Graph-based Machine Learning

Yiming Yang

Three Parts

• Part I. Relational Learning over Heterogeneous Graphs
  • Transductive Learning over Product Graphs (TOP) (H Liu and Y Yang, ICML’15, ICML’16)

• Part 2. Node Feature Inference [Yuexin]

• Part 3. Analogical Inference for Knowledge Base Completion
  • Graph-based Entity/Relation Embedding (ANALOGY) (H Liu, Y Wu and Y Yang, ICML 2017)
Part 1 Outline

• Motivation
• Semi-supervised learning over a single graphs
• Relational learning over heterogeneous graphs

Motivation

• Graphs have been widely studied in ML
  • Social network analysis (e.g., PageRank, HITS)
  • Clustering (via graph cutting)
  • Link prediction (collaborative filtering)
  • Semi-supervised classification (next slide)
  • Knowledge graph completion (entities, relations, combinations)
Simi-Supervised Learning (SSL) for Classification

**Learning Decision Boundary**
- based on the labeled data only (white and black)
- based on the manifolds of both labeled and unlabeled data (gray)

(from Wikipedia)

Graphical representation of node dependencies

- \( G = (V,E) \) with vertices \( V \) and edges \( E \) and adjacency matrix \( A \) of edge weights;
- e.g., each vertex is a document, each edge is the cosine-similarity of a document pair;
Graph-based SSL for Classification

- $f \in [0,1]^{|V|}$, system-learned values of vertices w.r.t. a class of interest;
- Optimize belief propagation over the graph
  \[
  \min_f l_\mathcal{O}(f) + \lambda f^T A^{-1} f
  \]
  - 1st term $l_\mathcal{O}(f)$ is the training-set loss, quantifying the differences b/w system-predicted $f$ and the true observations $\mathcal{O}$;
  - 2nd term $f^T A^{-1} f$ is the graph regularization term, reinforcing a smooth belief propagation over the connected nodes;
- $\lambda$ is a hyper-parameter, balancing the relative weights of the two terms.

Underlying Assumption

- Objective: $\min_f l_\mathcal{O}(f) + \lambda f^T A^{-1} f$
- Assuming a Gaussian prior of $f$ with covariance $A$ as
  \[
  f \sim N(0,A)
  \]
  i.e., $\log P(f) \propto -f^T A^{-1} f$
- Intuitively, adjacent nodes should have similar values.
Spectral Transform (ST) of the Graph

Original Objective: \[ \min_f l_O(f) + \lambda f^T A^{-1} f \]
Enhanced Objective: \[ \min_f l_O(f) + \lambda f^T \kappa(A)^{-1} f \]

- Different ways of using the graph
  \[ \kappa_{\text{rw}}(A) = A^k \] for \( k \)-step random walk
  \[ \kappa_{\text{von Neumann}}(A) = (I - A)^{-1} = I + A + A^2 + \cdots \] for infinite walk
  \[ \kappa_{\text{heat}}(A) = e^A = I + A + \frac{1}{2!} A^2 + \frac{1}{3!} A^3 + \cdots \] for weight-decayed walk

From Single Graph to \( k \)-Graphs for Label/Value Propagation

Graph L

User’s rating over items

Graph \( G \)

User-User Social Net

Graph H

Item-Item Similarities

Graph L

User’s rating over items

Recommendation Task: Predict the unknown links with scores (1-5)
Expert Finding as Bipartite Link Prediction

Task: Predict the relevant authors or related domain experts given a paper of interest

Cross-Language Topic Model Translation (R Xu et al., ACL’16)

Transductive Learning Over a Product Graph (TOP)
Transductive Link Prediction
Over a Product Graph (TOP)

G1: word similarity graph in L1
Bilingual Dictionary (with Link Prediction)
G2: word similarity graph in L2

e.g., The 1st "?" (from the top) will have a high score if the green dashed path is strong.
e.g., The 2nd "?" will have a high score if the both the green and red paths are strong.

Multi-Relational Learning (H Liu et al. ICML’16)
e.g., Who-What-Where in DBLP

“Hinton published CapsNet paper at NIPS”

Relational tuple: (Hinton,CapsNet_paper,NIPS).

Semi-supervised learning: Given labeled tuples (small set), predict the unknown tuples based on ...
Why Graphs?

- Labeled data are very limited.
- Unlabeled graphs (e.g., co-author graph, citation graph, knowledge graph) are widely available.
- Cross-graph label propagation should help.

![Figure: Cross-graph Inference for predicting known tuples](image)

How do we get there?

**Product Graph** ($\mathcal{P}$) induced from $G^{(1)}, \ldots, G^{(J)}$:

- Each node in $\mathcal{P}$ is a tuple;
- Each link in $\mathcal{P}$ combines the linkages in $G$'s.
Induce the link weights in $\mathcal{P}$

Tensor Product: $\mathcal{P}(G^{(1)}, G^{(2)}, G^{(3)}) = G^{(1)} \otimes G^{(2)} \otimes G^{(3)}$

Cartesian Product: $\mathcal{P}(G^{(1)}, G^{(2)}, G^{(3)}) = G^{(1)} \oplus G^{(2)} \oplus G^{(3)}$

- **Soft AND** (Tensor Product): If strong links in all the input graphs from node “a” to “b”, then we will have a strong link between “a” and “b” in $\mathcal{P}$;
- **Soft OR** (Cartesian Product): If strong link in any the input graph from node “a” to “b”, then we will have a strong link between “a” and “b” in $\mathcal{P}$.

Notation

- $G_j$ for $j = 1, 2, \cdots, J$ are the input individual graphs;
- $\mathcal{P} = \mathcal{P}(G_1, G_2, \cdots, G_J)$ is the output Product Graph;
- $n_j$ is the number vertices in input $G_j$;
- $N = n_1 \times \cdots \times n_J$ is the number of vertices in $\mathcal{P}$;
- $N^2$ is the number of edges in $\mathcal{P}$;
- $X \in \mathbb{R}^{n_1 \times \cdots \times n_J}$ is an $J$–way tensor of the vertex scores in $\mathcal{P}$;
- $C \in \mathbb{R}^{N \times N}$ is the matrix of link weights in $\mathcal{P}$. 
Why Product Graph?

Label propagation over $\mathcal{P}$:

- Given some labeled nodes (positive instances), we can propagate the labels to unlabeled notes via the links.

Optimization Problem

$$\min_f \quad l_0(f) + \lambda f^T C^{-1} f$$

- $f = vec(X) \in R^N$ is the flat vector representation of tensor $X$;
- $f^T C^{-1} f \equiv \|f\|_C^2$ is the L2-norm of $f$ adjusted by matrix $C$;
- Minimizing norm $\|f\|_C^2$ is equivalent to imposing a Gaussian prior on $f$.
- Vertices should have similar scores if they are strongly connected in $C$ (or $\mathcal{P}$).
- The (quadratic) objective function defines a convex optimization problem.
Computational Tractability

- # of nodes in $\mathcal{P}$ is $N = n_1 \times n_2 \times n_3 \cdots$
- # of edges is $N^2$ which is computationally prohibitive when $N$ is too large.
- We must solve the problem without explicitly constructing $\mathcal{P}$.
- We use Low-rank Tensor Decomposition to tackle this problem
  1) We focus on the Spectral Graph Products (SGP) as the space of $\mathcal{P}$.
  2) We compute the eigen-decomposition of each graph independently;
  3) We combine the top-$k$ eigenvectors & eigenvalues to approximate $\mathcal{P}$.

Spectral Graph Product (SGP)

- A family of product graphs
  - Whose eigenvectors $\{v(C)\} = \{v_i^{(G_1)} \otimes v_j^{(G_2)} \otimes v_k^{(G_3)} \cdots\}$
  - Whose eigenvalues $\{\lambda(C)\} = \{\kappa\left(\lambda_i^{(G_1)}, \lambda_j^{(G_2)}, \lambda_k^{(G_3)}, \cdots\right)\}$
- Examples
  - $\kappa\left(\lambda_i^{(G_1)}, \lambda_j^{(G_2)}, \cdots\right) = \lambda_i^{(G_1)} \times \lambda_j^{(G_2)} \times \cdots \implies \mathcal{P} \otimes = G_1 \otimes G_2 \otimes \cdots$ (Tensor Product)
  - $\kappa\left(\lambda_i^{(G_1)}, \lambda_j^{(G_2)}, \cdots\right) = \lambda_i^{(G_1)} + \lambda_j^{(G_2)} + \cdots \implies \mathcal{P} \oplus = G_1 \oplus G_2 \oplus \cdots$ (Cartesian Product)
  - $\kappa\left(\lambda_i^{(G_1)}, \lambda_j^{(G_2)}, \cdots\right) = e^{\lambda_i^{(G_1)} \lambda_j^{(G_2)}} \cdots \implies \mathcal{P}_e \otimes = I + \mathcal{P} \otimes + \frac{\mathcal{P}^2 \otimes}{2} + \cdots$ (Heat Diffusion)
Scalable Approximation via a Low-rank Tensor

Figure: Tucker Decomposition, where $\alpha$ is the core tensor.

- Tensor algebras are carried out on GPU.

Computing the Graph-based Regularization Term

$$\|f\|_C^2 \equiv f^T C^{-1} f$$

$$= \sum_{i_1, i_2, \ldots, i_J=1}^{n_1, n_2, \ldots, n_J} \left( \frac{\text{TensorProd}(f; v_{i_1}^{(G1)}, v_{i_2}^{(G2)}, \ldots, v_{ij}^{(GJ)})}{\kappa \left( \lambda_{i_1}^{(G1)}, \lambda_{i_2}^{(G2)}, \ldots, \lambda_{ij}^{(GJ)} \right)} \right)^2$$

$$\approx \sum_{k_1, k_2, \ldots, k_J}^{k_1, k_2, \ldots, k_J} \left( \frac{\text{TensorProd}(\alpha; v_{k_1}^{(G1)}, v_{k_2}^{(G2)}, \ldots, v_{k_J}^{(GJ)})}{\kappa \left( \lambda_{k_1}^{(G1)}, \lambda_{k_2}^{(G2)}, \ldots, \lambda_{k_J}^{(GJ)} \right)} \right)^2$$

$\leftarrow$ J-mode Product on the full tensor, taking time $O \left( \sum_{j=1}^J n_j \prod_{j=1}^J n_j \right)$

$\leftarrow$ J-mode Product on the core tensor, taking time $O(d^J)$, $d \ll n_j$
Smoothing Effect (a simulated example)

- The low-rank approximation prunes off the high-volatility factors (eigen-vectors) of $\mathcal{P}$

![Figure: Eigenvectors of $G_1$ (blue), $G_2$ (red) and $\mathcal{P}(G_2)$.]

Evaluation

**Datasets**

- **Enzyme** 445 compounds, 664 proteins.
- **DBLP** 34$K$ authors, 11$K$ papers, 22 venues.

**Representative Baselines**

- **TF/GRTF** Tensor Factorization/Graph-Regularized TF
- **NN** One-class Nearest Neighbor
- **RSVM** Ranking SVMs
- **LTKM** Low-Rank Tensor Kernel Machines
Evaluation

Our method: “TOP” (blue).

Figure: Performance on Enzyme (above) and DBLP (below).

Concluding Remarks for Part 1

- Challenge:
  - Relational learning over heterogeneous graphs

- Our contributions:
  - A unified framework to integrate heterogeneous input graphs into a product graph (PG).
  - Transductive multi-relational learning from both labeled data (sparse) and unlabeled data (massive).
  - A convex approximation for scalable inference over the combinatorially large number of tuples.
  - Strong performance on evaluation benchmarks
References

