AN EFFICIENT ALGORITHM TO INTEGRATE NETWORK AND ATTRIBUTE DATA FOR GENE FUNCTION PREDICTION

SHANKAR VEMBU

Donnelly Center for Cellular and Biomolecular Research,
University of Toronto, Toronto, ON, Canada
E-mail: shankar.vembu@utoronto.ca

QUAID MORRIS

Donnelly Center for Cellular and Biomolecular Research,
Banting and Best Department of Medical Research,
Department of Molecular Genetics,
Edward S. Rogers Sr. Department of Electrical and Computer Engineering,
Department of Computer Science,
University of Toronto, Toronto, ON, Canada
E-mail: quaid.morris@utoronto.ca

Label propagation methods are extremely well-suited for a variety of biomedical prediction tasks based on network data. However, these algorithms cannot be used to integrate feature-based data sources with networks. We propose an efficient learning algorithm to integrate these two types of heterogeneous data sources to perform binary prediction tasks on node features (e.g., gene prioritization, disease gene prediction). Our method, *LMGraph*, consists of two steps. In the first step, we extract a small set of "network features" from the nodes of networks that represent connectivity with labeled nodes in the prediction tasks. In the second step, we apply a simple weighting scheme in conjunction with linear classifiers to combine these network features with other feature data. This two-step procedure allows us to (i) learn highly scalable and computationally efficient linear classifiers, (ii) and seamlessly combine feature-based data sources with networks. Our method is much faster than label propagation which is already known to be computationally efficient on large-scale prediction problems. Experiments on multiple functional interaction networks from three species (mouse, fly, *C.elegans*) with tens of thousands of nodes and hundreds of binary prediction tasks demonstrate the efficacy of our method.

Keywords: gene function prediction; graph-based learning; label propagation; ensemble learning.

1. Introduction

Network-based prediction algorithms are widely used in biomedical prediction tasks.[1–3] These prediction tasks often share a number of properties – a small number of labeled nodes (e.g., genes or patients), a large number of unlabeled nodes, and sparse connectivity among the nodes – that make label propagation algorithms particularly well-suited to the domain. In particular, algorithms proposed by Zhu *et al.*[4] and Zhou *et al.*[5] have only a single free parameter and permit very efficient implementations, and can therefore be applied to very large prediction problems with very little labeled data. Despite their simplicity, these algorithms perform surprisingly well on prediction benchmarks, see for example, Refs. 6 and 7.

However, "feature-based" data are often available for individual nodes in the networks – for example, gene features could include the presence of particular protein domains, sequence conservation levels, associations with disease, phenotypes associated with its deletion mutants.

Representing this feature data by a network-based similarity measure requires grouping features and measuring similarity among feature profiles. This approach loses information about individual feature values, as well as generating a dense similarity network that slows down label propagation algorithms. In this paper, we describe a new algorithm related to label propagation which retains many of its advantages while also allowing heterogeneous feature and network data to be integrated into a common framework.

Although the algorithm we describe can be applied to any domain, for concreteness and because of the existence of comprehensive benchmark data, we consider the problem of predicting gene function from heterogeneous genomic and proteomic data sources.[8–11] Here, one is given a set of genes (query) with a given annotation, and asked to find genes similar to the query. The classic example of this type of problem is predicting Gene Ontology (GO) annotations but could also involve predicting disease associated genes. Functional interaction networks are a widely used representation to capture information about shared gene function present in genomic and proteomic data sources.[8,11] A popular approach to solving this problem is to combine these networks into a composite network[6,12] and, along with a set of labels that describe the gene function, use them as inputs to a graph-based learning algorithm such as label propagation.[4,5] The main advantage of these methods is that they are computationally efficient. Both label propagation and the method of Tsuda *et al.*[12] admit a solution of the form $P^{-1}q$, where P is a sparse matrix, and can be computed by solving a sparse linear system whose time complexity is almost linear in the number of non-zero entries in P.[13]

Despite being computationally efficient, the algorithms proposed by Tsuda *et al.*[12] and Mostafavi *et al.*[6] cannot be used to integrate feature-based data sources (attributes) with networks. A natural solution to this problem is to construct a similarity graph[a] (preferably sparse) from the feature-based data. This can be done by first computing a kernel matrix from the features, for example, using the dot-product kernel or the radial basis function (RBF) kernel, and then by using an appropriate method to sparsify the dense kernel matrix. However, as mentioned above, the main drawback of this approach is the potential loss of information during the graph construction step and the inability to produce interpretable models. By interpretable models, we mean linear prediction models learned from feature-based data sources that allow us to assess the importance of the learned weights/parameters.

Another solution is to use multiple kernel learning (MKL).[14] Given a set of kernels $\{K_d\}$, the goal of MKL is to learn a (linear) combination of kernels, $K = \sum_d \mu_d K_d$ (where $\mu_d \geq 0$ are the weights assigned to the individual kernels), along with the classifier parameters. Although there has been a lot of progress in designing efficient optimization methods for MKL (see, for example, Refs. 15 and 16, and references therein), these methods are not efficient to solve the specific problem of learning from multiple graphs for several reasons. In order to use MKL on graphs, we have to first compute a kernel on graphs.[17] Unfortunately, the resulting kernel matrix is dense and storing a pre-computed kernel matrix is infeasible for graphs with tens of thousands of nodes. Also, it is not possible to compute graph kernels "on-the-fly" unlike, for example, an RBF kernel, thereby forcing us to store the entire kernel matrix in memory. Furthermore, training a kernelized classifier (for example, non-linear SVMs) is computationally

[a]We use the terms graph and network interchangeably.

more expensive than training a linear classifier, and has the drawback of not being able to produce interpretable models. Although several advances have been made in machine learning to scale linear classifiers (see, for example, Ref. 18), large-scale learning of kernelized (non-linear) classifiers still remains a difficult problem to solve.

We propose a computationally efficient two-step procedure to integrate multiple functional interaction networks and feature-based data sources for gene function prediction. First, we extract a small set of discriminative features from the network nodes. Then, we apply a simple weighting scheme in conjunction with linear classifiers to combine these features. When compared to the methods proposed by Tsuda *et al.*[12] and Mostafavi *et al.*,[6] our method has the advantage of being able to combine networks with feature-based data sources. Furthermore, our method allows us to learn highly scalable and efficient linear classifiers for gene function prediction from tens of thousands of nodes and hundreds of GO biological process categories. Using our method, we were able to train classifiers much faster than label propagation which is already known to be computationally efficient on large-scale prediction problems.

1.1. *Preliminaries*

Given k undirected graphs, $G_d = (V, E_d)$, $d \in \{1, \ldots, k\}$, each of them having n nodes, and a set $V_\ell \subset V$ of labeled nodes, the goal is to learn a binary classifier $f : V \to \{0, 1\}$ to predict node labels by using (edge) information from all the graphs. For single graphs, the standard approach to learn such a classifier is to propagate labels in the graph.[4,5] Let $W = (w_{ij})_{i,j=1,\ldots,n}$ denote the weighted adjacency matrix of the graph, D denote the diagonal degree matrix whose entries are $d_{ii} = \sum_j w_{ij}$, $\forall i \in \{1, \ldots, n\}$. Let L denote the unnormalized graph Laplacian defined as $L = D - W$. Label propagation is reduced to the following optimization problem:

$$\begin{aligned}\hat{f} &= \operatorname*{argmin}_{f \in \mathbb{R}^n} \sum_{i \in V_\ell} (f_i - y_i)^2 + \lambda \sum_{i,j \in E} w_{ij} (f_i - f_j)^2 \\ &= \operatorname*{argmin}_{f \in \mathbb{R}^n} \sum_{i \in V_\ell} (f_i - y_i)^2 + \lambda f^\top L f \ ,\end{aligned} \tag{1}$$

where y is the label vector and $\lambda > 0$ is a regularization parameter. The estimate $\hat{f}$ can be used to score/rank the nodes where higher scores imply higher confidence in the classifier to assign a positive label to the nodes. Label propagation is a transductive learning algorithm where unlabeled examples are used for training. Since gene function prediction is a highly unbalanced classification problem, we redefine the labels to take one of three values from {-1,+1,u} and label all the unlabeled nodes with $u = (n^+ - n^-)/n$, where n^+ and n^- are the number of positive and negative labels respectively.[6] The solution to this problem can be computed by solving the sparse system of equations: $(L + \lambda \mathbf{I})\hat{f} = y$, where $\mathbf{I}$ denotes the identity matrix.

To combine multiple graphs, we construct a composite graph with adjacency matrix $W = \sum_{d=1}^{k} \mu_d W_d$, where $\mu_d \geq 0$ are the weights assigned to the individual graphs. If these weights are known, then we can compute the corresponding composite Laplacian and plug it into the optimization problem (1). Tsuda *et al.*[12] showed that the weights $\{\mu_d\}$ can be computed by solving for

$$\hat{\mu} = \underset{\mu}{\operatorname{argmin}}\, y^\top \left(\mathbf{I} + \sum_{d=1}^{k} \mu_d L_d \right)^{-1} y, \quad \text{s.t.} \quad \sum_d \mu_d \le \lambda \ .$$

Mostafavi *et al.*[6] proposed another algorithm to compute the network weights and showed that it performed better than the method of Tsuda *et al.*[12] In this algorithm, network weights are estimated by solving the following constrained linear regression problem:

$$\hat{\mu} = \underset{\mu}{\operatorname{argmin}}\, \|T - \sum_d \mu_d W_d\|_2^2, \quad \text{s.t.} \quad \mu_d \ge 0, \quad \forall d \in \{1, \dots, k\} \ , \tag{2}$$

where T is the target matrix whose entries are $t_{ij} = (n^-/n)^2$ if genes i and j are both positive and $-n^+n^-/n^2$ if genes i and j have opposite labels.

2. Methods

2.1. *Extracting features from graph nodes*

We first describe a method to extract discriminative features from graphs, where by discriminative we mean that the feature extraction method takes label information into account. Our feature extraction method is based on the 3Prop algorithm proposed by Mostafavi *et al.*[19] that labels the nodes of graphs using only three degrees of propagation. Let $P = D^{-1}W$ denote the transition probability matrix of the graph. The entries p_{ij} of P are the probability that a random walk of length one starting from node i ends at node j. The r-step probability matrix P^r can be similarly interpreted as the random walk probabilities of length r. Let y denote the 0/1 vector of labels used for training. Now, if we compute the matrix-vector product $x^{(r)} = P^r y$, then the i-th element $x_i^{(r)}$ can be interpreted as the probability that a random walk of length r from node i ends at a positively labeled node. Mostafavi *et al.*[19] argued and demonstrated empirically that random walks of length at most three suffice to propagate labels in biological networks. We use these probabilities as features for the nodes in the graph, i.e., for every node $i \in V$, we form the three-dimensional 3Prop feature vector $[x_i^{(1)}, x_i^{(2)}, x_i^{(3)}]$. Note that the probability matrix P is asymmetric. A symmetric version can be computed by setting $P = D^{-1/2}WD^{1/2}$. Algorithm 2.1 describes the feature extraction method. It is important to note that this feature extraction method is computationally highly efficient. As shown in Step 2 in Algorithm 2.1, it is not necessary to explicitly compute $P^{(r)}$ using dense matrix-matrix products; instead, it can be computed efficiently in a recursive manner using only sparse matrix-vector products. Given these features, we learn a binary linear classifier $h : \mathcal{X} \to \{0, 1\}$, where $\mathcal{X}$ denotes the feature space, parameterized by a weight vector $w \in \mathbb{R}^3$ and make predictions as $h(x) = w^\top x$. Note that $h(\cdot)$ is essentially a scoring function that can be used to score/rank nodes according to how confident the classifier is for the nodes to have a positive label.

2.2. *Combining multiple graphs*

We extract 3Prop features from the nodes of all the k graphs using the feature extraction method described in Algorithm 2.1. The next step is to learn a classifier by combining all

Algorithm 2.1 3Prop - Feature extraction from graph nodes

```
Input: Graph $G = (V, E)$ ($|V| = n$) with adjacency matrix $W$ and degree matrix $D$, label vector $y \in \{0,1\}^n$
Output: Feature matrix $X \in \mathbb{R}^{n\times 3}$
1: Compute $P = D^{-1}W$ (asymmetric) or $P = D^{-1/2}WD^{-1/2}$ (symmetric)
2: Compute $x^{(1)} = Py$, $x^{(2)} = Px^{(1)}$, $x^{(3)} = Px^{(2)}$
3: return $X = [x^{(1)}|x^{(2)}|x^{(3)}]$
```

these feature sets. A principled solution to this problem is to use multiple kernel learning (MKL). It is important to note that we do not have to design a kernel given the features extracted from the graph nodes. In other words, MKL can applied with a *linear* kernel. This greatly reduces the computational complexity of learning classifiers in the MKL framework and allows us to scale our method to graphs with thousands of nodes. Although several advances have been made to solve the MKL problem, it has been found that a uniform weighting of kernels is a hard baseline to outperform.[20,21] Cortes *et al.*[21] proposed a simple yet computationally efficient algorithm that was shown to perform better than uniform weighting and also traditional MKL methods.[14] The algorithm consists of two steps: first, independent classifiers are trained using each of the given kernels; then, the weights of these classifiers are determined using an appropriate weighting scheme. We use a similar approach in our algorithm and describe the two steps below.

We use regularized least-squares regression (RLSR) since the loss function minimized by label propagation is squared loss noting, however, that any loss function such as the hinge loss (SVM), the logistic loss (logistic regression) or the ranking loss (RankSVM[22]) can be used. We solve the following set of (independent) optimization problems to estimate the parameters of the classifiers, one for each of the k graphs:

$$\hat{w}_d = \underset{w}{\operatorname{argmin}} \sum_i (y_i - w^\top x_{id})^2 + \lambda \|w\|_2^2 \ , \quad \forall d \in \{1, \ldots, k\} \ ,$$

where we have used $x_{id} \in \mathbb{R}^3$ to denote the 3Prop feature vector for the i-th node in graph d. The solution to this problem can be computed in closed form as $\hat{w}_d = (X_d^\top X_d + \lambda \mathbf{I})^{-1} X_d^\top y$, where $X_d \in \mathbb{R}^{n\times 3}$ is the feature matrix corresponding to the graph G_d (cf. Algorithm 2.1). Note that computing this solution requires the inversion of a small 3×3 matrix. In practice, we found this method to be much faster than label propagation (1).

We then evaluate the performance of these classifiers on a separate validation set and use this performance measure directly as the network weights $\{\mu_d\}$. Specifically, we use the area under the ROC curve (AUC) as the performance measure, which is defined as follows:

$$\mathrm{AUC}(y, \hat{y}) \propto \sum_{(i,j):y_i > y_j} \left([\hat{y}_i > \hat{y}_j] + \frac{1}{2}[\hat{y}_i = \hat{y}_j] \right) \ , \tag{3}$$

where y and $\hat{y}$ are the target and the predicted label vectors respectively, and $[p] = 1$ if p is True and 0 otherwise. We use AUC since the data sets in this domain are typically highly unbalanced and the use of other performance measures such as 0/1 error is not useful in such

Algorithm 2.2 LMGraph - Learning from Multiple Graphs

Input: k undirected graphs, $G_d = (V, E_d)$ ($|V| = n$) with adjacency matrix W_d, label vector $y \in \{0,1\}^n$, training and validation set indices $t, v \subset \{1, \ldots, n\}$

Output: classifier parameters $\{w_1, \ldots, w_k\}$, network weights $\{\mu_1, \ldots, \mu_k\}$

```
1: For all $i \in \{1, \ldots, n\}$: $\tilde{y_i} = y_i$ if $i \in t$, 0 otherwise $\{$training labels$\}$
2: for $d = 1 \ldots k$ do
3:     $X_d = \mathrm{3Prop}(G_d, \tilde{y})$ $\{$extract features$\}$
4:     $w_d = \mathrm{RLSR}\left((X_d)_t, (\tilde{y})_t\right)$ $\{$train classifier$\}$
5:     For all $i \in \{1, \ldots, n\}$: $\bar{y_i} = y_i$ if $i \in v$, 0 otherwise $\{$validation labels$\}$
6:     $\mu_d = \mathrm{AUC}\left((\bar{y})_v, (X_d)_v w_d\right)$ $\{$estimate network weights$\}$
7: end for
8: return $\{w_1, \ldots, w_k\}$, $\{\mu_1, \ldots, \mu_k\}$
```

scenarios. We note that Cortes *et al.*[21] used a learning method (for example, SVM, Lasso or RLSR) to learn the weights $\{\mu_d\}$ of the independent models using their predictions on the validation set as "features." While it is indeed possible to optimize AUC[22] and learn the weights $\{\mu_d\}$, we refrained from following this approach because training a RankSVM is computationally expensive. Our own experience with learning these weights by solving the constrained least-squares regression problem: $\mathrm{argmin}_{0 \le \mu \le 1} \sum_i \left(\sum_d \mu_d h_d(x_i) - y_i\right)^2$ did not result in performance gains when compared to simply using AUC for the network weights.

Algorithm 2.2 describes our method, LMGraph, for gene function prediction from multiple graphs. The final predictions are made according to $\hat{y}(x) = \sum_d \mu_d h_d(x) = \sum_d \mu_d \cdot (w_d^\top x)$. Given this algorithm, it is straightforward to integrate feature-based data sources with the functional interaction networks – we simply combine any additional feature-based data with the 3Prop feature sets extracted using Algorithm 2.1 and train LMGraph described in Algorithm 2.2 using these feature sets.

3. Results and Discussion

3.1. *Data sets and experimental setup*

The data sets in our experiments consists of multiple functional interaction networks in three species – mouse, fly and *C.elegans*. The mouse data set consists of seven networks with 19,559 genes constructed from gene expression, protein interaction and domain composition data. To demonstrate the integration of feature-based data with networks, we also included 6,273 protein domain features extracted from Ensembl.[23] The fly data set consists of 28 networks with 13,457 genes constructed from genetic and physical interaction, co-expression, and co-localization data. The *C.elegans* data set consists of 30 networks with 18,946 genes constructed from co-expression and shared protein domain data. All the network and feature-based data sources were downloaded from the GeneMANIA prediction server.[24]

To evaluate the gene function prediction task, we use the GO biological process (BP) function categories[25] as target labels. In all the experiments, we use all the categories with at least 30 annotations. This resulted in 954, 963 and 724 categories (binary prediction tasks) for the mouse, fly and *C.elegans* data sets respectively.

We report results from three main experiments: (i) In the first experiment, we evaluate the predictive ability of classifiers trained with 3Prop features extracted from graphs in direct comparison to label propagation (cf. (1)). In this experiment, we combine the networks using uniform weights. (ii) In the second experiment, we compare our algorithm to learn from multiple graphs, LMGraph, as described in Algorithm 2.2 with both label propagation and regularized least-squares regression trained on a composite network where the networks are combined using uniform weights. We use 3Prop features in all the comparisons. (iii) In the final experiment, we combine feature-based data and networks from the mouse data set and evaluate the performance of LMGraph.

In all the experiments, we split the data sets in a stratified manner into training, validation and test sets in the ratio of 3:1:1. We report the performance of the algorithms measured in terms of the average ranking error, i.e., 1-AUC (cf. (3)), on the test data sets from five such trials in all the experiments. We use the validation sets to tune the regularization parameter, selected from the set $\{2^{-14}, 2^{-12}, \ldots, 2^{6}, 2^{8}\}$, in label propagation and regularized least-squares regression. We evaluate the statistical significance of the results based on the Wilcoxon signed-rank test when comparing the AUC for all the GO categories resulting from pairs of algorithms. In all the figures below, double asterisk ($**$) indicates that the predictive performance gains due to our method are significant when compared to the competing/baseline methods with $p \leq 0.005$; single asterisk ($*$) indicates that the differences in performance are not significant.

3.2. *Justification for 3Prop features*

We begin with an empirical justification for extracting node features from random walks of length at most three. As mentioned before, Mostafavi *et al.*[19] argued and demonstrated empirically that random walks of length three suffice to propagate labels in several types of networks, including functional interaction networks. In most of these networks, random walks quickly converge to the stationary distribution over the nodes, and therefore longer random walks do not carry any discriminative information useful for labeling the nodes. Let $\pi \in \mathbb{R}^n$ denote the stationary distribution with the property $\lim_{r\to\infty} P^r = \mathbf{1}\pi^\top$ ($\mathbf{1}$ is vector whose elements are all one), and is computed as $\pi = d/\sum_i d_i$, where d_i is the degree of the node i. The total variation distance between probability distributions p and q is defined as $\delta(p, q) := (1/2)\sum_i |p_i - q_i|$. This distance can be used to study the convergence of random walks. For a random walk starting from node i, we compute the total variation distance between the distribution $e_i^\top P^r$ (e_i is a vector with 1 at position i and 0 elsewhere) and the stationary distribution π. The smallest value of r at which this distance falls below a fixed value ϵ is known as the mixing time of the random walk, which gives us a measure of how close the distribution for random walk of length r is to the stationary distribution π. Typically, ϵ is chosen to be 0.25. The total variation distance computed for varying random walk lengths is shown in Figure 1 for mouse, fly and *C.elegans* networks for 100 different random walks starting from 100 randomly chosen nodes. Each gray line shows the effect of random walk length on the total variation distance for a random walk starting from a random node, and the red line shows the median of 100 such random walks. For each species, we combine all

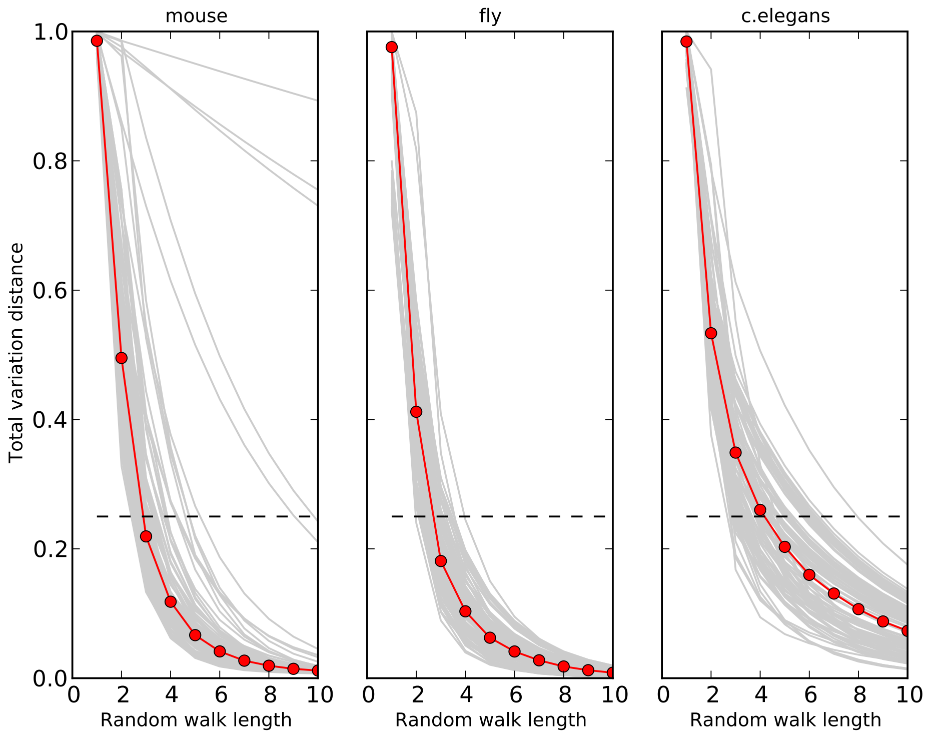

Fig. 1. Total variation distance for varying random walk lengths for mouse, fly and *C.elegans* networks, computed for 100 different random walks. Each gray line indicates a random walk starting from a random node, and the red line shows the median. The dashed line corresponds to a total variation distance of 0.25.

the networks with uniform weights and compute the transition probability matrix $P = D^{-1}W$ from this combined network. From the figure, we observe that for $r = 3$, the total variation distance has dropped below or close to 0.25 for all the networks, thus confirming the findings of Mostafavi *et al.*[19] for the data sets used in our experiments.

3.3. *3Prop vs. LProp*

We compare the performance of label propagation (LProp) with regularized least-squares regression (RLSR) trained with asymmetric and symmetric 3Prop features. In this experiment, we first combine all the networks for a given species with uniform weights to construct a single composite network. We then extract 3Prop features (cf. Algorithm 2.1) from this composite network and train RLSR with these features. For label propagation, we optimize the objective function (1) on the composite network. In Figure 2, we show the ranking errors[b] for both these methods trained on mouse, fly and *C.elegans* networks across all the GO biological process function categories. For all the networks, RLSR trained with both asymmetric and symmetric 3Prop features performs significantly better than or its performance is on par with label propagation.

[b]We do not show absolute AUC scores in the figures due to space constraints. In general, we observed percentage decrease in ranking errors across a wide range of AUC scores in all the experiments.

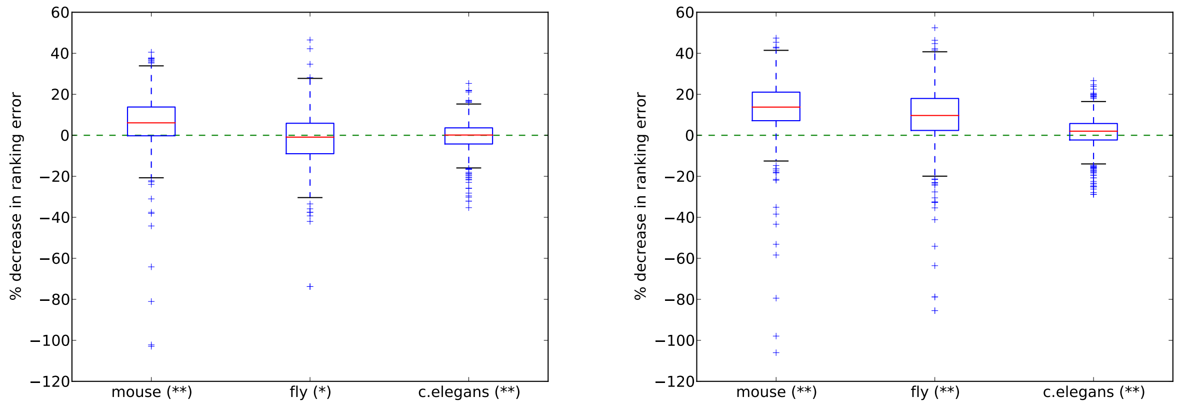

Fig. 2. Performance of label propagation and RLSR trained with asymmetric (*left*) and symmetric (*right*) 3Prop features on mouse, fly and *C.elegans* networks. The box plots show the percentage decrease in ranking error across all the GO biological process function categories.

This experiment clearly demonstrates the predictive ability of classifiers trained with 3Prop features and their potential as an alternative to label propagation for graph-based semi-supervised learning. We also observed that training a classifier with 3Prop features is computationally more efficient than label propagation even on sparse networks. As an example, on the mouse networks with 19,559 genes and a biological process function category with 100 annotations, the training time of RLSR with asymmetric and symmetric 3Prop features were 2.64s and 3.02s respectively for one trial which includes the tuning of regularization parameter, whereas label propagation took 35.29s for the same task on a 2 x 2.66 GHz dual-core processor with 4 GB of memory.

3.4. *LMGraph vs. LProp and 3Prop*

In this experiment, we first compare the performance of LMGraph with label propagation trained on composite networks combined using uniform weights. The results are shown in Figure 3 for asymmetric and symmetric 3Prop features. On all the networks, LMGraph performs significantly better than label propagation for both asymmetric and symmetric 3Prop features. On the *C.elegans* data set, we found that a slightly different version of Algorithm 2.2 wherein we first use the estimated weights $\{\mu_d\}$ to construct a composite network and then extract 3Prop features from the resulting network, followed by training an RLSR with these features performed better than the ensemble method described in Algorithm 2.2.

We also compare the performance of LMGraph with RLSR trained using 3Prop features extracted from the composite networks combined using uniform weights. The results are shown in Figure 4 for asymmetric and symmetric 3Prop features. Here, we found that LMGraph did not significantly boost the predictive performance, especially for symmetric 3Prop features and the *C.elegans* networks. However, we would like to emphasize that RLSR trained on composite networks combined using uniform weights, where we first combine the networks and then extract 3Prop features, is a strong baseline. In fact, this is a much stronger baseline than training RLSR with all the $3 \times K$ 3Prop features and this was verified by us in our

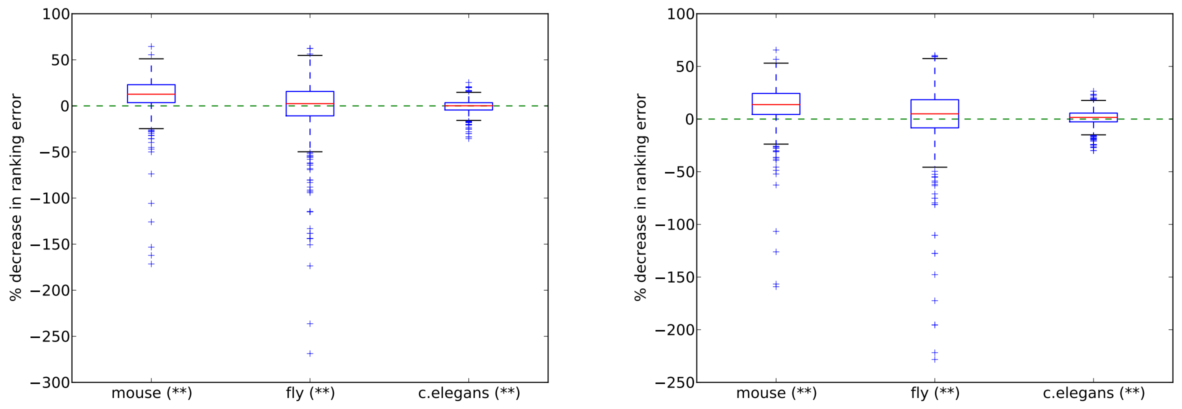

Fig. 3. Performance of LMGraph trained with asymmetric (*left*) / symmetric (*right*) 3Prop features and label propagation on mouse, fly and *C.elegans* networks.

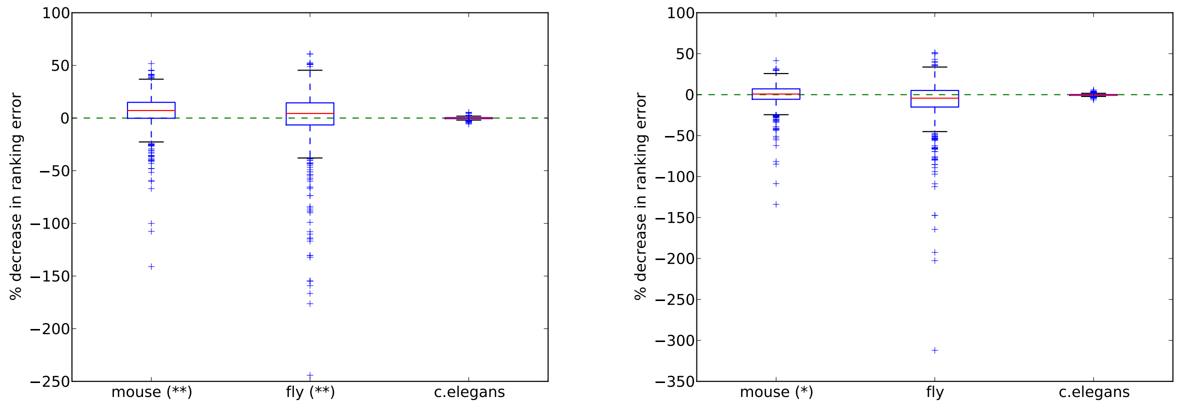

Fig. 4. Performance of LMGraph and RLSR trained with asymmetric (*left*) and symmetric (*right*) 3Prop features on mouse, fly and *C.elegans* networks.

experiments. Indeed, as is clearly evident from Figure 2 in the previous experiment, we see that these classifiers with a simple uniform weighting scheme performed significantly better than label propagation trained using the same composite networks.

3.5. *LMGraph with features*

In the final experiment, we integrate protein domain features into our learning algorithm to demonstrate the benefits of combining feature-based data sources with functional interaction networks. The results are shown in Figure 5 for the mouse networks. Integrating feature-based data into LMGraph results in significant improvements in AUC for both asymmetric and symmetric 3Prop features when compared to LMGraph trained using only the network data. Furthermore, when compared to RLSR trained with 3Prop features extracted from a composite network combined with uniform weights, we found that LMGraph trained with the protein domain features results in significant performance gains for both asymmetric and symmetric 3Prop features. LMGraph with features also performs significantly better than

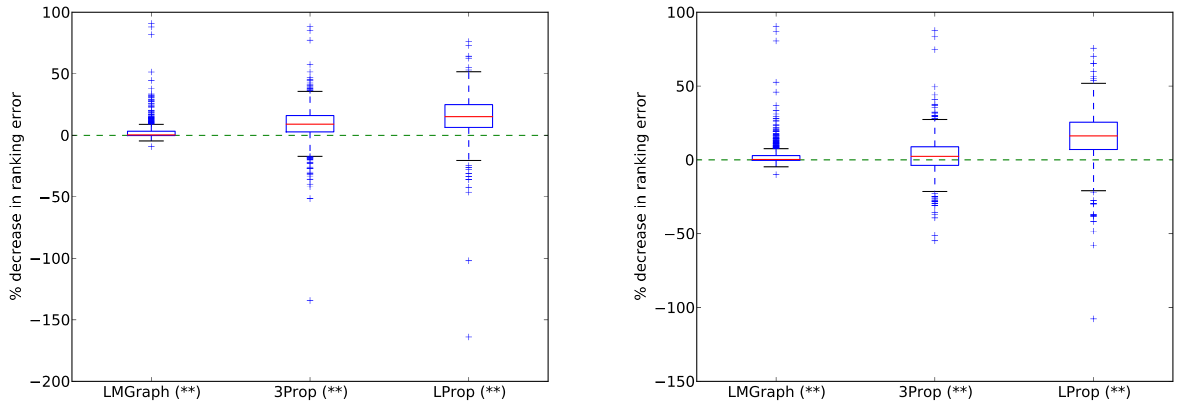

Fig. 5. Performance comparison of LMGraph trained using asymmetric (*left*) / symmetric (*right*) 3Prop and protein domain features with LMGraph, RLSR (3Prop) and label propagation (LProp) trained on a composite network combined with uniform weights on the mouse networks.

label propagation trained on a composite network combined with uniform weights; we note that it is not possible to combine features with label propagation using existing methods.[12,20]

4. Conclusions

We proposed a computationally efficient machine learning algorithm, *LMGraph*, for gene function prediction from multiple functional interaction networks. The crux and novelty of our algorithmic contribution lies in the computationally efficient two-step procedure that allows us to combine multiple graph-based and feature-based data sources, where in the first step we extract features from the nodes of graphs and in the second step we combine these feature sets and train *linear* predictors. Our feature extraction method is based on the method proposed by Mostafavi *et al.*,[19] which is known to work well on functional interaction networks. We used a variant of the ensemble method proposed by Cortes *et al.*[21] to combine multiple data sources since (i) it has been shown to perform better than the traditional MKL methods,[14] (ii) it has been shown to outperform the strong uniform weighting baseline, and (iii) it is extremely easy to implement as a machine learning practitioner in bioinformatics. However, we would like to emphasize that the user is free to design and use other relevant feature extraction methods on graphs such as spectral embeddings and also other standard multiple kernel learning methods (with a linear kernel) in these steps.

We have shown experimentally that training linear predictors with symmetric and/or asymmetric graph-based 3Prop features is a viable alternative to label propagation. Furthermore, using these features in LMGraph resulted in significant performance gains when compared to propagating labels on composite networks combined using uniform weights which is known to be a hard baseline.[12,20,21] We have also demonstrated that our method, LMGraph, can be used to combine attribute data with functional interaction networks and that this combination can result in significant performance gains for gene function prediction tasks.

Acknowledgments SV was supported by a Natural Science and Engineering Research Council of Canada operating grant to QM. We would like to thank Khalid Zuberi and Sara Mostafavi for help compiling the data sets.

References

1. A.-L. Barabási, N. Gulbahce and J. Loscalzo, *Nature Reviews Genetics* **12**, 56 (2011).
2. X. Wu, R. Jiang, M. Q. Zhang and S. Li, *Molecular Systems Biology* **4** (2008).
3. S. Aerts, D. Lambrechts, S. Maity, P. Van Loo, B. Coessens, F. De Smet, L.-C. Tranchevent, B. De Moor, P. Marynen, B. Hassan *et al.*, *Nature biotechnology* **24**, 537 (2006).
4. X. Zhu, Z. Ghahramani and J. D. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in *Proceedings of the International Conference on Machine Learning*, 2003.
5. D. Zhou, O. Bousquet, T. N. Lal, J. Weston and B. Schölkopf, Learning with local and global consistency, in *Advances in Neural Information Processing Systems 16*, 2003.
6. S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios and Q. Morris, *Genome Biology* **9**, p. S4 (2008).
7. L. Peña-Castillo, M. Tasan, C. L. Myers, H. Lee, T. Joshi, C. Zhang, Y. Guan, M. Leone, A. Pagnani, W. K. Kim *et al.*, *Genome Biol* **9**, p. S2 (2008).
8. E. M. Marcotte, M. Pellegrini, M. J. Thompson, T. O. Yeates and D. Eisenberg, *Nature* **402**, 83 (1999).
9. P. Pavlidis, J. Weston, J. Cai and W. S. Noble, *Journal of Computational Biology* **9**, 401 (2002).
10. G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan and W. S. Noble, *Bioinformatics* **20**, 2626 (2004).
11. S. Mostafavi and Q. Morris, *Proteomics* **12**, 1687 (2012).
12. K. Tsuda, H. Shin and B. Schölkopf, Fast protein classification with multiple networks, in *Proceedings of the Fourth European Conference on Computational Biology and the Sixth Meeting of the Spanish Bioinformatics Network (Jornadas de BioInformática)*, 2005.
13. D. A. Spielman and S.-H. Teng, Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.
14. G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui and M. I. Jordan, *Journal of Machine Learning Research* **5**, 27 (2004).
15. S. Sonnenburg, G. Rätsch, C. Schäfer and B. Schölkopf, *Journal of Machine Learning Research* **7**, 1531 (2006).
16. S. V. N. Vishwanathan, Z. Sun, N. Theera-Ampornpunt and M. Varma, Multiple kernel learning and the SMO algorithm, in *Advances in Neural Information Processing Systems 23*, 2010.
17. A. J. Smola and R. I. Kondor, Kernels and regularization on graphs, in *Proceedings of the Annual Conference on Computational Learning Theory and the Seventh Kernel Workshop*, 2003.
18. S. Shalev-Shwartz, Y. Singer, N. Srebro and A. Cotter, *Mathematical Programming, Series B* **127**, 3 (2011).
19. S. Mostafavi, A. Goldenberg and Q. Morris, *PLoS ONE* **7**, p. e51947 (12 2012).
20. S. Mostafavi and Q. Morris, *Bioinformatics* **26**, 1759 (2010).
21. C. Cortes, M. Mohri and A. Rostamizadeh, Ensembles of kernel predictors, in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2011.
22. R. Herbrich, T. Graepel and K. Obermayer, Large margin rank boundaries for ordinal regression, in *Advances in Large Margin Classifiers*, eds. A. Smola, P. Bartlett, B. Schölkopf and D. Schuurmans (MIT Press, Cambridge, MA, 2000).
23. Flicek *et al.*, *Nucleic Acids Research* **40**, D84 (2012).
24. Warde-Farley *et al.*, *Nucleic Acids Research* **38**, 214 (2010).
25. Ashburner *et al.*, *Nature genetics* **25**, 25 (2000).