BIOINFORMATICS ORIGINAL PAPER

Vol. 25 no. 22 2009, pages 2929–2936
doi:10.1093/bioinformatics/btp485

Systems biology

A boosting approach to structure learning of graphs with and without prior knowledge

Shahzia Anjum[1,*], Arnaud Doucet[2] and Chris C. Holmes[1,3,*]

[1]Medical Research Council, Harwell, UK, [2]The Institute of Statistical Mathematics, Tokyo, Japan and [3]Department of Statistics, University of Oxford, Oxford, UK

Received on April 7, 2009; revised on July 24, 2009; accepted on August 9, 2009

Advance Access publication August 20, 2009

Associate Editor: Olga Troyanskaya

ABSTRACT

Motivation: Identifying the network structure through which genes and their products interact can help to elucidate normal cell physiology as well as the genetic architecture of pathological phenotypes. Recently, a number of gene network inference tools have appeared based on Gaussian graphical model representations. Following this, we introduce a novel Boosting approach to learn the structure of a high-dimensional Gaussian graphical model motivated by the applications in genomics. A particular emphasis is paid to the inclusion of partial prior knowledge on the structure of the graph. With the increasing availability of pathway information and large-scale gene expression datasets, we believe that conditioning on prior knowledge will be an important aspect in raising the statistical power of structural learning algorithms to infer true conditional dependencies.

Results: Our Boosting approach, termed BoostiGraph, is conceptually and algorithmically simple. It complements recent work on the network inference problem based on Lasso-type approaches. BoostiGraph is computationally cheap and is applicable to very high-dimensional graphs. For example, on graphs of order 5000 nodes, it is able to map out paths for the conditional independence structure in few minutes. Using computer simulations, we investigate the ability of our method with and without prior information to infer Gaussian graphical models from artificial as well as actual microarray datasets. The experimental results demonstrate that, using our method, it is possible to recover the true network topology with relatively high accuracy.

Availability: This method and all other associated files are freely available from http://www.stats.ox.ac.uk/~anjum/.

Contact: s.anjum@har.mrc.ac.uk; cholmes@stats.ox.ac.uk

Supplementary information: Supplementary data are available at *Bioinfomatics* online.

1 INTRODUCTION

Learning the structure, or connectivity, of a graphical model is an important problem with a wide range of applications ranging from speech processing to genomics. For example, it is the concerted activity of thousands of genes and their products that primarily determines the cellular phenotypes. Thus, elucidation of their complex interaction networks can help to provide new insights into normal cell physiology as well as identify the genetic architecture of pathological phenotypes. In addition, the study of protein–protein interaction networks can also help to highlight key proteins that could potentially become interesting drug targets (Jeong *et al.*, 2001).

Consequently, this task of structure learning for graphical models has attracted considerable attention over the past few years. Previously published methods in this field include Bayesian networks (Friedman *et al.*, 2000a; Segal *et al.*, 2003) and Gaussian graphical models (Dobra *et al.*, 2004; Schafer and Korbinian, 2005). A sizeable amount of effort has also been directed towards supervised or semi-supervised approaches, where partial knowledge of the network structure is used to supplement the overall inference procedure. Methods based on this framework include the kernel canonical correlation analysis of Yamanishi *et al.* (2004) and the kernel metric learning method by Vert and Yamanishi (2005). Unfortunately, most of the methods are either limited by the small sample sizes of genomic datasets or are computationally too expensive in the high-dimensional case, though some interesting Bayesian methods are emerging (Lenkoski and Dobra, 2008; Mukherjee and Speed, 2008). Motivated by these challenges, we have developed a novel boosting approach for learning the undirected network structure, with and without prior knowledge, from high-dimensional data sets. We will limit ourselves here to the Gaussian case as have others (Friedman *et al.*, 2008; Schafer and Korbinian, 2005). We consider p-dimensional data, $y=\{y_1,\dots,y_p\}$, arising from a multivariate normal density

$$y \sim N(\mu,\Sigma)$$

with unknown mean μ and non-singular covariance matrix Σ. A graphical model for y can be represented by an undirected graph $G=(V;E)$, where V contains p nodes corresponding to each of the p coordinates and the edges $E=(e_{ij})$ for $i<j$ describe the conditional independence relationship among $(y_1,\dots,y_p)$. The edge between y_i and y_j is absent if and only if y_i and y_j are independent conditional on the other variables. It is well known that the graphical structure of G can be inferred from the precision matrix $\Omega=\Sigma^{-1}$. Indeed the partial correlation coefficients, ρ, satisfy

$$\rho_{ij} = -\frac{\Omega_{ij}}{\sqrt{\Omega_{ii}\Omega_{jj}}}$$

*To whom correspondence should be addressed.

where $\rho_{ij}=0$ implies conditional independence between y_i and y_j. That is, $y_i \perp y_j | y_{-(i,j)}$ where $\perp$ denotes conditional independence and $y_{-\{i,j\}}$ denotes all other variables than $\{y_i, y_j\}$, $y_{-\{i,j\}} = y \setminus \{y_i, y_j\}$. Thus, the partial correlations encode the structure (set of edges) of the graph G, such that

$$e_{ij} = \begin{cases} 0 & \text{if } \rho_{ij}=0 \\ 1 & \text{otherwise} \end{cases}$$

Given an i.i.d. sample of n data, we are interested here in estimating the structure of the undirected graph. This problem is also known as covariance selection and was first studied by Dempster (1972). In order to handle the high-dimensional models under this framework, one approach is to use a greedy forward/backward search, but the selection/deletion of an edge requires $O(p^2)$ operations and is too computationally intensive for high-dimensional graphs. Recently, many alternative algorithms have appeared to solve this problem. For example, in a Bayesian framework, the authors in Jones *et al.* (2005) employ a stochastic algorithm to manage tens of thousands of variables. However, most of the papers have focused on the use of faster L1 (Lasso) regularization techniques, where a L1 penalty is imposed on Ω to increase its sparsity (Tibshirani , 1996). Various optimization techniques have been developed to maximize the L1-penalized log-likelihood (Banerjee *et al.*, 2008; Friedman *et al.*, 2008; Li and Yang, 2005; Schmidt *et al.*, 2007; Yuan, 2006; Yuan and Lin, 2007). In particular, Friedman *et al.* (2008) have recently proposed the graphical Lasso procedure which is a simple yet computationally efficient procedure. Yuan (2006) have also proposed an approach to compute the regularization path for this problem. A simpler approach has also been presented in the seminal work of Meinshausen and Buhlmann (2006), which consists of using a Lasso procedure for each variable, while using the others as predictors. The edge between nodes i and j is included if the estimated regression coefficient of variable i on j and j on i are non-zero. The authors studied this approach in detail, and showed that the resulting estimator is consistent for sparse high-dimensional graphs.

In this article, we explore the use of boosting methods to learn the structure of large graphs, of order 5000 nodes or higher. Since the introduction of Freund and Schapire's Adaboost algorithm for classification, boosting algorithms have become one of the most important supervised machine learning methods. In the insightful paper (Friedman *et al.*, 2000b), Friedman, Hastie and Tibshirani pointed out the relationship between boosting and L1-penalized estimation and initiated the use of boosting in contexts other than classification. In particular, in a regression context, Buhlmann and Yu (2003) have proposed L2 boosting which is an iterative procedure refitting the residuals multiple times.

The algorithm we propose, termed BoostiGraph, for learning graph structures uses the same notion of iterated residual updating at the nodes and univariate autoregressions and can be interpreted as a componentwise L2 boosting alternative to the work of Meinshausen and Buhlmann (2006). For graphs of order 5000 nodes, BoostiGraph is able to map out full boosting paths for the conditional independence structure in few minutes on a standard PC at time of writing. The method is also conceptually simple, of around 50 lines of a single file Matlab code using a single *for* loop, and appears highly competitive with other graphical inference approaches. The Matlab code is available from http://www.stats.ox.ac.uk/∼anjum/.

2 METHODS

2.1 Boosting graphical structure

Our starting point is similar to that of Meinshausen and Buhlmann (2006) and relies on the use of the autoregressions

$$y_i = \sum_{j \neq i} \beta_{ij} y_j + \varepsilon_i$$

where $\rho_{ij}=0$ implies that the regression coefficient, $\beta_{ij}=0$. That is, by fitting the autoregressions at each node and considering the β_{ij} for all i,j we can detect whether $y_i \perp y_j | y_{-(i,j)}$, namely y_i is conditionally independent of y_j given the rest of the data, i.e. there is no edge between y_i and y_j.

Let the $n \times p$-dimensional matrix $Y=(Y_1,...,Y_p)$ contain the n independent observations of y so that the rows of the column vector Y_i correspond to the n independent observations of y_i and Y_{-i} corresponds to the n independent observations of the remaining $(p-1)$ nodes. In Meinshausen and Buhlmann (2006), the regression is solved under the Lasso-penalized likelihood framework

$$\widehat{\beta}_i = \underset{\beta_i}{\arg\max} \left\{ \|Y_i - Y_{-i}\beta_i\|_2^2 + \lambda \|\beta_i\|_1 \right\}$$

where $\|\beta_i\|_1 = \sum_{j \neq i} |\beta_{ij}|$ and $\lambda > 0$ is a regularization parameter. That is, $\widehat{\beta}_i$ is the optimal setting of β_i given the criteria on the right-hand side. Instead of using a Lasso-type approach, we favour here a boosting approach where at the m-th iteration residuals from the current model fit are regressed upon the univariate predictors given by other y's. This univariate update is extremely fast. For more details on the linear regression and boosting, readers are referred to Buhlmann (2006) and the references therein.

2.2 BoostiGraph

The method is perhaps most easily illustrated through the algorithm. Here, T is the number of boosting iterations, while $0<\eta\leq 1$ is a step-length factor. In all subsequent simulations, we select $\eta=0.1$. The indices i,j refer to different nodes in the network.

Boosting graph structure algorithm (BoostiGraph):

(1) Standardize Y_i to zero mean and unit standard deviation for all i.

(2) Set counter $m=0$, initialize the $p\times(p-1)$ coefficients $\beta_{ij}^{(m)}=0$ for all i,j.

(3) Initialize *working residuals* $Y^* \leftarrow Y$.

(4) Fit the $p\times(p-1)$ univariate regressions

$$\widehat{\beta}_{ij} = Y_j' Y_i^* \qquad \forall i \neq j$$

(5) Find best predictor

$$\hat{i},\hat{j} = \underset{i,j}{\arg\max} \left\{ |\widehat{\beta}_{ij}| \right\}$$

(6) Perform the boosting update

$$\widehat{\beta}_{\hat{i}\hat{j}}^{(m+1)} \leftarrow \widehat{\beta}_{\hat{i}\hat{j}}^{(m)} + \eta \widehat{\beta}_{\hat{i}\hat{j}}$$

$$Y_{\hat{i}}^* \leftarrow (Y_{\hat{i}}^* - \eta Y_{\hat{j}} \widehat{\beta}_{\hat{i}\hat{j}})$$

(7) Update the $(p-1)$ autoregressions on $Y_{\hat{i}}^*$ that have changed

$$\widehat{\beta}_{\hat{i}j} = Y_j' Y_{\hat{i}}^* \text{ for } j \neq \hat{i}$$

(8) Increment counter, $m \leftarrow m+1$, and repeat steps 5–7 until $m=T$.

(9) Return vector of matrices $\widehat{\beta}^{(1:T)}$.

(10) Report $e_{ij}^{(T)} = 1 - \delta_0\left(\widehat{\beta}_{ij}^{(T)} \widehat{\beta}_{ji}^{(T)}\right)$.

We can see from the above that BoostiGraph implements a forward stagewise fitting algorithm in the spirit of Friedman *et al.* (2000b). That is, at each stage the 'best' predictor (directed edge) is selected.

Computationally, the initialization of the algorithm involves $1/2p(p-1)$ pairwise node comparisons after which each boosting step requires $p-1$

updates. These evaluations are only univariate regressions as the working residuals store the current parameter values. The algorithm returns at iteration T an estimate $e_{ij}^{(T)}$ of the graph structure, but it is also possible to display the full boosting path $e_{ij}^{(1:T)}$. The run time is of order $(T+p/2)(p-1)$ for T boosting steps and p variables. Please note that in principle, unlike the approaches by Yuan and Lin (2007) and Friedman *et al.* (2008), the proposed method (as in Meinshausen and Buhlmann, 2006) does not guarantee the positive definiteness of the resulting estimator. However, given an estimate of the structure of the graph a conventional estimator of the precision matrix could be adopted.

2.3 Regularization by early stopping

As the number of boosting iterations increases, BoostiGraph will overfit and the false discovery rate of edge detection will increase. Stopping early allows us to regularize the solution. One way to determine the stopping time is to estimate the degrees of freedom in conjunction with an information criteria such as the Bayesian information criteria (BIC; Schwartz, 1978). We have explored the use of BIC which appears to give reasonable performance. The BIC for the model, say m, can be estimated as shown in the equation given below; and where, following Efron *et al.* (2004), the degrees of freedom (df) are calculated by the number of non-zero edges.

$$\mathrm{BIC}(m)=\frac{\mathrm{RSS}}{\widehat{\sigma}^2}+2\log(np)\mathrm{df}(m)$$

where RSS is the residual sum of squares for the estimated model and $\widehat{\sigma}^2$ is the usual unbiased estimate of the variance of the error terms (ϵ_i) based on the full model. The second term of the equation $[2\log(np)\mathrm{df}(m)]$ penalizes the model for the lack of parsimony; where, $\mathrm{df}(m)$ represents the degrees of freedom of the model m. The number of observations is represented by n and p is the number of parameters.

An alternative would be to use, say, 10-fold cross-validation on the autoregressions. We have not explored this last method. Readers interested in the theoretical properties of boosting methods are referred to Buhlmann (2003).

3 GRAPH INFERENCE WITH AUXILIARY INFORMATION

The majority of the mainstream graph/network inference algorithms, including the BoostiGraph method described above, follow the *unsupervised* learning framework. Although these methods have been used with varying degrees of success, predicting new edges with a reasonable false discovery rate still remains extremely challenging due the relatively small sample sizes.

The increasing wealth of large-scale genomic data and other useful biological information have enabled the network inference problem to be formulated as a *semi-supervised or supervised* learning task. The main aim, in this case, is to utilize the prior knowledge in such a way that it can help to predict the missing interactions and increase the overall performance.

Supervised graph inference has drawn considerable interest and a variety of methods under this framework have been proposed. For instance, the kernel canonical correlation analysis published by Yamanishi *et al.* (2004) and the kernel metric learning method by Vert and Yamanishi (2005). Both these methods map the genes or proteins onto the Euclidean space where connected nodes are close to each other. For example, if nodes A and B are connected, they will lie close to each other in the Euclidean space. However, a third node C, only connected to node A, will also be assigned an edge to node B based on the fact that A and B are connected. This underlying hypothesis that genes with similar genomic data are likely to have similar neighbours has not yet been justified nor supported by experimental evidence (Bleakley *et al.*, 2007).

Ben-Hur and Noble (2005) view the graph inference problem as a classical binary supervised classification task and solve it with a support vector machine (SVM) based on a specific kernel for all edges. However, building one unique *global* model that can be applied to all the nodes in the graph is not trivial. Instead, Bleakley *et al.* (2007) investigated the possibility of building *local* models. This proposed *local* method uses SVM to learn the individual subnetworks associated with each of the nodes in the known graph (the training set). The classification rules, thus, learnt for each of those nodes in the training set are then used individually to predict edges between that node and the nodes in the unknown part of the graph (the test set).

However, learning robust classification rules still requires a reasonably large number of positive examples, in the form of known edges, and negative examples in the form of known non-edges. This could be a problem if only a small amount of biological information regarding the regulatory network structure to be inferred is at hand. Moreover, the above mentioned methods, given a (labelled) training set and an (unlabelled) test set, infer edges only for the test set (Chapelle *et al.*, 2006). This means that targets of all examples in the training set are required and since this may not always be possible, we suggest using a *semi-supervised* framework. Under this framework, given a training set, we can infer edges over the entire input space.

In this section, we propose a simple modification to the BoostiGraph algorithm which will allow us to utilize auxiliary information and improve the graph inference especially when the number of predictors greatly exceeds the number of observations.

3.1 The proposed method

As before, $G=(V;E)$ represents the undirected graph, where $E\subset(V\times V)$ is a set of edges.

Suppose we are given the knowledge of a subgraph $G_m=(V_m;E_m)$ of G where $V_m\subset V$ and $E_m=\{(v,v')\in E|v,v'\in V_m\}$ which contains the *known edges* and *the known non-edges*. The goal of the semi-supervised network inference is to then determine the set of edges $E'\subset(V\times V)\backslash E_m$ from the knowledge of G_m. Information regarding the *known edges* can be extracted from a number of potential sources such as scientific literature or databases with information on transcriptional relationships (e.g. TRRD; http://wwwmgs.bionet.nsc.ru/mgs/gnw/trrd/). In contrast, prior knowledge on non-edges is harder to obtain mainly due to their absence from scientific literature. However, as shown in Segal *et al.* (2003), information on non-edges can be extracted based on the annotation of genes in terms of their cellular function or cellular localization and the assumption that genes that are not involved in a signal transduction pathway, are unlikely to directly regulate any other gene. This type of prior knowledge is likely to increase as more genes are annotated.

To utilize the information given in the subgraph, G_m, the main idea is to choose, at each iteration, the edge with the largest score which takes into consideration the likelihood of the edge given the data as well as the log-odds value as shown in the equation given below:

$$\log\ \text{score}_{ij} = \log\frac{L(e_{ij}=1)}{L(e_{ij}=0)} + \log\frac{\pi_{ij}}{1-\pi_{ij}}$$

We can think of $L(e_{ij}=1)$ as the approximate likelihood that there is an edge, between nodes i and j, given the data and also given the other edges. $L(e_{ij}=0)$ is the (approximate) likelihood that there is no edge, between nodes i and j, given the data and also given the other edges. Since the latter value, at each iteration, is equivalent to the current likelihood of the model and will be the same for each edge; it, therefore, suffices to rank the edges based on the (approximate) value of their $L(e_{ij}=1)$ plus the log-odds. Here, $\pi_{ij} \in [0,1]$ represents the prior information which is basically the users certainty or uncertainty, on the presence of an edge between nodes i and j.

3.1.1 Modified BoostiGraph The pseudo-algorithm shown in this section will help to illustrate the proposed modification to BoostiGraph. As before, T is the number of boosting iterations, while $0 \le \eta \le 1$ is a step-length factor. Let E_m represent the set of likely edges and E_n the set of unlikely edges. Let the prior information about the network structure be encoded in a $V \times V$ symmetric matrix such that

$$\pi_{ij} = \begin{cases} 0.5 < \pi_{ij} \le 1 & \text{if } i \neq j \text{ and } (i,j) \in E_m \\ 0 \le \pi_{ij} < 0.5 & \text{if } i \neq j \text{ and } (i,j) \in E_n \\ 0 & \text{if } i = j \\ 0.5 & \text{otherwise} \end{cases}$$

```
(1) Standardize $Y_i$ to zero mean and unit SD for all $i$.
(2) Set counter $m=0$, initialize the $p\times(p-1)$ coefficients $\beta_{ij}^{(m)}=0$ for all $i,j$.
(3) Initialize working residuals $Y^* \leftarrow Y$.
(4) Fit the $p\times(p-1)$ univariate regressions.
        $\widehat{\beta}_{ij} = Y_j' Y_i^* \quad \forall i \neq j$
(5) Calculate score for each potential edge
        $\log(\text{score}_{ij}) = \log(\text{likelihood}_{ij}) + \log(\text{odds}_{ij})$
    where,
        $\log(\text{likelihood}_{ij}) = -\frac{1}{2}(Y_i^* - Y_j\widehat{\beta}_{ij})'(Y_i^* - Y_j\widehat{\beta}_{ij})$
        $\log(\text{odds}_{ij}) = \log(\pi_{ij}) - \log(1-\pi_{ij})$
(6) Find best edge
        $\hat{i},\hat{j} = \arg\max_{i,j} \{\log(\text{score}_{ij})\}.$
(7) Once the edge has been added, reset the prior information for that edge to 0.5
        $\pi_{ij} < 0.5.$
(8) Follow steps 6 and 7 of the BoostiGraph algorithm 2.2.
(9) Increment counter, $m \leftarrow m+1$, and repeat steps 5–8 until $m=T$.
(10) Follow steps 9 and 10 of the BoostiGraph algorithm 2.2.
```

4 EXPERIMENTAL RESULTS

4.1 Graph inference using BoostiGraph

4.1.1 On sparse network structure Following the procedure from others (Friedman *et al.*, 2008, Yuan, 2006, Yuan and Lin, 2007), we generated data from a graphical model with sparse Markov structure $(\Omega)_{ij} = \alpha$ for $|i-j|=1$; $(\Omega)_{ii}=1\ \forall i$; $(\Omega)_{ij}=0$ otherwise.

We generated 100 datasets with $n=100$, $p=200$, $\alpha=\{0.5, 0.3\}$ and noise $\sigma=0.1$,

$$y \sim N(0, \Omega^{-1} + \sigma^2 I)$$

The resulting graphs had approximately 500 connections.
Tables 1 and 2 show the true positive rate (TPR) and false positive rate (FPR) for the top 300, 400 and 500 edges inferred by BoostiGraph, GeneNet (Schafer and Korbinian, 2005), glasso and Least Angle Regression (LARS; Efron *et al.*, 2004). LARS was run separately on each node with the remaining nodes as predictors in a similar manner to Meinshausen and Buhlmann (2006). The points on the receiver operating curves (ROC) curve for LARS correspond to the TPRs and FPRs across the entire network inferred under various cut-off points. In this case, the cut-off points, ranging from 1% to 100%, are the percentages of variance explained by the LARS solution.

The Tables 1 and 2 show the results over 100 simulated datasets. From these tables it can be seen that BoostiGraph slightly outperforms all three methods. However, its performance is closely followed by others. Please refer to the Supplementary Material for the full ROC curves.

Table 3 shows the runtime of all four methods, in minutes, for the sparse setting on a Quad Core processor. BoostiGraph is extremely fast and can compute the required statistics for large datasets with 5000–10 000 nodes within 1 min.

4.1.2 On non-sparse network structure We also considered a non-sparse graph which was generated by randomly removing some connections in the Cholesky decomposition of $\Omega_{ij}=\alpha^2$ for $i\neq j$, $\alpha=0.5$, $(\Omega)_{ii}=1$ and $\sigma=0.01$. The resulting graph has approximately 1000 connections for $p=200$ and we set $n=100$. The lower graph size, $p=200$, was needed to ensure the stability in the generation of the dataset, as the condition number of the covariance matrix became too high for larger p. We assessed the performance on 100 datasets drawn from this non-sparse network structure. Tables 4 and 5 show the performance of BoostiGraph, glasso, GeneNet and LARS for low ($\alpha=0.3$) and high ($\alpha=0.5$) effect sizes, respectively.

The BoostiGraph algorithm once again performs competitively against all three methods—glasso, GeneNet and LARS. Please refer to the Supplementary Material for the full ROC curves with error

Table 1. Average performance results for BoostiGraph, GeneNet, glasso and LARS on 100 datasets of sparse Markov structure, $\alpha=0.3, \sigma=0.1$

Method	Top 300 edges		Top 400 edges		Top 500 edges	
	TPR	FPR	TPR	FPR	TPR	FPR
BoostiGraph	0.417	0.004	0.490	0.008	0.545	0.011
GeneNet	0.416	0.004	0.488	0.008	0.541	0.011
glasso	0.390	0.005	0.455	0.009	0.506	0.012
LARS	0.308	0.004	0.369	0.008	0.419	0.009

Table 2. Average performance results for BoostiGraph, GeneNet, glasso and LARS on 100 datasets of sparse Markov structure, $\alpha = 0.5, \sigma = 0.1$

Method	Top 300 edges		Top 400 edges		Top 500 edges	
	TPR	FPR	TPR	FPR	TPR	FPR
BoostiGraph	0.553	0.0008	0.685	0.002	0.768	0.005
GeneNet	0.547	0.094	0.674	0.003	0.756	0.006
glasso	0.490	0.003	0.590	0.005	0.666	0.008
LARS	0.380	0.003	0.462	0.005	0.529	0.006

Table 3. Runtime, in minutes, for sparse networks on a Quad Core processor for BoostiGraph, GeneNet, glasso and LARS

Nodes	BoostiGraph	GeneNet	glasso	LARS
100	0.001	0.004	5e-04	0.196
500	0.002	0.068	0.037	3.540
1000	0.003	0.323	0.310	12.60
5000	0.038	7.87	>11	> 279
10000	0.158	34.52	> 60	>1173

Table 4. Average performance results for BoostiGraph, GeneNet, glasso and LARS on 100 datasets of non-sparse structure, $\alpha = 0.3, \sigma = 0.1$

Method	Top 900 edges		Top 1000 edges		Top 1500 edges	
	TPR	FPR	TPR	FPR	TPR	FPR
BoostiGraph	0.524	0.019	0.551	0.024	0.650	0.045
GeneNet	0.520	0.020	0.547	0.024	0.647	0.045
glasso	0.474	0.023	0.513	0.029	0.603	0.047
LARS	0.411	0.011	0.437	0.013	0.461	0.015

Table 5. Average performance results for BoostiGraph, GeneNet, glasso and LARS on 100 datasets of non-sparse structure, $\alpha = 0.5, \sigma = 0.1$

Method	Top 900 edges		Top 1000 edges		Top 1500 edges	
	TPR	FPR	TPR	FPR	TPR	FPR
BoostiGraph	0.613	0.015	0.650	0.019	0.751	0.0395
GeneNet	0.617	0.015	0.650	0.018	0.749	0.0396
glasso	0.453	0.024	0.481	0.028	0.600	0.0476
LARS	0.447	0.012	0.474	0.014	0.500	0.0163

bars. The timing comparison between all four methods on the setting is given in Table 6. Once again the runtime, reported in minutes, was observed on a Quad Core processor.

As can be seen from Table 6, the BoostiGraph algorithm is relatively computationally fast especially when the number of nodes is high, i.e. $\geq$1000, though the runtime gets comparatively slower for more edges and higher FDR as BoostiGraph has to be run for longer to realise such networks.

Table 6. Runtime, in minutes, for non-sparse networks on a Quad Core processor for BoostiGraph, GeneNet, glasso and LARS

Nodes	BoostiGraph	GeneNet	glasso	LARS
100	0.0017	0.0034	0.0015	0.192
500	0.0017	0.0688	4.9841	3.303
1000	0.0021	0.3185	>120	11.84

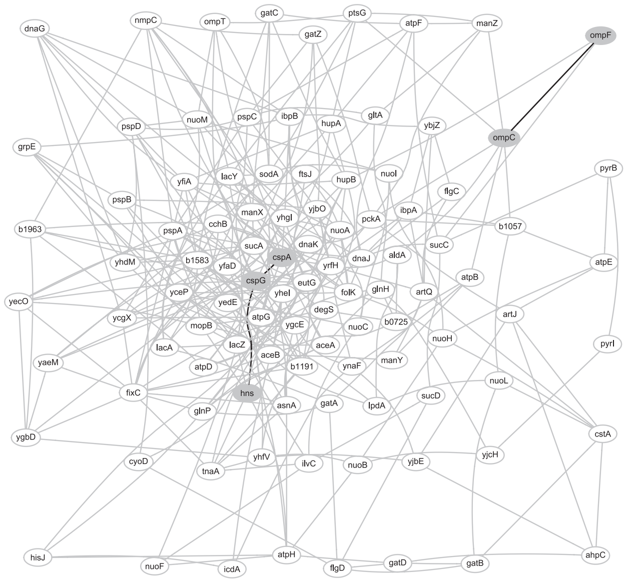

Fig. 1. Gene network inferred from *E.coli* data set using the BoostiGraph method; from example in section 4.1.3

4.1.3 Escherichia coli example We applied the BoostiGraph to the microarray dataset containing the stress response levels of *E.coli* during the expression of a recombinant protein SOD (human superoxide dismutase). This experiment was conducted by the Institute of Applied Microbiology, University of Agricultural Sciences of Vienna (Schmidt-Heck *et al.*, 2004). The initial experiment results monitor the expression levels of all 4289 *E.coli* genes at 8, 15, 22, 45, 68, 90, 150, and 180 min after induction of the recombinant protein. Schmidt-Heck *et al.* (2004) have identified 102 genes out of the 4289 genes as differentially expressed in one or more samples after induction. This preselected set of 102 genes and their expression levels across nine samples were used for our analysis.

Figure 1 shows the network inferred by the BoostiGraph method with early stopping using a BIC criteria. The runtime was 5.4 s to fit the graph. We have highlighted a few of the edges that demonstrate the ability of BoostiGraph to infer real structure. These include the edge between the genes *cspA* and *cspG* and between *cspG* and *hns*. The edge between the cold shock proteins *cspA* and *cspG* is supported by the study which reports a high sequence similarity, both in coding and non-coding regions, for these two proteins in *E.coli* (Nakashima *et al.*, 1996). While the *cspA* protein is known to be involved in the transcriptional regulation of the cold shock gene *hns* (Jones *et al.*, 1992; La Teana *et al.*, 1991), the high sequence similarity between *cspA* and *cspG* provides some support for the edge between *cspG* and *hns*. This edge is particularly interesting because none of the other approaches, namely GeneNet, LARS and Lasso regression, were able to infer an edge between *cspA* and *hns* or *cspG* and *hns*. Due to limited space, we have relegated the details on the networks inferred by the other three methods to the Supplementary Material. Another notable edge is that between genes *ompF* and *ompC* which has been inferred only by the BoostiGraph approach and the LARS approach. Both of the genes, *ompF* and *ompC*, are porin proteins and are known to show differential expression depending on the growing media for the *E.coli* (Waukau and Forst, 1992). This provides some evidence that our BoostiGraph approach is able to infer real structure.

4.2 Graph inference using BoostiGraph and prior knowledge of partial network structure

4.2.1 On sparse network structure We assessed the performance of our method and compared it to the *local model* approach proposed by Bleakley *et al.* (2007) on simulated data drawn from the sparse network structure described in Section 4.1.1. Using that network structure, we generated 100 datasets with $n=100$, $p=200$, $\alpha=0.5$ and noise $\sigma=0.1$. The performance on all 100 datasets were tested across a 5-fold cross-validation and at each fold, information on all pairwise interactions between a randomly selected set comprising of 20% (80% in the second case) of the nodes were provided as the training set. For the *local model*, following Kato *et al.* (2005), we transform the datasets into positive semi-definite matrices of similarities between genes using a RBF Gaussian kernel. To emulate different levels of prior information, we simulated prior values uniformly using $\pi_{ij}\in[0.5,1]$ for edges and uniformly using $\pi_{ij}\in[0,0.5]$ for non-edges.

From the ROC curves in Figure 2a, it is quite clear that the *modified* BoostiGraph has an advantage over the *local model* approach when the network to be inferred is sparse. Although the *local model* uses balanced penalization when positive and negative training sets are of unequal sizes, in this sparse setting with only 500 edges and approximately 19 000 non-edges, there was a large difference between the number of positive and negative examples. This discrepancy made it difficult for the SVM classifier to learn robust classification rules.

4.2.2 On non-sparse network structure For this non-sparse example, we considered the network structure described in Section 4.1.1. The performance of our *modified* BoostiGraph and the *local model* were compared on all 100 datasets with 5-fold cross-validation and at each fold, information on all pairwise interactions between a randomly selected set comprising 20% (80% in the second case) of the nodes were provided as the training set. As in the previous example, the RBF Gaussian kernel was used as the kernel matrix function for input to the *local model*.

The ROC curves in Figure 2b shows the performance of our *modified* BoostiGraph approach in comparison with that of the *local model*. Once again, the *modified* BoostiGraph approach out-performs the *local model* by a large margin regardless of the amount of information provided in the training set. Please refer to the Supplementary Material for experiments on the sensitivity

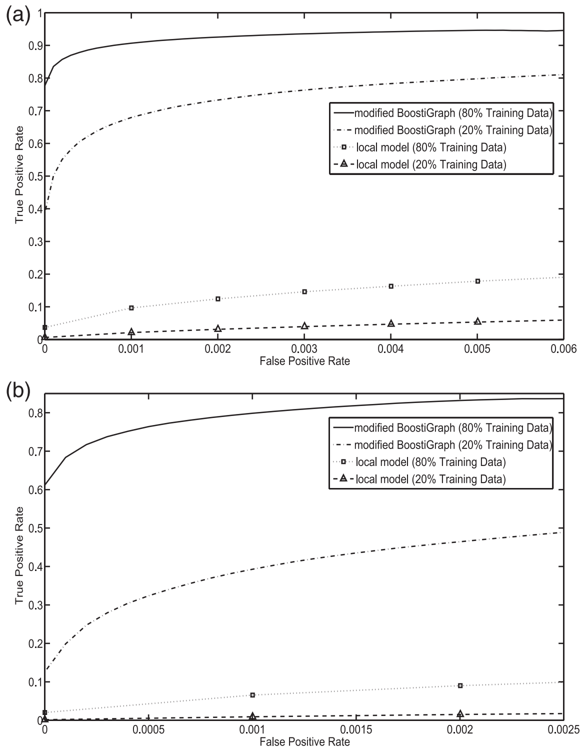

Fig. 2. ROC curves: *modified* BoostiGraph vs *local model* on sparse and non-sparse structures. (**a**) ROC curves for BoostiGraph (solid lines) and Local Model using SVM (dashed lines), for sparse Markov structure, $\alpha=0.5$, $\sigma=0.1$. (**b**) ROC curves for BoostiGraph (solid lines) and Local Model using SVM (dashed lines) , for non-sparse structure, $\alpha=0.5$, $\sigma=0.1$.

of the *modified* BoostiGraph algorithm and also a performance comparison between the BoostiGraph algorithms with and without prior information.

4.2.3 Analysis of cell signalling data We evaluated the performance of our *modified* BoostiGraph approach and that of the *local model* proposed by Bleakley *et al.* (2007) on the RAF signalling network. This network, shown in Figure 3, has been well studied in the literature (e.g. Sachs *et al.*, 2005, Dougherty *et al.*, 2005) and has been used by both Friedman *et al.* (2008) and Werhli *et al.* (2006) as the gold standard network for their analyses.

The expression levels for all the 11 proteins of this network, namely RAF, MEK, PLCg, PIP2, PIP3, ERK, AKT, PKA, PKC, P38 and JNK, were measured by Sachs *et al.* (2005) by applying flow cytometery experiments. The original dataset consisted of 5400 data points for each of the 11 proteins and Sachs *et al.* (2005) have demonstrated that with such a large dataset, the RAF signalling network can be successfully reverse engineered. However, most of the microarray expression datasets have small sample sizes and therefore, to make it a more realistic performance assessment of the methods, we used the data from Werhli *et al.* (2006) who have sampled the data from Sachs *et al.* (2005) into five individual sets with 100 data points for each protein.

Also, since the RAF signalling network is a relatively small network, as part of our prior information, we selected the network between 6 of the 11 proteins, namely PIP3, PIP2, PKC, PKA, RAF, ERK (coloured nodes in Fig. 3). This was done to ensure that the

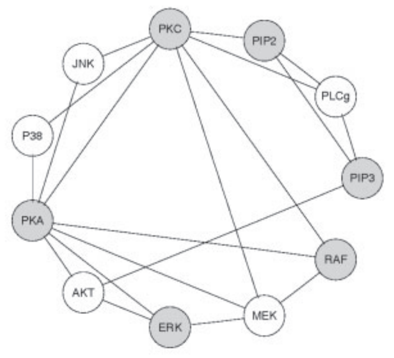

Fig. 3. The undirected RAF signalling pathway from Werhli *et al.* (2006) and drawn using RGraphviz (Gentry *et al.*, 2004).

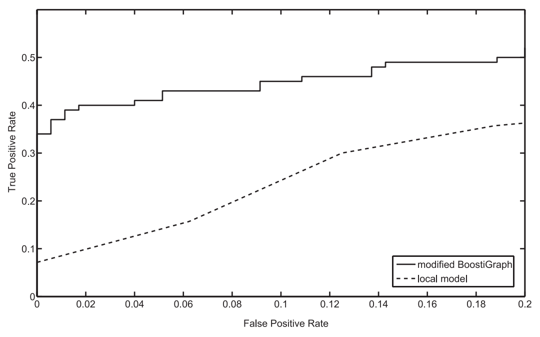

Fig. 4. The ROC curves for both *modified* BoostiGraph (solid line) and *local model* (dashed line) on the flow cytometery data from Werhli *et al.* (2006).

local model approach had at least one positive example for each node to train on and also that there was at least one edge between each node in the training set and the test set for the *local model* to infer. Once again, for the *modified* Boostigraph, the edges between the chosen proteins were randomly assigned probabilities between 0.5 and 1; while the non-edges were randomly assigned probabilities between 0 and 0.5.

Figure 4 shows the ROC curves for both the *modified* BoostiGraph approach and the *local model* approach. The performance results for both methods were averaged over the five individual datasets. From the ROC curves, it can be seen that our *modified* BoostiGraph approach performed slightly better than the *local model* and was able to infer almost 35% and 50% of the true edges with zero and 1% false positives, respectively.

5 DISCUSSION

In this article, we have proposed BoostiGraph: a boosting approach to graph structure learning. The resulting graph structure is an undirected network that allows exploratory analysis of gene expression data. Please note that the gene interactions in the inferred network should not be interpreted as pathways due to the lack of information on the underlying translation of mRNA to proteins and the activation states of these proteins. The BoostiGraph algorithm is conceptually simple and computationally very efficient as it relies only on univariate regressions. We have demonstrated in simulations that BoostiGraph appears highly competitive with GeneNet, glasso and LARS. However, there are still several open questions. In particular, it would be interesting to study the theoretical properties of the resulting algorithm and to establish a stopping rule allowing us to obtain a consistent estimate of the graph. Furthermore, it would also be worth investigating that how the BoostiGraph and other regression-based approaches fare against those based on entirely different philosophies, e.g. Belief Propagation Networks (Braunstein *et al.*, 2008), as well as fully Bayesian and non-approximate methods (Dobra *et al.*, 2003; Mukherjee and Speed, 2008). However, in this article, our focus was to introduce a novel boosting approach for network inference and to demonstrate its performance in comparison with other linear methods.

We have also shown the how the BoostiGraph algorithm can be extended to incorporate prior knowledge about the network structure. We compared the performance of our method with the *local model* approach proposed by Bleakley *et al.* (2007). The main advantage of this method is that through the use of kernels this method is capable of handling vectorial as well as non-vectorial data to represent the information on the nodes by the use of the *kernel trick* (Vapnik, 1998). For instance, in the case of inferring biological networks, this *local model* can handle a variety of data types, including biological sequences and molecular structures (Bleakley *et al.*, 2007). However, selecting the appropriate kernel for the data at hand is also one of its main challenges.

In addition, this *local model* approach does require complete information on all nodes in the training set and can only infer new edges between nodes in the training set and the test set. In contrast, the *modified* BoostiGraph algorithm is able to work with any amount of prior information regarding the pertinent network structure and is capable of inferring new edges over the entire input space.

From a methodological point of view, it would be interesting to extend the BoostiGraph approach to non-Gaussian mixed variable graphical models, as the univariate nature would allow for efficient fitting of the generalized autoregressions (for different variable types). An effective approach proposed in Heckerman *et al.* (2000) relies on learning a dependency network; that is, we independently learn the conditional distribution of a node given the remaining modes. In the discrete case, we can represent these distributions using logistic regression. It is fairly straightforward to extend BoostiGraph to this and other classes of problems such as automatic speech recognition (Bilmes, 2004), image analysis (Freeman *et al.*, 2000) and for industrial planning on complex domains (Gebhardt *et al.*, 2006).

ACKNOWLEDGMENTS

The authors would like to acknowledge support from the Oxford-Man Institute, Oxford, UK.

Funding: Medical Research Council, Harwell, Oxfordshire, OX11 0RD, U.K.

Conflict of Interest: none declared.

REFERENCES

Banerjee,O. *et al.* (2008) Model selection through sparse maximum likelihood estimation. *J. Mach. Learn. Res.*, **9**, 485–516.

Ben-Hur,A. and Noble,W.S. (2005) Kernel methods for predicting protein-protein interactions. *Bioinformatics*, **21**, i38–i46.

Bleakley,K. *et al.* (2007) Supervised reconstruction of biological networks with local models. *Bioinformatics*, **23**, i57–i65.

Bilmes,J. (2004) Graphical models and automatic speech recog- nition. In Johnson,M. *et al.* (eds) *Mathematical Foundations of Speech and Language Processing*. Springer, New York.

Braunstein,A. *et al.* (2008) Gene-network inference by message passing. *J. Phys.*, **95**, 012016.

Bühlmann,P. (2003) Boosting methods: why they can be useful for high-dimensional data. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. National Academy Press, Austria-Vienna.

Bühlmann,P. (2006) Boosting for high-dimensional linear models. *Ann. Stat.*, **34**, 559–583.

Bühlmann,P. and Yu,B. (2003) Boosting with the L2 loss: regression and classification. *J. Am. Stat. Assoc.*, **98**, 324–339.

Chapelle,O. *et al.* (2006) *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Dempster,A.P. (1972) Covariance selection. *Biometrika*, **32**, 95–108.

Dobra,A. *et al.* (2003) Sparse graphical models for exploring gene expression data. *J. Multivar. Anal. (special issue on Multivariate Methods in Genomic Data Analysis)*, **90**, 196–212.

Dobra,A. *et al.* (2004) Sparse graphical models for exploring gene expression data. *J. Multivar. Anal.*, **90**, 196–212.

Dougherty,M. *et al.* (2005) Regulation of Raf-1 by direct feedback phosphorylation. *Mol. Cell*, **17**, 215–224.

Efron,B. *et al.* (2004) Least angle regression. *Ann. Stat.*, **32**, 407–499.

Freeman,W.T. *et al.* (2000) Learning low-level vision. *Int. J. Comput. Vis.*, **40**, 2547.

Friedman,N. *et al.* (2000a) Using Bayesian networks to analyze gene expression data. *J. Comput. Biol.*, **7**, 601–620.

Friedman,J. *et al.* (2000b) Additive logistic regression: a statistical view of boosting. *Ann. Stat.*, **28**, 337–407.

Friedman,J. *et al.* (2008) Sparse inverse covariance estimation with the graphical lasso. *Ann. Stat.*, **9**, 432–441.

Gebhardt,J. *et al.* (2006) Graphical models for industrial planning on complex domains. In Riccia,G.D. *et al.* (eds) *Decision Theory and Multi-Agent Planning, CISM Courses and Lectures*, **482**. Springer-Verlag New York, Inc, Secaucus, pp. 131–143.

Gentry,J. *et al.* (2004) Laying out pathways with Rgraphviz. *R News*, **4**, 14–18.

Heckerman,D. *et al.* (2000) Dependency networks for density estimation, collaborative filtering and data visualization. *J. Mach. Learn. Res.*, **1**,49–75.

Jeong,H. *et al.* (2001) Lethality and centrality in protein networks. *Nature*, **411**, 41–42.

Jones,B. *et al.* (2005) Experiments in stochastic computation for high-dimensional graphical models. *Stat. Sci.*, **20**, 388–400.

Jones,P. *et al.* (1992) DNA gyrase, CS7.4 and the cold shock response in Escherichia coli. *J. Bacteriol.*, **174**, 5798–5802.

Kato,T. *et al.* (2005) Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, **21**, 2488–2495.

La Teana,A. *et al.* (1991) Identification of a cold shock transcriptional enhancer of the Escherichia coli gene encoding nucleoid protein H-NS. *Proc. Natl Acad. Sci. USA*, **88**, 10907–10911.

Lenkoski,A. and Dobra,A. (2008) Bayesian structural learning and estimation in Gaussian graphical models. *Technical Report No. 545*, Department of Statistics, University of Washington.

Li,F. and Yang,Y. (2005) Using modified lasso regression to learn large undirected graphs in a probabilistic framework. In *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005*, Pittsburgh, PA, pp. 801–806.

Meinshausen,N. and Bühlmann,P. (2006) High-dimensional graphs and variable selection with the lasso. *Ann. Stat.*, **34**, 1436–1462.

Mukherjee,S. and Speed,T.P. (2008) Network inference using informative priors. *Proc. Natl Acad. Sci. USA*, **105**, 14313–14318.

Nakashima,K. *et al.* (1996) A novel member of the cspA family of genes that is induced by cold shock in Escherichia coli. *J. Bacteriol.*, **178**, 2994–2997.

Sachs,K. *et al.* (2005) Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, **308**, 523–529.

Schäfer,J. and Korbinian,S. (2005) An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, **21**, 754–764.

Schmidt,M. *et al.* (2007) Learning graphical model structure using l1-regularization paths. In *The Twenty-Second Conference on Artificial Intelligence(AAAI)*. Vancouver, British Columbia, pp. 1278–1283.

Schmidt-Heck,W. *et al.* (2004) Reverse engineering of the stress response during expression of a recombinant protein. In *Proceedings of the EUNITE Symposiumé*. Verlag Mainz, Aachen, pp. 407–412.

Schwartz,G. (1978) Estimating the dimension of a model. *Ann. Stat.*, **6**, 461–464.

Segal,E. *et al.* (2003) Module networks:identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat. Genet.*, **34**, 166–176.

Toh,H. and Horimoto,K. (2002) Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. *Bioinformatics*, **18**, 287–297.

Tibshirani,R. (1996) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B*, **58**, 267–288.

Vapnik,V.N. (1998) *Statistical Learning Theory.* Wiley, New York.

Vert,J.P. and Yamanishi,Y. (2005) Supervised graph inference. In Lawrence,K. *et al.* (eds) *Advances in Neural Information Processing Systems 17 (NIPS 2004)*. MIT Press, Cambridge, MA, pp. 1433–1440.

Yamanishi,Y. *et al.* (2004) Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, **20**, 363–370.

Yuan,M. (2006) Efficient computation of the l1 regularized solution path in Gaussian graphical models. *J. Comput. Graph. Stat.*, **17**, 809–826.

Yuan,M. and Lin,Y. (2007) Model selection and estimation in the Gaussian graphical model. *Biometrika*, **94**, 19–35.

Waukau,J. and Forst,S. (1992) Molecular analysis of the signaling pathway between EnvZ and OmpR in Escherichia coli. *J. Bacteriol.*, **174**, 1522–1527.

Werhli,A.V. *et al.* (2006) Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks. *Bioinformatics*, **22**, 2523–2531.