AutoNE: Hyperparameter Optimization for Massive Network Embedding

Ke Tu       Jianxin Ma       Peng Cui       Jian Pei       Wenwu Zhu
Tsinghua U  Tsinghua U  Tsinghua U  SFU&JD  Tsinghua U
Networks are widely used to represent the rich pairwise relationships of data objects.
Network Embedding

Networks embedding aims to learn a low-dimensional representation for each node
Existing Embedding Methods

- Link Prediction
- Community Detection
- Node Classification
- Network Distance
- Node Importance
- …

Various network properties
- High-order
- Transitivity
- Global position

Various applications

- Leading to a large number of hyperparameters
  - E.g. Deepwalk: number of walks, walk length, window size
- Must be carefully tuned

AutoML
AutoML

• Ease the adoption of machine learning and reduce the reliance on human experts
  • e.g., hyperparameter optimization

• Network data remains largely unexplored

• Large scale issue:
  • Complexity of Network Embedding is usually at least $O(E)$
    • $E$ is the number of edges (can be 10 billion)

  • Total complexity: $O(ET)$, $T$ is the times searching for optimal hyperparameter

How to incorporate AutoML into massive network embedding efficiently? (reduce $E$ and $T$)
incorporating AutoML into NE

- A straightforward way: configuration selection on sampled sub-networks

- **Transferability**
  - $\theta \neq$ optimal configuration on origin network

- **Heterogeneity**
  - several highly heterogeneous components $\Rightarrow$ carefully designed sampling
AutoNE

Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network
AutoNE

Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network
Transfer the **knowledge** about optimal hyperparameters from the sub-networks to the original massive network.
Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network.
Sampling Module

• **Goal**: Sample a series of representative sub-networks that share similar properties with the original large-scale network

• **Heterogeneity** issue: preserve diversity of the origin network

• Origin network $G = (V, E)$
• Random walk $w = (v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n)$
• Sub-graph $G_s$ based random walk $w$:
  • Nodes: $V' = \{v_1, v_2, \ldots, v_n\}$
  • Edges: $E' = \{(v_i, v_j) | i, j \in [1, n], (v_i, v_j) \in E\}$

• The starting points $v_1$
  • Supervised: several nodes with different labels
  • Unsupervised: from the different communities, e.g., a greedy algorithm that maximizes modularity
Signature Extraction Module

• The signature of a network is a vector descriptor that encodes the various properties of the whole network

• Graph signature $h(G)$
  
  \[ \text{Similarity}(G_1, G_2) = \text{Distance}(h(G_1), h(G_2)) \]

• **Challenge**: The signatures should be comprehensive enough
  
  • Based on spectral graph theory, a large number of network properties are decided by the spectrum of a network, e.g. the normalized cuts used by spectral clustering
  
  • We choose NetLSD [Tsitsulin et al. KDD18]
    
    • heat diffusion process on a network
    
    \[ h_t(G) = tr(H_t) = tr(e^{-tL}) = \sum_j e^{-t\lambda_j} \]
Meta-Learning Module

• Performance function $f_M(\theta, G) \rightarrow P$

• **Transferability** issue:
  • Assumption: Two similar network will has similar optimal hyperparameter on the same network embedding method

• Learning Performance function on small sampling networks, and predict on origin massive network

• We choose $f_M$ as **Gaussian Process** like Bayesian optimization
• The log likelihood: (K is the kernel function)

$$\ln p(f \mid X) = -\frac{1}{2} f^T K(X, X)^{-1} f - \frac{1}{2} \ln \det(K(X, X)) + \text{constant}.$$
Meta-Learning Module

- Given a new test point $x_\ast = (\theta_\ast, h(G_\ast))$ and the observed values $f$
  - The predicted performance $f_\ast$ and $f$ follow a joint normal distribution
  - The posterior distribution $p(f_M(\theta_\ast, h(G_\ast) | x_\ast, f, X)$ is a normal distribution:
    \[
    f_M(\theta_\ast, G_\ast) | x_\ast, f, X \sim N(\mu_\ast, \sigma^2_\ast),
    \]
    \[
    \mu_\ast = K(x_\ast, X)K(X, X)^{-1}f, \]
    \[
    \sigma^2_\ast = K(x_\ast, x_\ast) - K(x_\ast, X)K(X, X)^{-1}K(X, x_\ast).
    \]
- Optimize $\theta_\ast$: Upper confidence bound: $\arg \max_{\theta_\ast} \mu_\ast + \kappa \sigma_\ast$.
- Kernel function $K$:
  - Measure the similarity between two sets of hyperparameters and the similarity between two networks
    \[
    k((\theta_1, G_1), (\theta_2, G_2)) = k_\theta(\theta_1, \theta_2) \cdot k_g(h(G_1), h(G_2)).
    \]
**Experiment Setting --- datasets**

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Node</th>
<th>Edge</th>
<th>Label</th>
<th>Feature (For GCN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlogCatalog</td>
<td>10,312</td>
<td>333,983</td>
<td>39</td>
<td>-</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>4,777</td>
<td>184,812</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19,717</td>
<td>44,338</td>
<td>3</td>
<td>500</td>
</tr>
<tr>
<td>TopCat</td>
<td>1,791,489</td>
<td>28,511,807</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Experiment Setting --- NE

- Network Embedding method: Three classes
  - **Sampling-based NE: Deepwalk** [Perozzi, Bryan, et.al. KDD14]
    - Number of random walks
    - Length of each random walk
    - Windows size
  - **Factorization based NE: AROPE** [Zhang, Ziwei, et al. KDD18]
    - The weights of the different orders (Max 4)
    - Learning rate
    - Size of each hidden layer
    - Number of training epochs
    - The dropout rate
    - The weight decay
Experiment Setting --- Baselines and Tasks

- **Baselines**
  - **Random search**
    - find the optimal solution as long as the time budget is large enough
    - Instead of grid search
      - explore larger configuration space more efficiently
      - find better solutions faster
  - **Bayesian optimization (BayesOpt)**
    - the most used method in AutoML framework
    - performs many trials on the original data
      - makes it inefficient at handling large-scale networks.
- **Tasks**
  - Link prediction and node classification
Experiment --- Sampling-Based NE

The performance achieved within various time thresholds.

The number of trials to reach a certain performance threshold
Experiment---Factorization-Based NE

The performance achieved within various time thresholds.

The number of trials to reach a certain performance threshold
Experiment --- Deep NN-Based NE

(a) The performance achieved by each method within various time thresholds.  
(b) The number of trials required to reach a certain performance threshold.

Node classification on Pubmed.
Experiment --- Large-Scale

Table 1: Results on a massive network with around thirty million edges, where we can only afford to run a NE algorithm on the whole network for a few times.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trial 1</th>
<th></th>
<th>Trial 2</th>
<th></th>
<th>Trial 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUC</td>
<td>Time(s)</td>
<td>AUC</td>
<td>Time(s)</td>
<td>AUC</td>
<td>Time(s)</td>
</tr>
<tr>
<td>AutoNE</td>
<td>0.717</td>
<td>1067.9</td>
<td>0.726</td>
<td>1856.2</td>
<td>0.769</td>
<td>2641.9</td>
</tr>
<tr>
<td>Random</td>
<td>0.714</td>
<td>698.3</td>
<td>0.727</td>
<td>1426.3</td>
<td>0.715</td>
<td>2088.6</td>
</tr>
<tr>
<td>BayesOpt</td>
<td>0.715</td>
<td>702.5</td>
<td>0.714</td>
<td>1405.1</td>
<td>0.727</td>
<td>2307.7</td>
</tr>
</tbody>
</table>
Summary

• Investigate the pressing problem of incorporating AutoML into NE
• Propose a novel framework AutoNE
  • Automate hyperparameter optimization for NE.
• Can scale up to massive real-world networks
  • Utilizing the meta-knowledge transferred from sampled sub-networks.
• Extensive experiment on real-world networks
  • Four real-world dataset
  • Three representative network embedding methods
  • Two baselines
Thanks!

Ke Tu, Tsinghua University
tuke1993@gmail.com

AutoNE: Hyperparameter Optimization for Massive Network Embedding