in multi-layer networks

Solve maximum likelihood problem:

\[
\max_{f_1, f_2, \ldots, f_{|M|}} \sum_{i \in \mathcal{T}} \Omega_i - \lambda \sum_{j \in \mathcal{M}} C_j .
\]

- Per-layer network objectives
- Hierarchical dependency objectives
Proteins are worker molecules
- Understanding protein function has great biomedical and pharmaceutical implications
- Function of proteins depends on their tissue context
  [Greene et al., Nat Genet ‘15]
Protein functions are tissue-specific

- The precise function of proteins depends on their tissue context (Greene et al., Nat Genet 2015)
- Diseases result from the failure of tissue-specific processes (Hu et al., Nat Rev Genet 2016)
- Current models assume that protein functions are constant across tissues
A multi-layer tissue network has many network layers (tissues)
Each layer corresponds to one tissue-specific protein interaction network
Hierarchy $M$ encodes biological similarities between the tissues at multiple scales
107 genome-wide tissue-specific protein interaction networks

- 584 tissue-specific cellular functions
- Examples (tissue, cellular function):
  - (renal cortex, cortex development)
  - (artery, pulmonary artery morphogenesis)
Brain Tissues

9 brain tissue PPI networks in two-level hierarchy
Meaningful Node Embeddings

Brainstem

- Cerebellum
- Medulla oblongata
- Substantia nigra

- Frontal lobe
- Temporal lobe
- Pons

Brain

- Parietal lobe
- Occipital lobe
- Midbrain
Experimental setup

- Cellular function prediction is a multi-label node classification task

- Every node (protein) is assigned one or more labels (cellular functions)

Setup:
- We apply OhmNet, which for every node in every layer learns a separate feature vector in an unsupervised way.
- For every layer and every function we then train a separate one-vs-all regularized linear classifier using the modified Huber loss.
- During the training phase, we observe only a certain fraction of proteins and all their cellular functions across the layers.
- The task is then to predict the tissue-specific functions for the remaining proteins.
Protein Function Prediction
Protein Function Prediction

- 42% improvement over state-of-the-art on the same dataset
Transfer Learning

Transfer functions to unannotated tissues

- **Task:** Predict functions in target tissue without access to any annotation/label in that tissue

<table>
<thead>
<tr>
<th>Target tissue</th>
<th>OhmNet</th>
<th>Tissue non-specific</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placenta</td>
<td>0.758</td>
<td>0.684</td>
<td>11%</td>
</tr>
<tr>
<td>Spleen</td>
<td>0.779</td>
<td>0.712</td>
<td>10%</td>
</tr>
<tr>
<td>Liver</td>
<td>0.741</td>
<td>0.553</td>
<td>34%</td>
</tr>
<tr>
<td>Forebrain</td>
<td>0.755</td>
<td>0.632</td>
<td>20%</td>
</tr>
<tr>
<td>Blood plasma</td>
<td>0.703</td>
<td>0.540</td>
<td>40%</td>
</tr>
<tr>
<td>Smooth muscle</td>
<td>0.729</td>
<td>0.583</td>
<td>25%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.746</td>
<td>0.617</td>
<td>21%</td>
</tr>
</tbody>
</table>

Reported are AUC values