Network Prediction with Degree Distributional Metric Learning

Bert Huang, Blake Shaw, and Tony Jebara
Department of Computer Science, Columbia University, New York, NY 10027
{bert, blake, jebara}@cs.columbia.edu

Introduction. Real-world networks often consist of nodes with informative attributes as well as links. To properly model these networks, it is necessary to learn how attributes of the nodes relate to the connectivity structure. Metric learning is a natural framework for transforming the raw node features to match the structural properties of a graph. Traditional metric learning algorithms primarily model the similarity between nodes and not structural properties, such as degree distributions. Degree distributions play a central role in graph structure analysis [1]. The degree distribution for some nodes may be non-stationary and depend on their attributes, particularly if some attributes naturally relate to connectedness. For example, 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 propose *degree distributional metric learning* (DDML), a method for simultaneously learning a metric and degree preference functions such that the combination captures the structure of the input graph and allows for more accurate link prediction from only node features.

Algorithm Description. The learning algorithm is given training data consisting of N pairs of node feature vectors and corresponding adjacency matrices $\{(\mathbf{X}^1, \mathbf{A}^1), \ldots, (\mathbf{X}^N, \mathbf{A}^N)\}$, where each row i of $\mathbf{X}^k \in \mathbb{R}^{n_k \times D}$ represents one of n_k D-dimensional real-valued node feature vectors denoted by $(\mathbf{x}_i^k)^\top$, and each $\mathbf{A}^k \in \mathbb{B}^{n_k \times n_k}$ is a directed adjacency matrix. DDML then outputs a similarity function $f : \{\mathbb{R}^D, \mathbb{R}^D\} \mapsto \mathbb{R}$ that takes two vectors as input and outputs a real value, and a degree preference function $g : \{\mathbb{R}^D, \mathbb{N}\} \mapsto \mathbb{R}$ takes a node descriptor vector and a candidate degree d and outputs a real valued preference score for that node having degree d.

Matrices $\mathbf{M} \in \mathbb{R}^{D\times D}$, and $\mathbf{T}, \mathbf{S} \in \mathbb{R}^{n \times D}$, where $n = \max_k n_k$, define a degree distributional metric. The similarity function is [1] $f(\mathbf{x}_i, \mathbf{x}_j; \mathbf{M}) = \mathbf{x}_i^\top \mathbf{M} \mathbf{x}_j$. Using the notation that $\mathbf{s}_c$ is the $1 \times D$ dimensional c'th row of $\mathbf{S}$, the degree preference function is $g(\mathbf{x}_i^k, b; \mathbf{S}) = \sum_{c=1}^{n_k - b} \mathbf{s}_c \mathbf{x}_i^k$. A graph is predicted by maximizing $F(\mathbf{A}|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) = \sum_{ij|A_{ij}=1} f(\mathbf{x}_i^k, \mathbf{x}_j^k; \mathbf{M}) + \sum_i g\left(\mathbf{x}_i^k, \sum_j A_{ij}^k; \mathbf{S}\right) + \sum_j g\left(\mathbf{x}_j^k, \sum_i A_{ij}^k; \mathbf{T}\right)$. This optimization is computable by a reduction to a maximum weight b-matching [3]. Using normalized Hamming distance $\Delta(\mathbf{A}^k, \tilde{\mathbf{A}}) = \sum_{ij|A_{ij}^k \neq \tilde{A}_{ij}} 1/(n_k^2 - n_k)$, i.e., the proportion of misclassified edges, as a loss function and the Frobenius ℓ_2-norm of the parameter matrices as a regularizer, learning is done by solving

$$\min_{\mathbf{M},\mathbf{S},\mathbf{T},\xi\geq 0} \frac{1}{2}\left(||\mathbf{M}||_{\mathrm{Fro}} + ||\mathbf{S}||_{\mathrm{Fro}} + ||\mathbf{T}||_{\mathrm{Fro}}\right) + C\xi, \text{ s.t.}$$
$$\frac{1}{N}\sum_{k=1}^{N}\left[F(\mathbf{A}^k|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T}) - F(\tilde{\mathbf{A}}^k|\mathbf{X}^k, \mathbf{M}, \mathbf{S}, \mathbf{T})\right]$$
$$\geq \frac{1}{N}\sum_{k=1}^{N}\Delta(\mathbf{A}^k, \tilde{\mathbf{A}}) - \xi, \quad \forall\{\tilde{\mathbf{A}}^1, \ldots, \tilde{\mathbf{A}}^n\}. \quad (1)$$

The optimization is a quadratic program with exponentially many linear constraints, and is of the same form as a *structural support vector machine* (SVM) [4]. Thus, the established cutting-plane approach (and efficiency guarantees) can be applied. The solution to (1) is found by maintaining a working set of constraints, solving for the optimal $\mathbf{M}$, $\mathbf{S}$, and $\mathbf{T}$, then adding the worst-violated constraint by the current solution and repeating. The worst violated constraint is found by a *separation oracle*, $\tilde{\mathbf{A}}^k = \mathrm{argmax}_{\mathbf{A}} F(\mathbf{A}|\mathbf{X}, \mathbf{M}, \mathbf{S}, \mathbf{T}) + \Delta(\mathbf{A}^k, \mathbf{A})$. This is computed by adding the decomposed loss to the primary edge weights of the b-matching input.

Experiments. We consider comparisons against two baseline models of varying richness. The simplest model classifies node-pairs using a support vector machine (SVM). The SVM receives training data as pairs of inputs and outputs (binary labels) $\{[\mathbf{x}_i^k(1)\mathbf{x}_j^k(1), \ldots, \mathbf{x}_i^k(D)\mathbf{x}_j^k(D)], (\mathbf{A}^k)_{ij}\}$, and then estimates a weight vector $\mathbf{w}$. The second model, $\mathbf{M}$-learning, learns a linear transform matrix $\mathbf{M}$ without any degree information, predicting the presence an edge if $\mathbf{x}_i \mathbf{M} \mathbf{x}_j$ is a positive quantity. We learn $\mathbf{M}$ by the same optimization as DDML except with the $\mathbf{S}$ and $\mathbf{T}$ matrices fixed at zero. The SVM, $\mathbf{M}$-learning and DDML approaches bring increasing model richness. The SVM approach is equivalent to learning an $\mathbf{M}$ matrix that is only nonzero along the diagonal. Similarly, $\mathbf{M}$-learning is equivalent to DDML with no degree distribution information.

[1] If $\mathbf{M}$ is positive semi-definite (PSD), it can be used to describe a metric. Omitting the PSD requirement allows the similarity function to be asymmetric, which allows representation of directed graphs. Nevertheless, we always refer to the algorithm as degree distributional metric learning.

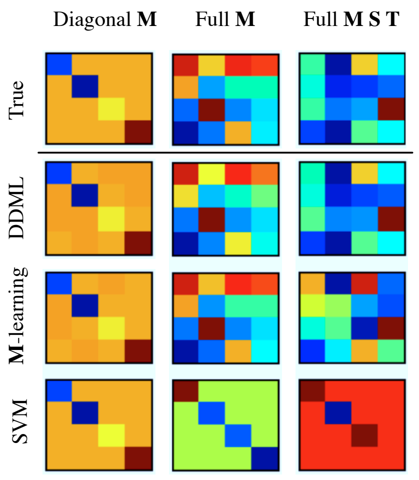

Figure 1: True and Learned **M** matrices from Synthetic Tests. For each sampling scheme, the models that do not learn the active sampling parameters inadequately model the feature interactions.

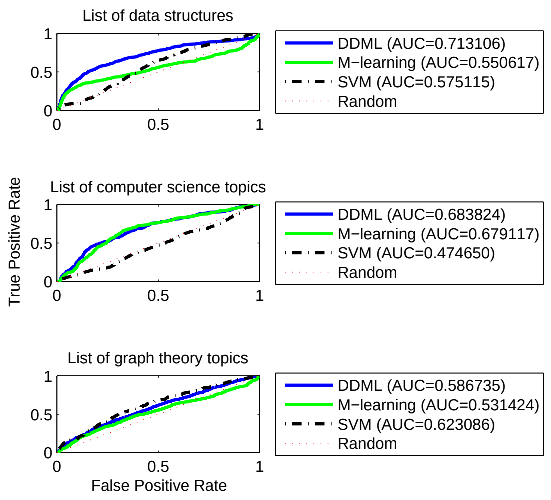

Figure 2: ROC Curves for Wikipedia Link Prediction. These plots compare the true and false positive rates of the rankings returned by the three learning algorithms for predicting the edges of held out graphs.

Synthetic Graphs. We generate data and graphs from three sampling schemes. For each scheme, we train on five graphs and test on five new graphs. To generate graphs, we randomly sample 50 data vectors from $\mathbb{R}^4$ uniformly from $[0, 1]^4$ and predict different **M**, **S** and **T** matrices. First, we use a diagonal **M** matrix and zero degree preference. Second, we use a full **M** matrix and zero degree preference. Third, we generate a random **M** matrix and random **S** and **T** matrices. All methods perform well when data is generated from their corresponding models, but the baselines fail on graphs from the richer generative processes. E.g., in the third scheme, only DDML predicts near-perfectly. See Fig. 1.

Wikipedia Lists. We conducted an experiment to predict the link structure between Wikipedia articles in predefined categories using bag-of-words features for each article. 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.

We also compared the ranking of edges obtained by the three models. Since the DDML model is richer than a simple ranking of edges, we greedily select edges in order according to the gain in current overall weight, which takes into account the edge weight itself as well as the reward for the change in degree induced by adding each edge. For **M**-learning and SVM, the ranking is the ordering of prediction values. The receiver order statistics (ROC) curves for each of the held-out test graphs are in Fig. 2.

Discussion. Metric learning is a natural framework for modeling graph data containing both connectivity information and node attributes. We have demonstrated that it is insufficient to learn only a metric that defines node similarity merely pairwise. To properly model how nodes connect, it is necessary to estimate both a metric and a set of degree preference functions which allow the model to better match the structural properties of real networks.

References

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

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

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

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