scientific reports

OPEN

KBoost: a new method to infer gene regulatory networks from gene expression data

Luis F. Iglesias-Martinez[1]✉, Barbara De Kegel[1,2] & Walter Kolch[1,3]

Reconstructing gene regulatory networks is crucial to understand biological processes and holds potential for developing personalized treatment. Yet, it is still an open problem as state-of-the-art algorithms are often not able to process large amounts of data within reasonable time. Furthermore, many of the existing methods predict numerous false positives and have limited capabilities to integrate other sources of information, such as previously known interactions. Here we introduce KBoost, an algorithm that uses kernel PCA regression, boosting and Bayesian model averaging for fast and accurate reconstruction of gene regulatory networks. We have benchmarked KBoost against other high performing algorithms using three different datasets. The results show that our method compares favorably to other methods across datasets. We have also applied KBoost to a large cohort of close to 2000 breast cancer patients and 24,000 genes in less than 2 h on standard hardware. Our results show that molecularly defined breast cancer subtypes also feature differences in their GRNs. An implementation of KBoost in the form of an R package is available at: https://github.com/Luisiglm/KBoost and as a Bioconductor software package.

Gene regulatory networks (GRNs) are models that describe how transcription factors (TFs) orchestrate the expression of other genes[1–3]. In GRNs the nodes are TFs and genes, and the edges represent regulatory interactions. TFs bind to the promoters of genes to activate or silence the production of mRNA[4]. By doing so, TFs help control gene expression and thus modulate or enable cellular processes[5]. Most existing approaches of studying gene regulation focus on the effect of a single TF on the rest of the genes. However, TFs can regulate the expression of other TFs and thus a perturbation of a single TF can propagate information throughout the whole system. The potential for a single TF to control a system depends on certain network features: a TF that regulates many genes will obviously have a large influence, but the contrary is not necessarily true. Even if a TF regulates only a small number of genes directly, it can still play a major role in the system if these genes in turn regulate many others. This is the concept of closeness centrality in graph theory. Therefore, understanding a system's GRN architecture can reveal the TFs that are important for a specific phenotype and thus can explain molecular pathogenesis and reveal potential drug targets.

Several methods have been developed to predict GRNs using gene expression data[2,3,6–9]. These methods formulate the problem of inferring a GRN as an unsupervised classification problem. Typically, a GRN is formulated as a weighted graph, trained on unlabeled data, whose edge values represent the predicted probability that each TF regulates other genes (including other TFs).

Several groups have used different algorithms based on different mathematical formulations to infer GRNs from gene expression data. These include Bayesian networks, correlation metrics, mutual information methods and parametric and non-parametric regression. A seminal paper published in 2012 showed that correlation, mutual information and Bayesian networks tended to perform far worse than methods based on regression[3]. For this reason, in this work we focused only on regression based GRN inference methods.

Regression based GRN inference methods build a mathematical model of the expression levels of a target gene given the expression levels of different TFs. The central assumption in these methods is that if the expression level of a TF predicts the expression level of a target gene it is likely regulating it. The model goodness of fit is used to estimate the probability that the TFs in a model regulate the target gene. That is, if the expression of a TF poorly predicts the expression of a gene, then it is not likely to regulate it and vice versa. Some of the popular regression based GRN inference algorithms are summarized in Fig. 1.

[1]Systems Biology Ireland, School of Medicine, University College Dublin, Belfield, Dublin 4, Republic of Ireland. [2]School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland. [3]Conway Institute of Biomolecular and Biomedical Research, University College Dublin, Belfield, Dublin 4, Republic of Ireland. ✉email: luis.iglesiasmartinez@ucd.ie

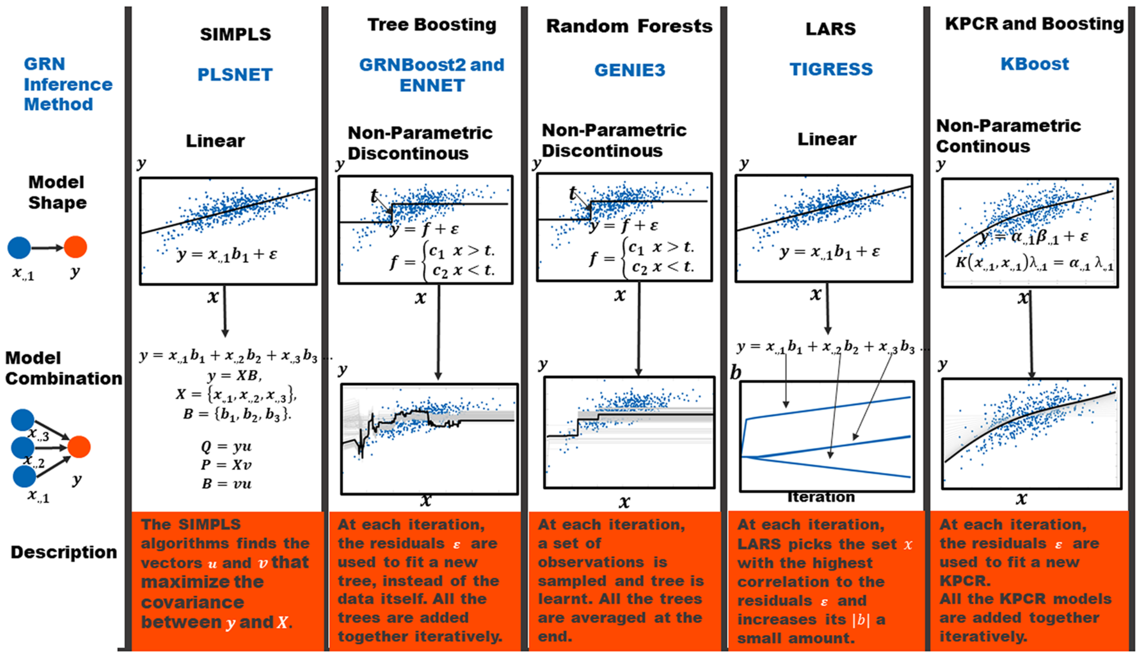
GRN Inference Method
Model Shape
Model Combination
Description
SIMPLS
PLSNET
Linear
$y = x_{.1}b_1 + \varepsilon$
$y = x_{.1}b_1 + x_{.2}b_2 + x_{.3}b_3$
$y = XB$, $X = \{x_{.1}, x_{.2}, x_{.3}\}$, $B = \{b_1, b_2, b_3\}$.
$Q = yu$
$P = Xv$
$B = vu$
The SIMPLS algorithms finds the vectors u and v that maximize the covariance between y and X.
Tree Boosting
GRNBoost2 and ENNET
Non-Parametric Discontinous
$y = f + \varepsilon$
$f = \begin{cases} c_1 & x > t. \\ c_2 & x < t. \end{cases}$
At each iteration, the residuals ε are used to fit a new tree, instead of the data itself. All the trees are added together iteratively.
Random Forests
GENIE3
Non-Parametric Discontinous
At each iteration, a set of observations is sampled and tree is learnt. All the trees are averaged at the end.
LARS
TIGRESS
Linear
Iteration
At each iteration, LARS picks the set x with the highest correlation to the residuals ε and increases its $|b|$ a small amount.
KPCR and Boosting
KBoost
Non-Parametric Continous
$y = \alpha_{.1}\beta_{.1} + \varepsilon$
$K(x_{.1}, x_{.1})\lambda_{.1} = \alpha_{.1}\lambda_{.1}$
At each iteration, the residuals ε are used to fit a new KPCR. All the KPCR models are added together iteratively.

Figure 1. Regression-based Gene Regulatory Network Inference Algorithms. Schematic describing how different regression based GRN inference methods work. These methods are based on different machine learning algorithms. We show six methods based on different machine learning algorithms that differ on the model shape and the way models for different TFs are combined. They were selected because they represent major types of machine learning methods used for GRN reconstruction and because of their high performance in the DREAM 4 and DREAM 5 challenges. PLSNET uses partial least squares and fits a linear model between TFs and targets. TIGRESS uses a linear model with different lasso parameters They both rely on the assumption that the expression of a gene is proportional to the expression levels of the TFs that regulate it. GRNBoost2 and ENNET use boosting to learn different tree models between TFs and targets. GENIE3 also uses tree models, however they iteratively resample different subsets of observations and potential TFs per target and create an ensemble of tree models. Unlike linear models, tree models do not rely on any assumption between the relationship of a TF and a target, however they are not continuous models.

Parametric GRN inference algorithms use a mathematical formulation that describes gene regulation using a predefined expected form. The most popular algorithms belonging to this class are linear regression based. In other words, they assume that the expression of a target gene is a linear combination of the expression levels of the TFs that regulate it[8–11]. PLSNET and TIGRESS both use a linear regression approach[8,9]. PLSNET uses a form of partial least squares known as SIMPLS[8], while TIGRESS uses the least angle regression algorithm (LARS)[9]. LARS is an effective approach for obtaining different solutions for the linear regression coefficients with different absolute norms[12]. This algorithm takes an iterative approach: at each step, the residuals between the response variable and the linear regression model are compared with the explanatory variables, and because the models are trained iteratively on the residuals it can be described as a boosting algorithm[12]. TIGRESS couples LARS with stability selection and uses the frequency with which each TF is chosen in iterations of LARS as a proxy for the predictive probability that a TF regulates a gene[9]. The effectiveness of linear model-based approaches relies on how well a target gene's expression levels can be approximated using linear functions.

For this reason, nonparametric regression methods have been proposed. In particular, regression trees have been very successful for GRN inference in different datasets[2,6,7]. Regression trees separate the explanatory variables into regions and give the response variable a value depending on what region of the explanatory variables each observation resides in[13,14]. The GENIE3 method uses the regression version of the random forest algorithm[2,13]—this algorithm iteratively samples different observations and explanatory variables and builds regression trees with a fixed depth. The different trees' predictions are then combined to form the final prediction[13]. GENIE3 runs the random forest algorithm once per target gene. At each step, the variable importance of each TF is calculated as the proportion of explained variance of the target gene. The GENIE3 algorithm won two important GRN inference challenges, namely the DREAM 4 multifactorial sub-challenge and the DREAM 5 challenge[2]. The datasets for these challenges are now widely used benchmarks to test the performance of different GRN inference algorithms. Besides the random forest approach, regression tree models can also be trained using a stochastic gradient boosting approach. In the context of GRN inference, ENNET and GRNBoost2 are based on regression tree stochastic gradient boosting[6,7]. Stochastic gradient boosting is a form of a gradient based function optimization applied on model space[14]. Very generally speaking, tree boosting is similar to LARS except that it uses tree models instead of linear models[12].

Here we present KBoost, a method that uses kernel PCA (Principal Component Analysis) regression and gradient boosting to reconstruct gene regulatory networks. Kernel PCA regression (KPCR) is a nonparametric regression technique that does not require the data to follow a strict form[15,16]. This is accomplished by

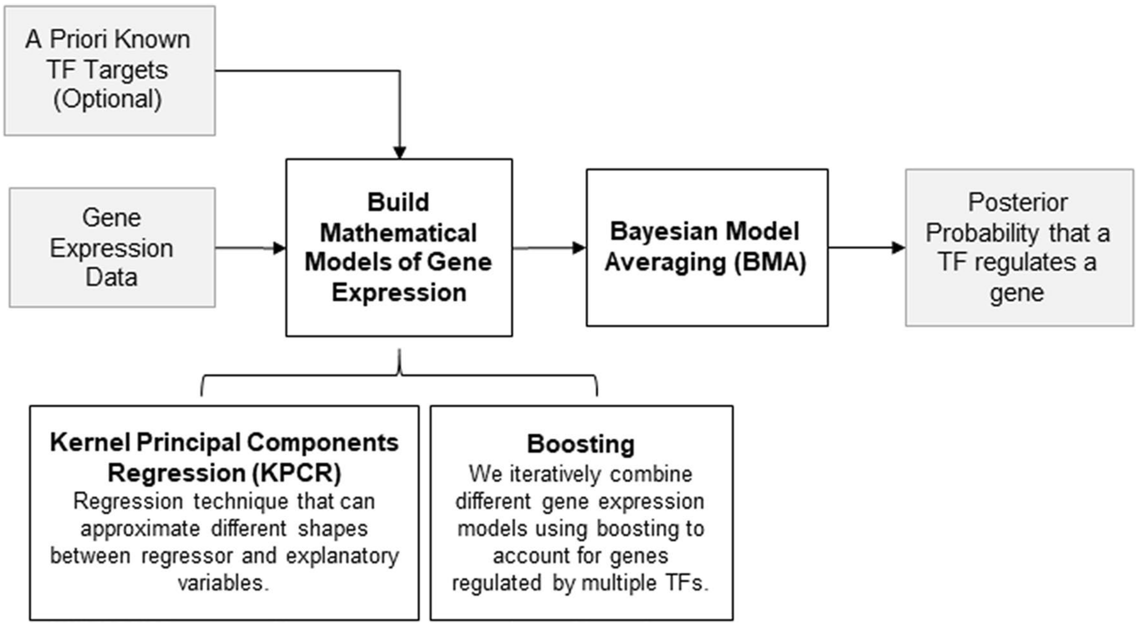
A Priori Known TF Targets (Optional)
Gene Expression Data
Build Mathematical Models of Gene Expression
Bayesian Model Averaging (BMA)
Posterior Probability that a TF regulates a gene
Kernel Principal Components Regression (KPCR)
Regression technique that can approximate different shapes between regressor and explanatory variables.
Boosting
We iteratively combine different gene expression models using boosting to account for genes regulated by multiple TFs.

Figure 2. Overview of KBoost. KBoost uses KPCR in a boosting framework to infer GRNs from gene expression data. KBoost uses a list of predefined TFs and builds KPCR models to predict the expression of other genes (including TFs). These models are combined using gradient boosting. The residuals are used to estimate the probability that a TF regulates a gene.

transforming the explanatory variables, in our case the TF expression levels, using a kernel function. The kernel function yields a symmetric matrix that corresponds to the dot product of nonlinear features that are a function of the explanatory variables. For many kernel functions these implicit nonlinear features correspond to polynomials of the explanatory variables[17]. In fact, the radial basis kernel function (RBF) is the dot product of an infinite number of polynomials without having to calculate them explicitly[18]. Using polynomials for regression is an attractive approach, as according to the Weierstrass approximation theorem any continuous real function can be uniformly approximated by polynomials. Furthermore, kernel PCA lets us obtain the principal components of this large number of polynomials by using the kernel matrix with much smaller dimensions. Thus, with KPCR we can use the first principal components of a large number of implicit features to approximate any continuous function—in our case the relationship between a TF and a target gene.

For simplicity, instead of calculating KPCs of the expression levels of multiple TFs, we build an ensemble of models of KPCs. This is done through gradient boosting, which iteratively combines predictions from different models, in our case KPCRs from different TFs[7,14]. For each gene, we start by selecting a TF whose KPCs predict its expression levels the best according to the posterior. Then, we search among every TF's KPCs to see which can predict the residuals, that is the difference with the original prediction and the actual gene expression values. At the end of each iteration, we update the predictions, by adding the model fit on the residuals. This greedy model search that allows us to drastically reduce the running time. The posterior probabilities of the models queried are used to estimate the GRN. Because of this Bayesian formulation, we can include information from previous experimental data and increase the accuracy of our predictions[11,19]. We tested KBoost against other algorithms on several benchmark datasets. The results show that KBoost performs better than other algorithms even in the absence of prior information.

Methods

KBoost overview.

The aim of KBoost is to provide a fast and scalable algorithm for the accurate inference of GRNs (Fig. 2). It takes mRNA expression (microarray or RNA-seq) and, optionally, previously found TF-target interactions (e.g. ChIPseq) as input. KBoost uses KPCR and boosting coupled with Bayesian model averaging (BMA) to estimate the probabilities of genes regulating each other, and thereby reconstructs GRNs[10,11,20]. We use a greedy approach to sample the model space. That is, for every gene we build a mathematical model that predicts its expression using the kernel PCA of the expression levels of a likely subset of TFs. We then compare different KPCR TF-gene models and estimate the probability that a TF regulates a gene using BMA. The model fit is used as a proxy of the probability that a set of TFs regulate a specific gene. If there is information from other sources on likely TF targets, it can be combined with the model fit in the form of a prior. The output of KBoost is then an estimate of the probability that each TF regulates each gene.

Modelling gene expression. We formulate the data as n independent observations $\boldsymbol{x} = [x_1, \ldots, x_n]^T$, each with real-valued measurements of G genes: $x_i = (x_i^{(1)}, \ldots, x_i^{(G)})$. We assume that each gene's j expression, $x_i^{(j)}$, is a linear combination of non-linear functions of the expression of a subset of the gene's TFs, noted as A_j, plus some random noise $\varepsilon_i^{(j)}$, which comes from a normal distribution with zero mean and variance σ_j^2:

$$x_i^{(j)} = \sum_{p \in A_j} f(x_i^{(p)}) + \varepsilon_i^{(j)}. \tag{1}$$

Gene regulatory network inference as unsupervised classification. The gene regulatory network, $\boldsymbol{C}$, would then be defined as a directed graph with G nodes, where each node is a gene. We write $\boldsymbol{C}$ as a $G \times P$ matrix, where P is the number of genes in G that are TFs. The entry $C_{j,p}$ gets the value 1 if TF p regulates gene j and 0 otherwise. As many other algorithms, we formulate the problem of estimating $\boldsymbol{C}$ from $\boldsymbol{x}$ as an unsupervised classification problem, where the fit of different subsets of TFs in Eq. (1) is used to estimate the posterior probability that any TF regulates any gene; in other words, how well different combinations of TFs, A_j, fit the data $x_i^{(j)}$. We can then consider all the combinations of TF subsets $A_j = (A_j^{(1)}, \ldots, A_j^{(2^P)})$, where $A_j^{(d)}$ contains the integer indexes of the genes that are TFs and are part of subset d. The predictive probability of an element of $C_{j,p}$ to be 1:

$$P\left(C_{j,p} = 1|x\right) = \frac{\sum_{d:p \in A_j^{(d)}} P\left(A_j^{(d)}\right) \pi\left(A_j^{(d)}\right)}{\sum_{t=1}^{2K} P\left(A_j^{(t)}\right) \pi\left(A_j^{(t)}\right)}. \tag{2}$$

Here, $P\left(A_j^{(d)}\right)$ is the marginal likelihood of the model in Eq. (1) given the subset $A_j^{(d)}$, and $\pi\left(A_j^{(d)}\right)$ is the prior that the TFs in this subset $A_j^{(d)}$ regulate gene j. We formulate the prior probability of a subset of TFs $A_j^{(d)}$ to regulate a gene, j, as coming from a binomial distribution:

$$\pi\left(A_j^{(d)}\right) = \prod_{p=1}^{P} w_{j,p}^{1_{k \in A^{(d)}{}_j}} \left(1 - w_{j,p}\right)^{1_{k \notin A^{(d)}{}_j}}. \tag{3}$$

Here, the parameters of the prior are in a matrix $\boldsymbol{W} = [w_1, \ldots, w_G]^T$ with the same dimensions as the gene regulatory network $\boldsymbol{C}$. For simplicity the marginal likelihood, $P\left(A_j^{(d)}\right)$, is obtained by assuming $\sum_{p \in A_j} f(x_i^{(k)})$ to be fixed. This assumption facilitates the inference process and reduces the computation time required in KBoost. We give the variance of the noise Jeffreys' prior, $\left(\sigma_j^2\right) \propto \frac{1}{\sigma_j^2}$. This yields the following expression for the marginal likelihood:

$$P\left(A_j^{(d)}\right) \propto \left(\sum_{i=1}^{n} \left(x_i^{(j)} - \sum_{p \in A_j} f\left(x_i^{(k)}\right)\right)^2\right)^{\left(-\frac{n-1}{2}\right)}. \tag{4}$$

Greedy model selection and boosting. So far, we have omitted the details of two crucial tasks, first how we calculate $\sum_{p \in A_j} f\left(x_i^{(k)}\right)$, and second how we go through the different subsets $\mathrm{A}_j^{(d)}$. For the second task, we chose a greedy approach where we iteratively select the TF subsets with the highest posteriors $\mathrm{P}\left(\mathrm{A}_j^{(d)}\right)$ and expand them for each gene. This choice is motivated by two goals: (1) to reduce the computational burden of calculating $\mathrm{P}\left(\mathrm{A}_j^{(d)}\right)$ for all possible combinations of TFs, and (2) to reduce the number of false positives. Genes that are regulated by the same set of TFs will have a high correlation between their expression levels. We assume that for each gene j, there are several $\mathrm{A}_j^{(d)}$ which yield a relatively high $\mathrm{P}\left(\mathrm{A}_j^{(d)}\right)$ due to coregulation and thus we perform a greedy search keeping only the TFs that yield the highest $\mathrm{P}\left(\mathrm{A}_j^{(d)}\right)$ at each boosting iteration. This is similar to the LARS procedure, except that in Eq. (2) we include all explored models, like in the Occam Razor version of BMA[12,21]. Therefore, Eq. (5) uses only the subsets $\widehat{A}_j$ explored by our greedy search, i.e.:

$$\widehat{C}_j, p = P\left(C_{j,p} = 1|\boldsymbol{x}\right) \cong \frac{\sum_{d:p \in \widehat{A_j^{(d)}}} P\left(A_j^{(d)}\right) \pi\left(A_j^{(d)}\right)}{\sum_{\widehat{A_j^{(q)}}} P\left(A_j^{(q)}\right) \pi\left(A_j^{(q)}\right)}. \tag{5}$$

Kernel principal components. Regarding the task of calculating $\sum_{p \in A_j} f\left(x_i^{(k)}\right)$, we chose to use KPCR because it has three important advantages in a boosting framework. First, KPCR is a kernel-based regression method which can approximate any potential shape that the relationship between a TF a target gene might have. We use the RBF kernel function which is the exponential of minus the squared Euclidean distance between observations divided by a width hyperparameter. KPCR is a technique that allows to compute the principal components in the infinite dimensional feature space directly from the kernel matrix. A more detailed description is presented in the appendix. Second, unlike other kernel regression methods, which would need an $n \times n$ kernel matrix per TF, in KPCR we only need a few principal components per TF. This means a substantial reduction in memory usage while trying out different subsets $A_j^{(d)}$. Finally, the boosting framework allows us to effectively combine principal components from kernels of different TFs. The formulation is shown below.

For each TF p, we first calculate the RBF kernel with width parameter γ:

$$K^{(p)} \in \mathbb{R}^{n \times n}, K_{i,u}^{(p)} = exp\left(-\frac{\left(x_i^{(p)} - x_u^{(p)}\right)^2}{\gamma}\right) \tag{6}$$

Then, we normalize the covariance matrix of the infinite dimensional features implicitly:

$$\overline{K^{(p)}} = K^{(p)} - 1_{nn}K^{(p)} - K^{(p)}1_{nn} + 1_{nn}K^{(p)}1_{nn} \tag{7}$$

here, 1_{nn} is an $n \times n$ matrix with all the values equal to $\frac{1}{n}$. Next, we perform an eigendecomposition of $\overline{K}$:

$$\overline{K^{(p)}}\alpha^{(p)} = \overline{K^{(p)}}\lambda^{(p)}, \tag{8}$$

where α are the eigenvectors and λ is a diagonal matrix with the eigenvalues in descending order The first z KPCs are then obtained by

$$g\left(x_i^{(p)}\right) = \overline{K_i^{(p)}}\alpha_{1:z}^{(p)}. \tag{9}$$

In KBoost we select KPCs by using a threshold on their corresponding eigenvalues. The KPCs are then normalized so that they have unit norm. After obtaining $g\left(x_i^{(p)}\right)$, the matrices $\overline{K^{(p)}}$ and $K^{(p)}$ can be overwritten and thus this procedure allows us to reduce the amount of memory required. Note that the columns of $g\left(x_i^{(p)}\right)$ are orthogonal to each other. This means that for a model with one TF, $A_j^{(d)} = \{p\}$, we can fit the model in Eq. (1) by:

$$\begin{aligned} x_i^{(j)} &= \sum_{p \in A_j} f\left(x_i^{(p)}\right) + \varepsilon_i^{(j)} = \widehat{\beta}_j^{(p)T} g\left(x_i^{(p)}\right) + \varepsilon_i^{(j)}, \\ \widehat{\beta}_j^{(p)} &= \nu\lambda^{*(p)} g\left(x^{(p)}\right)^T \varepsilon^{(j)}. \end{aligned} \tag{10}$$

here, $\widehat{\beta}_j^{(p)}$ are the coefficients of the KPC regression multiplied by a constant $0 < \nu \leq 1$, and $\lambda^{*(p)}$ is a diagonal matrix whose entries are the reciprocals of the eigenvalues that correspond to the KPCs. The constant ν is known as a shrinkage parameter. We propose using ν to counteract the effect that the number of observations n has on the marginal likelihood. Since the marginal likelihood changes exponentially to the power of $(n-1)/2$, the resulting GRNs would change dramatically depending on n. Namely, with a large n, most TFs posterior probability would be close to zero while a few would be close to one. This can lead to an overestimation of the confidence we have on the predicted GRN. By shrinking the estimates of $\widehat{\beta}_j^{(p)}$, we introduce a heuristic measure that will make the resulting sum of squares less different between TF subsets, $A_j^{(d)}$, and thus reduce the effect of n.

Boosting. The last equation (Eq. 10) shows how we fit the model in Eq. (1) in the case that $A_j^{(d)}$ contains 1 TF. For simplicity we then follow a greedy boosting approach that is very similar to the least angle regression method. In KBoost, we first fit every single TF subset $A_j^{(d)}$ (i.e. only one TF in each model) and calculate their unnormalized posteriors. Then, we select the TF with the highest unnormalized posterior, p^*, and calculate its residuals, $\varepsilon^{(j)}$. Next, we form new subsets by iteratively adding a TF, and fit the TFs to the residuals:

$$\begin{aligned} &\varepsilon_i^{(j)} = x_i^{(j)} - \widehat{\beta}_j^{(p^*)} g\left(x_i^{(p^*)}\right), \\ &\hat{\beta}_j^{(p)} = \nu\lambda^{*(p)} g\left(x^{(p)}\right)^T \varepsilon^{(j)}, \\ &A_j^{(d)} = \{p^*, p\}, \\ &\sum_{p \in A_j} f\left(x_i^{(p)}\right) = \widehat{\beta}_j^{(p)T} g\left(x_i^{(p)}\right) + \widehat{\beta}_j^{(p^*)T} g\left(x_i^{(p^*)}\right). \end{aligned} \tag{11}$$

Similarly, to the previous step, we calculate their unnormalized posterior and select the TF with the highest posterior to expand next (Fig. 3). When we use a ν under 1, there is a possibility that the same TF will be chosen more than once. This might reflect a TF having a very strong relationship with a particular target gene. At the

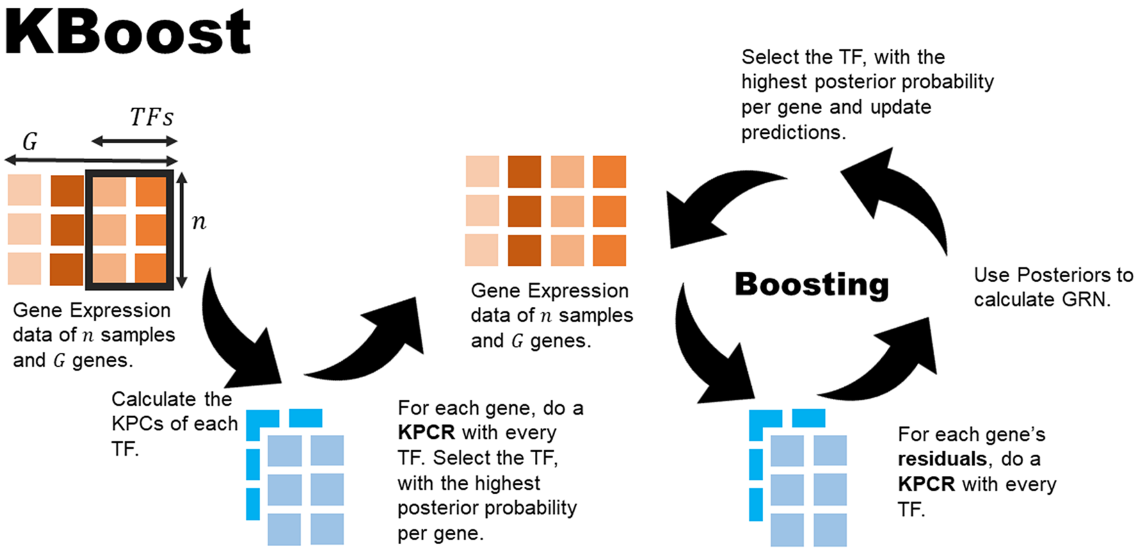
KBoost
TFs
G
n
Gene Expression data of n samples and G genes.
Calculate the KPCs of each TF.
For each gene, do a **KPCR** with every TF. Select the TF, with the highest posterior probability per gene.
Gene Expression data of n samples and G genes.
Select the TF, with the highest posterior probability per gene and update predictions.
Boosting
Use Posteriors to calculate GRN.
For each gene's **residuals**, do a **KPCR** with every TF.

Figure 3. Gene Expression Modelling with KPCR and Boosting. The procedure starts by selecting the TF whose KPCR model has the highest posterior for each gene. Then we combine different KPCR models by boosting. At each boosting iteration we fit a new KPCR model to each gene expression's residuals instead of the actual values. We then select the TF whose KPCR model has the highest performance in each case and update the KPCR model ensemble.

end we estimate $P\left(C_{j,p} = 1|x\right)$ by performing BMA on all explored models. In KBoost, the number of iterations represents also the maximum number of TFs per gene to be considered. The resulting network is tuned using the heuristic proposed by Slawek and Arodz[7], in which we multiply each column p of $P\left(C_{j,p} = 1|x\right)$ by the sample variance of the whole column. This has the effect of preserving the sparse outdegree distribution found in most biological networks. KBoost is available as an R package at: https://github.com/Luisiglm/KBoost and as a Bioconductor software package.

Performance assessment. We compared KBoost against five other GRN reconstruction methods that were chosen because of their high performance in previous benchmarking exercises, such as the DREAM 4 and DREAM 5 challenges. Specifically, the algorithms included are GENIE, PLSNET, ENNET, TIGRESS, and GRNBoost2[2,6–8]. All algorithms were run using their default parameters, including KBoost $(if\ n > 10 : v = \frac{10}{n}, else\ v = 0.5,, \gamma = 60$ and 3 iterations). We used three different datasets to assess the comparative performance of our algorithm: IRMA[22], the in silico networks from the DREAM 4 multifactorial sub-challenge[2], and the DREAM 5 dataset, which contains three sets of gene expression data—one from a simulated network and two from real-life networks[3]. Before the analysis we standardized each gene's expression to a sample variance of one and a sample mean of zero. For this comparison, we did not use any prior information on the relationships between TFs and potential targets. Therefore, we gave every TF a probability of 0.5 of regulating any gene, so that every possible combination of TFs had the same prior probability.

Algorithm 1. KBoost

Input: Zero-centered and unit variance gene expression data $\boldsymbol{x}$ for G genes in N samples, $\{P\}$ a subset of G which are TFs, shrinkage factor $0 < v < 1$, maximum number of iterations ite, RBF-kernel parameter γ, and a matrix $\boldsymbol{W}$ with the weights of the prior probability that a TF j regulates gene i, $w_{j,i}$.
Output: A GRN $\widehat{\boldsymbol{C}}$, with the posterior probability that each TF regulates each gene in the system.

```
For $p$ in the indexes of G that are TFs do:
  Calculate the feature-centralized RBF kernel $\overline{K_\iota^{(p)}}$
  Perform the eigen decomposition of $\overline{K^{(p)}}\alpha^{(p)} = \overline{K^{(p)}}\,\lambda^{(p)}$
  Calculate the KPCs for $p$, $g\left(x_i^{(p)}\right) = \overline{K_\iota^{(p)}}\alpha_{1:z}^{(p)}$
End for

Initialize $\varepsilon = x$ and $A^* = [A^{*(0)}_1, \dots, A^{*(0)}_G]$, $A^{*(0)}_j = (\emptyset)$ (at each iteration new subsets will be created).
For t ite do:
  For $j$ in the indexes of G do:
    For p in indexes of G that are TFs and $p \neq j$ in case $j$ is a TF:
      Create a new subset $S_j^{(p)} = A^{*(t)}_j \cup \{p\}$.
      Calculate the regression coefficients $\hat{\beta}_j^{(p)} = \nu\lambda^{*(p)}\, g(x^{(p)})^{T} \varepsilon^{(p)}$
      Calculate expression estimate $\sum_{p\in S^{(p)}_j} f(x^{(p)}) = \sum_{q\in A^{*(t)}_j} f(x^{(q)}) + g(x^{(p)})\hat{\beta}_j^{(p)}$
      Calculate the unnormalized posterior probability $P(S_j^{(p)})\pi(S_j^{(p)})$
    End for
    Update $A_j^{(t+1)} = [A_j^{(t)}, S_j]$
    Define a new $A^{*(t+1)}_j = argmax_{S_j}(P(S^{(p)}{}_j)\pi(S^{(p)}{}_j))$
  End for
End for

Perform the Bayesian Model Averaging $\widetilde{C_{J,p}} = \frac{\sum_{d:p\in \widehat{A_j^{(d)}}} P\left(A_j^{(d)}\right)\pi\left(A_j^{(d)}\right)}{\sum_{\widehat{A_j^{(q)}}} P\left(A_j^{(q)}\right)\pi\left(A_j^{(q)}\right)}$.

Perform Slawek and Arodz Adjustment $\widehat{C_{,p}} = \tilde{}, p\frac{1}{(P-1)}€\sum_{j=1}^{G}(\widetilde{C_{J,p}} - \sum_{j=1}^{G}(\widetilde{C_{J,p}}))^2$
```

Benchmarking consisted of comparing a predicted GRN with the real GRN using common performance metrics such as the area under the precision recall curve (AUPR) and the area under the receiver operator curve (AUROC). Both methods work by changing a threshold of probability from close to 0 to 1. Regulatory interactions with a probability higher than the threshold are kept while those below the threshold are discarded. The inferred regulatory interactions are compared with the real GRN. To obtain AUPR we calculate the precision (number of correctly predicted interactions divided by the total number of predicted interactions) and the recall (number of correctly predicted interactions divided by the total number of actual interactions) at each threshold. Then, both precision and recall are plotted and the area under the curve is calculated. Similarly, for AUROC, at each threshold we calculate the recall and the false positive rate (number of incorrectly predicted interactions divided by the actual number of non-interactions). The recall is plotted against the false positive rate and the area under the curve is calculated. The AUROC value ranges from 0 to 1; 0.5 represents the number that would be achieved by random guessing and 1 is the perfect score[23]. Besides using the AUROC and AUPR, we also use the specific scores developed for the DREAM 4 and DREAM 5 challenges. The authors of the DREAM 4 and DREAM 5 challenge devised a score to benchmark different methods on their datasets. They calculated these scores using the mean of the log10 of the empirical P-value to guess a network randomly of the same accuracy as the one inferred by each specific method[3]. For consistency, we used the DREAM 4 and DREAM 5 evaluation scripts to compare the different algorithms.

GRN inference method	IRMA switch on		IRMA switch off	
	AUROC	AUPR	AUROC	AUPR
KBoost (2020)	0.67	0.31	**0.83**	**0.52**
GRNBoost2 (2019)	0.58	0.21	0.63	0.29
PLSNET (2016)	**0.74**	**0.34**	0.76	0.51
ENNET (2013)	0.56	0.25	0.82	0.49
TIGRESS (2012)	0.61	0.28	0.71	0.39
GENIE3 (2010)	0.67	0.30	0.82	0.46

Table 1. IRMA dataset benchmark. The best performance in each column is in bold.

GRN inference method	Net1		Net2		Net3		Net 4		Net 5		
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	Score
KBoost (2020)	0.74	0.15	**0.85**	**0.31**	**0.84**	0.25	**0.82**	0.27	**0.85**	**0.32**	**55.93**
GRNBoost2 (2019)	0.64	0.08	0.65	0.10	0.70	0.17	0.69	0.14	0.72	0.11	20.78
PLSNET (2016)	0.70	0.13	0.83	0.28	0.79	0.21	**0.82**	0.21	0.78	0.18	49.08
ENNET (2013)	0.73	**0.17**	0.81	0.26	0.81	**0.29**	**0.82**	**0.29**	0.82	0.28	52.66
TIGRESS (2012)	0.61	0.06	0.60	0.08	0.60	0.06	0.69	0.09	0.67	0.11	12.66
GENIE3 (2010)[a]	**0.75**	0.16	0.73	0.16	0.78	0.23	0.79	0.21	0.80	0.20	37.72

Table 2. DREAM 4 dataset benchmark. [a]Winner of the DREAM 4 challenge which included 11 algorithms. The best performance in each column is in bold.

Results

Performance assessment of KBoost.

IRMA dataset. The IRMA dataset is a small experimental synthetic network in yeast, which is composed of five genes that regulate each other and are responsive to galactose and glucose[22]. To implement IRMA in vivo, yeast cells were transduced with expression vectors that contained five single TF promoters linked to TF encoding genes. The five genes used were ASH1, GAL80, SWI5, GAL4 and CBF1. The network can be activated by the addition of galactose and silenced by glucose. Thus, this dataset contains two small steady state sub-datasets: IRMA switch on, and IRMA switch off, respectively. The expression of the five genes was measured by q-PCR. Both IRMA switch on and IRMA switch off contain 24 observations[22]. We used the IRMA switch off dataset to select the default values of the parameters for KBoost, and these were kept fixed for the rest of the benchmark performance assessments (see "Supplementary data S1"). The results show that KBoost performs similarly well in both datasets (Table 1). Furthermore, it had the highest performance in both metrics for the IRMA switch off dataset.

DREAM 4 multifactorial challenge dataset. The DREAM 4 dataset contains several different network inference challenges. We focused only on the multifactorial sub-challenge as it more closely resembles gene expression data from patients[2]. This challenge consists of five simulated steady-state expression profiles of 100 genes in 100 samples that were obtained by perturbing all genes simultaneously. The expression levels in the dataset correspond to steady state levels after perturbations with added normal and log-normal noise to resemble a real-life microarray experiment. The results show that KBoost, using the default parameters, generally achieved the highest performance, compared to state-of-the-art algorithms (Table 2). GENIE3, the original sub-challenge winner, had obtained a score of 37.5, whereas KBoost scores 55.93. This score is also clearly higher than that achieved by the other algorithms, which were developed after GENIE3. The results show that for several networks KBoost achieved the highest AUROC and AUPR.

DREAM 5 dataset. The DREAM 5 dataset contains three sets of gene expression data, one from a simulated network (Net 1) and two from real-life networks. The three gene expression sets contain data simulated or obtained from different kinds of experiments such as drug perturbations, genetic perturbations (gene deletions or overexpression), time series sets and steady-state sets. The first network is a simulated network with 195 TFs and 1643 genes. The second network is from *Escherichia coli* with 334 TFs and 4511 genes (Net 3), and the third network is from *Saccharomyces cerevisiae* and has 333 TFs and 5950 genes (Net 4)[3]. The results show that KBoost compares favorably to most algorithms and has a similar overall performance as ENNET, a tree gradient boosting algorithm (Table 3). The results also show that most algorithms did not perform well on the experimental networks, Net 2 and Net 3.

Running time. The boosting framework of KBoost allows us to dramatically reduce the number of computational operations. That is because KBoost adds only one TF at a time, and the maximum likelihood solution to the regression of KPCs of only one TF does not require an inversion, since KPCs coming from one kernel are

	Net1		Net2[b]		Net3[b]		
GRN inference method	**AUROC**	**AUPR**	**AUROC**	**AUPR**	**AUROC**	**AUPR**	**Score**
KBoost (2020)	**0.88**	0.43	0.57	0.04	0.51	**0.02**	**> 300**
GRNBoost2 (2019)	0.82	0.33	**0.63**	**0.10**	**0.52**	**0.02**	54.30
PLSNET (2016)	0.85	0.24	0.57	0.06	0.51	0.02	37.03
ENNET (2013)	0.85	**0.44**	0.61	0.05	0.51	0.02	**> 300**
TIGRESS (2012)	0.75	0.29	0.58	0.06	0.51	0.02	22.63
GENIE3 (2010)[a]	0.82	0.29	0.62	0.09	**0.52**	**0.02**	40.74

Table 3. DREAM 5 dataset benchmark. [a]Winner of the DREAM 5 challenge which included 351 algorithms. [b]In the original DREAM 5 dataset, Net 2 and 3 are labeled 3 and 4. The best performance in each column is in bold.

	IRMA on		IRMA off						
GRN inference method	**AUROC**	**AUPR**	**AUROC**	**AUPR**	**Total running time (mins)**	**DREAM 4 M challenge**	**Total running time (mins)**	**DREAM 5**	**Total running time (mins)**
KBoost (Matlab, 2020)	0.67	0.31	**0.83**	**0.52**	**0.001**	**55.93**	**0.025**	**> 300**	**8.6**
GRNBoost2 (Python, 2019)	0.58	0.21	0.63	0.29	0.12	20.78	0.42	54.30	68.48
PLSNET (Matlab, 2016)	**0.74**	**0.34**	0.76	0.51	0.02	49.08	1.63	37.03	141.36
ENNET (R, 2013)	0.56	0.25	0.82	0.49	**0.001**	52.66	0.86	**> 300**	382.46
TIGRESS (Matlab, 2012)	0.61	0.28	0.71	0.39	0.06	12.66	0.65	22.63	455.77
GENIE3 (Matlab, 2010)	0.67	0.30	0.82	0.46	0.003	37.72	1.60	40.74	806.75

Table 4. Results summary of running time and performance. The best performance in each column is in bold.

orthogonal. The resulting worst case computational complexity is governed by the eigendecomposition of the KPCA $O(p \cdot (n^3))$ n the number of observations and p the number of TFs. This might seem extreme but typically GRN inference datasets have at most a few hundred of observations.

We ran all algorithms on the same desktop with an Intel® Core™ i7-8700 CPU with 370 GHz, 6 cores and 12 logical cores, and 32 GB of RAM. The results show that KBoost is much faster for all datasets compared to all the other algorithms. The summary of all the benchmark analysis is shown in Table 4. KBoost has the best performance for all datasets except for the IRMA switch on dataset, where it performed second best. However, KBoost dramatically shortens running times, particularly for the DREAM 5 dataset.

Effect of an informative prior network. For several organisms, many years of research have linked certain TFs with specific genes. This information can be included into KBoost in the form of the prior $\boldsymbol{W}$. In the previous sections, we used a $\boldsymbol{W}$ with 0.5 for every of its element, giving every TF-gene relationship the same prior probability. In this section, we explore what could happen in a real-life setting, if the user had prior information from previous research studies linking certain TFs to certain genes. We used the gold standards of the DREAM 4 multifactorial dataset as prior networks and added some noise by randomly modifying different fractions of the edges. We gave the edges present in the prior networks a weight of 0.6 and the edges that were absent a weight of 0.4. The results show that even with a prior network with relatively high levels of noise (up to 50% of the edges were added or deleted from the original network) the average AUPR and AUROC are higher than when a noninformative prior was used (Fig. 3). All the posterior networks had a higher AUPR than the corresponding prior networks, while for the AUROC only the prior with 5% of the edges modified had a higher AUROC in the prior than the posterior network. In general, this shows the potential to increase the accuracy and precision of the predictions by including prior information (Fig. 4).

Application to the METABRIC dataset. To examine the usefulness of KBoost to biological research we applied KBoost to the METABRIC dataset of breast cancer downloaded from cBioportal[24]. This dataset contains data for 1904 patients, of which 1898 have been stratified into subtypes. Breast cancer was one of the first cancer types where gene expression profiles were used for a molecular stratification of cancer subtypes, which correlate with disease phenotypes, prognosis and treatment response[25,26]. Studies have identified 4 major breast cancer subtypes (luminal A, luminal B, HER2-enriched and basal), a claudin-low subtype and a normal-like group. The claudin-low is sometimes described as a particularly aggressive form of the basal subtype subtype[27,28]. We wanted to test whether the subtypes have characteristic GRNs that can be used to distinguish them. For this, we reconstructed a network for each of the six subtypes using KBoost with a prior based on the GRN obtained from ChIP-Seq experiments on several TFs[29]. We gave the edges seen in the prior network a prior probability of 0.6, while for TFs in the prior network but not on our system, we gave 0.4 probability to the edges that were not observed in the ChIP-Seq. The dataset contains microarray experiments with expression levels for 24,368 genes. To identify TFs we overlapped the list of 24,368 genes in METABRIC with a previously published list of human TFs[30]. KBoost was ran on a desktop with an Intel® Core™ i7-8700 CPU with 370 GHz, 6 cores and 12 logical cores,

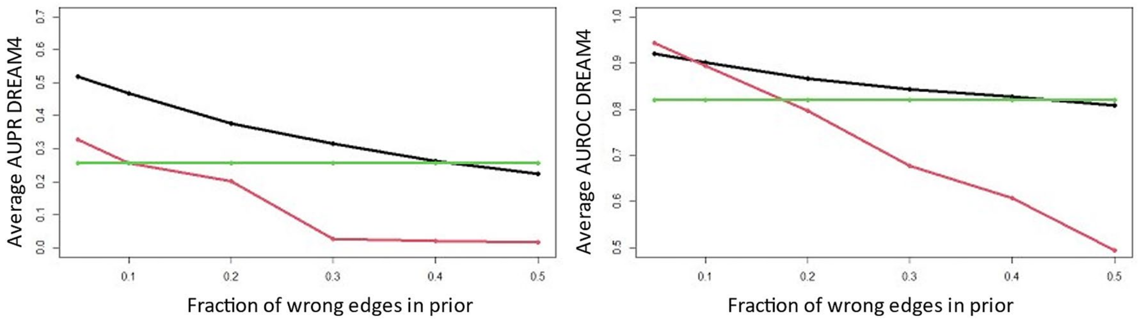

Figure 4. Effect of an Informative Prior into KBoost on the DREAM 4 Multifactorial Dataset The red line represents the average of the prior networks, the green line the average results with a non-informative prior and the black line the average of the posterior networks with an informative prior. The results highlight the advantage of having prior information even if it is limited to a few TF—target gene pairs. It also shows that when the prior information is not accurate it can have a detrimental effect on performance.

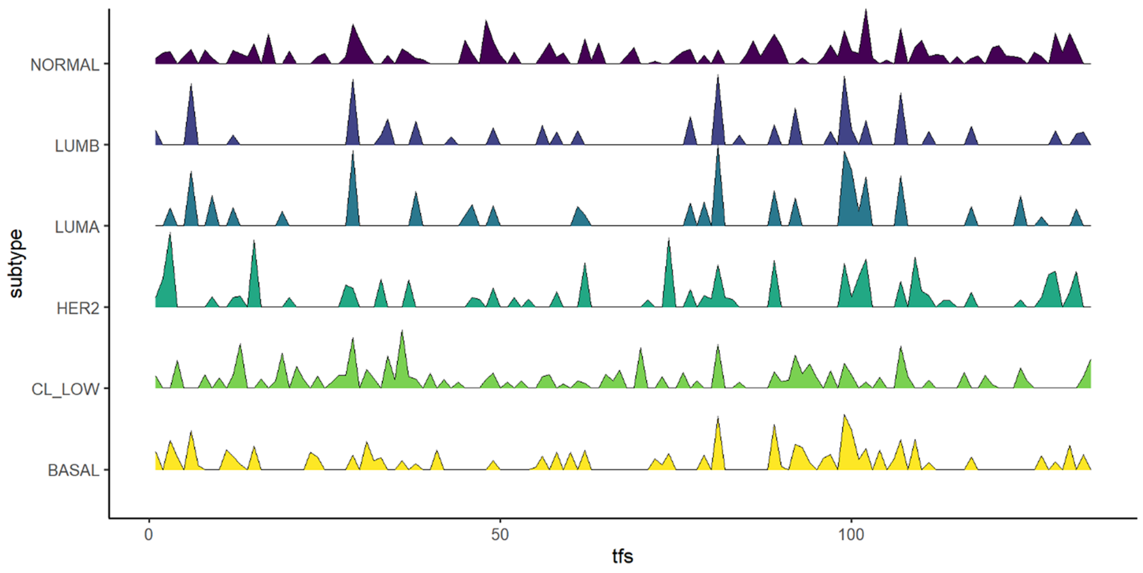

Figure 5. Closeness Centralities of Breast Cancer Subtypes. The peaks correspond to the closeness centrality of different TFs among the subtypes. The results show that breast cancer subtypes have differences in their underlying GRNs.

and 32 GB of RAM. The resulting network was filtered by keeping edges with a posterior probability higher than 0.2. This threshold was found by using the F1 metric on the IRMA Off dataset. The whole dataset of 24,368 genes by 1345 TFs and 1898 patients was run in 104 min. This is less than what most algorithms took for the DREAM 5 challenge which is significantly smaller.

The results show that KBoost reconstructed GRNs that distinguished the five main subtypes, Normal-like, Luminal A, Luminal B, HER2 enriched, and Basal/Triple negative breast cancer (TNBC) (Fig. 5 and Table 5). KBoost. We analyzed the GRNs using the closeness centrality measure, which is an indication of the number of genes a TF regulates directly or indirectly.

Discussion

Inferring GRNs from gene expression data is an important task in bioinformatics. KBoost represents a novel algorithm that uses KPCR, boosting and BMA coupled with greedy model selection. Both boosting and BMA have been previously used in this setting. ENNET and GRNBoost2 use regression tree boosting, while TIGRESS uses a least angle regression which can be considered a form of boosting. On the other hand, BMA has mainly been used with linear regression. KPCR has advantages over using regression trees and linear regression. Both KPCR and regression trees do not need the relationship between a TF and its target gene to be linear, unlike linear regression. However, regression trees produce discontinuous functions which can be added together after several iterations to approximate a continuous function by using random forests or boosting. In contrast, KPCR directly yields continuous functions.

We show that KBoost has a very competitive performance using three different datasets. KBoost had the highest score on the multifactorial subset of DREAM 4 and the IRMA switch off dataset and tied for first place

Breast cancer subtype	Running time (mins)	No. of patients	TF closeness centrality hub
Claudin Low	13.0	199	FOXM1
Luminal A	32.1	679	REPIN1
Luminal B	21.6	461	REPIN1
HER2 enriched	13.5	220	AKAP8L
Basal/TNBC	12.9	199	TBX21
Normal-like	11.5	140	TSC22D1

Table 5. METABRIC summary.

with ENNET in DREAM 5. Furthermore, though we did not use any prior information for the benchmarking, we show that in the presence of prior information the performance of KBoost can increase dramatically. This is particularly relevant as there are several datasets with predicted or observed TF binding sites, ChIP-seq networks and other forms of prior information available. In terms of running time, the advantage of using KBoost is especially clear, as it was the fastest among all algorithms across all datasets. In addition, we were able to run KBoost on the METABRIC dataset containing over 24,000 genes with over 1300 TFs and over 1800 patients in less than two hours on a standard desktop computer.

We compared the GRNs from different breast cancer subtypes using the closeness centrality. Closeness centrality is a measure of a node's importance in a network and is calculated using the number of other nodes to which it is directly and indirectly connected. Different subtypes showed differences in TF centrality. The patterns can easily be distinguished by the naked eye and correlate with increasing aggressiveness. According to the current transcriptome based classification[24,28] the following phenotypes can be discerned: normal or normal-like is the most benign phenotype that may simply reflect normal breast tissue due to a low tumor cell content in the samples; luminal A and luminal B, which are relatively benign; HER2 enriched, which is more aggressive; and basal which has the worst prognosis. In the claudin-low subgroup, the gene FOXM1 had the highest closeness centrality. FOXM1 regulates the cell cycle and is required for the proliferation of normal breast epithelia but also promotes the proliferation of malignant cells and is often overexpressed in breast cancer[31]. The Luminal A and B GRNs were similar. In both GRNs, the TF with highest closeness centrality was REPIN1/AP4. This gene is a zinc finger protein that has been previously associated with enhanced proliferation, migration, and cisplatin resistance in breast cancer cells[32]. Interestingly, REPIN1/AP4 also was reported to reduce breast cancer cell proliferation[33], which may be related to the slow growing phenotype of luminal A/B breast cancers. The gene AKAP8L had the highest closeness centrality in HER2 enriched breast cancer. Previous studies have found AKAP8L to be associated with metastasis suppression in Luminal A, B and HER2 enriched breast cancer[34]. For the basal subtype, the gene TBX21 had the highest closeness centrality. This gene is often overexpressed in breast cancer indicating poor prognosis[35]. Finally, we found the gene TSC22D1 (TGFβ stimulated clone-22) to be important for the normal-like subtype GRN. TSC22D1 seems to have a dual role. It was identified as tumor suppressor in several cancer types including breast cancer[36]. However, it also was described that its expression correlates with tamoxifen resistance and breast cancer recurrence[37]. TSC22D1 functions in the TGFβ pathway, which can suppress tumorigenesis in early stages but promote it in later stages[38]. Together these networks highlight the gene expression differences between the breast cancer subtypes that might contribute to their observed differences in outcome. As this molecular classification of breast cancer subtypes is based on gene expression data[25,26], it is not surprising that the subtypes differ in their GRNs. However, the identification of the TFs responsible is useful. It suggests pathogenetic mechanisms, and the dimensionality reduction associated with classifying molecular phenotypes based on GRN reconstruction facilitates the analysis and identification of potentially actionable targets.

Received: 17 May 2021; Accepted: 6 July 2021
Published online: 29 July 2021

References

1. Alon, U. *An Introduction to Systems Biology: Design Principles of Biological Circuits* (Chapman & Hall/CRC, 2007).
2. Huynh-Thu, V. A., Irrthum, A., Wehenkel, L. & Geurts, P. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*. https://doi.org/10.1371/journal.pone.0012776 (2010).
3. Marbach, D. *et al.* Wisdom of Crowds for Robust Gene Network Inference. *Nat. Methods* **9**, 796. https://doi.org/10.1038/nmeth.2016 (2012).
4. Whitmarsh, A. J. & Davis, R. J. Regulation of transcription factor function by phosphorylation. *Cell. Mol. Life Sci.* **57**, 1172–1183. https://doi.org/10.1007/pl00000757 (2000).
5. Fischer, M., Grossmann, P., Padi, M. & DeCaprio, J. A. Integration of TP53, DREAM, MMB-FOXM1 and RB-E2F target gene analyses identifies cell cycle gene regulatory networks. *Nucleic Acids Res.* **44**, 6070–6086. https://doi.org/10.1093/nar/gkw523 (2016).
6. Moerman, T. *et al.* GRNBoost2 and Arboreto: Efficient and scalable inference of gene regulatory networks. *Bioinformatics* **35**, 2159–2161. https://doi.org/10.1093/bioinformatics/bty916 (2019).
7. Slawek, J. & Arodz, T. ENNET: Inferring large gene regulatory networks from expression data using gradient boosting. *BMC Syst. Biol.* https://doi.org/10.1186/1752-0509-7-106 (2013).
8. Guo, S., Jiang, Q. S., Chen, L. F. & Guo, D. H. Gene regulatory network inference using PLS-based methods. *BMC Bioinform.* https://doi.org/10.1186/s12859-016-1398-6 (2016).

9. Haury, A. C., Mordelet, F., Vera-Licona, P. & Vert, J. P. TIGRESS: Trustful Inference of Gene REgulation using Stability Selection. *BMC Syst. Biol.* https://doi.org/10.1186/1752-0509-6-145 (2012).
10. Iglesias-Martinez, L. F., Kolch, W. & Santra, T. BGRMI: A method for inferring gene regulatory networks from time-course gene expression data and its application in breast cancer research. *Sci. Rep.* https://doi.org/10.1038/srep37140 (2016).
11. Young, W. C., Raftery, A. E. & Yeung, K. Y. Fast Bayesian inference for gene regulatory networks using ScanBMA. *BMC Syst. Biol.* https://doi.org/10.1186/1752-0509-8-47 (2014).
12. Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. Least angle regression. *Ann. Stat.* **32**, 407–451 (2004).
13. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32. https://doi.org/10.1023/a:1010933404324 (2001).
14. Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **29**, 1189–1232. https://doi.org/10.1214/aos/1013203451 (2001).
15. Scholkopf, B., Smola, A. & Muller, K. R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* **10**, 1299–1319. https://doi.org/10.1162/089976698300017467 (1998).
16. Rosipal, R., Girolami, M., Trejo, L. J. & Cichocki, A. Kernel PCA for feature extraction and de-noising in nonlinear regression. *Neural Comput. Appl.* **10**, 231–243. https://doi.org/10.1007/s521-001-8051-z (2001).
17. Scholkopf, B. & Smola, A. J. A short introduction to learning with kernels. *Adv. Lect. Mach. Learn.* **2600**, 41–64 (2002).
18. Sch©œlkopf, B. & Smola, A. J. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, 2002).
19. Madigan, D. & Raftery, A. E. Model selection and accounting for model uncertainty in graphical models using Occams window. *J. Am. Stat. Assoc.* **89**, 1535–1546. https://doi.org/10.2307/2291017 (1994).
20. Raftery, A. E., Madigan, D. & Hoeting, J. A. Bayesian model averaging for linear regression models. *J. Am. Stat. Assoc.* **92**, 179–191. https://doi.org/10.2307/2291462 (1997).
21. Hoeting, J. A., Madigan, D., Raftery, A. E. & Volinsky, C. T. Bayesian model averaging: A tutorial. *Stat. Sci.* **14**, 382–401 (1999).
22. Cantone, I. *et al.* A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell* **137**, 172–181. https://doi.org/10.1016/j.cell.2009.01.055 (2009).
23. Saito, T. & Rehmsmeier, M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE* https://doi.org/10.1371/journal.pone.0118432 (2015).
24. Curtis, C. *et al.* The genomic and transcriptomic architecture of 2000 breast tumours reveals novel subgroups. *Nature* **486**, 346–352. https://doi.org/10.1038/nature10983 (2012).
25. Prat, A. & Perou, C. M. Deconstructing the molecular portraits of breast cancer. *Mol. Oncol.* **5**, 5–23. https://doi.org/10.1016/j.molonc.2010.11.003 (2011).
26. Perou, C. M. *et al.* Molecular portraits of human breast tumours. *Nature* **406**, 747–752. https://doi.org/10.1038/35021093 (2000).
27. Herschkowitz, J. I. *et al.* Identification of conserved gene expression features between murine mammary carcinoma models and human breast tumors. *Genome Biol.* **8**, R76. https://doi.org/10.1186/gb-2007-8-5-r76 (2007).
28. Fougner, C., Bergholtz, H., Norum, J. H. & Sørlie, T. Re-definition of claudin-low as a breast cancer phenotype. *Nat. Commun.* **11**, 1787. https://doi.org/10.1038/s41467-020-15574-5 (2020).
29. Gerstein, M. B. *et al.* Architecture of the human regulatory network derived from ENCODE data. *Nature* **489**, 91–100. https://doi.org/10.1038/nature11245 (2012).
30. Lambert, S. A. *et al.* The human transcription factors. *Cell* **172**, 650–665. https://doi.org/10.1016/j.cell.2018.01.029 (2018).
31. Saba, R., Alsayed, A., Zacny, J. P. & Dudek, A. Z. The role of forkhead box protein M1 in breast cancer progression and resistance to therapy. *Int. J. Breast Cancer* **2016**, 9768183. https://doi.org/10.1155/2016/9768183 (2016).
32. Wang, L., Meng, Y., Xu, J. J. & Zhang, Q. Y. The transcription factor AP4 promotes oncogenic phenotypes and cisplatin resistance by regulating LAPTM4B expression. *Mol. Cancer Res. MCR* **16**, 857–868. https://doi.org/10.1158/1541-7786.Mcr-17-0519 (2018).
33. Amin, S., Kumar, A., Nilchi, L., Wright, K. & Kozlowski, M. Breast cancer cells proliferation is regulated by tyrosine phosphatase SHP1 through c-jun N-terminal kinase and cooperative induction of RFX-1 and AP-4 transcription factors. *Mol. Cancer Res. MCR* **9**, 1112–1125. https://doi.org/10.1158/1541-7786.Mcr-11-0097 (2011).
34. Hu, X. *et al.* The RNA-binding protein AKAP8 suppresses tumor metastasis by antagonizing EMT-associated alternative splicing. *Nat. Commun.* **11**, 486–486. https://doi.org/10.1038/s41467-020-14304-1 (2020).
35. Yu, H. *et al.* T-box transcription factor 21 expression in breast cancer and its relationship with prognosis. *Int. J. Clin. Exp. Pathol.* **7**, 6906–6913 (2014).
36. Kester, H. A., van der Leede, B. M., van der Saag, P. T. & van der Burg, B. Novel progesterone target genes identified by an improved differential display technique suggest that progestin-induced growth inhibition of breast cancer cells coincides with enhancement of differentiation. *J. Biol. Chem.* **272**, 16637–16643. https://doi.org/10.1074/jbc.272.26.16637 (1997).
37. Meijer, D. *et al.* TSC22D1 and PSAP predict clinical outcome of tamoxifen treatment in patients with recurrent breast cancer. *Breast Cancer Res. Treat.* **113**, 253–260. https://doi.org/10.1007/s10549-008-9934-3 (2009).
38. Syed, V. TGF-β signaling in cancer. *J. Cell. Biochem.* **117**, 1279–1287. https://doi.org/10.1002/jcb.25496 (2016).

Acknowledgements
We thank Dr. Colm Ryan for helpful discussions in the preparation of this manuscript.

Author contributions
L.F.I.M. and W.K. designed the study. L.F.I.M. wrote the R package for KBoost and B.D.K. reviewed the code and wrote the vignette. L.F.I.M. wrote the main manuscript text and Figs. 1, 3 and 4. B.D.K. prepared Fig. 1. All authors reviewed the manuscript.

Funding
This work has been supported by the Irish Cancer Society CCRC BREAST-PREDICT Grant CCRC13GAL, and by Science Foundation Ireland under Grants 14/IA/2395 and 18/SPP/3522. B.D.K. was funded through an Irish Research Council 2017/2018 Laureate Award awarded to Dr. Colm J. Ryan.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-021-94919-6.

Correspondence and requests for materials should be addressed to L.F.I.-M.