Short Papers

A Fast Ranking Algorithm for Predicting Gene Functions in Biomolecular Networks

Matteo Re, Marco Mesiti, and Giorgio Valentini

Abstract—Ranking genes in functional networks according to a specific biological function is a challenging task raising relevant performance and computational complexity problems. To cope with both these problems we developed a transductive gene ranking method based on kernelized score functions able to fully exploit the topology and the graph structure of biomolecular networks and to capture significant functional relationships between genes. We run the method on a network constructed by integrating multiple biomolecular data sources in the yeast model organism, achieving significantly better results than the compared state-of-the-art network-based algorithms for gene function prediction, and with relevant savings in computational time. The proposed approach is general and fast enough to be in perspective applied to other relevant node ranking problems in large and complex biological networks.

Index Terms—Gene function prediction, gene ranking, biological networks, kernel functions

1 Introduction

Investigations carried out using high throughput biomolecular technologies highlighted that most biological functions rely on complex relationships between numerous biomolecular components such as proteins, DNA, RNA, and many other small molecules. This is the reason why in gene function prediction (GFP), as in many other computational biology research fields, the development of approaches suitable for the analysis of data represented in the form of a graph is of critical importance [1].

GFP is a complex task [2] with several distinctive features and poses challenging problems from a machine learning standpoint [25]. The first attempts to predict gene functions were based on algorithms able to quantify the similarities between protein sequences [4]. More general approaches to GFP collect for each protein a set of features characterizing it, and apply machine-learning algorithms to infer annotation rules based on those features [5].

The availability of large-scale networks of gene interactions constructed using different types of data, such as protein-protein interactions (PPI), genes coexpression and coregulation just to cite a few, allowed us to investigate gene functions using network-based algorithms [1].

Network-based GFP methods usually represent each data set through an undirected graph $G = (V, E)$, where nodes $v \in V$ correspond to genes, and edges $e \in E$ are weighted according to the evidence of cofunctionality implied by data sources [7]. By exploiting proximity relationships between connected nodes, these algorithms are able to transfer annotations from previously annotated (labeled) nodes to unannotated (unlabeled) ones through a learning process inherently transductive in nature. Indeed, these methods are based on algorithms that rank genes or predict labels of unannotated examples without using a global predictive model. They include guilt-by-association (GBA) methods [6], methods that integrate local learning strategies with simple weighted combination of diverse information [8], approaches based on the evaluation of the functional flow in graphs [7], methods based on Hopfield networks [9], methods that exploit relationships between homologous proteins to connect networks of different species [10], and label propagation algorithms based on Markov [11], and Gaussian Random Fields [12].

Despite their proved effectiveness, network-based GFP methods suffer of serious limitations inherent to both prediction performances (due to the challenging nature of the GFP problem) and scalability (due to the rapidly increasing size of the biomolecular networks produced by recent high-throughput technologies).

To tackle these problems, on the one hand we generalize the guilt-by-association approach [6] by introducing fast and efficient local learning strategies based on an extended notion of functional distance between genes, and on the other hand we adopt also a global learning strategy by using kernel functions able to exploit the relationships and the overall topology of the underlying biological network.

More precisely, we propose a semi-supervised transductive method that generalizes the notion of average, nearest neighbor and k-nearest neighbor distance from the set of "positive" genes annotated to a specific functional class, and embeds a general kernel to model the functional similarity between genes. Our approach can be seen as a general algorithmic scheme: by introducing different local score functions and choosing different kernels to model the similarity between genes, we can derive different network-based GFP algorithms. For instance, by adopting graph kernels [13], both direct and indirect relationships between genes can be exploited, thus taking into account the overall topology of the network.

Our ranking method is fast and scalable, since no model learning is required, but only a computation of scores, approximately linear in the number of genes. We compared our approach with several state-of-the-art GFP ranking methods, by integrating multiple sources of biomolecular data in the context of a whole-ontology GFP problem in the yeast model organism.

2 Ranking of Genes with Kernelized Score Functions

Our method rank genes in a biological network according to their likelihood of belonging to a specific functional class. At first, we introduce score functions based on different notions of kernelized distance (Section 2.1): they are defined in terms of general kernel functions and are used to rank genes. Then, we choose kernels well suited for network-oriented score functions, in order to fully exploit the topology and the graph structure of biological networks (Section 2.2).

2.1 Score Functions for Gene Ranking in Functional Networks

Let $G = (V, E)$ be an undirected weighted graph, where V is the set of vertices representing genes and E the set of edges representing some notion of functional relationships between pairs of genes/vertices. We represent both vertices of the graph and genes with natural numbers $1, 2, \ldots, n$, since each vertex of G is univocally associated with a gene. Let $\boldsymbol{W}$ be the corresponding adjacency matrix with weights w_{ij} representing the "strength" of the relationship between vertices $i, j \in V$, and $V_C \subset V$ a subset of "positive" vertices belonging to a specific functional category C

• *The authors are with the Dipartimento di Informatica, Università degli Studi di Milano, Via Comelico 39/41, I-20135 Milano, Italia. E-mail: {re, mesiti, valentini}@di.unimi.it.*

Manuscript received 17 Feb. 2012; revised 29 July 2012; accepted 31 July 2012; published online 18 Aug. 2012.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2012-02-0039. Digital Object Identifier no. 10.1109/TCBB.2012.114.

(e.g., a term of the Gene Ontology or FunCat). A set of features $\boldsymbol{x}_i \in X$ can be associated with a gene i. For instance, $\boldsymbol{x}_i$ could represent the expression or the phylogenetic profile of a gene i or whatever available data for a given gene/vertex i.

Our aim is to derive score functions $S : V \longrightarrow \mathbb{R}^+$ based on properly chosen kernels, by which we can directly rank vertices according to the values of $S(i)$: the higher the score, the higher the likelihood that a gene belongs to a given functional class. Score functions are based on distance measures defined in a suitable Hilbert space $\mathcal{H}$. More precisely, let $\phi : X \to \mathcal{H}$, be a mapping to a given universal reproducing kernel Hilbert space $\mathcal{H}$, and $K : X \times X \to \mathbb{R}$ its associated kernel function, such that $<\phi(\cdot), \phi(\cdot)>_{\mathcal{H}} = K(\cdot, \cdot)$, where $<\cdot, \cdot>_{\mathcal{H}}$ represents an internal product in $\mathcal{H}$.

We introduce distance measures $D(i, V_C, X)$ in the Hilbert space between a given vertex/gene i and the set of genes V_C belonging to a specific functional class C, according to the data X associated with each gene. We chose to define a distance measure on Hilbert space, since by exploiting the classical "kernel-trick" [14] we can embed any valid kernel into the distance measure itself, thus resulting in a modular approach by which existing graph kernels, or in perspective graph kernels properly designed for GFP, can be applied to rank genes according to their functions.

If there is no ambiguity about the data X, for the sake of simplicity we denote $D(i, V_C, X)$ as $D(i, V_C)$. By choosing different distance measures, diverse score functions can be derived. In the following, we introduce the *Average score*, the *Nearest Neighbors*, and the *K-Nearest Neighbors* scores, that are based on the *guilt-by-association* principle: the label or the score associated with a given node depend on the label or the scores associated with the neighboring nodes [6]. As an example, consider the majority voting scheme, by which we choose for a node the most represented label within its neighbors.

2.1.1 Average Score

We can define a distance measure $D_{AV}(i, V_C)$ of a vertex $i \in V$ w.r.t. to a set of nodes V_C, simply as the average distance in the mapped Hilbert space $\phi : X \to \mathcal{H}$ between i and the set of nodes included in V_C:

$$D_{AV}(i, V_C) = \left\| \phi(\boldsymbol{x}_i) - \frac{1}{|V_C|} \sum_{j \in V_C} \phi(\boldsymbol{x}_j) \right\|^2. \tag{1}$$

By developing the square (1) we obtain

$$\begin{aligned} D_{AV}(i, V_C) = <\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i)> - \frac{2}{|V_C|} \sum_{j \in V_C} <\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)> \\ + \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} <\phi(\boldsymbol{x}_k), \phi(\boldsymbol{x}_j)>, \end{aligned} \tag{2}$$

where $<\phi(\cdot), \phi(\cdot)>$ represents an internal product in the feature space $\mathcal{H}$. By recalling that (2) represents a distance measure and $<\phi(\cdot), \phi(\cdot)> = K(\cdot, \cdot)$, we can obtain a similarity measure simply by changing the sign:

$$\begin{aligned} Sim_{AV}(i, V_C) = -K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \frac{2}{|V_C|} \sum_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ - \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} K(\boldsymbol{x}_k, \boldsymbol{x}_j). \end{aligned} \tag{3}$$

By observing that the third term of (3) is equal for all $i \in V$, we can obtain the following *average score* S_{AV}:

$$S_{AV}(i, V_C) = -K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \frac{2}{|V_C|} \sum_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{4}$$

This score represents the average similarity of the gene i w.r.t. to the genes belonging to the V_C set. If all $K(\boldsymbol{x}_i, \boldsymbol{x}_i)$ are equal for each $i \in V$ (i.e., the "autosimilarity" of genes does not matter), we can further simplify (4) by removing its first term. It is worth noting the S_{AV} score resembles the one proposed by Borgwardt et al. in the context of gene function prediction from synthetic lethality networks: from this standpoint our approach can be viewed as an extension of the algorithm proposed in [15].

2.1.2 Nearest-Neighbors Score

If instead of considering the average distance (1) between a vertex i and V_C we consider the minimum distance between i and V_C, we can obtain the *nearest neighbors score* S_{NN}. To this end let consider

$$D_{NN}(i, V_C) = \min_{j \in V_C} \| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \|^2. \tag{5}$$

The distance (5) is the minimum distance in the Hilbert space of the vertex v w.r.t. the set of vertices belonging to V_C. By developing the square (5) we obtain

$$\begin{aligned} D_{NN}(i, V_C) = \min_{j \in V_C} [<\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i)> \\ + <\phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_j)> - 2<\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)>]. \end{aligned} \tag{6}$$

From (6) we can easily derive the following similarity measure:

$$Sim_{NN}(v, V_C) = -\min_{j \in V_C} \big[K(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j) + K(\boldsymbol{x}_j, \boldsymbol{x}_j)\big]. \tag{7}$$

If $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$ is equal for all $j \in V$, we can simplify (7), thus achieving the *nearest neighbors score* S_{NN}:

$$S_{NN}(i, V_C) = -\min_{j \in V_C} -2K(\boldsymbol{x}_i, \boldsymbol{x}_j) = 2 \max_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{8}$$

If it does not hold that $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$ is equal for all $j \in V$, then we can set $S_{NN}(i, V_C) = Sim_{NN}(i, V_C)$.

2.1.3 K-Nearest Neighbors Score

A natural extension of the S_{NN} score can be obtained by introducing a different notion of distance based on *k-nearest neighbors distance*:

$$D_{kNN}(i, V_C) = \sum_{j \in I_k(i)} \| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \|^2, \tag{9}$$

where $I_k(i) = \{j \in V_C | j \text{ is ranked among the first k in } V_C\}$. The distance (9) is the sum of distances of the vertex i from the set of the k-nearest vertices included in V_C in the Hilbert space to which $\boldsymbol{x}_i$ is mapped by ϕ. By developing the square (9) we can obtain a similarity measure:

$$Sim_{kNN}(i, V_C) = -\sum_{j \in I_k(v)} \big(K(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j) + K(\boldsymbol{x}_j, \boldsymbol{x}_j)\big). \tag{10}$$

This similarity measure can be directly used as a *k-nearest neighbors score* S_{kNN}, but in the case that $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$ is equal for all $j \in V$, we can simplify (10):

$$S_{kNN}(v, V_C) = 2 \sum_{j \in I_k(i)} K(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{11}$$

The proposed distance measures and the corresponding scores are motivated by an extension of the guilt-by-association principle. Indeed, all the proposed kernelized score functions share the common principle that the score of each node depends only by its neighbors. It is worth noting that we extend the notion of neighbor through the kernel K: by choosing an appropriate kernel, node j can be in the neighbor of node i even if there is no edge between them in the original graph G (i.e., $w_{ij} = 0$), but $K(\boldsymbol{x}_i, \boldsymbol{x}_j) > 0$. From this standpoint the Gram matrix $\boldsymbol{K}$ can be interpreted as a novel weighted adjacency matrix in the projected Hilbert space induced by the mapping $\phi : X \to \mathcal{H}$.

We emphasize that we propose a general algorithmic scheme: by choosing different score functions and/or designing novel kernels able to capture the overall topology of the network, we can in principle derive different network-based gene ranking algorithms for GFP.

The proposed method is very fast, since no model learning is required, but only a computation of scores based on kernelized distances: once the kernel matrix has been computed, the score computation has a complexity $\mathcal{O}(|V| \cdot |V_C|)$, that is approximately linear when the number of "positive" nodes is largely lower than the overall number of vertices. We remark that this is just the usual setting of GFP problems.

2.2 Random Walk (RK) Kernels for Network-Oriented Score Functions

All the score functions introduced in Section 2.1 are based on a generic kernel function $K(\cdot,\cdot)$. It is worth noting that in principle any valid kernel can be used (e.g., Gaussian, Laplacian or polynomial kernels), but in the context of network-based GFP we need to apply meaningful kernel functions able to capture the functional similarity between genes connected in an undirected graph. From this standpoint graph-based kernels represent a natural choice, since they are able to take into account the topology of the network as well as the strength of the similarities between vertices [13].

Among them, *random walk kernels* [16] can capture not only relationships coming from direct neighborhoods between genes, similarly to *guilt by association* methods [6], but also relationships coming from shared and more in general indirect neighbors between genes. This is of paramount importance in the context of network-based GFP, since, while it is quite obvious that functional relationships are coded into direct neighbors, important functional relationships between genes can also be coded through indirect neighbors [8]: for instance, proteins belonging to the same complex may also not directly interact, but can both contribute to the realization of the biological function mediated by the entire protein complex, or enzymes belonging to the same biological process may not share the same links, since their catalyzed reactions can be linked through other intermediate reactions belonging to the same pathway.

Random walk kernels represent the kernelized version of *Markov Random Walks*, by which random trajectories across graphs can be exploited to investigate the relationships between nodes and to score or to label each node with respect to a specific property of the vertices [21]. We adopt the same term kernel to refer to both the kernel function $K(\cdot,\cdot)$ and its corresponding Gram matrix $\boldsymbol{K}$ whose elements are $k_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Given a symmetric adjacency matrix $\boldsymbol{W}$ of the undirected graph $G = (V, E)$, the unnormalized graph Laplacian is $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is a diagonal matrix with elements $d_{ii} = \sum_j w_{ij}$; this matrix is named "degree" matrix, since its diagonal elements represent the (weighted) degree of the corresponding node. The name of the $\boldsymbol{L}$ matrix is derived from its analogy with the Laplacian operator Δ on continuous spaces and it can be shown that $-\boldsymbol{L}$ up to a constant is exactly the finite difference discretization of Δ on a regular lattice [16]. The normalized graph Laplacian is

$$\begin{aligned}\tilde{\boldsymbol{L}} &= \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{D}^{-\frac{1}{2}} \\ &= \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{D}\boldsymbol{D}^{-\frac{1}{2}} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}},\end{aligned} \tag{12}$$

where $\boldsymbol{I}$ is the identity matrix. In other words, to obtain the normalized graph Laplacian, each element of the adjacency matrix $\boldsymbol{W}$ is divided by the square root of the product of the sum of the weights of its corresponding row and column: from this standpoint $\tilde{\boldsymbol{L}}$ can be regarded as a sort of regularization of the original adjacency matrix $\boldsymbol{W}$ of the graph. The *one-step random walk kernel* [16] can be defined in terms of the normalized graph Laplacian:

$$\begin{aligned}\boldsymbol{K}_{rw} &= a\boldsymbol{I} - \tilde{\boldsymbol{L}} = a\boldsymbol{I} - \boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}} \\ &= (a-1)\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}},\end{aligned} \tag{13}$$

with $a > 1$. The *q-step random walk kernel* is a slight generalization of (13) [16]:

$$\boldsymbol{K}_{rw}^{q} = (a\boldsymbol{I} - \tilde{\boldsymbol{L}})^{q}, \tag{14}$$

where $q \geq 1$ is an integer representing the number of steps of the random walk across the graph. The name of the kernel derives from the fact that (14) is up to scaling terms equivalent to a q-step random walk on the graph with random restarts, a well-known algorithm used for scoring webpages in the Google search engine [17]. Indeed, consider a random walk with random restart on a graph with adjacency matrix $\boldsymbol{W}$, the vector of probabilities $\boldsymbol{p} = [p_1, p_2, \ldots, p_n]$ of being on a certain node $i, 1 \leq i \leq n$, the degree matrix $\boldsymbol{D}$ associated with the graph, and the probability α of a random restart at an arbitrary node. Then, the probability distribution over states obeys the following discrete time evolution equation:

$$\boldsymbol{p}^{t+1} = [\alpha\boldsymbol{I} + (1-\alpha)\boldsymbol{D}^{-1}\boldsymbol{W}]\boldsymbol{p}^{t}. \tag{15}$$

The stationary distribution of $\boldsymbol{p}$ is determined by the largest eigenvalue/eigenvector pair of the transition matrix $\boldsymbol{Q} = [\alpha\boldsymbol{I} + (1-\alpha)\boldsymbol{D}^{-1}\boldsymbol{W}]$ in (15), and values of $\boldsymbol{p}$ at convergence determine the ranking of the nodes. It can be shown that the spectrum of the 1-step random walk kernel (13) rescaled by $\frac{1}{a}$ is the same of the matrix $\boldsymbol{Q}$ if we set $\alpha = \frac{1-a}{a}$ [16]. Hence, the random walks on graphs and the stationary distribution arising from them are closely related to the eigensystem of $\tilde{\boldsymbol{L}}$. The main difference between these two methods is that with classical random walk we may have directed graphs leading to asymmetric $\boldsymbol{W}$, while with random walk kernels we need to deal with symmetric positive semidefinite Gram matrices, thus requiring undirected graphs, and consequently a symmetric adjacency matrix $\boldsymbol{W}$, leading to symmetric $\boldsymbol{L}$ and $\tilde{\boldsymbol{L}}$ matrices.

To implement a *q-step random walk kernel* we can adopt a step-by-step strategy: we can at first compute $\boldsymbol{K}_{rw}^{2}$ by using the one-step random walk kernel (13), and then we can compute $\boldsymbol{K}_{rw}^{3}$ by using the computed $\boldsymbol{K}_{rw}^{2}$, adopting a recursive strategy for each $q \geq 2$:

$$\boldsymbol{K}_{rw}^{q} = \boldsymbol{K}_{rw}^{q-1}\boldsymbol{K}_{rw}. \tag{16}$$

In *q-step random walk kernels* the parameter a allows to balance the weight of direct and indirect connections between nodes: by tuning a and the number q of steps we can modulate the influence of direct and indirect connections between genes in the network.

3 Experimental Setup

3.1 Prediction of FunCat Classes in Yeast

We tested our proposed methods on gene ranking tasks with respect to FunCat (Functional Catalogue) classes with the yeast model organism [3]. The tree-like structure of FunCat, with up to six levels of increasing specificity accounts for different functional characteristics of genes and gene products.

3.2 Data

For gene ranking in yeast, we combined six biomolecular data sets previously used for the related task of gene classification [18]. The data sets include protein-protein interaction, protein domain, gene expression, and pairwise sequence similarity data.

We considered only yeast genes common to all data sets, and in order to get a not too small set of positive examples for training, for each data set we selected only the FunCat-annotated genes,[1] and

1. Our experiments build on annotations coded in the funcat-2.1 scheme, and funcat-2.1_data_20070316 data, available from the MIPS web site (http://mips.gsf.de/projects/funcat).

the classes with at least 20 positive examples, using the *HCgene* R package [19]. This selection process yielded 1,901 yeast genes annotated to 168 FunCat classes distributed across 16 trees and five hierarchical levels.

From each data set, we computed the kernel matrix using a linear kernel, thus obtaining six square symmetric matrices with equal number of elements. Then, we integrated the data simply by summing the corresponding kernel matrices (unweighted averaging kernel integration).[2]

As a preprocessing step, to obtain a sparse version of the corresponding graph, and to avoid "singleton" disconnected nodes, we set a threshold for the edges in order to guarantee at least one neighbor for each node. In practice, for each node we computed the maximum weight of its associated edges, and among these maxima we chose the minimum, and we set it as a threshold for the edges of the graph. In this way, we obtain a graph with 1,901 genes/nodes and 489,338 edges, with a graph density equal to 0.1354.

The resulting integrated matrix corresponds to the weighted adjacency matrix $\boldsymbol{W}$ of the graph G considered in Section 2. According to (13) we obtained from $\boldsymbol{W}$ the corresponding $\boldsymbol{K}_{rw}$ 1-step random walk kernel, and according to (14) we constructed the two and three steps random walk kernel.

3.3 Compared Methods

We compared our proposed kernelized score functions with several state-of-the-art ranking methods for GFP. In particular, we considered *GeneMANIA* [12], an algorithm based on Gaussian Random Fields, that ranked among the best methods in the MouseFunc competition for mouse GFP [20]. Since our score functions are based on random walk kernels, we compared our method also with the classical random walk and its variant *RW 2 steps*, that simply stops after two random steps instead of running till the convergence condition is satisfied [21]. We evaluated also another variant of *RW*, i.e., the random walk with restart (*RWR*) algorithm, just successfully applied in gene prioritization problems [22]: at each step of the random walk in the graph the random walker can move to one of its neighbors or can restart from its initial condition with probability $0 < \theta < 1$. We considered also a classical inductive method, i.e., the Support Vector Machine (SVM) algorithm, largely applied in computational biology and in GFP. More precisely, we considered a probabilistic version of SVM [23], in order to obtain a probabilistic score to rank genes with respect to functional classes. Finally, as a baseline, we implemented a simple version of the guilt-by-association algorithm [6], by which a score for each node is computed by averaging the weights $w_{ij} \in \boldsymbol{W}$ of the edges connecting the node i with positive labeled nodes j in the neighborhood of i.

3.4 Performance Measures and Software Implementation

To estimate the generalization performances of all the considered methods we adopted a classical stratified fivefold cross validation repeated 10 times (averaging results across the repetitions), and we applied the Wilcoxon signed-ranks test to compare the overall results between methods.

The performances of the compared ranking methods have been assessed using the Area Under the ROC Curve (AUC) and the precision at different levels of recall, since usually the FunCat classes are highly unbalanced, with negative examples (genes not annotated for a given class) that largely outnumber positives (genes annotated for a given class).

Our proposed methods have been implemented in R and C language (software is available upon request from the authors), as well as the code for *RW*, *RWR*, and *GBA*, while for the probabilistic SVM we used the C++ LIBSVM library [24], and for *GeneMANIA* the MATLAB code available from the authors.

2. The integrated kernel matrices and the labels of the 168 FunCat classes are downloadable from http://homes.dsi.unimi.it/~valenti/DATA/Yeast-GeneRank.

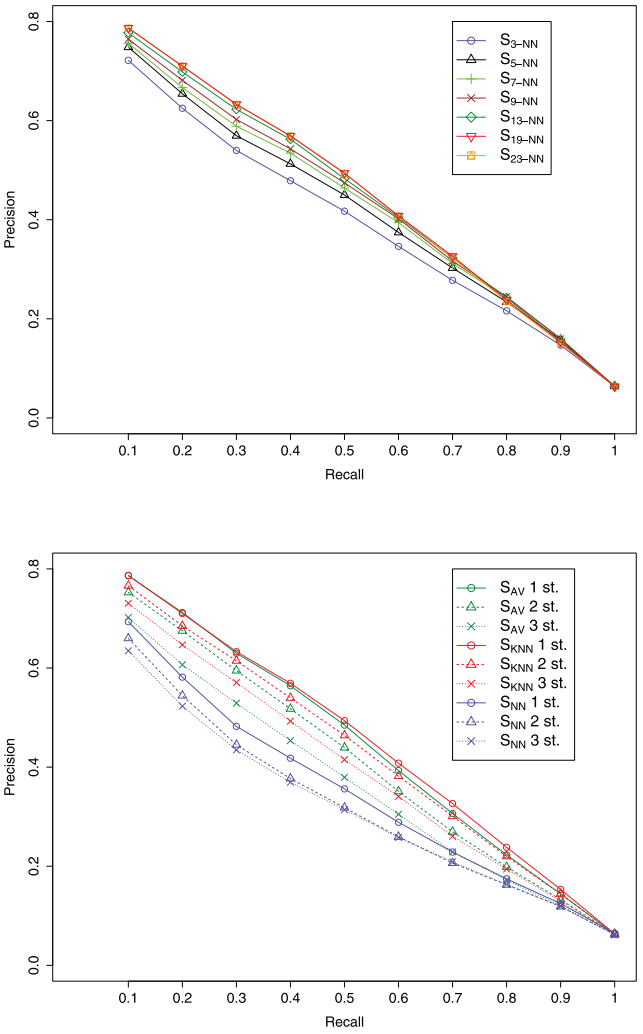

Fig. 1. Precision at different levels of recall with kernelized score functions (results averaged across classes). Above: S_{kNN} results with k varying between 3 and 23. Below: Compared results between $S_{kNN}, k = 19$ (red lines), S_{AV} (green) and S_{NN} (blue) using random walk kernels with 1 (continuous lines with circles), 2 (dashed lines with triangles), and 3 (dotted lines with crosses) steps.

4 Results

At first we analyzed the results of our proposed kernelized score functions using different types of random walk kernels (Section 4.1), and then we compared our proposed methods with other state-of-the-art ranking methods for GFP (Section 4.2).

4.1 Comparing Kernelized Score Functions

We evaluated the proposed score functions (Section 2.1) using random walk kernels (Section 2.2). To understand the impact of the number k of neighbors in S_{kNN} score function, we compared the precision at different levels of recall by varying k between 3 and 23. The overall results show that the largest values of k, i.e., $k = 19$ or $k = 23$ achieve the best results: red and orange lines, quite overlapped, lie above all the other precision/recall curves (Fig. 1, above).

Then, we compared the different score functions and at the same time the impact of the choice of 1, 2, or 3-steps random walk kernels. Cross-validation results averaged across classes show that in terms of precision/recall curves S_{kNN} and S_{AV} largely outperform S_{NN} independently of the number of steps of the random

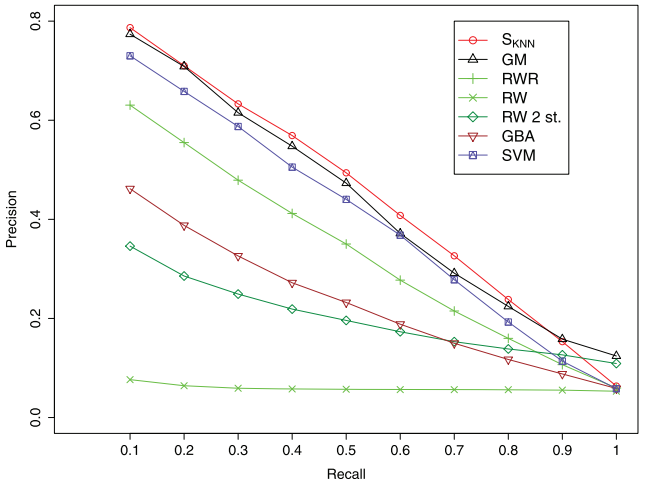

Fig. 2. Comparison of precision at different levels of recall between different ranking methods (results averaged across classes). S_{kNN}: $19-NN$-score random walk kernel; GM: *GeneMANIA*; RWR: random walk with restart; RW: random walk; $RW\ 2$ st: 2-steps random walk; GBA; guilt-by-association; SVM: probabilistic support vector machine.

walk kernels used in the score functions (Fig. 1 below). Moreover S_{kNN} outperforms S_{AV} with 2 or 3-steps random walk kernels, at almost all the recall levels (Wilcoxon signed-ranks sum test, *p-value* $< 10^{-5}$), but no significant difference, according to the Wilcoxon test, can be found at any recall level when 1-step random walk kernels are used instead. Independently of the choice of the score function, the best results are obtained with 1-step random walk kernels (Fig. 1): in this context, direct "functional similarities" between genes are more informative than indirect ones, even if for some classes also indirect connections play a significant role to uncover functions of genes. For instance, at 0.2 recall level, with S_{kNN} we obtain better results with 2-steps w.r.t. to 1-step random walk kernel for 24 FunCat classes, and for 21 classes with 3-steps w.r.t to 2-steps random walk kernels. For some classes the difference in precision is very large: with the class 42.25 "vacuole or lysosome" we move from 0.57 to 0.80 precision when we use 2-steps instead of 1-step random walk kernel. These results suggest that by properly choosing the kernel for each functional class (e.g., by internal cross validation), we could further improve the overall performances.

4.2 Comparison with State-of-the-Art Ranking Methods

Fig. 2 shows the precision at different levels of recall averaged across classes achieved by S_{kNN} compared with other six gene ranking methods (Section 3.3). The red precision/recall curve with circles denoting S_{kNN} is above all the other curves at any recall level (except for $\text{recall} = 1$) and the difference is always statistically significant (Wilcoxon signed-ranks test, *p-value* $< 10^{-6}$) w.r.t. all the other compared methods, except *GeneMANIA*, where the difference is significant for recall levels between 0.2 and 0.8 (Wilcoxon test, *p-value* $< 10^{-4}$). Also linear SVMs with fitted sigmoid achieve good results, even if significantly worse than those obtained by both S_{kNN} and *GeneMANIA*. On the contrary, the classical random walk algorithm (*RW*) completely fails in this task. This is likely due to the fact that the convergence of the algorithm requires for most classes tens or hundreds of iterations, thus leading to explore too remote and indirect similarities between genes and to "forget" the a priori knowledge coded in the initial probabilities of the genes. This interpretation is also confirmed by the fact that random walks with restart and in part also 2-steps Random walks achieve reasonable results, since they, in different ways, "remember" the initial probabilities of known positive genes, by explicitly recalling them (*RWR*) or considering only direct or common neighbors between genes (2-steps *RW*). These results, according to the experiments of Section 4.1, show that for most classes direct neighbors are more informative than indirect neighbors. Our proposed score functions, and in particular S_{kNN} (and also S_{AV} that achieves comparable results with S_{kNN}, Fig. 1) significantly outperform both classical *RW* and *RWR* algorithms, showing that embedding random walk kernels in properly defined score functions is the key to improve the performances in this difficult gene ranking task.

These overall results are also confirmed through the AUC measures: Table 1 shows that our proposed kernelized score functions achieve the best results in terms of average AUC across classes. More precisely, both S_{kNN} and S_{AV} accomplish significantly better results than all the other compared methods (Wilcoxon test, *p-value* $< 10^{-5}$). Also S_{NN} significantly outperform the other methods (but with no significant difference w.r.t. *GeneMANIA*). Note that also *RWR* and the simple *GBA* method achieve significantly better results than SVMs in terms of AUC. Also AUC results confirm that the "vanilla" *RW* algorithm fails in this gene ranking task ($\text{AUC} \simeq 0.5$), but if we limit the random

TABLE 1
Comparison of AUC Values Averaged across Classes between Different Ranking Methods

S_{kNN}	S_{AV}	S_{NN}	GM	RWR	RW	RW 2 st	GBA	SVM
0.8840	0.8782	0.8649	0.8614	0.8383	0.5118	0.7660	0.8014	0.7770

TABLE 2
Average Precision at 0.2 Recall and Average AUC at the Five Levels of the FunCat Taxonomy

Precision at 20 % recall across levels										
level	S_{kNN}	S_{AV}	S_{NN}	GM	RWR	RW	RW 2 st	GBA	SVM	n. classes
1	0.7111	0.6641	0.6156	0.6704	0.4921	0.1857	0.5223	0.3824	**0.8806**	16
2	**0.7299**	0.7174	0.6015	0.6926	0.5145	0.0722	0.3024	0.3301	0.6914	49
3	0.7243	**0.7405**	0.5920	**0.7405**	0.5958	0.0446	0.2367	0.4264	0.6823	65
4	0.6682	**0.6917**	0.5310	0.6781	0.5690	0.0390	0.2744	0.4011	0.4835	30
5	0.6247	0.6263	0.4965	**0.7429**	0.5531	0.0228	0.1418	0.4011	0.4690	8
AUC across levels										
level	S_{kNN}	S_{AV}	S_{NN}	GM	RWR	RW	RW 2 st	GBA	SVM	n. class
1	0.8276	0.8069	0.7754	0.7920	0.7466	0.5194	0.7608	0.7006	**0.8752**	16
2	**0.8620**	0.8535	0.8424	0.8370	0.8071	0.4902	0.7389	0.7649	0.7849	49
3	**0.9006**	0.8983	0.8874	0.8807	0.8614	0.5146	0.7712	0.8279	0.8058	65
4	**0.9052**	0.9030	0.8880	0.8860	0.8717	0.5216	0.7888	0.8419	0.7248	30
5	**0.9206**	**0.9206**	0.9164	0.9067	0.9064	0.5733	0.8173	0.8682	0.4954	8

Numbers in boldface refer to the best results for a given level.

TABLE 3
Empirical Computational Time Complexity of the Compared Methods

	S_{kNN}	S_{AV}	S_{NN}	GM	RWR	RW	RW 2 st	GBA	SVM
time (sec.)	10	7	7	118	435	3297	85	6	21805

The computational time (sec.) refers to a single fivefolds cross validation performed on the entire FunCat ontology at genome-wide level.

walk to only 2 steps a certain learning can be registered (AUC $\simeq 0.77$, Table 1).

To better understand the behavior of the compared methods w.r.t. the specificity of the functional classes, we reported the precision at 20 percent recall and the AUC averaged per each level of the FunCat Taxonomy (Table 2). Level 1 nodes are the root nodes, that is the more general classes (e.g., "metabolism"), level 2 nodes are their children classes, till to the fourth and fifth levels corresponding to more specific functional classes (e.g., "metabolism of methionine"). Our proposed score functions obtain the best results at levels 2, 3, and 4 for both AUC and precision at 20 percent recall, and also at level 5 in terms of AUC. SVMs achieve the best results at the first level of the hierarchy, where the largest set of positive examples is available: having enough information on a class, inductive algorithms such as SVMs outperform the other transductive methods. Quite interestingly, whenever we consider more specific classes, characterized by a lower number of positive examples, the performance of SVM substantially decay, confirming recent results [25], while our proposed score functions maintain or also increment the precision and the AUC (Table 2). This is of paramount importance, since the biologists are usually interested to the most specific classes, that better characterize the functional role of genes.

We note that S_{kNN} obtains better results at the first two levels of the hierarchy, while S_{AV} at the lower levels. This behavior suggests that simple strategies that use different score functions at different levels, or more in general methods able to select the best score function for each functional class (e.g., by internal cross validation), could lead to significantly better results. However, for the methods compared in the proposed experiments, we did not perform any fine tuning of their parameters by internal cross validation.

The proposed score functions are very fast, and are able to perform the ranking tasks at genome-wide and whole ontology level in yeast in less than 10 seconds on a Xeon quad processor W3520 2.67 GHz, with 8 Gb RAM (Table 3). Note that all the other compared methods have a larger empirical time complexity except for the simple *GBA* method that is even slightly faster. Indeed, both *kernelized score functions* and *GBA* have linear time complexity in GFP problems, while the other network-based compared methods have a quadratic complexity with sparse graphs [26]. These results show that our methods could be in perspective efficiently applied to larger genomes (i.e., mouse, human, fly, or also *Arabidopsis*), and using also ontologies with more classes, such as the Gene Ontology.

5 Conclusions

Our proposed score functions adopt both local learning strategies based on a generalized notion of distance in a universal reproducing kernel Hilbert space, and global learning strategies based on the choice of proper graph kernels to exploit the overall topology of the underlying biological network. Whole-ontology experiments with yeast show that the proposed methods achieve significantly better results than the compared state-of-the art ranking algorithms for GFP. Moreover a comparable AUC and precision at fixed recall rate is obtained at each level of the FunCat ontology, with no significant decay (at least in terms of AUC) also for the most specific classes. Compared results between S_{kNN} and S_{AV} suggest that selection strategies able to properly choose the best score function for each level of the taxonomy or for each functional class could further improve the overall performance of the proposed gene ranking system. The transductive nature of the method and the efficient computation of the score functions consistently reduce the computational costs and allow to nicely scale with large biomolecular networks.

For possible future work, we observe that even if in the context of the proposed score functions we applied random walk kernels, other kernels could in principle be designed to better characterize the functional similarities between genes. Moreover, we observe that our proposed method is general enough to be in perspective adapted to other relevant node ranking problems in computational biology, and recent results on drug repositioning and gene prioritization with respect to tumoral diseases seem to confirm this hypothesis [27], [28].

Acknowledgments

The authors would like to thank the editor and the reviewers for their comments and suggestions, and gratefully acknowledge partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

[1] R. Sharan, I. Ulitsky, and R. Shamir, "Network-Based Prediction of Protein Function," *Molecular Systems Biology*, vol. 3, article 88, 2007.
[2] R. Rentzsch and C. Orengo, "Protein Function Prediction-the Power of Multiplicity," *Trends Biotechnology*, vol. 27, no. 4, pp. 210-219, 2009.
[3] A. Ruepp et al., "The FunCat, a Functional Annotation Scheme for Systematic Classification of Proteins from Whole Genomes," *Nucleic Acids Research*, vol. 32, no. 18, pp. 5539-5545, 2004.
[4] S. Altschul et al., "Basic Local Alignment Search Tool," *J. Molecular Biology*, vol. 215, pp. 403-410, 1990.
[5] P. Pavlidis et al., "Learning Gene Functional Classification from Multiple Data," *J. Computational Biology*, vol. 9, pp. 401-411, 2002.
[6] M. Mayer and P. Hieter, "Protein Networks - Guilt by Association," *Nature Biotechnology*, vol. 18, no. 12, pp. 1242-1243, 2000.
[7] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani, "Global Protein Function Prediction from Protein-Protein Interaction Networks," *Nature Biotechnology*, vol. 21, pp. 697-700, 2003.
[8] H. Chua, W. Sung, and L. Wong, "An Efficient Strategy for Extensive Integration of Diverse Biological Data for Protein Function Prediction," *Bioinformatics*, vol. 23, no. 24, pp. 3364-3373, 2007.
[9] A. Bertoni, M. Frasca, and G. Valentini, "COSNet: A Cost Sensitive Neural Network for Semi-Supervised Learning in Graphs," *Proc. European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD '11)*, pp. 219-234, 2011.
[10] A. Mitrofanova, V. Pavlovic, and B. Mishra, "Prediction of Protein Functions with Gene Ontology and Interspecies Protein Homology Data," *IEEE/ACM Trans Computational Biology and Bioinformatics*, vol. 8, no. 3, pp. 775-784, May/June 2011.
[11] M. Deng, T. Chen, and F. Sun, "An Integrated Probabilistic Model for Functional Prediction of Proteins," *J. Computational Biology*, vol. 11, pp. 463-475, 2004.
[12] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris, "GeneMANIA: A Real-Time Multiple Association Network Integration Algorithm for Predicting Gene Function," *Genome Biology*, vol. 9, article S4, 2008.
[13] H. Kashima, K. Tsuda, and A. Inokuchi, "Kernels for Graphs," *Kernel Methods in Computational Biology*, pp. 155-170, MIT Press, 2004.
[14] B. Scholkopf, K. Tsuda, and J. Vert, *Kernel Methods in Computational Biology*. MIT Press, 2004.
[15] G. Lippert, Z. Ghahramani, and K. Borgwardt, "Gene Function Prediction form Synthetic Leathality Networks via Ranking on Demand," *Bioinformatics*, vol. 26, no. 7, pp. 912-918, 2010.
[16] A. Smola and I. Kondor, "Kernel and Regularization on Graphs," *Proc. Ann. Conf. COLT*, pp. 144-158, 2003.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," technical report, Stanford Digital Library Technologies Project, Stanford Univ., CA, Nov. 1998.
[18] G. Valentini, "True Path Rule Hierarchical Ensembles for Genome-Wide Gene Function Prediction," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 8, no. 3, pp. 832-847, May/June 2011.
[19] G. Valentini and N. Cesa-Bianchi, "Hcgene: A Software Tool to Support the Hierarchical Classification of Genes," *Bioinformatics,* vol. 24, no. 5, pp. 729-731, 2008.
[20] L. Pena-Castillo et al., "A Critical Assessment of Mus Musculus Gene Function Prediction Using Integrated Genomic Evidence," *Genome Biology,* vol. 9, article S1, 2008.
[21] L. Lovasz, "Random Walks on Graphs: A Survey," *Combinatorics, Paul Erdos Is Eighty,* vol. 2, pp. 1-46, 1993.
[22] M. Re and G. Valentini, "Random Walking on Functional Interaction Networks to Rank Genes Involved in Cancer," *Proc. Second Artificial Intelligence Applications in Biomedicine Workshop,* pp. 66-75, 2012.
[23] H. Lin, C. Lin, and R. Weng, "A Note on Platt's Probabilistic Outputs for Support Vector Machines," *Machine Learning,* vol. 68, pp. 267-276, 2007.
[24] C. Chang and C. Lin, "LIBSVM : A Library for Support Vector Machines," *ACM Trans. Intelligent Systems and Technology,* vol. 2, no. 3, pp. 27:1-27:27, 2011.
[25] N. Cesa-Bianchi, M. Re, and G. Valentini, "Synergy of Multi-Label Hierarchical Ensembles, Data Fusion, and Cost-Sensitive Methods for Gene Functional Inference," *Machine Learning,* vol. 88, no. 1, pp. 209-241, 2012.
[26] Y. Bengio, O. Delalleau, and N. Le Roux, "Label Propagation and Quadratic Criterion," *Semi-Supervised Learning,* pp. 193-216, MIT Press, 2006.
[27] M. Re and G. Valentini, "Cancer Module Genes Ranking Using Kernelized Score Functions," *BMC Bioinformatics,* vol. 13 (Suppl 14), article S3, 2012.
[28] M. Re and G. Valentini, "Large Scale Ranking and Repositioning of Drugs with Respect to Drugbank Therapeutic Categories," *Proc. Eighth Int'l Conf. Bioinformatics Research and Applications (ISBRA '12),* pp. 225-236, 2012.