Gene function prediction from synthetic lethality networks via ranking on demand

Christoph Lippert1,*, Zoubin Ghahramani2 and Karsten M. Borgwardt1

1Machine Learning & Computational Biology Research Group, Max Planck Institutes, Tübingen, Germany and
2Department of Engineering, University of Cambridge, Cambridge, UK

ABSTRACT

INTRODUCTION

Synthetic lethality—concept and mechanisms: Synthetic lethal interactions have received much attention in the genetics community over recent years. A synthetic lethal interaction refers to a pair of genes which interact in the following sense: while mutating each of the two genes individually does not cause lethality, a double mutation of both genes does show a lethal effect on the organism. Hence, synthetic lethality seems to indicate a compensatory effect of two genes, with one gene compensating for the deletion of the other and with lethal consequences only if both genes are deleted jointly.

The two main hypotheses to explain synthetic lethality between two genes A and B are the within- and between-pathway models (Boone et al., 2007; Ye et al., 2005b). The within-pathway hypothesis assumes that both genes A and B are part of the same pathway, and that the function of this pathway is diminished by their double mutation. The between-pathway hypothesis assumes that A and B act in parallel pathways that can compensate for defects in the other. These two main hypotheses can be extended in more complex models of synthetic lethality (e.g. see Ma et al., 2008), and algorithms for detecting pathways within genetic networks have been defined based on these hypotheses (Kelley and Ideker, 2005; Ma et al., 2008; Ulitsky and Shamir, 2007).

Goals and scope of this article: While previous studies focused on examining whether there is a correlation between genetic network structure and functional associations (Qi et al., 2008; Tong et al., 2004; Ye et al., 2005a, b) or in discovering pathways (Kelley and Ideker, 2005; Ma et al., 2008; Ulitsky and Shamir, 2007), our goal in this article is to define algorithmic machinery to rank all the genes in a genetic network based on their likelihood of belonging to a particular functional class, given a set of examples from this class. This ranking provides guidance in choosing promising targets for experimental function determination.

First, we study this problem of gene function prediction in yeast and for all three definitions of gene function provided by the GO (Ashburner et al., 2000): biological process, molecular function and cellular component. Second, we assess the prediction accuracy of our method in comparison with that of state-of-the-art methods. Third, we study whether combining predictions based on genetic
and congruence networks improves accuracy, or whether both types of networks provide redundant information.

Related work: the approaches closest to the one presented here are those which unify several different data sources, including genetic networks, into a joint prediction of gene function or co-complex membership (Deng et al., 2004; Lee et al., 2006; Letovsky and Kasif, 2003; Qiu and Noble, 2008; Tian et al., 2006). The approach presented here, however, differs conceptually from these earlier studies (Sharan et al., 2007), as we describe in the following. Ranking on demand ranks all genes in a given network based on their likelihood of belonging to a set of example genes from the same functional class.

Related work on methods for ranking on demand is scarce. The family of techniques for transductive semi-supervised learning does not apply here, as we are dealing with one class, while these are designed for discriminative learning on two or more classes (Chapelle et al., 2006; Tsuda et al., 2005). A notable exception are Bayesian sets (BS; Ghahramani and Heller, 2006) that use a model-based concept of a cluster (i.e. functional class of genes) to rank items (i.e. genes) using a score that evaluates the marginal probability that each item belongs to a cluster containing the query items. Another ranking on demand approach is RankProp (Weston et al., 2004), which is designed for detecting remote homologs in a network of protein sequence similarities by propagating this sequence similarity through the graph. For a special choice of kernel, the random walk kernel, our kernel-based ranking procedure can imitate this information propagation idea from RankProp. Ranking on demand has been performed before for loss-of-function RNAi phenotype prediction on biological networks (Lee et al., 2008). This approach is based upon the guilt by association (GBA) principle: the more edges between a node and the labeled examples, the higher its position in the ranking and the larger its probability of belonging to the same functional class. However, there are justified doubts that this successful technique will work on synthetic lethal interaction networks as well, because it only considers direct neighbors in a graph (nodes connected by an edge), and does not take the number of common neighbors into account, which has been shown to be correlated with functional similarity of two genes in genetic networks (Ye et al., 2005b). A central point in our experiments is to find out how this GBA approach compares with our novel approaches to function prediction. In our experiments, we compare our kernel-based technique with BS and GBA, and explore the use of diffusion and random walk kernels within our framework.

2 ALGORITHM

Our algorithm for function prediction on synthetic lethality networks focuses on the following scenario: given a network \( G \) of synthetic lethal interactions, one has experimentally determined a subset \( D_C \) of genes that exhibit a particular function. The goal is then to rank all remaining genes in the genetic network based on their likelihood of belonging to the same functional class. In algorithmic terms, the problem of ‘ranking of demand’ can be defined as follows.

**Problem statement 1 (Ranking on demand).** Given a network \( G \) with \( N \) nodes \( V = \{v_1, \ldots, v_N\} \) and a symmetric, undirected and unweighted adjacency matrix \( A \), and a subset of nodes \( D_C \) belonging to the same functional class \( C \). The problem of ranking on demand is to find a permutation \( \pi: N \to \{1, \ldots, N\} \) on the vertices in \( V \setminus D_C \) such that \( \pi(j) \geq \pi(i) \iff P(C|v_i) \geq P(C|v_j) \),

\[
\pi(i) \geq \pi(j) \iff P(C|v_j) > P(C|v_i), \quad (1)
\]

i.e. \( \pi \) ranks the nodes in \( V \setminus D_C \) based on their likelihood of belonging to \( C \) themselves.

A kernel approach to ranking on demand: it seems attractive to define a kernel-based approach for ranking on demand due to the efficiency of kernel machines in dealing with graph-structured data (Smola and Kondor, 2003) and in combining several different data sources (Schölkopf et al., 2004). The key observation that led to our novel algorithm was that BS, the Bayesian inference approach to ranking on demand (Ghahramani and Heller, 2006), computes a two-sample test statistic to score the likelihood of a gene (an item) to belong to a particular functional class (a cluster). A two-sample test tries to decide whether two samples, in our case \( x \) and \( D_C \), have been generated by the same distribution or not. In more detail, BS compute a Bayes factor that compares two hypotheses: the hypothesis \( H_1 \) that \( x \) and \( D_C \) were generated by the same distribution versus the hypothesis \( H_2 \) that they were generated by two different distributions.

The key idea in designing a kernel algorithm for ranking on demand is to compute a score that is based on a kernel-based two-sample test statistic rather than a Bayes factor. Such a kernel-based test statistic for the two-sample problem is the Maximum Mean Discrepancy (MMD) by Gretton et al. (2007).

**Ranking criterion:** the MMD is a criterion for measuring similarity between two distributions (population MMD) or between samples from two distributions (empirical MMD). The empirical MMD was shown to be equivalent to the distance between the means of two samples in a universal reproducing kernel Hilbert Space (Gretton et al., 2007).

**Theorem 1.** Let \( p \) and \( q \) be distributions and \( x = \{x_1, \ldots, x_m\} \) and \( y = \{y_1, \ldots, y_n\} \) be samples from \( p \) and \( q \) respectively. Let \( \Phi: X \to H \) be a mapping from input space to feature space, and let \( k: X \times X \to \mathbb{R} \) be the associated kernel function. Then the empirical MMD can be computed as

\[
\text{MMD}(H, X, Y) := \left\| \frac{1}{m_1} \sum_{i=1}^{m_1} \phi(x_i) - \frac{1}{m_2} \sum_{j=1}^{m_2} \phi(y_j) \right\|
\]
Algorithm 1: Kernel-based ranking on demand (kernelROD)

**background:** A graph $G$ with nodes $V$ and adjacency matrix $A$, a kernel function $k(\cdot,\cdot)$ between nodes in $G$.

**input:** a query $x_0 \in V$.

**for all** $x \in V \setminus \{x_0\}$ **do**

score($x$) $= -\frac{1}{|\mathcal{D}_C|} \sum_{k \in \mathcal{D}_C} \phi(x_i)$

**end for**

**output:** return elements of $V \setminus \{x_0\}$ sorted by descending score

$$
\text{score}(x) = -\frac{1}{|\mathcal{D}_C|} \sum_{k \in \mathcal{D}_C} \phi(x_i)\|^2

\frac{1}{|\mathcal{D}_C|} \sum_{k \in \mathcal{D}_C} \phi(x_i)\|^2
$$

The kernel-based estimator (2) follows from the fact that a kernel is an inner product between objects in feature space, i.e. $k(x, y) = \phi(x) \cdot \phi(y)$. The intuitive definition of a kernel is that it represents a similarity measure between objects $x$ and $y$.

Using this test statistic, Gretton et al. (2007) defined two-sample tests whose key concept can be summarized as follows: the larger the empirical MMD between $X$ and $Y$, the larger the probability that $p$ and $q$ are not the same. Hence $-\text{MMD}$ (negative MMD), and equivalently $-\text{MMD}^2$, shows exactly the same behavior as the score in BS: the larger $-\text{MMD}$, the more likely it is that the two samples were generated by the same distribution.

**Computation:** to determine the ranking score of a gene $x$, we compute the squared distance between $x$ and the set of examples $\{x_i\}$ in feature space,

$$
|\phi(x)| - \frac{1}{|\mathcal{D}_C|} \sum_{k \in \mathcal{D}_C} |\phi(x_i)|^2.
$$

We then multiply this score by $-1$, as we want scores to be the largest for $x$ that are likely to fit into the set of examples $\mathcal{D}_C$. In terms of kernels, this score(s) can be rewritten as

$$
-k(x, x) - \frac{2}{|\mathcal{D}_C|} \sum_{i \neq j} k(x, x_i) + \frac{1}{|\mathcal{D}_C|} \sum_{i \neq j} k(x_i, x_j)
$$

We can drop the third term from (4) as it does not depend on $x$ and hence does not affect the ranking. Hence, we have to compute $|\mathcal{D}_C|+1$ kernel values for each of the $N$ objects in our dataset to perform kernel-based ranking on demand. The entire procedure is summarized in Algorithm 1.

**Kernel design for function prediction on genetic networks:** the choice of kernel $k(\cdot,\cdot)$, which can be thought of as a similarity measure between two genes, is crucial for applying kernel-based ranking on demand to function prediction on genetic networks. We used different kernel functions in our experiments and provide here biological interpretations of how they measure similarity between two genes in a genetic network.

**Random walk kernels:** as synthetic lethal interactions have been shown to occur directly between functionally related genes (Tong et al., 2004), we designed a kernel that captures direct neighborhood between genes. The $p$-step random walk kernel exhibits this property (Smola and Kondor, 2003). It is defined as $K_{rw} = (\alpha I - \mathbf{L})^p$, where $\alpha$ is a scalar, $I$ is the identity matrix and $\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \alpha \mathbf{I})\mathbf{D}^{1/2}$ with $\mathbf{D}(i,i) = \sum_j A(i,j)$ the normalized graph Laplacian. This kernel measures similarity between two nodes $x_i$ and $x_j$ in a graph in terms of the number of random walks from $i$ to $j$ in $p$ steps. The term $\alpha$ allows the random walk to remain in the same node in the next step with some probability (we will refer to $\alpha$ as the restart parameter). The larger $\alpha$, the larger this probability.

For $p=1$ the random walk only takes one step, which means it reaches only the direct neighbors of a node and hence the associated kernel deems direct neighbors in the graph similar, which fits the observation by Tong et al. (2004). For $p=2$ one may visit nodes that are direct or indirect neighbors of a given node, which means that the corresponding kernel deems direct neighbors and nodes with similar neighborhoods similar, combining both the observations by Tong et al. (2004) and Ye et al. (2005b). For $p > 2$, the kernel leads to a diffusion-like exploration of the graph and resembles the graph diffusion kernel by Qi et al. (2008), whose kernel values have been shown to correlate with similarity in gene function, and the information propagation approach RankProp (Weston et al., 2004).

**Diffusion kernels:** for comparison, we also considered the diffusion kernel as defined by Kondor and Lafferty (2002) and used by Qi et al. (2008), which is defined as $K_{diffusion} = \exp(-\beta \mathbf{L})$, where $\beta$ is a scalar, $\mathbf{L}$ the normalized graph Laplacian as defined above and $\exp$ is the matrix exponentiation. This diffusion kernel deems nodes similar if they are in close proximity in the graph and connected by several paths. Qi et al. (2008) summed even- and odd-length path separately in their approach.

### 3 METHODS AND MATERIALS

**Dataset:** the BioGRID repository (Stark et al., 2006), version 2.0.51, was parsed for a total of 11.998 directed synthetic lethal interaction pairs between $N=2579$ *Saccharomyces cerevisiae* genes. By correcting for double counting bait-hit pairs, this results in a network of 10.791 undirected synthetic lethal interactions. Functional annotations for these genes were obtained from the *Saccharomyces Genome Database* (Cherry et al., 1998) and follow the GO nomenclature (Ashburner et al., 2000). Functional categories in the GO follow a general-to-specific ordering between terms of different depths in the GO graph. The annotations are transitive, meaning that if a gene is annotated with a term, this includes all terms that are more general. We make this explicit by annotating each gene by its GO terms and all GO terms that are more general than the annotated ones. We repeat this procedure for each of the three main GO branches: molecular function, biological process and cellular component.

**Congruence network:** in addition to the network of direct synthetic lethal interactions, we also computed a genetic congruence network based on the undirected synthetic lethal interactions as described by Ye et al. (2005a, b). They define a genetic congruence score of two nodes $x_i$ and $x_j$, as the negative logarithm of the probability that the number of shared true neighbors, $|N_{x_i} \cap N_{x_j}|$, in the graph is equal to or higher than the observed number of shared neighbors, $|N_{x_i, x_j}| = |N_{x_i} \cap N_{x_j}|$, of $x_i$ and $x_j$, that is

$$
\text{congruence}(x_i, x_j) = -\log_{10} P\left(|N_{x_i} \cap N_{x_j}| \geq |N_{x_i, x_j}|\right)
$$

where $P$ is hypergeometrically distributed:

$$
P\left(|N_{x_i} \cap N_{x_j}| \geq |N_{x_i, x_j}|\right) = \sum_{l=|N_{x_i, x_j}|}^{\min(|N_{x_i}|, |N_{x_j}|)} \binom{|N_{x_i}|}{l} \binom{|N_{x_j}|}{|N_{x_i, x_j}| - l} / \binom{|N_{x_i}| + |N_{x_j}| - |N_{x_i, x_j}|}{l}.
$$
In order to retain only significant edges, all edges \((x_i, x_j)\) with
\[ P(x_i, x_j) > \frac{1}{\sqrt{N}} \]
were deleted, where the denominator equals the total number of possible edges in the congruence graph and corrects for multiple testing.

**Experimental setting:** Gene function prediction experiments were performed on each of the three main GO branches for GO terms of GO depth (or GO level) 1, 2, 3 and 4, where depth is measured as the shortest distance to the root of the GO hierarchy. Depth 1 comprises the most general terms. For each of these terms, i.e. for each node on this level of the GO, we performed 5-fold stratified cross-validation (SCV). The genes were randomly divided into five sets of equal size and equal proportions of members and non-members of a functional class. Then in each of five iterations, four sets were used for training and the remaining set was used as the query database. This ensures that each gene was used for testing exactly once.

**Assumptions:** Our experiment might suffer from an ascertainment bias (Supplementary Section 2.1), as bait genes from the same functional classes have a higher probability of being connected by an edge, and might artificially boost prediction accuracy if they appear in both training and test sets. To avoid this effect, we make sure that bait genes with the same functional class are either all in the training dataset or all in the test set.

**Comparison methods and parameter optimization:** We run kernelROD using a random kernel on the synthetic interaction network (synth), the congruence network (cong) and using the sum of both kernels (sum). We conduct the same three experiments for the diffusion kernel. We compare kernelROD with two ranking on demand approaches, GBA and BS, and to a two-class support vector machine (SVM) and a one-class SVM (SVM1) for a full description—including the choice of negative examples for the SVM—see Supplementary Section 1. We set the parameters for all methods by SCV on the training set. Parameters include the number of steps in the \(p\)-step-random walk kernel \((p \in \{1, ..., 4\})\), the restart parameter \(a\) which controls the influence of shorter random walks \((a \in \{2^1, ..., 2^10\})\), the diffusion parameter \(\beta\) (except \(\beta \in \{1, ..., 10\}\)), weights for the convex combination \((\alpha \in \{0, ..., 9\})\) and in the SVM a parameter defining the size of outlier quantile \((\epsilon \in \{0^0.01, ..., 0^1\})\).

**Evaluation criteria:** In order to assess the quality of the rankings, we computed the area under the receiver operator characteristic curve (AUC) and the area under the receiver operator characteristic for up to 50 false positives (AUC50) for each ranking. The AUC score reflects the probability that a randomly drawn positive example is ranked higher than a randomly drawn negative example. AUC50 reflects the probability that a randomly chosen positive example is ranked higher than one random example out of the 50 highest ranked negative examples. The latter criterion is more appropriate in a setting in which one is only interested in a small set of high confidence predictions that can be further evaluated experimentally. We report the AUC results in the main paper and the AUC50 results in Supplementary Material.

**Results:** To quantify the performance of a method across tasks, we compute the \(L_2\)-distance between the results of this method and the best method on each task. A low \(L_2\)-distance indicates that a method is always close to the best performing one on each task.

**4 RESULTS**

We perform gene function prediction via cross-validation on a synthetic lethality network from yeast using kernelROD and several comparison methods: BS, GBA and SVM1 and two-class SVM (see Table 1 and Fig. 2, and Supplementary Tables 2 and 3 and Supplementary Figs 6 and 7 for all results).

**All methods are significantly better than random:** as a first check, we examined whether the results we had obtained with kernelROD, GBA and BS were better than random. For this purpose, we generated 1000 random rankings on each task and computed the AUC values for these random rankings. We then used this approximately null distribution of AUC values from random rankings to compute a \(P\)-value of the results of our methods. If the methods worked only as good as a random ranker, one would expect to obtain a distribution over \(P\)-values that is close to uniform. As can be seen from Figure 3 and Supplementary Figure 8, the distribution of \(P\)-values of all methods is skewed toward small \(P\)-values, and highly significantly different from that of a random ranker.

**KernelROD improves GO term prediction by ranking on demand:** We compare kernelROD based on the combination of a random walk kernel on both the synthetic and the congruence network gives the best results across all GO branches and GO levels, that is, the lowest \(L_2\) distance to the best method on each task in terms of AUC.

The improvement achieved by kernelROD is largest when looking at AUC50 scores (Supplementary Fig. 7): Here kernelROD with a random walk kernel achieves the lowest \(L_2\) distance on all three networks, that is, the synthetic lethal interaction network, the congruence graph and on both. In terms of AUC, BS perform similarly well as kernelROD on the synthetic interaction network. This indicates that the top-ranked predictions of kernelROD tend to be more accurate than those of BS.

**Indirect interactions improve rankings:** to further understand how kernelROD leads to improved AUC scores, we examine the impact of the number of steps taken in its random walk kernel on its AUC scores. We compare kernelROD with a one-step random walk kernel and kernel with a two-step random walk kernel on the synthetic interaction network, for all GO branches and GO depths from 1 to 4 (Table 1). The two-step random walk kernel gives significantly better results than the one-step random walk kernel, reaching a higher AUC value in 10 out of 12 settings \((P = 0.0032\); Binomial distribution with \(n = 12, P = 0.5\)). This indicates that it is useful to consider more than just direct interactions for gene ranking.

**Combination of networks improves prediction of GO terms:** in 8 out of 12 experiments, the best AUC result is achieved using kernelROD with a sum of kernels on both the synthetic and the congruence network. In three out of the other four experiments, a random walk kernel on the synthetic network or the congruence network yields the highest AUC score. These results indicate that both data integration is beneficial for function prediction and that direct synthetic lethal interactions and shared neighbors in the synthetic lethal interaction network are both indicative of joint function.

**Performance varies for different GO levels and branches:** next we examine whether our prediction results differ significantly between different GO branches and for different levels in the GO hierarchy (see Table 1 and the plots of AUC versus GO level in Fig. 2). In Figure 2, we compare kernelROD to GBA and BS.

The AUC results across GO branches do not differ significantly, they are all clustered in the range from 0.55 to 0.65. On GO molecular function (Fig. 2a), results on the congruence graph improve with GO depth, whereas the methods on the synthetic interaction graph show no uniform trend. On GO molecular function (Fig. 2b), results of all methods on the synthetic network deteriorate from GO level 1 to 3, whereas the methods on the congruence graph improve simultaneously. On GO biological process (Fig. 2c), we observe an increase in AUC for deeper levels of the GO hierarchy for all methods except for GBA on the synthetic interaction network. This indicates that across all GO branches, the gene function classes tend to be the more clustered in the congruence network, the deeper
C.Lippert et al.

Table 1. Results for GO term prediction using genetic interaction networks: Average AUC and standard deviations for GO term prediction on three GO branches

<table>
<thead>
<tr>
<th>GO level</th>
<th>Prokaryotic Shigella ( ++ )</th>
<th>GO null model</th>
<th>GO background p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average AUC</td>
<td>Standard deviation</td>
<td>p-value</td>
</tr>
<tr>
<td>depth 1</td>
<td>0.619 ± 0.038</td>
<td>0.618 ± 0.038</td>
<td>0.617 ± 0.038</td>
</tr>
<tr>
<td>depth 2</td>
<td>0.637 ± 0.038</td>
<td>0.634 ± 0.036</td>
<td>0.638 ± 0.038</td>
</tr>
<tr>
<td>depth 3</td>
<td>0.638 ± 0.038</td>
<td>0.635 ± 0.036</td>
<td>0.639 ± 0.039</td>
</tr>
<tr>
<td>depth 4</td>
<td>0.635 ± 0.038</td>
<td>0.632 ± 0.036</td>
<td>0.635 ± 0.039</td>
</tr>
</tbody>
</table>

The GO level we look at. On the synthetic interaction network, we could not detect a similar trend. Choice of kernel: finally, we assess to which degree the choice of kernel affects our gene prediction performance (Table 1). In terms of $L^2$-distance to the best performing method, the random walk kernel is better than the diffusion kernel on the synthetic network, the congruence network and their combination.

5 DISCUSSION

In this article, we have presented kernelROD, a novel ranking approach for gene function prediction from synthetic lethality networks. In function prediction in yeast, we observe that our kernel-based approach kernelROD outperforms state-of-the-art methods and that a combined random walk kernel on genetic networks and on congruence networks (Ye et al., 2005a) often improves, and never harms prediction accuracy. Considering indirect interactions (walks of length 2) in the synthetic interaction network results in improved rankings compared with considering only direct interactions.

We could confirm that congruence networks are useful for function prediction from genetic networks, as reported by Ye et al. (2005a, b). We could also confirm that diffusion or random walk-based kernels are a promising approach to function prediction on genetic networks, as reported by Qi et al. (2008). Furthermore, we also established that random walk kernels achieve even better performance in the synthetic network.
results on congruence networks. Our best performing kernel, both on genetic and on congruence networks, is the random walk kernel, and it achieves its best results when integrating information both from the synthetic and the congruence network.

We make the following interesting observations which will motivate future research efforts: our ranking on demand approach outperforms discriminative SVMs in function prediction, reaching better results in terms of L2 on the congruence graph and the combined graph, and similar ones on the synthetic graph. This indicates that within-class similarity seems to be more important than between-class dissimilarity for our task: genes with the same function seem to be in close proximity in the network (within-class similarity), but there are also connected genes with different functions (lack of between-class dissimilarity). The within-class similarity allows our ranking method to obtain good results, because genes with similar function will appear high in the ranking. The lack of between-class dissimilarity is likely to be the cause for the worse performance of discriminative methods, because functional classes cannot be distinguished based on proximity in the network. At the same time, this lack of between-class dissimilarity also keeps our method from reaching higher accuracy levels, as it leads to genes from other functional classes being ranked high. A topic of future research will be to add further data sources to our prediction system that increase the between-class dissimilarity in our feature representation of genes. Due to the closure properties of kernels, our kernel-based ranking algorithm will be a convenient framework for this task of data integration.

Conflict of Interest: none declared.

REFERENCES


