GNNRank: Learning Global Rankings from Pairwise Comparisons via Directed Graph Neural Networks

Yixuan He, Quan Gan,
David Wipf, Gesine Reinert,
Junchi Yan, Mihai Cucuringu

ICML International Conference On Machine Learning

Motivation

- Recovering global rankings from pairwise comparisons is an important problem with many applications, ranging from time synchronization to sports team ranking.
- Pairwise comparisons corresponding to matches in a competition can naturally be construed as edges in a directed graph (digraph), whose nodes represent competitors with an unknown rank or skill strength.
- However, existing methods addressing the rank estimation problem have thus far not utilized powerful neural network architectures to optimize ranking objectives.
- Hence, we augment a certain ranking algorithm with neural networks, in particular graph neural networks (GNN) for its intrinsic connection to the problem at hand.

Problem Definition

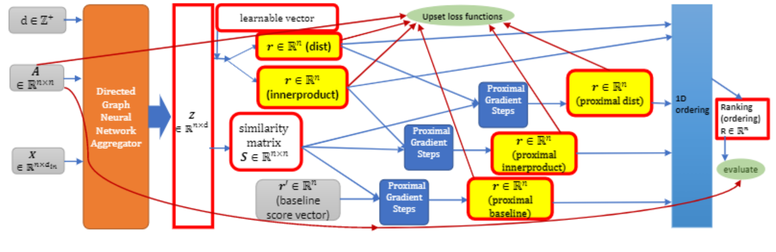
Without loss of generality, we consider pairwise comparisons in a competition, which can be encoded in a directed graph (digraph) $G = (V, E)$.

- The node set V: competitors.
- The edge set E: pairwise comparisons.
- The adjacency matrix: outcomes of the matches.
- Recovering global rankings from pairwise comparisons amounts to assigning an integer R_i to each node $v_i \in V$, denoting its position among competitors, where the lower the rank, the stronger the node is.
- We attempt by first learning a skill-level vector r for all nodes, where a higher skill value corresponds to a lower rank ordering.

SerialRank [1]

- First compute the Laplacian of a certain similarity matrix $\boldsymbol{S}'$.
- The corresponding Fiedler vector of $\boldsymbol{S}'$ then serves as the final ranking estimate, after a global sign reconciliation.
- While often effective in practice, SerialRank depends on the quality of the similarity matrix $\boldsymbol{S}'$.
- To address this issue, we introduce a parameterized GNN model that allows us to compute trainable measures of similarity that are useful for subsequent ranking.

GNNRank Framework Overview



Obtaining Directed Graph Embeddings

Any GNN method which is able to take into account directionality and output node embeddings could be applied. E.g., DIMPA [2], DiGCN [3].

Denoting the final node embedding $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$, the embedding vector z_i for a node v_i is $\boldsymbol{z}_i \coloneqq (\boldsymbol{Z})_{(i,:)} \in \mathbb{R}^d$, the i –th row of $\boldsymbol{Z}$.

Obtaining Final Scores and Rankings

To obtain the final ranking score, we unfold the calculation of a Fiedler vector for the graph constructed from our symmetric similarity matrix $\boldsymbol{S}$ with proximal gradient steps.

From the high-dimensional embedding matrix $\boldsymbol{Z}$, we calculate the symmetric similarity matrix $\boldsymbol{S}$ with $S_{i,j} = \exp(-\left|z_j - z_i\right|^2/(\sigma^2 d))$.

We then apply proximal gradient steps to approximate a Fiedler vector of $\boldsymbol{S}$ which serves as the skill-level vector $\boldsymbol{r}$.

Algorithm 1 Proximal Gradient Steps

Input: Initial score $\mathbf{r}' \in \mathbb{R}^n$, Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$ and $\mathbf{Q} \in \mathbb{R}^{n\times n}$
Parameter: (Initial) learning rate set $\{\alpha_\gamma > 0\}_{\gamma=1}^{\Gamma}$ that could either be fixed or trainable (default: trainable).
Output: Updated score vector $\mathbf{r} = [r_1, \ldots, r_n]^\top$.

Let $\mathbf{y} = \mathbf{r}' - \sum_{i=1}^n r_i'/n$;
$\mathbf{y} \leftarrow \mathbf{Q}'\mathbf{y} \in \mathbb{R}^{n-1}$ ($\mathbf{Q}'$ is $\mathbf{Q}$ with the first row removed);
$\mathbf{y} \leftarrow \mathcal{P}_{\mathcal{S}^{n-1}}(\mathbf{y})$ to have unit 2-norm;
$\widetilde{\mathbf{L}} \leftarrow \left[\mathbf{Q}\mathbf{L}\mathbf{Q}^\top\right]_{2:n,2:n}$.
for $\gamma < \Gamma$ **do**
  $\mathbf{y} \leftarrow \mathbf{y} - \alpha_\gamma(2\widetilde{\mathbf{L}})\mathbf{y}/n$;
  $\mathbf{y} \leftarrow \mathcal{P}_{\mathcal{S}^{n-2}}(\mathbf{y})$ to have unit 2-norm;
  $\gamma \leftarrow \gamma + 1$.
end for
$\mathbf{y} \leftarrow \mathrm{CONCAT}(0, \mathbf{y}) \in \mathbb{R}^n$;
$\mathbf{r} = \mathbf{Q}^\top\mathbf{y}$
return $\mathbf{r}$;

Typical Loss and Objective Functions

- $\mathbf{M}' = \mathbf{A} - \mathbf{A}^\top$
- $\mathbf{r} = [r_1, \ldots, r_n]^\top$: *real-valued* ranking scores as entries
- t: the number of nonzero elements in $\mathbf{M}'$
- $\mathbf{1}$: an all-one column vector
- $\mathbf{T}' = \mathbf{r}\mathbf{1}^\top - \mathbf{1}\mathbf{r}^\top \in \mathbb{R}^{n\times n} \Rightarrow \mathbf{T}'_{i,j} = r_i - r_j, \forall i,j \in \{1, \cdots, n\}$

$$\mathcal{L}_{\text{upset, naive}} = \sum_{i,j:\mathbf{M}'_{i,j}\neq 0} (\text{sign}(\mathbf{T}'_{i,j}) \neq \text{sign}(\mathbf{M}'_{i,j}))/t.$$

$\mathcal{L}_{\text{upset, naive}}$ piecewise constant $\Rightarrow$ not useful in gradient descent.

Solution: $\mathbf{M} = \frac{\mathbf{A}-\mathbf{A}^\top}{\mathbf{A}+\mathbf{A}^\top}, \mathbf{T} = \frac{\mathbf{r}\mathbf{1}^\top - \mathbf{1}\mathbf{r}^\top}{\mathbf{r}\mathbf{1}^\top + \mathbf{1}\mathbf{r}^\top} \in \mathbb{R}^{n\times n} \Rightarrow \mathbf{T}_{i,j} = \frac{r_i - r_j}{r_i + r_j}$
With $\tilde{\mathbf{T}}$ where $\tilde{\mathbf{T}}_{i,j} = \mathbf{T}_{i,j}$ if $\mathbf{M}_{i,j} \neq 0$ and $\tilde{\mathbf{T}}_{i,j} = 0$ if $\mathbf{M}_{i,j} = 0$, the differentiable upset loss is then defined as

$$\mathcal{L}_{\text{upset, ratio}} = \left\|\tilde{\mathbf{T}} - \mathbf{M}\right\|^2_{\text{Frobenius}}/t(\mathbf{M}),$$

where $t(\mathbf{M})$ is the number of nonzero elements in $\mathbf{M}$.

Algorithm 2 GNNRank: Proposed Ranking Framework

Input: Digraph adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, node feature matrix $\mathbf{X}$, variant name var, (optional) initial guess from baseline $\mathbf{r}'$.
Parameter: Learnable vector $\mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}, \sigma \in \mathbb{R}$, GNN parameters, parameters from Algo. 1.
Output: Score vector $\mathbf{r}$.

$\mathbf{Z} \in \mathbb{R}^{n\times d} = \text{GNN}(\mathbf{A}, \mathbf{X})$;
if $var \in$ {"dist", "proximal dist"} **then**
  Compute $\mathbf{r} : r_i = \exp(-|\mathbf{a} - \mathbf{z}_i|_2^2/(\sigma^2 d))$;
else if $var \in$ {"innerproduct", "proximal innerproduct"} **then**
  Compute $\mathbf{r} : r_i = \text{sigmoid}(z_i \cdot \mathbf{a} + b)$;
end if
if $var \in$ {"proximal dist", "proximal innerproduct", "proximal baseline"} **then**
  Compute $\mathbf{S} : \mathbf{S}_{i,j} = \exp(-|\mathbf{z}_j - \mathbf{z}_i|_2^2/(\sigma^2 d))$;
  Compute Laplacian $\mathbf{L}$ and $\mathbf{Q}$ by Sec. 3.5;
  if $var ==$ {"proximal baseline"} **then**
    $\mathbf{r} = \mathbf{r}'$;
  end if
  $\mathbf{r} \leftarrow$ Proximal Gradient Steps$(\mathbf{r}, \mathbf{L}, \mathbf{Q})$ from Algo. 1;
end if
return $\mathbf{r}$;

Main Results

For some dense synthetic digraphs, SerialRank (which motivated our proximal gradient steps) attains leading performance, while for some other cases it fails.

We observe across all data sets that our proximal methods:

- usually outperform non-proximal variants;
- can improve on existing baseline methods when using them as initial guesses, and never perform significantly worse than the corresponding baseline, hence they can be used to enhance existing methods;
- do not rely on baseline methods for an initial guess but can instead use non-proximal outcomes;
- can outperform SerialRank by unfolding its Fiedler vector calculations with a trainable similarity matrix and proximal gradient steps.

References

[1] Fogel, F., d'Aspremont, A., & Vojnovic, M. (2014). Serialrank: Spectral ranking using seriation. *Advances in Neural Information Processing Systems*, *27*.

[2] He, Y., Reinert, G., & Cucuringu, M. (2021). DIGRAC: Digraph Clustering Based on Flow Imbalance. *arXiv preprint arXiv:2106.05194*.

[3] Tong, Z., Liang, Y., Sun, C., Li, X., Rosenblum, D., & Lim, A. (2020). Digraph inception convolutional networks. *Advances in neural information processing systems*, *33*, 17907-17918.