Large-margin Structured Learning for Link Ranking

Stephen H. Bach **Bert Huang** **Lise Getoor**
Department of Computer Science
University of Maryland
College Park, MD 20742
{bach, bert, getoor}@cs.umd.edu

Abstract

We introduce a new approach to learning for the common problem of link prediction. Since link prediction is a structured prediction task that is naturally viewed as a ranking problem, we propose to learn to optimize a ranking metric directly in a large-margin structured prediction framework. This approach is appealing because it directly encodes the desired behavior into the learning objective. We show how to compute subgradients of the learning objective efficiently, and we demonstrate the effectiveness of our approach on real-world networks.

1 Introduction

A fundamental problem in network analysis is *link prediction*, the task of predicting which nodes in a network link to which. Link prediction differs from many other machine-learning tasks in a number of ways. First, underlying network phenomena are intrinsically structured, e.g., the presence or absence of a link usually depends on other links in the network. Consequentially, accurate models for link prediction should also be structured. Rather than predicting each link independently, modeling unknown links jointly often improves predictive performance, e.g., [1, 2]. Second, it is large scale. In a network with n nodes, there are $O(n^2)$ possible links, so models for link prediction must be able to scale to such large prediction spaces. Third, it is best viewed as a ranking problem. In practice, most real-world networks of interest are sparse, i.e., few of the $O(n^2)$ possible links actually exist. This imbalance makes traditional 0-1 loss evaluation metrics and training objectives inappropriate, because hypotheses that predict that no links exist can achieve very high accuracy. Instead, researchers have used ranking metrics such as area under the receiver operating characteristic (ROC) or precision-recall curve to evaluate models for link prediction.

In this work, we introduce a new approach to learning for link prediction designed to address these facets. We propose to directly train structured models to optimize a ranking metric within a large-margi estimation framework [3]. While researchers have considered large-margin structured prediction with 0-1 losses, e.g., [4, 5, 6, 7], large-margin ranking of independent elements, e.g., [8, 9, 10], and large-margin collaborative filtering using matrix factorization [11], we consider general large-margin structured prediction with a ranking loss and the unique computational challenges that arise. We present preliminary results that demonstrate the promise of our approach.

2 Background

We first describe large-margin learning (Section 2.1), which we will later extend to learn joint ranking models (Section 3). We then describe hinge-loss Markov random fields (Section 2.2), which are a particular family of models well-suited to link prediction. We will train them in our evaluation (Section 4) to demonstrate the effectiveness of our approach.

2.1 Large-margin structured learning

In structured prediction tasks, such as link prediction, the goal is to predict an output $\mathbf{Y}$ from a space of structures $\mathcal{Y}$ given input $\mathbf{X}$. A common approach to learning for structured prediction is large-margin learning, which seeks a function $f_\lambda(\mathbf{Y}, \mathbf{X})$ that discriminates the training data by a large margin from all other $\mathbf{Y} \in \mathcal{Y}$ given $\mathbf{X}$. We follow the common assumption that $f_\lambda(\mathbf{Y}, \mathbf{X})$ takes the form $\lambda^\top \phi(\mathbf{Y}, \mathbf{X})$ where λ is a vector of weights to be learned, and ϕ is a vector of features of the input and candidate outputs.

Since we will make predictions using $\arg\max_{\mathbf{Y}} f_\lambda(\mathbf{Y}, \mathbf{X})$, the ground-truth state should have a higher score than any alternate state by a large margin. Specifically, we seek weights λ such that, for any valid output state $\tilde{\mathbf{Y}}$,

$$f_\lambda(\mathbf{Y}, \mathbf{X}) \geq f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X}) + L(\mathbf{Y}, \tilde{\mathbf{Y}}), \;\; \forall \tilde{\mathbf{Y}} \in \mathcal{Y} \, .$$

where $L(\mathbf{Y}, \tilde{\mathbf{Y}})$ measures the disagreement between a state $\tilde{\mathbf{Y}}$ and the training data $\mathbf{Y}$. Since we do not expect all problems to be perfectly separable, we relax this constraint by penalizing its violation. This leads to a convex learning objective for a large-margin solution

$$\min_{\lambda} \;\; \frac{1}{2}\|\lambda\|^2 + C \max_{\tilde{\mathbf{Y}} \in \mathcal{Y}} \Big(f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X}) - f_\lambda(\mathbf{Y}, \mathbf{X}) + L(\mathbf{Y}, \tilde{\mathbf{Y}}) \Big) \, . \tag{1}$$

Such structured learning objectives can be solved in a few different ways. We use an online subgradient method [3], which iteratively updates the parameters by taking steps in a subgradient direction. The subgradient at current parameter λ is

$$\nabla_\lambda = \lambda + C \left(\phi(\mathbf{Y}^\star, \mathbf{X}) - \phi(\mathbf{Y}, \mathbf{X}) \right)$$

where

$$\mathbf{Y}^\star = \arg\max_{\tilde{\mathbf{Y}}} f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X}) + L(\mathbf{Y}, \tilde{\mathbf{Y}}) \, . \tag{2}$$

Given a subgradient, we attempt to improve the objective (1) by updating according to

$$\lambda \leftarrow \lambda - \eta_t \nabla_\lambda,$$

where η_t is a decreasing step size schedule such as $\eta_t = \frac{1}{t}$ for the t'th iteration.

The computational challenge of large-margin learning therefore comes down to the subgradient computation, which takes a similar form to inference itself and is often referred to as *loss-augmented inference* [12]. The difficulty of solving objective (2) depends on both the scoring function $f_\lambda(\mathbf{Y}, \mathbf{X})$ and the loss function $L(\mathbf{Y}, \tilde{\mathbf{Y}})$. The scoring function should be expressive, yet tractable, and the loss function should accurately characterize more and less desirable predictions. In Section 2.2, we describe one such scoring function well-suited to link prediction, before we show how to learn with a ranking loss for the loss function in Section 3.

2.2 Hinge-loss Markov random fields

In this section, we give an overview of hinge-loss Markov random fields (HL-MRFs) [7]. HL-MRFs are log-linear probabilistic graphical models parameterized by constrained hinge-loss energy functions. The energy function is factored into hinge-loss potentials, which are functions of the continuous variables. For completeness, a formal definition of HL-MRFs is as follows [7].

Definition 1. *Let $\mathbf{Y} = (Y_1, \ldots, Y_n)$ be a vector of n variables and $\mathbf{X} = (X_1, \ldots, X_{n'})$ a vector of n' variables with joint domain $\mathbf{D} = [0,1]^{n+n'}$. Let $\phi = (\phi_1, \ldots, \phi_m)$ be m continuous potentials of the form*

$$\phi_j(\mathbf{Y}, \mathbf{X}) = [\max\{\ell_j(\mathbf{Y}, \mathbf{X}), 0\}]^{p_j},$$

where ℓ_j is a linear function of $\mathbf{Y}$ and $\mathbf{X}$ and $p_j \in \{1, 2\}$. Let $C = (C_1, \ldots, C_r)$ be linear constraint functions associated with index sets denoting equality constraints $\mathcal{E}$ and inequality constraints $\mathcal{I}$, which define the feasible set

$$\tilde{\mathbf{D}} = \left\{ \;\; \mathbf{Y}, \mathbf{X} \in \mathbf{D} \;\; \middle| \;\; C_k(\mathbf{Y}, \mathbf{X}) = 0, \forall k \in \mathcal{E}, \quad C_k(\mathbf{Y}, \mathbf{X}) \geq 0, \forall k \in \mathcal{I} \;\; \right\} .$$

For $\mathbf{Y}, \mathbf{X} \in \tilde{\mathbf{D}}$, given a vector of nonnegative free parameters, i.e., weights, $\lambda = (\lambda_1, \ldots, \lambda_m)$, a constrained hinge-loss energy function f_λ is defined as

$$f_\lambda(\mathbf{Y}, \mathbf{X}) = -\sum_{j=1}^{m} \lambda_j \phi_j(\mathbf{Y}, \mathbf{X}) \, .$$

Definition 2. *A hinge-loss Markov random field P over random variables $\mathbf{Y}$ and conditioned on random variables $\mathbf{X}$ is a probability density defined as follows: if $\mathbf{Y}, \mathbf{X} \notin \tilde{\mathbf{D}}$, then $P(\mathbf{Y}|\mathbf{X}) = 0$; if $\mathbf{Y}, \mathbf{X} \in \tilde{\mathbf{D}}$, then*

$$P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(\lambda)} \exp\left[f_\lambda(\mathbf{Y}, \mathbf{X})\right] \;, \textit{ where } \; Z(\lambda) = \int_{\mathbf{Y}} \exp\left[f_\lambda(\mathbf{Y}, \mathbf{X})\right] .$$

Inference of the *most probable explanation* (MPE) in HL-MRFs is a convex optimization, since the hinge-loss potentials are each convex and the linear constraints form a convex feasible region. The fastest known method for HL-MRF inference uses the alternating direction method of multipliers (ADMM) [13], which decomposes the full objective into subproblems each with their own copy of the variables and uses augmented Lagrangian relaxation to enforce consensus between the independently optimized subproblems [7].

3 Learning to rank links

To train a function $f_\lambda(\mathbf{Y}, \mathbf{X})$ to predict links in a network, we propose to use a ranking loss directly during large-margin learning as the loss function $L(\mathbf{Y}, \tilde{\mathbf{Y}})$. Previously in large-margin learning for structured prediction, research has focused on loss functions that decompose over individual components of the structure, such as the Hamming and L1 losses, e.g., [4, 5, 6, 7]. However, since link predictions are best evaluated as rankings, we propose to learn to optimize a ranking loss instead. A standard ranking loss is the area under the receiver operating characteristic curve (ROC), which is defined in terms of the number of swapped pairs in a ranking. Let $\mathcal{P}$ be the set of links that are labeled as existing (positive links), and let $\mathcal{N}$ be the set of links that are labeled as not existing (negative links). Then,

$$\text{ROC}(\mathbf{Y}, \tilde{\mathbf{Y}}) \equiv 1 - \frac{\left|\left\{(i,j) \;|\; Y_i > Y_j \bigwedge \tilde{Y}_j > \tilde{Y}_i\right\}\right|}{|\mathcal{P}||\mathcal{N}|} \;,$$

which can be changed to a loss function

$$L^{\text{ROC}}(\mathbf{Y}, \tilde{\mathbf{Y}}) \equiv \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{(i,j)|Y_i > Y_j} \mathbb{I}\left[\tilde{Y}_j > \tilde{Y}_i\right] \;.$$

$L^{\text{ROC}}(\mathbf{Y}, \tilde{\mathbf{Y}})$ is a combinatorial function that makes optimizing objective (2) intractable. However, we define a more tractable convex surrogate, *pseudo ROC*,

$$L^{\text{pROC}}(\mathbf{Y}, \tilde{\mathbf{Y}}) \equiv \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{(i,j)|Y_i > Y_j} \max\left\{0, \tilde{Y}_j - \tilde{Y}_i\right\} \;.$$

We can now write the loss-augmented inference objective (2) specifically for this ranking loss:

$$\arg\max_{\tilde{\mathbf{Y}}} f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X}) + \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{(i,j)|Y_i > Y_j} \max\left\{0, \tilde{Y}_j - \tilde{Y}_i\right\} \;. \tag{3}$$

As we explained in Section 2.1, the computational challenge of large-margin learning comes down to optimizing this objective. Using this convex relaxation of the ROC loss augments a maximization with a convex function, so in settings where inference is a maximization of a convex or linear function, this loss-augmented inference is no longer concave. To address this non-concavity, we propose a procedure that is applicable to any concave energy function $f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X})$. Useful examples include hinge-loss Markov random fields, linear and quadratic programming relaxations of Markov random fields, and convex variational approximations to marginal inference. Further, there are $|\mathcal{P}||\mathcal{N}|$ terms

Algorithm 1 Gradient Computation for L^{pROC}

```
Input: model $f_\lambda$, input $\mathbf{X}$, training output $\mathbf{Y} = \mathcal{P} + \mathcal{N}$, initial guess $\tilde{\mathbf{Y}}$
Output: $\mathbf{Y}^\star = \arg\max_{\tilde{\mathbf{Y}}} f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X}) + L^{\text{pROC}}(\mathbf{Y}, \tilde{\mathbf{Y}})$
while not converged do
  $\mathbf{c} \leftarrow \mathbf{0}$
  $\#Pos, \#Neg \leftarrow 0$
  $\tilde{\mathbf{Y}} \leftarrow \texttt{PessimisticSort}_{\mathbf{Y}}(\tilde{\mathbf{Y}})$
  for $\tilde{Y}_i \in (\tilde{Y}_1, \ldots, \tilde{Y}_n)$ do
    if $Y_i \in \mathcal{P}$ then
      $c_i \leftarrow -1 \cdot \#Neg$
      $\#Pos \leftarrow \#Pos + 1$
    end if
    if $Y_i \in \mathcal{N}$ then
      $c_i \leftarrow |\mathcal{P}| - \#Pos$
      $\#Neg \leftarrow \#Neg + 1$
    end if
  end for
  $\tilde{\mathbf{Y}} \leftarrow \arg\max_{\tilde{\mathbf{Y}}} f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X}) + \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{i=1}^{n} c_i \tilde{Y}_i$
end while
$\mathbf{Y}^\star \leftarrow \tilde{\mathbf{Y}}$
```

in $L^{\text{pROC}}(\mathbf{Y}, \tilde{\mathbf{Y}})$. Since link-prediction tasks are large scale, explicitly enumerating all of these terms can be impractical. We have derived a procedure to efficiently compute the loss without having to enumerate all pairs, collapsing the loss function to $|\mathcal{P}| + |\mathcal{N}|$ terms during optimization.

Algorithm 1 describes our procedure for optimizing objective (3). At each iteration we optimize the energy function $f_\lambda(\tilde{\mathbf{Y}}, \mathbf{X})$ plus a linear approximation to the loss function $L^{\text{pROC}}(\mathbf{Y}, \tilde{\mathbf{Y}})$. Since the objective is a difference of convex functions, and our linear approximation is a subgradient of the loss function, we use the difference of convex functions (DCA) algorithm, which is guaranteed to converge to a local optimum of the objective [14]. Since we use a linear approximation to the piecewise-linear loss function, we are able to collapse its $|\mathcal{P}||\mathcal{N}|$ terms to $|\mathcal{P}| + |\mathcal{N}|$ terms, one for each of the unknown links in the network. The coefficient c_i for link $\tilde{Y}_i$ is the number of terms in $L^{\text{pROC}}(\mathbf{Y}, \tilde{\mathbf{Y}})$ in which $\tilde{Y}_i$ participates that have a non-zero subgradient at the current point $\tilde{\mathbf{Y}}$, and it is negative in sign if $Y_i \in \mathcal{P}$, i.e., Y_i is a positive link, and it is positive in sign if $Y_i \in \mathcal{N}$, i.e., Y_i is a negative link.

Algorithm 1 updates the vector $\mathbf{c}$ of coefficients without iterating over all $|\mathcal{P}||\mathcal{N}|$ terms by sorting the elements (links) of the current point $\tilde{\mathbf{Y}}$ in descending order, counting how many positive and negative links are ahead of each link, and comparing that with how many there are at each point in the true ranking. The operation $\texttt{PessimisticSort}_{\mathbf{Y}}(\tilde{\mathbf{Y}})$ sorts elements of $\tilde{\mathbf{Y}}$ in descending order and then breaks ties in *ascending* order by the corresponding elements of $\mathbf{Y}$. In this way, links that are tied at the current point $\tilde{\mathbf{Y}}$ are sorted such that they are in the incorrect order relative to the true ranking. This means that, if $Y_i > Y_j$ and $\tilde{Y}_i = \tilde{Y}_j$, then $\tilde{Y}_j$ will be placed before $\tilde{Y}_i$ in the sorted list. Consequently, the loss term $\max\left\{0, \tilde{Y}_j - \tilde{Y}_i\right\}$ will be included in the linear approximation, allowing the algorithm to consider the benefit of points where $\tilde{Y}_i < \tilde{Y}_j$.

The algorithm has converged when no elements of $\mathbf{c}$ are changed. At that point we compute the updated gradient ∇_λ at the final point $\tilde{\mathbf{Y}}$ and update the weights λ.

4 Evaluation

We test our approach on link prediction using real Web data, jointly predicting links between web pages given only the page content. We collected web sites for the featured articles from Wikipedia[1]

[1]http://en.wikipedia.org/wiki/Wikipedia:Featured_articles

Table 1: Average area under ROC and precision-recall curves and 0-1 accuracy with different loss functions during large-margin learning. Standard deviations are listed in parentheses. Scores statistically equivalent to the best scoring method by metric are typed in bold. We label the perceptron learner as "None" in the loss type column.

	ROC	P-R	Acc.
pROC	**0.869 (0.046)**	**0.601 (0.106)**	0.804 (0.112)
L1	0.843 (0.047)	0.556 (0.111)	**0.852 (0.086)**
None	0.842 (0.049)	0.524 (0.105)	0.667 (0.145)

as listed in February 2013, which were categorized by Wikipedia into thirty distinct categories. We encode each article's content using a bag-of-words vector. We also use a thresholded cosine similarity between the TF-IDF representations of pages to prune candidates for collective inference. The goal is to learn a model that can accurately and collectively rank potential links between a set of new articles.

We construct hinge-loss Markov random fields over link-existence variables using *probabilistic soft logic* (PSL) [15, 16] to template our models. We define PSL predicates for the target link existence (Link), labeled Wikipedia category (HasCat), and thresholded TF-IDF similarity (Candidate), which is true for pairs with greater than 0.4 similarity. Finally, we include a predicate (HasWord) for word occurrence from dictionary set $\mathcal{W}$. The model is then constructed using the following rules:

$$\begin{aligned}
w^{\text{same}} &: \textsc{HasCat}(A, C) \wedge \textsc{HasCat}(B, C) \rightarrow \textsc{Link}(A, B), \\
w^{\text{anti-same}} &: \textsc{HasCat}(A, C) \wedge \textsc{HasCat}(B, C) \rightarrow \neg\textsc{Link}(A, B), \\
w^{\text{diff}} &: \textsc{HasCat}(A, C_1) \wedge \textsc{HasCat}(B, C_2) \wedge (C_1 \neq C_2) \rightarrow \textsc{Link}(A, B), \\
w^{\text{anti-diff}} &: \textsc{HasCat}(A, C_1) \wedge \textsc{HasCat}(B, C_2) \wedge (C_1 \neq C_2) \rightarrow \neg\textsc{Link}(A, B), \\
w^{\text{triad}} &: \textsc{Link}(A, B) \wedge \textsc{Link}(B, C) \wedge \textsc{Candidate}(A, C) \rightarrow \textsc{Link}(A, C), \\
w_i^{\text{word}} &: \textsc{HasWord}(A, W_i) \wedge \textsc{HasWord}(B, W_i) \rightarrow \textsc{Link}(A, B), \forall W_i \in \mathcal{W}, \\
w^{\text{prior}} &: \neg\textsc{Link}(A, B).
\end{aligned}$$

The various weighted rules in this model enable a learning algorithm to adjust how much weight to place on links being implied by category-to-category linking tendencies (w^{same}, $w^{\text{anti-same}}$, w^{diff}, $w^{\text{anti-diff}}$), transitivity (w^{triad}), shared-word-occurrence linking tendencies (w^{word}), and overall prior sparsity (w^{prior}).

For each trial, we randomly sample two small, distinct networks each of 50 articles by *snowball sampling*, i.e., a randomized breadth-first search with random jumps. We train a model on one network and measure ranking accuracy on the other. We compare our ranking learning objective against existing strategies for learning structured predictors: L1-loss large margin learning and a voted-perceptron approximate maximum likelihood.

Over 30 folds, we measure the area under the ROC and precision-recall curves. The results indicate that using pROC loss produces better rankings according to both metrics. We also measure the 0-1 accuracy when predictions are rounded (i.e., thresholded at 0.5). Recall that the sparsity of real networks tends to make very high 0-1 accuracies trivial to achieve because predicting that no links exist is often better than any informative estimate. Consistent with its loss function, large-margin estimation using L1 loss scores higher accuracy at the cost of ranking quality.

5 Conclusion

In this work, we introduce a new approach to learning for link prediction. We propose learning to optimize a ranking loss during large-margin learning and introduce a novel algorithm for computing subgradients of the resulting objective. Our preliminary experimental results are encouraging, showing that our approach can outperform other learning methods for link prediction.

There are many interesting directions for future work. We aim to further analyze our algorithm to understand how well the DCA algorithm solves the subgradient calculation. Further, our method is applicable to other models beyond hinge-loss Markov random fields, and it can be applied to other structured ranking problems beyond link predicition.

Acknowledgments

This work was supported by NSF grant CCF0937094 and IARPA via DoI/NBC contract number D12PC00337. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

[1] Richardson, M., P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006.

[2] Getoor, L., B. Taskar, eds. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[3] Ratliff, N. D., J. A. Bagnell, M. A. Zinkevich. (online) subgradient methods for structured prediction. In *AISTATS 2007*. 2007.

[4] Taskar, B., C. Guestrin, D. Koller. Max-margin Markov networks. In *Neural Information Processing Systems*. 2004.

[5] Joachims, T., T. Finley, C. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[6] Huynh, T., R. Mooney. Online max-margin weight learning for Markov logic networks. In *SIAM International Conference on Data Mining*. 2011.

[7] Bach, S., B. Huang, B. London, et al. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Uncertainty in Artificial Intelligence*. 2013.

[8] Joachims, T. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*. 2005.

[9] Boyd, S., C. Cortes, M. Mohri, et al. Accuracy at the Top. In *Advances in Neural Information Processing Systems (NIPS)*, pages 962–970. 2012.

[10] Le, Q., A. Smola. Direct Optimization of Ranking Measures, 2007.

[11] Weimer, M., A. Karatzoglou, Q. V. Le, et al. COFI rank-maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1593–1600. 2007.

[12] Taskar, B., V. Chatalbashev, D. Koller, et al. Learning structured prediction models: a large margin approach. In *International Conference on Machine learning (ICML)*. 2005.

[13] Boyd, S., N. Parikh, E. Chu, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1), 2011.

[14] An, L., P. Tao. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

[15] Broecheler, M., L. Mihalkova, L. Getoor. Probabilistic similarity logic. In *Uncertainty in Artificial Intelligence*. 2010.

[16] Kimmig, A., S. H. Bach, M. Broecheler, et al. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*. 2012.