Network Prediction with Degree Distributional Metric Learning

Bert Huang, Blake Shaw, and Tony Jebara
bert@cs.columbia.edu, blake@cs.columbia.edu, jebara@cs.columbia.edu
Computer Science Department, Columbia University

Overview

- Given information about n nodes, what is a likely network structure?
- An ideal solution should be aware of structure, not just $n \times n$ independent estimates.
- One critical structural measure in networks is the degree distribution, which has played an important role in many network analyses [1].
- We propose modeling structure using a *degree distributional metric*:
  - A similarity function for pairs of nodes,
  - A non-stationary degree preference function dependent on the attributes of each node:
  - E.g., in the LinkedIn network, an individual whose job area is "Software Sales" is likely to have more connections than an individual whose area is "Software Programmer".
- We learn the parameters algorithmically for these functions from training data.

Degree Distributional Metric Learning

- The similarity function $f(\mathbf{x}_i, \mathbf{x}_j; \mathbf{M}) = \mathbf{x}_i^\top \mathbf{M} \mathbf{x}_j$ takes two nodes $\mathbf{x}_i$, and $\mathbf{x}_j$ and outputs a score, parameterized by matrix $\mathbf{M}$.
- The degree preference function $g(\mathbf{x}_i^k, b; \mathbf{S}) = \sum_{c=1}^{b} \mathbf{s}_c \mathbf{x}_i^k$ takes a node $\mathbf{x}_i$ and a degree b and outputs a score, parameterized by matrix $\mathbf{S}$.
- Together, the score function for any directed graph encoded with adjacency matrix $\mathbf{A}$ for data matrix $\mathbf{X}$ is

$$F(\mathbf{A}|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) = \sum_{ij|A_{ij}=1} f(\mathbf{x}_i^k, \mathbf{x}_j^k; \mathbf{M}) +$$

$$\sum_i g\left(\mathbf{x}_i^k, \sum_j A_{ij}^k; \mathbf{S}\right) + \sum_j g\left(\mathbf{x}_j^k, \sum_i A_{ij}^k; \mathbf{T}\right)$$

- Goal: learn parameters $\mathbf{M}$, $\mathbf{S}$, $\mathbf{T}$ that are consistent with a training data.
- Regularize with L_2 Frobenius norm, and try to find parameters such that the true graphs score at least $\Delta(\mathbf{A}^k, \tilde{\mathbf{A}}) = \sum_{ij|A_{ij}^k \neq \tilde{A}_{ij}} 1/(n_k^2 - n_k)$ better than all possible false graphs.
- The optimization is a form of a *structural support vector machine* (SVM), which is proven to be efficiently solvable with a *cutting-plane method* [4]:
  - Iteratively add the worst violated constraint (the highest-scoring false graph), $\tilde{\mathbf{A}}^k = \operatorname{argmax}_{\mathbf{A}} F(\mathbf{A}|\mathbf{X}, \mathbf{M}, \mathbf{S}, \mathbf{T}) + \Delta(\mathbf{A}^k, \mathbf{A})$
  - Computed using procedure in [3].
  - Re-optimize with newly added constraint.

Algorithm 1 Degree Distributional Metric Learning.

input $\{(\mathbf{X}^1, \mathbf{A}^1), \ldots, (\mathbf{X}^N, \mathbf{A}^N)\}, C$

Initialize $\mathbf{M}, \mathbf{S}, \mathbf{T}$ {e.g., $\mathbf{M} \leftarrow \mathbf{I}$ and $\mathbf{S}, \mathbf{T} \leftarrow [0]$}

Constraint set $\mathcal{C} \leftarrow \emptyset$ {or optionally add concavity constraints)}

repeat

$\quad (\mathbf{M}, \mathbf{S}, \mathbf{T}, \xi) \leftarrow \operatorname{argmin}_{\mathbf{M}, \mathbf{S}, \mathbf{T}, \xi \geq 0} \frac{1}{2} (||\mathbf{M}|| + ||\mathbf{S}|| + ||\mathbf{T}||) + C\xi$ s.t. $\mathcal{C}$

$\quad$ (Optional) Project $\mathbf{M}$ onto PSD cone

$\quad$ **for** $k = 1$ to N **do**

$\quad\quad \tilde{\mathbf{A}}^k \leftarrow \operatorname{argmax}_{\mathbf{A}} F(\mathbf{A}|\mathbf{X}, \mathbf{M}, \mathbf{S}, \mathbf{T}) + \Delta(\mathbf{A}^k, \mathbf{A})$

$\quad$ **end for**

$\quad \mathcal{C} \leftarrow \mathcal{C} \cup \frac{1}{N} \sum_{k=1}^{N} \left[F(\mathbf{A}^k|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) - F(\tilde{\mathbf{A}}^k|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) \right] \geq \frac{1}{N} \sum_{k=1}^{N} \Delta(\mathbf{A}^k, \tilde{\mathbf{A}}) - \xi$

until $\frac{1}{N} \sum_{k=1}^{N} \Delta(\mathbf{A}^k, \tilde{\mathbf{A}}) - \frac{1}{N} \sum_{k=1}^{N} \left[F(\mathbf{A}^k|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) - F(\tilde{\mathbf{A}}^k|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) \right] < \epsilon + \xi$

	DDML	M-learning	SVM
Diagonal **M**			
Hamming dist.	**0.002**	**0.001**	0.025
F_1-score	**0.990**	**0.993**	0.888
Full **M**			
Hamming dist.	0.009	**0.002**	0.169
F_1-score	0.966	**0.991**	0.467
Full **M**, **S**, **T**			
Hamming dist.	**0.020**	0.191	0.424
F_1-score	**0.970**	0.698	0.566
Averages			
Hamming dist.	**0.0103**	0.0648	0.2060
F_1-score	**0.9755**	0.8939	0.6405

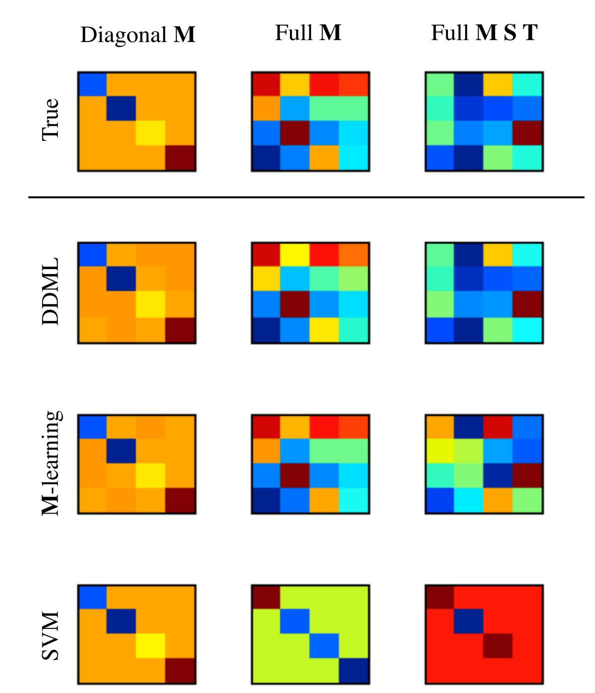

Figure 1. Left: reconstruction performance metrics on synthetic graph experiments. Right: True **M** matrices and learned **M** matrices .

Experiments

- We compare against two weaker algorithms: vanilla SVM and **M**-*learning*:
  - SVM receives the element-wise product of node-node pairs and classifies these according to whether they share and edge or not:

$$\{[\mathbf{x}_i^k(1)\mathbf{x}_j^k(1), \ldots, \mathbf{x}_i^k(D)\mathbf{x}_j^k(D)], (\mathbf{A}^k)_{ij}\},$$

  - **M**-learning only estimates the similarity parameter **M**, not leveraging degree information.

Synthetic Graphs

- Using randomly sampled node data, we generate graphs under models with increasing complexity:
  - Graphs generated according to inner product of feature vectors (corresponds to SVM),
  - graphs generated according to full **M** matrix (corresponds to **M**-learning),
  - graphs generated according to full **M** and degree preference functions (corresponds to full DDML).
- Since DDML generalizes each of the simpler models, it is able to learn nearly perfectly the true parameters we used to generate graphs, whereas each other model fails.

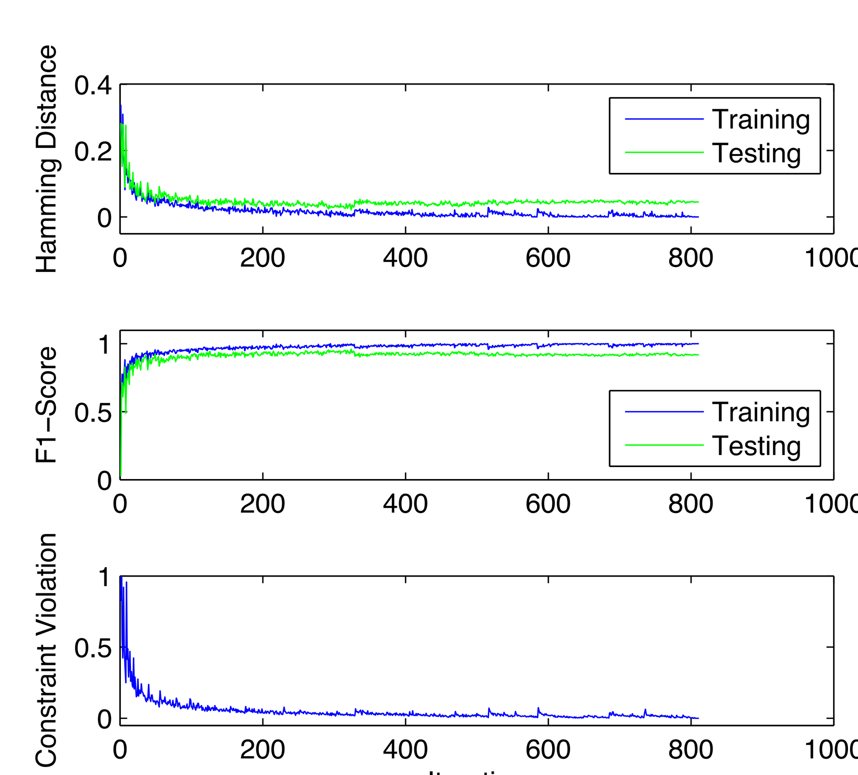

Figure 2. Convergence of DDML algorithm.

Wikipedia Categories

- We learn from a set of Wikipedia categories and their interconnections and try to predict the graphs of new categories.
- For each category, we collected the count of word-occurrences in articles listed on the main category page and directed links between the articles within each category. We squash the word counts with the square root function and reduce dimensionality to 20 by applying non-negative matrix factorization [2].
- We train the algorithms on categories "linear algebra topics", and "mathematical functions", and test on "computer science topics", "data structures", and "graph theory topics".
- DDML predictions produce an average F_1-score of 0.1255, **M**-learning scores 0.0930, and SVM scores 0.0534 and a fully-connected graph scores 0.0561.

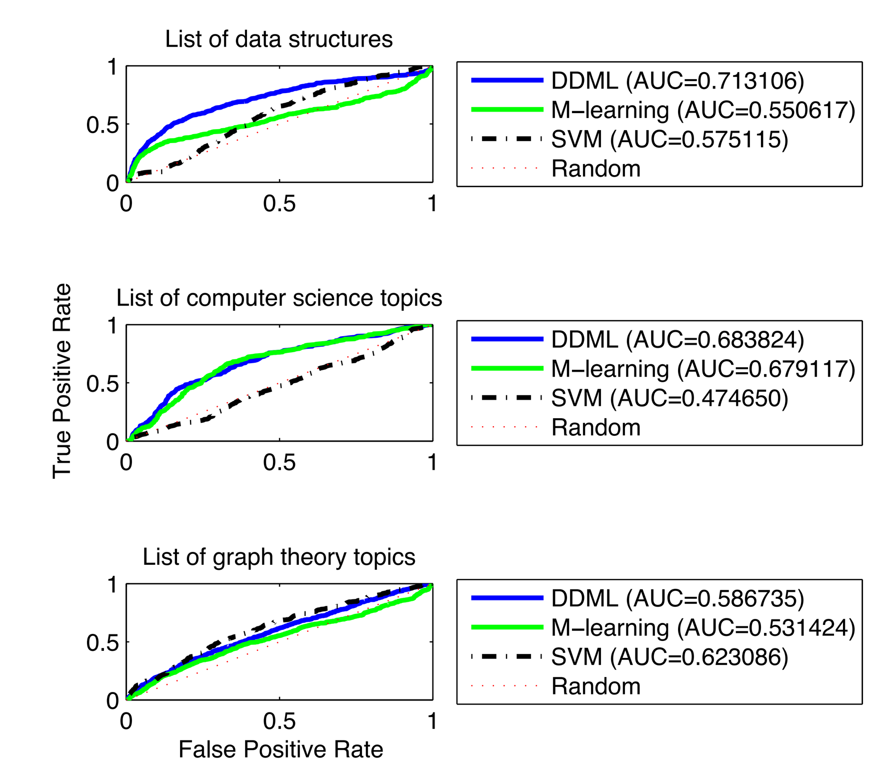

Figure 3. Receiver Order Characteristic curves of various algorithms on Wikipedia lists

Latest Updates and Future Work

- We have developed an alternate learning algorithm that uses stochastic gradient descent to optimize the objective instead of cutting-plane.
  - This variant proves theoretical guarantees that the running time to a certain solution-quality **does not depend on the size of the input**.
  - Stochastic strategy. Iterate:
    - $*$ Find a random triplet of nodes: (1) any node, (2) its neighbor, and (3) a disconnected node.
    - $*$ If swapping the edge to the true neighbor (1 to 2) with an edge to the disconnected node (1 to 3) scores higher using the current parameters, take a small gradient step in the direction of the violation.
  - Running on a Wikipedia lists, we obtain a solution in minutes of computation time on a consumer computer.

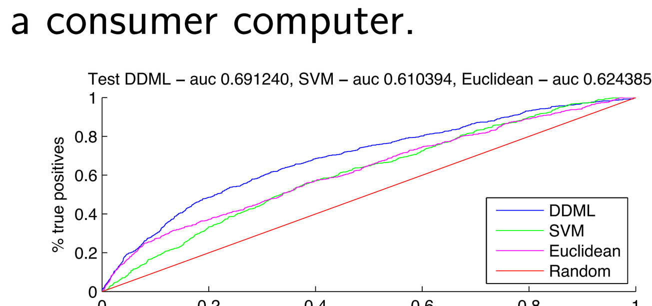

Figure 4. ROC results using stochastic DDML.

- We plan to run stochastic DDML on massive-scale networks (e.g., the full Wikipedia set), exploiting the independence of running time on network size, and learn models of node distance and degree likelihood for graphs.

References

[1] A. Barabási. Linked: The new science of networks. *J. Artificial Societies and Social Simulation*, 6(2), 2003.

[2] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155 – 173, 2007.

[3] B. Huang and T. Jebara. Exact graph structure estimation with degree priors. In *ICMLA*, pages 111–118, 2009.

[4] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural svms. *Mach. Learning*, 77(1):27–59, 2009.