BIOINFORMATICS

Vol. 00 no. 00 2007
Pages 1–8

Kernel-based data fusion for gene prioritization

Tijl De Bie [a,b], Léon-Charles Tranchevent [c], Liesbeth M. M. van Oeffelen [c], Yves Moreau [c]

[a]Dept. of Engineering Mathematics, University of Bristol, University Walk, BS8 1TR, Bristol, UK
[b]OKP Research Group, Katholieke Universiteit Leuven, Tiensestraat 102, 3000 Leuven, Belgium
[c]ESAT-SCD, Kathlieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium

ABSTRACT

Motivation: Hunting disease genes is a problem of primary importance in biomedical research. Biologists usually approach this problem in two steps: first a set of candidate genes is identified using traditional positional cloning or high-throughput genomics techniques; second, these genes are further investigated and validated in the wet lab, one by one. To speed up discovery and limit the number of costly wet lab experiments, biologist must test the candidate genes starting with the most probable candidates. So far, biologists have relied on literature studies, extensive queries to multiple databases, and hunches about expected properties of the disease gene to determine such an ordering. Recently, we have introduced the data mining tool ENDEAVOUR (Aerts *et al.*, 2006), which performs this task automatically by relying on different genomewide data sources, such as Gene Ontology, literature, microarray, sequence, and more.

Results: In this paper, we present a novel kernel method that operates in the same setting: based on a number of different views on a set of training genes, a prioritization of test genes is obtained. We furthermore provide a thorough learning theoretical analysis of the method's guaranteed performance. Finally, we apply the method to the disease data sets on which ENDEAVOUR (Aerts *et al.*, 2006) has been benchmarked, and report a considerable improvement in empirical performance.

Availability: The MATLAB code used in the empirical results will be made publicly available.

Contact: tijl.debie@gmail.com, yves.moreau@esat.kuleuven.be

1 INTRODUCTION

Identifying genes whose disruption causes congenital or acquired disease in humans is a major goal of genetics and molecular biology, both towards diagnosis and understanding the biology of disease processes. These genes are called *disease genes*—an example being the BRCA1 gene whose mutation is responsible for cases of familial breast cancer. Several biological strategies are available to identify disease genes. Positional cloning strategies aim at identifying the position of the gene on its chromosome (linkage analysis, linkage disequilibrium, association studies, study of chromosomal aberrations). Most of the time these studies can only restrict the location of the disease gene to a region containing tens to hundreds of *candidate genes*. High-throughput genomic studies (microarray analysis, proteomics, and so on) often consider biological samples from patients or animal models and try to identify which key genes or proteins are disrupted in the disease process. Again, these strategies often deliver long laundry lists of hundreds of candidate genes.

In both cases, the candidate genes need to be further investigated to identify the disease causing genes. Because this work is time consuming and expensive, biologists must prioritize the genes from most to least promising when carrying out the validation process—this is called *gene prioritization*.

A main strategy to prioritize candidate genes is to compare the candidate genes (called here the *test genes*) to genes already known to cause the same disease or closely related disease processes (called here the *training genes*). Hence, the problem faced by the biologist to determine the implicated gene among the test genes can potentially be simplified, by concentrating on those test genes that are in some sense similar to the training genes.

With the advent of high-throughput technologies, many sources of information, or *views* on genes may be useful and relevant in defining what is 'similar'. Therefore, this task has become extremely challenging for biologists. For this reason, we have recently developed the tool ENDEAVOUR (Aerts *et al.*, 2006). It makes use of statistics to compute a ranking of test genes according to their similarity to the training genes, and this once on each of a number of data sources. In a subsequent step, these rankings are integrated into a single ranking by making use of order statistics.

1.1 Formal problem setting

In the current paper, we formulate the problem in machine learning terms, and we develop a kernel-based method to solve it (see Shawe-Taylor and Cristianini (2004) for an introduction to kernel methods). As for the formalization, several avenues can be followed. First, one may cast it into the *classification* framework, regarding the training genes as belonging to the positive class, and the rest of the genome to the negative. However, the assumption that the rest of the genome contains only negatives is false, even though in gene hunting the proportion of positives is usually small. This means that label noise is unavoidable, which is detrimental from a robustness point of view. Moreover, the set of positives is usually extremely small (a few to a couple tens) and is drawn with major biases from the underlying positive class, which compromises uniform generalization performance over the whole space. On the other hand, the large size of the negative training set would pose computational challenges to data fusion approaches.

The second possible approach formalizes the problem as *novelty detection*, where one tries to model (the support of the distribution of) the training genes only. Several approaches to novelty detection have been described in literature (Tax and Duin, 1999; Schölkopf *et al.*, 2001), and relations between them have been established. One approach tightly fits a hypersphere around a vector representation

© Oxford University Press 2007.

of the data, and considers the inner volume of the hypersphere as the support of the distribution. Another approach finds a hyperplane separating the positive data from the origin.

The approach proposed in this paper is reminiscent mostly of the latter: find a hyperplane that separates the vector representations of the disease genes from the origin with the largest possible margin, and consider a gene more likely to be a disease gene if it lies farther in the direction of this hyperplane. However, we have an additional problem to be dealt with: while all methods described so far make use of just one view on the data, our method should be capable of taking into account several different views on the genes.

1.2 Data fusion by learning the kernel

The main source of inspiration for our algorithmic and theoretical contributions is in our previous work in Lanckriet *et al.* (2004b), and in Lanckriet *et al.* (2004c,a); Herrmann and Bousquet (2003). They describe a methodology for learning the kernel matrix relying on a quadratically constrained linear program (QCLP) for classification in a transduction setting. Both approaches rely on strong statistical foundations and performance guarantees are provided. A number of recent publications have carried this work further, generalizing this approach towards other problems besides classification (Ong *et al.*, 2005), working on algorithmic improvements to reduce time and memory requirements (Bach *et al.*, 2004), or contributing to both these aspects (Sonnenburg *et al.*, 2006). Still, thus far data fusion approaches to novelty detection have remained understudied. To our knowledge, a statistical study of the problem is still lacking. Furthermore, empirical studies have remained limited to the method in Ong *et al.* (2005), which is based on the elegant framework of kernel-learning with hyperkernels, but which is computationally extremely challenging as it relies on a semi-definite program with a number of variables that is quadratic in the training set size.

1.3 Results

We present an approach for gene prioritization based on a novel kernel-based algorithm capable of integrating various sources of information in a natural way. Our approach leads to fast algorithms relying on a QCLP, and we show how it is explicitly guided by a rigorous statistical study. We demonstrate it on a large number of disease gene hunting problems, outperforming ENDEAVOUR (Aerts *et al.*, 2006), which is the first tool to combine many data sources for gene prioritization and to have provided new candidate genes that have been successfully biologically validated.

2 DATA FUSION, KERNEL COMBINATIONS, AND NOVELTY DETECTION

We first discuss a variant of a well-known kernel method for novelty detection, which makes use of a single view on the data only (Schölkopf *et al.*, 2001). Let us assume a gene x has an associated vector representation $\mathbf{x}$. Then, this method finds a hyperplane parameterized by a unit norm weight vector $\mathbf{w}$, defined by the equality $f(x) \triangleq \mathbf{x}'\mathbf{w} = M$, which separates all training genes from the origin.

Then, for a *test* gene x, the function f measures its distance from the origin in the direction of the hyperplane, and can be used to prioritize the genes: the larger $f(x)$, the higher gene x in the prioritization. See the lower part of Figure 1 for a schematic clarification.

Subsequently, we discuss how different views on the genes can be integrated naturally and efficiently by convexly combining the kernels of each of these data sources. The statistical study which theoretically supports the use of the function f as a way to prioritize is left to Section 3.

2.1 Novelty detection

Let us represent the vector representation of the set of training genes by a matrix $\mathbf{X}$, with the ith row of $\mathbf{X}$ containing the feature vector $\mathbf{x}_i$ of the ith training gene. As above, we define a function f of gene x as $f(x) \triangleq \mathbf{x}'\mathbf{w}$. Then, we search for the weight vector $\mathbf{w}$ with $\|\mathbf{w}\|^2 \leq 1$ such that for all genes x_i in the training set the function $f(x_i)$ is larger than a margin M, with M as large as possible (i.e., it searches for a hyperplane parameterized by $\mathbf{w}$, such that all training data lie on one side of the hyperplane and the perpendicular distance M between the origin and the hyperplane is maximized). Formally, this leads to the optimization problem,

$$\max_{M,\mathbf{w}} p(M) = M \quad \text{s.t.} \quad \mathbf{w}'\mathbf{w} \leq 1, \;\; f(x_i) \triangleq \mathbf{x}_i'\mathbf{w} \geq M \; (\forall i).$$

The dual of this (convex) optimization problem can be given by

$$\min_{\boldsymbol{\alpha}} d(\boldsymbol{\alpha}) = \sqrt{\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\boldsymbol{\alpha}} \quad \text{s.t.} \quad \alpha_i \geq 0 \; (\forall i), \;\; \mathbf{1}'\boldsymbol{\alpha} = 1. \; (1)$$

Thanks to strong duality, the primal and dual optima (achieved for $M^\star$ and $\boldsymbol{\alpha}^\star$) are equal to each other: $p(M^\star) = d(\boldsymbol{\alpha}^\star)$. Furthermore, duality relations show that the optimal value of the weight vector can be expressed in terms of the dual variables as $\mathbf{w}^\star = \mathbf{X}'\boldsymbol{\alpha}^\star / \sqrt{\boldsymbol{\alpha}^{\star\prime}\mathbf{X}\mathbf{X}'\boldsymbol{\alpha}^\star}$. Note that the square root is a monotonic function and can hence be ignored in the objective of the dual optimization problem (1).

It is a crucial recurring fact in kernel methods that the dual formulation can be written solely in terms of inner products between feature vectors $\mathbf{x}_i$. Indeed, the matrix $\mathbf{X}\mathbf{X}'$ contains the inner product $\mathbf{x}_i'\mathbf{x}_j$ on its ith row and jth column, and we denote $\mathbf{X}\mathbf{X}' = \mathbf{K}$, the so-called kernel matrix. As a consequence, the actual representation $\mathbf{x}$ of gene x does not need to be known, as long as the inner product between any pair of genes in this representation is specified by a kernel function $k(x_i, x_j)$.

Equally importantly, instead of the representation $\mathbf{x}'\mathbf{w}$ of the prioritization function $f(x)$, we can use the following equivalent dual formulation, relying on kernel evaluations only:

$$f(x) = \frac{1}{\sqrt{\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}}} \sum_{i=1}^{n} \alpha_i k(x, x_i) \quad (2)$$

2.2 Data fusion

In the method discussed above only a single kernel function k and corresponding kernel matrix $\mathbf{K}$ on the training set are given. However, in our application m different kernel functions k_j and kernel matrices $\mathbf{K}_j$ are available, each of which is based on a certain representation or view on the genes. The availability of these different views leaves us the freedom to design the kernel matrix such that the penalized margin is maximized. The challenge is to exploit this in a statistically and algorithmically sound way.

2.2.1 Averaging the kernels A first and trivial approach is to combine the kernel matrices $\mathbf{K}_j$ by simply computing an average

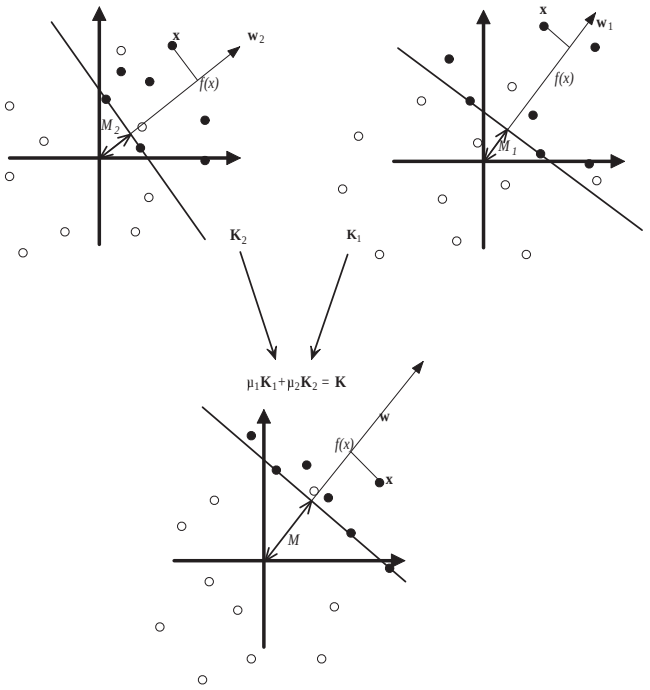


Fig. 1. Schematic representation of the hyperplane separating the (positive) training genes (filled circles) from the origin, along with the negative genes (open circles). Combining two kernels in an optimal way leads to a new space (right figure) where the distance of the positive genes to the origin is larger.

of all kernels normalised by positive constants $\beta_j > 0$:

$$\mathbf{K} = \frac{1}{m}\sum_{j=1}^{m}\frac{\mathbf{K}_j}{\beta_j}.$$

The choice of the constants β_j may be arbitrary and at the user's discretion. However, here we suggest the following simple choice. The concern to address using these weights is that different kernels may have different scales, such that their importance in the linear combination may be overly small or large. In order to correct for this, we choose β_j to be proportional to the trace of $\mathbf{K}_j$. Essentially, this value for β_j is chosen in order to make the kernels comparable to each other. Statistical arguments for this choice will be given in Section 3. This is the first and most simple data fusion approach we propose here, and we compare it with ENDEAVOUR in the experimental Section. There we will see that it significantly outperforms ENDEAVOUR.

Note that, if appropriate, one could hand-tune the weights of the kernels in the kernel combination based on expert knowledge. I.e., instead of using $\mathbf{K}$ as defined above, one can additionally weigh the kernels with a hand-tuned weight $\mu_j \geq 0$:

$$\mathbf{K} \quad = \quad \sum_{j=1}^{m}\mu_j\frac{\mathbf{K}_j}{\beta_j}$$

A kernel for a data source deemed relevant for a certain disease could then be given a larger μ_j, and hence a larger vote in the linear combination. Since this requires expert knowledge on individual diseases, we choose not to follow this avenue in the present paper.

Nevertheless, as we will see below, there are other ways to tune the weights μ_j automatically in a data-dependent but agnostic way (i.e., without taking disease characteristics into account). The goal of these approaches is to reduce the influence of noisy (or irrelevant) information sources, and of double counting of information that is present in more than one of the information sources. Hence, as we will see in the experimental Section, in cases where noise influences are large or where redundant information is provided, such methods may perform better. Let us discuss these methods now in greater detail.

2.2.2 Optimal convex kernel combination As a first method to achieve automatic tuning of the kernel weights, and in the same spirit as Lanckriet *et al.* (2004c,b), we propose to convexly combine the m different kernel matrices $\mathbf{K}_j$ so as to maximize the margin M between the data points and the origin. More specifically, with β_j positive constants, we choose the 'summarizing kernel' $\mathbf{K}$ as the one from the set $\mathcal{K} = \left\{\sum_j \mu_j(\mathbf{K}_j/\beta_j) : \boldsymbol{\mu}'\mathbf{1} = 1, \boldsymbol{\mu} \geq \mathbf{0}\right\}$ that maximizes the optimum of optimization problem (1):

$$\max_{\mathbf{K}}\min_{\boldsymbol{\alpha}}\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad \text{s.t.} \quad \alpha_i \geq 0\ (\forall i),\ \mathbf{1}'\boldsymbol{\alpha} = 1,$$
$$\mathbf{K} \in \left\{\sum_j \mu_j\frac{\mathbf{K}_j}{\beta_j} : \boldsymbol{\mu}'\mathbf{1} = 1, \boldsymbol{\mu} \geq \mathbf{0}\right\}.$$

which can be shown to be equivalent to

$$\min_{t,\boldsymbol{\alpha}} t \quad \text{s.t.} \quad \alpha_i \geq 0\ (\forall i),\ \mathbf{1}'\boldsymbol{\alpha} = 1, \qquad (3)$$
$$t \geq \boldsymbol{\alpha}'\frac{\mathbf{K}_j}{\beta_j}\boldsymbol{\alpha}\ (\forall j).$$

This is a QCLP problem that is efficiently solvable using general purpose software.

2.2.3 A regularized intermediate solution In some cases, the freedom allowed to the optimization problem in this way may be so large that overfitting occurs, resulting in a bad generalization performance. Therefore, we propose an approach intermediate to the simple averaging of the kernels and their optimal combination using convex optimization as explained above. This can be achieved by specifying a lower bound $0 < \mu_{\min} \leq \frac{1}{m}$ on μ_j in the specification of $\mathcal{K}$, i.e. $\mathcal{K} = \left\{\sum_j \mu_j(\mathbf{K}_j/\beta_j) : \boldsymbol{\mu}'\mathbf{1} = 1, \boldsymbol{\mu} \geq \mathbf{1}\mu_{\min}\right\}$. Increasing $\mu_{\min}$ reduces the size of $\mathcal{K}$, which amounts to regularizing the problem, and hence reduces the risk of overfitting. For $\mu_{\min} = \frac{1}{m}$, the simple method that computes the average of the normalised kernels is obtained.

2.2.4 A unifying method Interestingly, the last method contains the first and the second as a special case. Indeed, by taking $\mu_{\min} = \frac{1}{m}$, the first method is obtained (the uniform, equal weighting of the kernels normalized by β_j). By taking $\mu_{\min} = 0$, the second method is obtained. Therefore, in the remainder of the paper, we can refer to the different methods by choosing the value for $\mu_{\min}$.

2.2.5 The function f For all these data fusion approaches, the evaluation of the function $f(x) = \mathbf{x}'\mathbf{w}$ can now be expressed in

terms of the β_j, μ_j, and α_i, as

$$f(x) = \frac{1}{\sqrt{\boldsymbol{\alpha}' \mathbf{K} \boldsymbol{\alpha}}} \sum_{i=1}^{n} \alpha_i \left[\sum_{j=1}^{k} \mu_j \frac{k_j(x, x_i)}{\beta_j} \right]. \quad (4)$$

In the next Section we will provide a rigorous theoretical evidence motivating these approaches, which will furthermore point us to good possible choices for values of β_j.

3 STATISTICAL GUARANTEES AND MOTIVATIONS

While earlier approaches to novelty detection based on different kernel-views exist (e.g. Ong *et al.* (2005); Sonnenburg *et al.* (2006)), very few experimental results have been reported (only in Ong *et al.* (2005), and only in an informal qualitative way). Furthermore, to our knowledge, none of these were based on statistical foundations. Still, it is clear that in our above formulation, certain parameters need to be chosen, in particular the values of β_j. Additionally, we will see that other choices need to be made, regarding normalizations and centering of the data. In order to make these choices in a principled way, a statistical study is indispensable. Can we choose the parameters such that the number of false positives and false negatives are controlled? Here we present our analysis which achieves that goal. As in Lanckriet *et al.* (2004c), our study relies on the use of Rademacher complexities.

In our discussion, we focus on the unregularized version, where $\mu_{\min} = 0$. All results can quite easily be adapted to the regularized case, and we will point out consequences of this regularization where relevant.

3.1 Controlling the number of false negatives

We will assume that the training genes are sampled *iid* from the distribution of the positive class (the class of disease genes). Admittedly, this is not exactly true, but it is probably a good approximation given the large number of genes in the genome. We derive a bound for the probability that $f(x) \leq M - \gamma$ for an *iid* test gene x from the positive class. This is the probability to make an error of a certain magnitude in evaluating whether the test point x is a novelty or not, i.e., the probability that a test point from the positive distribution lies a distance γ at the negative side of the hyperplane. We will see that this probability quickly decreases for increasing γ, which means that the probability that for a true disease gene x the function $f(x)$ can be expected to be large. Let us now state the Theorem; for brevity, we provide its proof in a Supplement.

THEOREM 1. *Given a set X of n genes x_i sampled* iid *from an unknown distribution $\mathcal{D}$. Let $\lambda(\mathbf{K}_j)$ denote the largest eigenvalue of $\mathbf{K}_j$. Then, for any $M, \gamma \in \Re_+$ and for any $\delta \in (0,1)$, with probability of at least $1-\delta$ the following holds for the function f in Equation (4) as found by optimization problem (3):*

$$P_{\mathcal{D}}\left(f(x) \leq M - \gamma\right) \leq \frac{1}{n\gamma} C(\mathbf{K}_j, \beta_j) + \frac{1}{\sqrt{n}} \sqrt{2 \ln \frac{2}{\delta}}.$$

where $C(\mathbf{K}_j, \beta_j)$ is a complexity term equal to

$$C(\mathbf{K}_j, \beta_j) = 4 \sqrt{\min \left[n \max_j \frac{\lambda(\mathbf{K}_j)}{\beta_j}, \sum_{j=1}^{m} \frac{\operatorname{trace}(\mathbf{K}_j)}{\beta_j} \right]}.$$

While the Theorem holds for any specific value of M, it does not hold uniformly over M. However, as outlined in Lanckriet *et al.* (2004c) and references therein, the theorem can easily be adapted to yield uniform bounds over M. Note that the complexity term involves the square root of the minimum of two quantities, the first of which grows quadratically with n, while the second grows only proportionally with n as the training set grows. Hence, asymptotically, the bound decreases as $1/\sqrt{n}$.

A similar Theorem holds for the regularized version, where a lower bound is imposed on the values of μ_j. Interestingly, in that case the bound is generally tighter, as the Rademacher complexity of the smaller function class is smaller. We omit the Theorem here for readability.

In some sense, Theorem 1 bounds the false negative probability by a value that depends on the value of β_j relative to $\lambda(\mathbf{K}_j)$ and to $\operatorname{trace}(\mathbf{K}_j)$, which should therefore be kept under control. We will now show how an approach to controlling the false positives motivates a choice for the β_j that ensures this requirement is fulfilled in practical situations.[1]

3.2 Controlling the number of false positives

Given the full genome, and the probability of false negatives being bounded, we could control the number of false positives by bounding the total number of positives. Recall that the algorithm tries to separate the data as far from the origin as possible. With this in mind, we suggest the following strategy. First, use centered kernel matrices (or kernel functions), i.e. in gene hunting the kernels are defined by centering the kernels on the full genome. And second, equate the value of β_j to the trace of the jth centered *genome-wide* kernel matrix divided by the total number of genes in the genome. Such a choice for β_j can be expected to yield tight capacity terms in Theorem 1 in practice, assuming that positive and negative genes are not *too* different in norm and in distribution (a reasonable assumption, as it appears so in practice and it is the fact motivating this work).

This strategy ensures that the trace of the kernel matrix obtained by linearly combining all genome-wide kernel matrices weighted by β_j has a trace equal to the number of genes in the genome, such that the norm $k(x,x)$ of a gene is equal to 1 on average. Hence, for a function $f(x)$ as found by any of our 3 data fusion methods, the centering and choice of β_j imply that $E_x(f(x)) = 0$ and $E_x(f(x)^2) = (\mathbf{w}'\mathbf{x})^2 \leq \|\mathbf{w}\|^2 \|x\|^2 = \|\mathbf{w}\|^2 k(x,x) \leq 1$. In this way, for a large margin between the training points and the origin, one can expect that relatively few data points lie at the positive side of the hyperplane, as quantified by e.g. the one-tailed Chebyshev's inequality:

$$P\left[f(x) - E_x f(x) \geq k \sqrt{E_x(f(x)^2)} \right] \leq \frac{1}{1+k^2},$$

[1] In this context we would like to note that, while the theory and the algorithm for transduction in Lanckriet *et al.* (2004c,b) is never explicitly expressed in terms of such β_j that weigh the kernel matrices, also there a similar choice has been made. In that paper, what would be the equivalent of our β_j is chosen to be equal (or proportional) to the trace of the kernel matrix on the training and test points together. This choice is also implicitly motivated by the statistical study.

where the expectations are over the total gene distribution. Applied to our problem:

$$P(f(x) \geq k) \leq \frac{1}{1+k^2}.$$

In practice, the number of genes in the genome is so large that *iid* assumptions concerning test and training genes become realistic. In such cases it is possible (and more convenient) to carry out the kernel-centering and determination of β_j on a smaller number of genes, such as on the combination of test and training-genes. This is the approach we take in the experiments below.

4 GENE PRIORITIZATION RESULTS

We will now apply our algorithms on a series of actual biological data on disease gene hunting, taken from a large-scale cross-validation study from Aerts *et al.* (2006). In this paper, 29 diseases are investigated, and to each disease a number of genes between 4 and 113 are known to be associated, with 624 as the total number of disease genes in the study. To assess the performance of a gene-hunting method the following strategy is used. Note that for comparability, we choose to use an essentially identical assessment strategy as the one used in Aerts *et al.* (2006). *For each disease*, do the following:

1. Choose a set of 99 genes, randomly selected from the genome.
2. Perform leave-one-out cross-validation: Regard one of the training genes as a test gene. This test gene is further referred to as the hold-out gene. Then train the disease gene hunting method on the reduced training set which is obtained by omitting the hold-out gene, and rank all test genes plus the hold-out gene. Ideally, the hold-out gene will be unveiled as a disease gene, which means that it will be on top of the list. To verify this, record the rank of the hold out gene in the test set. Since there are 99 test genes and 1 hold-out gene hidden among them, this rank will be in between 1 and 100.

Now, based on all these ranks in the cross-validation (over all diseases and all disease genes for these diseases), we can construct a ROC-like curve in the following way. Plot the fraction of genes that, when held out, rank among the top $x\%$ of test genes, and this as a function of x. If in each hold out experiment the hold out gene ranks first, the ROC-like curve will be 1 for all x, and the area under the ROC curve, further called AUC (Area Under Curve) is equal to 1. For a random training and test set combination, the AUC is 0.5 in expectation.

4.1 Data sources and kernels used

4.1.1 The data sources We are using the following data sources, based on Ensembl v39 (Hubbard *et al.*, 2005): microarray data (MA), DNA sequence (Seq), EST data (EST), Gene Ontology annotations (GO), InterPro domains (IP), KEGG pathways (KEGG), motifs (Motif), binding data (BIND), and literature (Text), just as in Aerts *et al.* (2006). Not to obfuscate the comparison, in our method we deal with missing values in the most naive way, e.g. by equating them to genome-wide averages. We should note that in the ENDEAVOUR paper (Aerts *et al.*, 2006) data from a previous version of Ensembl was used. Therefore, to ensure a fair comparison, we reran their experiments with ENDEAVOUR based on the most recent Ensembl version v39 as well. Hence, we will compare our proposed methods with the performance of ENDEAVOUR on exactly the same data.

4.1.2 The kernel matrices For each data source except for Seq, we use three different kernels:

1. the linear kernel followed by normalization,
2. a Radial Basis Function (RBF) kernels with kernel width equal to twice the average distance of a data point to its nearest neighbor in the union of the training and the test set,
3. and an RBF kernel with kernel width equal to four times the average distance of a data point to its nearest neighbor in the complete data.

The kernel widths are chosen heuristically according to rules of thumb that often yields good results in practice. For the sequence data, we also used three different kernels: the 2-mer, 3-mer, and 4-mer kernels as defined in Leslie and Kuang (2003). Hence, in total 27 kernels are used, 3 for each of the 9 data sources.

4.1.3 Noise data sources As explained below, we have also carried out a robustness analysis by constructing random noise data sources to be included as additional data. These noise data sources consist of 10-dimensional normally distributed random vectors (variance equal to 1). For each noise data source, we constructed 3 kernels to use in the algorithm, a linear one and two RBF kernels, exactly as for the other vectorial data sources. Constructed in this way, the noise data sources should quite accurately mimic real-life data with no relevance to the problem. Note that a comparison with ENDEAVOUR in terms of noise robustness is hard to design, since true noise models cannot as easily be generated as we can generate noise kernels. Therefore, we will exclude ENDEAVOUR from the noise robustness analyses below.

4.2 Disease genes hunting: results

We have carried out a number of experiments to assess the following:

1. the performance gain when compared to ENDEAVOUR of the simple method with uniformly weighted kernels ($\mu_{\min} = \frac{1}{m}$),
2. the use of automatically tuned weights when noisy data sources are taken into account, or when a small number of data sources is much more informative than the others ($\mu_{\min} = 0$),
3. the use of automatically tuned weights with a lower bound in the same scenario ($\mu_{\min} = \frac{0.5}{m}$).

In order to assess noise resilience, we examined the performance as a function of the number of noisy data sources, ranging from 4, over 8, to 16 noise sources, yielding 12, 24, and 48 kernels respectively.

Lastly, we performed each of these same experiments in three scenarios: (i) based on all data sources listed above, (ii) based on all but Text, (iii) based on all but Text and GO, and (iv) based on all but Text, GO and KEGG. We have performed these exclusions in order to investigate to what extent the methods are capable of extracting information from data that may lead to novel discoveries, as opposed to for example Text data that may contain known clues of disease implications. We will now discuss the results in detail.

In order to obtain stable results, we performed 10 randomization for each experiment reported below. In each of these randomizations, a different set of test genes has been chosen, randomly selected from

Table 1. Comparison of the simple, uniformly weighted, kernel method with ENDEAVOUR. Different scenarios are considered, taking into account all 9 data sources, all but Text, all but Text and GO, and all but Text, GO, and KEGG. The first two lines show 1-AUC (lower is better) for both methods, averaged over 10 random selections of the test genes. The last line shows a p-value computed by means of a paired t-test, testing the null hypothesis that the expected 1-AUC is not smaller for the kernel method than for ENDEAVOUR. Clearly, the difference is highly significant for all but the last set of data sources used. Furthermore, the AUC value differs considerably for the first 3 scenarios considered.

1-AUC	All	No Text	No Text, GO	No Text, GO, KEGG
ENDEAVOUR	0.0833	0.1290	0.1698	0.1698
Kernel method	**0.0686**	**0.1043**	**0.1491**	**0.1675**
p-value	7.4e-10	7.5e-11	3.3e-7	2.4e-1

the genome. The same random test set was used in the different methods being compared.

4.2.1 Comparison of the uniformly weigthed method with ENDEAVOUR We compared the performance of ENDEAVOUR with our method with $\mu_{\text{min}} = \frac{1}{m}$. The results are summarized in Table 1, and clearly show that the proposed method outperforms ENDEAVOUR significantly, and by a large margin. This is the case for all (sub)sets of data sources investigated.

Furthermore, we should note that the proposed method is computationally extremely fast: finding the optimal $\boldsymbol{\alpha}$ takes a negligible time for up to hundreds of training genes, and computing the ranking function f on a test gene (the testing phase) is extremely fast as well.

4.2.2 Performance in the presence of one or few 'dominant' informative information sources It can be assumed (and it is observed) that in general, Text and GO contain the most accessible information relevant to disease gene hunting. While this may not always be the case, it is possible that in other cases another data source is much more relevant than any of the others. Therefore, it is of interest to assess to what extent the different methods are able to disregard the less informative data sources.

To assess this, in Table 2 we summarized the 1-AUC scores for different subsets of data sources, and this for our proposed methods and for ENDEAVOUR. We can conclude that if there is a clear best information source (e.g. Text or GO), it pays off to tune the weights automatically (either with $\mu_{\text{min}} = 0$ or, better, with $\mu_{\text{min}} = \frac{0.5}{m}$). If all or most information sources are roughly equally good, the uniform weighting performs better than with the automatically tuned weights. In all cases except when all data sources including Text are being used, $\mu_{\text{min}} = \frac{1}{m}$ seems to perform comparably well to the other kernel methods, and in all cases it performs better than ENDEAVOUR (see higher).

4.2.3 Investigation of the noise sensitivity Table 3 reveals that for increasing amounts of noise, the uniformly weighted method degrades much more rapidly than the methods with tuned weights. This can be explained by the fact that the tuned weights are usually lower for noise kernels than for the informative kernels. Similarly, a larger number of (approximate) copies of the same bad kernel would degrade the performance of the naive method with uniform weights, while the methods with tuned weights are insensitive to this. For a discussion of similar observations in a related context, see Lanckriet *et al.* (2004c,b).

Overall, when large amounts of noise are to be expected or when one or few of the data sources is much more discriminative than the others, the regularized method ($\mu_{\text{min}} = \frac{0.5}{m}$) seems the most robust and performant method. If less than half of the information sources are suspected to be irrelevant, it is better to use the uniformly weighted kernel method ($\mu_{\text{min}} = \frac{1}{m}$).

4.2.4 Performance of individual kernels versus the overall performance Besides comparing the different proposed data fusion methods, we should assess whether it makes sense at all to perform data fusion. To this end, consider Figure 2. We now only consider the regularized method with $\mu_{\text{min}} = \frac{0.5}{m}$, which seems to be a safe approach, though the method with $\mu_{\text{min}} = \frac{1}{m}$ performs even slightly better in most noiseless cases we investigated. Nevertheless, it is interesting to investigate the weights attributed to the individual kernels by the method with $\mu_{\text{min}} = \frac{0.5}{m}$. Our findings are:

1. When using all data sources, the performance is not significantly different from the performance based on single-kernel novelty detection on Text, in particular for the linear kernel. The most likely reason is that the genes in this study have been well-described, such that Text is likely to be most informative by far. Then, taking other less informative (or more 'noisy') data into account may be expected to worsen the result. However, it is encouraging that this is not the case. (See Figure 2 above.) One may argue that it is as good to simply pick the Text data source (if it is available), and discard the others. It should be noted however that in general it is not known a priori which data source is clearly the better. Hence, also the task to pick the best kernel has to be made in a data driven way, and imperfections in this selection process would degrade the result. Hence, comparing the data fusion methods with any of the single kernels would unfairly disadvantage data fusion.
2. When applied to all data sources except the suspectedly richer ones such as Text, GO and KEGG, our method based on kernel combinations is clearly better than any of the separate kernels. This effect is stronger if more informative data sources are excluded.
3. The weights μ_i attributed to each of the kernels are summarized in the bottom four bar plots of Figure 2. Clearly, when taken into account, Text, and to a lesser extent GO data, get the largest part of the weight. When Text data, or Text and GO, are absent the weights are more evenly distributed. Overall, linear kernels yield better results and get higher weights on this data set. The Seq data gets small weights all over, despite its good individual performance. A potential explanation is that interesting aspects of the sequence information are contained in other information sources, such as InterPro.

In summary, our method effectively integrates complementary information from different sources.

4.3 Guidelines on the choice of μ_{min}

Based on the experiments described above, we can suggest the following guidelines:

Table 2. In this Table we show 1 minus the mean AUC performances (smaller is better) for each of the following three methods: automatic tuning with $\mu_{\min} = 0$, regularized automatic tuning with $\mu_{\min} = \frac{0.5}{m}$, and uniformly weighted (equivalent with using $\mu_{\min} = \frac{1}{m}$). ENDEAVOUR's performance is shown for comparison. Clearly, overall the simple kernel-based method with $\mu_{\min} = 1$ comes out best, despite a somewhat worse performance than the methods with $\mu_{\min} = 0$ and $\mu_{\min} = \frac{0.5}{m}$ for when the Text data is included (but still significantly better than ENDEAVOUR).

1-AUC	$\mu_{\min} = 0$	$\mu_{\min} = \frac{0.5}{m}$	$\mu_{\min} = \frac{1}{m}$	ENDEAVOUR
All data	0.0505	**0.0477**	0.0686	0.0833
No Text	0.1241	0.1121	**0.1043**	0.1290
No Text/GO	0.1902	0.1644	**0.1491**	0.1698
No Text/GO/KEGG	0.2121	0.1828	**0.1675**	0.1698

Table 3. In this Table we show 1 minus the mean AUC performances (smaller is better) for each of the following three methods: automatic tuning with $\mu_{\min} = 0$, regularized automatic tuning with $\mu_{\min} = \frac{0.5}{m}$, and uniformly weighted (equivalent with using $\mu_{\min} = \frac{1}{m}$), and this for varying number of noise sources. Clearly, the method with $\mu_{\min} = \frac{0.5}{m}$ is the most robust against noise. However, also with $\mu_{\min} = \frac{1}{m}$ a reasonable robustness is achieved.

1-AUC		$\mu_{\min} = 0$	$\mu_{\min} = \frac{0.5}{m}$	$\mu_{\min} = \frac{1}{m}$
All data sources	No noise	0.0505	**0.0477**	0.0686
	4× noise	0.0596	**0.0579**	0.0950
	8× noise	0.0656	**0.0644**	0.1144
	16× noise	0.0702	**0.0694**	0.1420
No Text	No noise	0.1241	0.1121	**0.1043**
	4× noise	0.1411	**0.1330**	0.1395
	8× noise	0.1520	**0.1444**	0.1629
	16× noise	0.1624	**0.1566**	0.1943
No Text, no GO	No noise	0.1902	0.1644	**0.1491**
	4× noise	0.2186	0.2034	**0.2005**
	8× noise	0.2375	**0.2257**	0.2275
	16× noise	0.2554	**0.2496**	0.2599
No Text, no GO, no KEGG	No noise	0.2121	0.1828	**0.1675**
	4× noise	0.2410	**0.2245**	0.2296
	8× noise	0.2626	**0.2500**	0.2612
	16× noise	0.2825	**0.2770**	0.2963

1. When all data sources (kernels) are likely to be informative to some extent, and not too redundant with each other, it is advisable to use uniform weighting ($\mu_{\min} = \frac{1}{m}$). This comes with the additional benefit of an extremely small computational cost.
2. When it is strongly suspected that some of the data sources may be irrelevant, or that there is a lot of redundancy among them (i.e. a subset of kernels containing the same information on the genes), it is advisable to tune the weights automatically to account for this, by using $\mu_{\min} = \frac{0.5}{m}$. The returned optimal weights will provide feedback as to which data sources are irrelevant or redundant with others.
3. If sufficient training genes were available, we would recommend to tune $\mu_{\min}$ using cross-validation. (Note that often the number of disease genes will be too small for this, so we opted not to include this in the current empirical evaluation.)

5 CONCLUSIONS AND OUTLOOK

We have presented a new approach and its theoretical analysis, to provide an adequate answer to a recently identified highly relevant problem in bioinformatics. The newly proposed method is shown empirically to outperform the method that has currently been most successful in this setting. Two aspects contribute to its success. First, the kernel method on itself seems to ensure a good performance in this context. Second, a uniform linear combination of all kernel matrices appears to be a robust and highly performant method for data fusion for disease gene hunting. And third, if appropriate, the data-dependent automatic weighting procedures ensure robustness against irrelevant or too noisy data sources.

A particularly appealing aspect of the method is its computational efficiency. The kernels can be computed offline, which makes their computation time less relevant. All other tasks require a negligible computation effort: training is extremely fast (with relatively small training sets of up to a few hundreds), and also the subsequent prioritization of genes can be carried out extremely efficiently, easily

scalable to a genome-wide scale. For example, on a laptop with a 2GHz centrino processor and 1GB RAM, an entire cross-validation experiment with all data sources as described in the experiments takes roughly 1 hour for $\mu_{\min} \neq \frac{1}{m}$ and only 2 minutes for $\mu_{\min} = \frac{1}{m}$ (implemented in MATLAB). On the other hand, for the same cross-validation task, ENDEAVOUR needs 16 hours on a workstation with 4GB RAM and a 2GHz dual opteron processor.

As further work, we plan to investigate whether a hand-tuning of the weights is a feasible and useable approach in practice. The main question to be answered here is whether the optimal values for the kernel weights represent some intuitive notion of relevance of the kernels.

ACKNOWLEDGMENTS

YM is a professor at the Katholieke Universiteit Leuven, Belgium. This research is supported by:

- Research Council KUL: GOA/2005/04, GOA AMBioRICS, CoE EF/05/007 SymBioSys
- Flemish Government:
  - FWO: projects G.0407.02 (support vector machines), G.0302.07 (SVM/Kernel), research community (MLDM);
  - IWT: TAD-BioScope-IT
- Belgian Federal Science Policy Office: IUAP P6/25 (BioMaGNet, Bioinformatics and Modeling: from Genomes to Networks, 2007-2011) ;
- EU-RTD: FP6-NoE Biopattern, FP6-MC-EST Bioptrain

REFERENCES

Aerts, S., Lambrechts, D., Maity, S., Loo, P. V., Coessens, B., Smet, F. D., Tranchevent, L.-C., Moor, B. D., Marynen, P., Hassan, B., Carmeliet, P., and Moreau, Y. (2006). Gene prioritization through genomic data fusion. *Nature Biotechnology*, **24**, 537–544.

Bach, F. R., Lanckriet, G. R. G., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML04)*.

Herrmann, D. and Bousquet, O. (2003). On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems 15 (NIPS02)*.

Hubbard, T., Andrews, D., Caccamo, M., Cameron, G., Chen, Y., Clamp, M., Clarke, L., Coates, G., Cox, T., Cunningham, F., Curwen, V., Cutts, T., Down, T., Durbin, R., Fernandez-Suarez, X., Gilbert, J., Hammond, M., Herrero, J. abd Hotz, H., Howe, K., Iyer, V., Jekosch, K., Kahari, A., Kasprzyk, A., Keefe, D., Keenan, S., Kokocinsci, F., London, D., Longden, I., McVicker, G., Melsopp, C., Meidl, P., Potter, S., Proctor, G., Rae, M., Rios, D., Schuster, M., Searle, S., Severin, J., Slater, G., Smedley, D., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Storey, R., Trevanion, S., Ureta-Vidal, A., Vogel, J., White, S., Woodwark, C., and E., B. (2005). Ensembl 2005. *Nucleic Acids Res.*, **1**(33), D447–D453.

Lanckriet, G., Deng, M., Cristianini, N., Jordan, M., and Noble, W. S. (2004a). Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing (PSB04)*.

Lanckriet, G., Bie, T. D., Cristianini, N., Jordan, M., and Noble, W. S. (2004b). A statistical framework for genomic data fusion. *Bioinformatics*, **20**(16), 2626–2635.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., El Ghaoui, L., and Jordan, M. I. (2004c). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, **5**, 27–72.

Leslie, C. and Kuang, R. (2003). Fast kernels for inexact string matching. In *Conference on Learning Theory and Kernel Workshop (COLT03)*, pages 114–128.

Ong, C. S., Smola, A., and Williamson, R. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, **6**, 1043–1071.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, **13**, 1443–1471.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for Pattern Analysis*. Cambridge University Press, Cambridge, U.K.

Sonnenburg, S., Rätsch, G., and Schäfer, C. (2006). A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processing Systems 18 (NIPS05)*.

Tax, D. and Duin, R. (1999). Support vector domain description. *Pattern Recognition Letters*, **20**(11-13), 1191–1199.

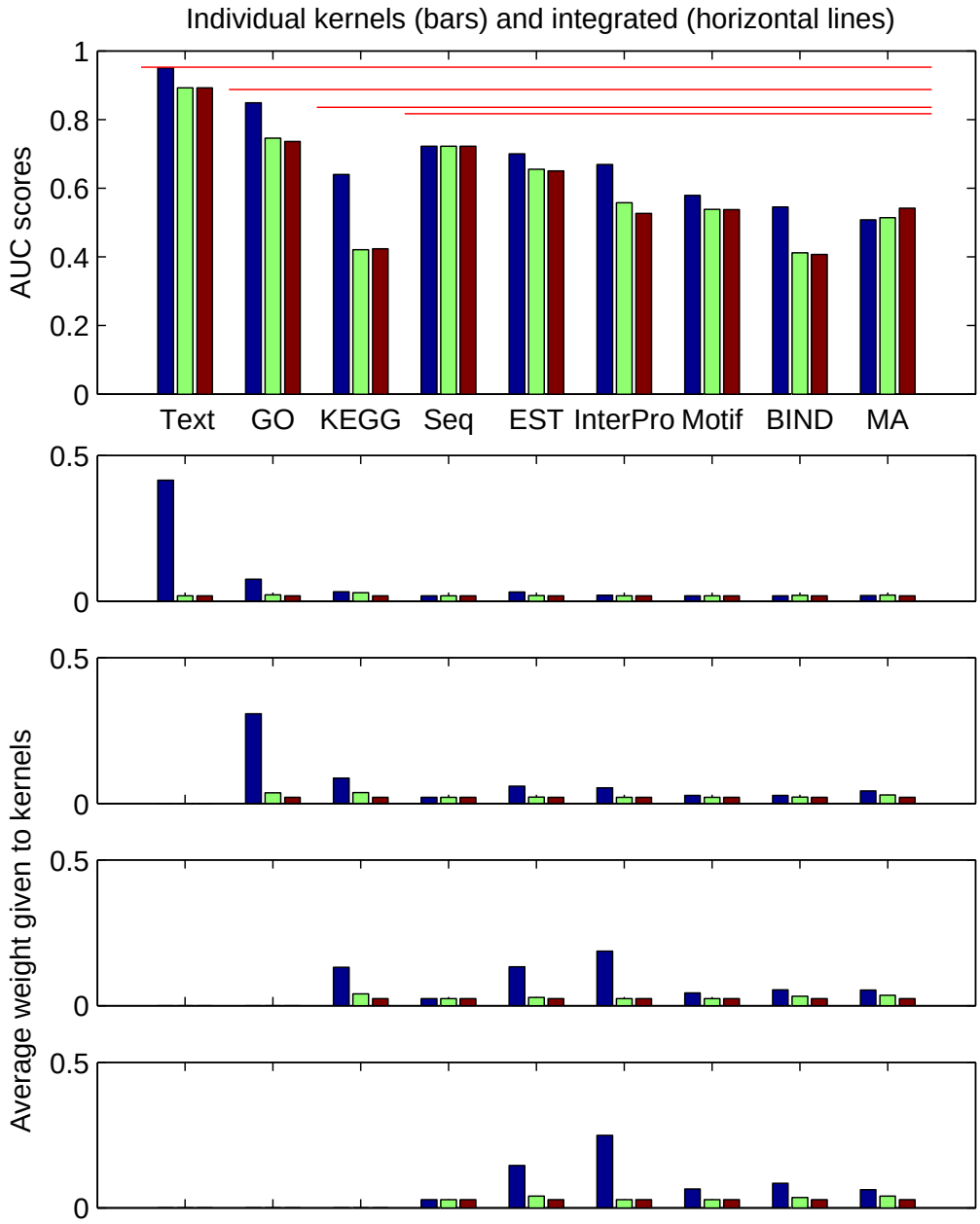


Fig. 2. The top figure shows the AUC performances of each individual kernel, three for each data source (in each triplet the left bar corresponds with the linear kernel, the middle one with the RBF with small kernel width, the right one with large kernel width). For comparison, the 4 full red lines indicate the performance of the kernel combination method with $\mu_{\min} = \frac{0.5}{m}$, using all data sources, all but Text, all but Text and GO, and all but Text, GO and KEGG (in order of decreasing AUC). The lower 4 bar plots show the weights μ_j attributed to each of the kernels by the kernel combination method with $\mu_{\min} = \frac{0.5}{m}$, using each of these 4 (sub)sets of data sources.