A Bayesian Approach to Graph Regression with Relevant Subgraph Selection

Silvia Chiappa[1]   Hiroto Saigo[2]   Koji Tsuda[1]

[1]Max-Planck Institute for Biological Cybernetics
Spemannstraße 38, 72076 Tübingen, Germany
{silvia.chiappa,koji.tsuda}@tuebingen.mpg.de
[2]Max-Planck Institute for Informatics
Campus E1 4, 66123 Saarbrücken, Germany
hiroto.saigo@mpi-inf.mpg.de

Abstract

Many real-world applications with graph data require the solution of a given regression task as well as the identification of the subgraphs which are relevant for the task. In these cases graphs are commonly represented as high dimensional binary vectors of indicators of subgraphs. However, since the dimensionality of such indicator vectors can be high even for small datasets, traditional regression algorithms become intractable and past approaches used to preselect a feasible subset of subgraphs. A different approach was recently proposed by a Lasso-type method where the objective function optimization with a large number of variables is reformulated as a dual mathematical programming problem with a small number of variables but a large number of constraints. The dual problem is then solved by column generation, where the subgraphs corresponding to the most violated constraints are found by weighted subgraph mining. This paper proposes an extension of this method to a Bayesian approach in which the regression parameters are considered as random variables and integrated out from the model likelihood, thus providing a posterior distribution on the target variable as opposed to a point estimate. We focus on a linear regression model with a Gaussian prior distribution on the parameters. We evaluate our approach on several molecular graph datasets and analyze whether the uncertainty in the target estimate given by the target posterior distribution variance can be used to improve model performance and therefore provides useful additional information.

1 Introduction

Graphs are general and powerful data types that can be used to represent many kinds of real-world objects, including biological sequences, semi-structured texts such as HTML and XML, chemical compounds, RNA secondary structures, and so forth. This paper focuses on regression problems with graphs that require the identification of the parts of the graphs that are relevant
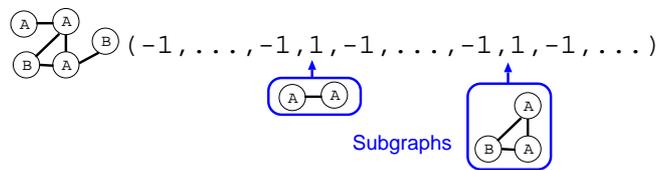

Figure 1: A graph and its representation as a binary vector of subgraph indicators: 1 indicates the presence while -1 indicates the absence of a certain subgraph.

for solving the task. For example, in drug discovery it is important to identify the subgraphs which can explain why a molecular graph candidate is supposed to work as a drug [1, 2].

In these cases, it is common to represent each graph as a binary vector of indicators of subgraphs (see Fig. 1) such that subgraph selection can be obtained by selecting the corresponding part of the vector. However, since the dimensionality of such indicator vectors can be high even for small datasets, this representation makes direct application of traditional regression algorithms intractable and past approaches in the literature used to first obtain a subset of subgraphs of feasible dimensionality by a mining method and then apply a separate regression algorithm. This was achieved by frequent substructure mining methods such as AGM [3], gSpan [4] or Gaston [5] that enumerate all subgraphs whose frequency is above a minimum threshold. The most frequent subgraphs were then used as inputs to a regularized regression algorithm such as L1-SVM [6] or Lasso [7] which enables relevant subgraph selection. Other approaches replaced frequent mining by mining methods that select subgraphs on the basis of statistical criteria, such as information gain, and then used a separate regression algorithm [8, 9, 10].

All these methods share the same limitation that

regression and subgraph selection are based on criteria that are not necessarily well related. Indeed, the most relevant subgraphs for a certain regression task are not necessarily a subset of the most frequent subgraphs, as assumed when frequent mining methods and L1-SVM or Lasso are used. Similarly, it is not clear how different statistical criteria relate to different regression models.

Inspired by [12, 13], [11] recently proposed a new approach based on column generation that overcomes this limitation. In this approach, a Lasso-type[1] optimization problem with a large number of variables is re-expressed as a dual mathematical programming problem with a small number of variables but a large number of constraints. The dual problem is then solved by column generation, in which at each iteration of the algorithm a subset of most violated constraints is identified and a restricted optimization problem over the (growing) set of selected constraints is solved. Only the subgraphs corresponding to selected constraints can contribute to the estimation of the target variable.

In this paper we investigate using a similar column generation approach to obtain a regression model that, in addition to performing subgraph selection, gives a posterior distribution on the regression parameters, as opposed to a point estimate. The advantage is that, in addition to the target variable estimate, the posterior distribution can provide extra information, such as uncertainty in the estimate.

Our approach is to use a Bayesian method where the parameters are treated as random variables and integrated out from the model. We consider the simpler case of a linear model with a Gaussian prior distribution on the parameters and focus on molecular graph applications. We compare the performance of this approach with the Lasso-type approach on several datasets, and analyze whether the uncertainty in the target estimate given by the target posterior distribution variance can be used to improve model performance and therefore provides useful additional information.

The rest of the paper is organized as follows. In Section 2 we describe a Bayesian approach to linear regression and present an algorithm for learning the hyperparameters recently introduced in [16]. In Section 3 we combine this algorithm with a column generation approach for the intractable case in which the inputs are binary vectors of indicators of subgraphs. In Section 4 we describe the Lasso-type method introduced in [11] that we use for comparison with our method. We then present some results on molecular graph data in Section 5 and draw some conclusions in Section 6.

2   Bayesian Linear Regression

Given n pairs of input-target variables $(x_i, y_i) \in \mathcal{R}^d \times \mathcal{R}$, we consider the following linear regression model

$$y = Xw + \epsilon,$$

where X and y are the $n \times d$ matrix and the $n \times 1$ vector formed by stacking all inputs and targets respectively, w is a vector of unknown parameters and[2] $\epsilon \sim \mathcal{N}(0, \lambda I)$ is a Gaussian noise vector with unknown covariance element λ. An equivalent probabilistic formulation of this model is given by[3]

$$p(y|w, \lambda) = \mathcal{N}(Xw, \lambda I).$$

In our Bayesian approach, we introduce a zero-mean Gaussian prior distribution on w, $p(w|\gamma) = \mathcal{N}(0, \Gamma)$, where Γ is a diagonal matrix with unknown hyperparameters γ_i on the diagonal, and form the marginal likelihood

$$p(y|\lambda, \gamma) = \int_w p(y|w, \lambda) p(w|\gamma) = \mathcal{N}\big(0, \underbrace{X\Gamma X^{\mathsf{T}} + \lambda I}_{\Sigma}\big).$$

As opposed to a point estimate of standard linear regression methods, this approach provides a posterior distribution on w, given by

$$(2.1)\quad p(w|y, \lambda, \gamma) = \mathcal{N}\big(\underbrace{\Gamma X^{\mathsf{T}} \Sigma^{-1} y}_{\mu_w}, \underbrace{\Gamma - \Gamma X^{\mathsf{T}} \Sigma^{-1} X\Gamma}_{\Sigma_w}\big).$$

The optimal values of γ_i are learned by type-II maximum likelihood [17], i.e., by maximization of the marginal likelihood, whilst λ is selected by validation. This choice has the advantage of making all quantities of interest computationally tractable and thus avoids the need to introduce approximations.

To predict an unknown target variable y^* from a new input variable x^*, we can use the predictive distribution

$$(2.2)\quad p(y^*|y, \lambda, \gamma) = \mathcal{N}\big((x^*)^{\mathsf{T}}\mu_w, (x^*)^{\mathsf{T}}\Sigma_w x^* + \lambda\big).$$

The mean in (2.2) gives an estimate of the target variable, while the variance gives a measure of uncertainty in the estimate. Due to the linear relation between inputs and targets and to the choice of a Gaussian prior distribution on w, the information provided by the predictive variance is restricted. In particular, the Gaussian prior makes the variance independent on the targets y. Nonlinear approaches and other prior distributions

---

[1]With Lasso-type we mean that the same objective function as in Lasso [7] is employed.

[2]$\mathcal{N}(m, S)$ denotes a Gaussian with mean m and covariance S. I is the identity matrix.

[3]To simplify the notation, we omit the conditioning on X.

Algorithm 1

1: $z = 1$
2: **repeat**
3:     $w \leftarrow \mathrm{argmin}_w \sum_{j=1}^{n} (y_j - \sum_{i=1}^{d} X_{ji} w_i)^2$
        $+ 2\lambda \sum_{i=1}^{d} z_i^{1/2} |w_i|$
4:     $\gamma_i = z_i^{-1/2} |w_i|, \quad i = 1, \ldots, d$
5:     $\Sigma^{-1} = (X \Gamma X^{\mathsf{T}} + \lambda I)^{-1}$
6:     $z = diag \left[X^{\mathsf{T}} \Sigma^{-1} X \right]$
7: **until convergence**

---

could be employed to obtain a richer variance. However, this would introduce intractability issues and make the development of the model for the case of high dimensional graph representation more complex.

The use of a zero-mean Gaussian prior on w and type-II maximum likelihood enforces a sparse parametrization. This allows to select the elements of the input vectors which are relevant for solving the given regression task. Indeed, during training, the model is biased towards learning a zero value for γ_i such that the corresponding elements of the input vectors do not contribute to the model (e.g., from (2.1) we can see that, if $\gamma_i = 0$, the i-th element of μ_w and the i-th row and column of Σ_w become zero so that the i-th elements of the input variables do not contribute to the predictive distribution (2.2)). This pruning effect is explained in details in [18].

2.1 Finding the Optimal Hyperparameters γ
The optimal set of hyperparameters γ can be found by maximizing the marginal likelihood $p(y|\lambda, \gamma)$, or equivalently by setting

$$\gamma \leftarrow \underset{\gamma}{\mathrm{argmin}} \, \mathcal{L}(\gamma) \equiv \underset{\gamma}{\mathrm{argmin}} \log |\Sigma| + y^{\mathsf{T}} \Sigma^{-1} y.$$

Whilst this minimization problem can be solved with gradient-based methods or Expectation Maximization, here we describe a different approach, recently introduced in [16], that will be useful for dealing with our high dimensional graph representation.

In this approach, the minimization of $\mathcal{L}(\gamma)$ with respect to γ is reformulated by introducing a new set of auxiliary variables z in order to obtain a convex optimization problem. More specifically, since $\log |\Sigma|$ is concave in γ it can be expressed as

$$\log |\Sigma| = \min_z z^{\mathsf{T}} \gamma - g^*(z),$$

where $g^*(z)$ is the concave conjugate of $\log |\Sigma|$, defined by

$$g^*(z) = \min_{\gamma} z^{\mathsf{T}} \gamma - \log |\Sigma|.$$

We can then construct an upperbounding function

$$\mathcal{L}(\gamma, z) \equiv z^{\mathsf{T}} \gamma - g^*(z) + y^{\mathsf{T}} \Sigma^{-1} y \geq \mathcal{L}(\gamma).$$

Minimizing $\mathcal{L}(\gamma)$ can be achieved by iteratively minimizing $\mathcal{L}(\gamma, z)$ over z for fixed γ and vice-versa. For fixed γ, the optimal z is given by[4]

$$z = diag \left[X^{\mathsf{T}} \Sigma^{-1} X \right].$$

On the other hand, for fixed z, the optimal γ can be found as

$$\gamma \leftarrow \underset{\gamma}{\mathrm{argmin}} \, z^{\mathsf{T}} \gamma + y^{\mathsf{T}} \Sigma^{-1} y,$$

which is equivalent to solving the following quadratic convex minimization problem

(2.3)

$$w \leftarrow \underset{w}{\mathrm{argmin}} \sum_{j=1}^{n} \left(y_j - \sum_{i=1}^{d} X_{ji} w_i \right)^2 + 2\lambda \sum_{i=1}^{d} z_i^{1/2} |w_i|,$$

and then set $\gamma_i = z_i^{-1/2} |w_i|$.

This optimization procedure is summarized in Algorithm 1.

3 Bayesian Linear Regression with Graphs

In our regression problem with graphs each input $x_i \in \mathcal{R}^d$ is a binary vector of indicators of subgraphs. Even for small datasets, the high dimensionality of such vector renders Algorithm 1 intractable.

In this section we show how Algorithm 1 can be modified to obtain a tractable method. We start by reformulating the minimization problem (2.3) with a large number of variables as a minimization problem with a small number of variables but a large number of constraints. We then introduce a column generation approach where, at each iteration, (2.3) is solved only over $\tilde{d}$ elements of w corresponding to a set of most violated constraints at the current solution and to the constraints selected at the previous iterations. This way we obtain an algorithm that is tractable, since it operates in a $\tilde{d}$-dimensional input space, with the exception of the search of the most violated constraints, for which we introduce an efficient mining method.

3.1 Reformulation of the Minimization Problem (2.3) By defining $w_i \equiv w_i^+ - w_i^-$, the minimization

---

[4] With $diag \, [A]$ we indicate a vector formed by the diagonal elements of the matrix A.

Algorithm 2

1: $\tilde{d} = 0$
2: **for** $k = 1, \ldots, K$ **do**
3:     Select the $\hat{d}$ most violated constraints
4:     $\tilde{d} \leftarrow \tilde{d} + \hat{d}$
5:     Collect the input elements corresponding to the selected constraints in a $n \times \tilde{d}$ matrix X
6:     **repeat**
7:         $z_i = \sum_{j,k=1}^n X_{ji} \Sigma_{jk}^{-1} X_{ki}$ (if $k=1$, $z_i = 1$) $i = 1, \ldots, \tilde{d}$
8:         $(w_1, \ldots, w_{\tilde{d}}) \leftarrow \operatorname{argmin}_{w^+, w^-, \xi} \ \sum_{j=1}^n \xi_j^2 + 2\lambda \sum_{i=1}^{\tilde{d}} z_i^{1/2} (w_i^+ + w_i^-),$
                $s.t. -\xi_j \leq y_j - \sum_{i=1}^{\tilde{d}} X_{ji}(w_i^+ - w_i^-) \leq \xi_j \ \ j = 1, \ldots, n$
                    $w_i^+ \geq 0, \ \ w_i^- \geq 0 \ \ i = 1, \ldots, \tilde{d}$
9:         $\gamma_i = z_i^{-1/2} |w_i| \ i = 1, \ldots, \tilde{d}$
10:       $\Gamma \leftarrow$ Diagonal matrix with $\gamma_1, \ldots, \gamma_{\tilde{d}}$ on its diagonal
11:       $\Sigma^{-1} = (X \Gamma X^{\mathsf{T}} + \lambda I)^{-1}$
12:     **until convergence**
13: **end for**

problem (2.3) can be rewritten as the following constrained quadratic programming problem

(3.4)

$$\min_{w^+, w^-, \xi} \ \sum_{j=1}^n \xi_j^2 + 2\lambda \sum_{i=1}^d z_i^{1/2}(w_i^+ + w_i^-)$$

$$s.t. -\xi_j \leq y_j - \sum_{i=1}^d X_{ji}(w_i^+ - w_i^-) \leq \xi_j \ \ j = 1, \ldots, n$$

$$w_i^+ \geq 0, \ \ w_i^- \geq 0 \ \ i = 1, \ldots, d.$$

By setting the derivatives of the Lagrangian with respect to w_i^+, w_i^- and ξ_j to zero and substituting back the optimal lagrange multipliers, we obtain the following dual problem

(3.5)

$$\min_{u^+, u^-} \ -\sum_{j=1}^n (u_j^+ - u_j^-) y_j + \frac{1}{16} \sum_{j=1}^n (u_j^+ + u_j^-)^2$$

$$s.t. -2\lambda z_i^{1/2} \leq \sum_{j=1}^n (u_j^+ - u_j^-) X_{ji} \leq 2\lambda z_i^{1/2} \ \ i = 1, \ldots, d$$

$$u_j^+ \geq 0, \ \ u_j^- \geq 0 \ \ j = 1, \ldots, n,$$

where u^+, u^- are dual variables. Unlike the original problem, the dual has a small number of variables but a large number of constraints.

3.2 Computationally Tractable Algorithm Using this reformulation, we introduce a new computationally tractable algorithm based on column generation. The idea of column generation is to iteratively solve a relaxation of the dual (3.5) where only a growing subset of constraints are considered, which is equivalent to solving (3.4) only over the corresponding elements of w^+, w^-. At each iteration the most violated constraints given the current solution are added to (3.5). Optimality is reached when no more violated constraints can be found.

Our recursive procedure is summarized in Algorithm 2. At iteration k, we select the $\hat{d}$ most violated constraints in (3.5) and collect the elements of the input vectors corresponding to the $\tilde{d}$ constraints selected up to iteration k in a $n \times \tilde{d}$ matrix X. Then, until convergence, we compute the optimal $z_i, i = 1, \ldots, \tilde{d}$, (see step 7 in Algorithm 2), solve the optimization problem (3.4) over the $\tilde{d}$ variables corresponding to the selected constraints, and compute $\gamma_i, i = 1, \ldots, \tilde{d}$, and Σ^{-1} (see steps 9-11 in Algorithm 2).

Except when finding the most violated constraints, Algorithm 2 operates in a $\tilde{d}$-dimensional input space. Below we describe an efficient mining method for identifying such constraints.

3.3 Selecting the Most Violated Constraints
Before getting into the main discussion, we provide formal definitions of graphs and subgraphs and describe how we represent a graph as a vector of subgraph indicators. We consider undirected labeled and connected graphs.

DEFINITION 3.1. *A labeled graph is a 4-tuple $G = (V, E, \mathcal{L}, l)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, $\mathcal{L}$ is a set of labels, and $l : V \cup E \to \mathcal{L}$ is*
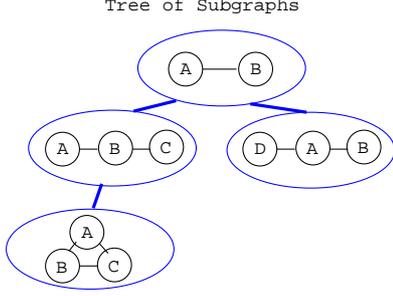
Tree of Subgraphs

Figure 2: Schematic figure of the tree-shaped search space of graph patterns (i.e., the DFS code tree). To find the optimal pattern efficiently, the tree is systematically expanded by rightmost extensions.

a mapping that assigns labels to the vertices and edges. A labeled connected graph is a labeled graph such that there is a path between any pair of vertices.

DEFINITION 3.2. *Let $G' = (V', E', \mathcal{L}', l')$ and $G = (V, E, \mathcal{L}, l)$ be labeled connected graphs. G' is a subgraph of G ($G' \subseteq G$) if the following conditions are satisfied: (1) $V' \subseteq V$, (2) $E' \subseteq E$, (3) $\mathcal{L}' \subseteq \mathcal{L}$, (4) $\forall v' \subseteq V', l(v') = l'(v')$ and (5) $\forall e' \subseteq E', l(e') = l'(e')$. If G' is a subgraph of G, then G is a supergraph of G'.*

Let p be a subgraph pattern in a graph, and $\mathcal{P}$ be the set of all patterns, i.e., the set of all subgraphs included in at least one graph. Given n graphs in the database, each graph G_j is represented as a $d \equiv |\mathcal{P}|$-dimensional vector x_j and we form a matrix X where

$$X_{jp} = \begin{cases} 1 & \text{if } p \subseteq G_j, \\ -1 & \text{otherwise} \end{cases}$$

The violation of the dual constraint given in (3.5) is described as

$$g(p) = \left| \sum_{j=1}^{n} u_j X_{jp} \right| - 2\lambda \sqrt{\sum_{j=1}^{n} \sum_{k=1}^{n} \Sigma_{jk}^{-1} X_{jp} X_{kp}},$$

where $u_j \equiv u_j^+ - u_j^-$ and $\sum_{j=1}^{n} \sum_{k=1}^{n} \Sigma_{jk}^{-1} X_{jp} X_{kp}$ is the optimal value of z_p. We call it the gain function of p. An efficient algorithm that finds the best $\hat{d}$ patterns in terms of the gain function is necessary. In [19] a similar mining problem is solved, but this gain function is nontrivially different from theirs in that our gain contains cross terms $X_{jp} X_{kp}$.

Optimal Pattern Search Our search strategy to find the best $\hat{d}$ patterns is a branch-and-bound algorithm that requires a canonical search space in which a whole

set of patterns are enumerated without duplication. As the search space, we adopt the DFS code tree [4]. The basic idea of the DFS code tree is to organize patterns as a tree, where a child node has a supergraph of the pattern in its parent node (see Fig. 2). A pattern is represented as a text string called the DFS (depth first search) code. The patterns are enumerated by generating the tree from the root to leaves using a recursive algorithm. To avoid duplications, node generation is systematically done by rightmost extensions.

For efficient search, it is important to minimize the size of the search space. To this aim, we perform tree pruning by using the bound defined by the following theorem

THEOREM 3.1. *Define*

$$s^+(p) = 2 \sum_{\{j | u_j \geq 0, p \subseteq G_j\}} |u_j| - \sum_{j=1}^{n} u_j$$

$$s^-(p) = 2 \sum_{\{j | u_j < 0, p \subseteq G_j\}} |u_j| + \sum_{j=1}^{n} u_j$$

and $R_k = \sum_{j=1}^{n} \Sigma_{jk}^{-1}$. For any supergraph p' of p, the following bound holds

$$g(p') \leq \max\{s^+(p), s^-(p)\}$$

$$(3.6) \qquad -2\lambda \sqrt{\sum_{k=1}^{n} R_k - 4 \sum_{R_k > 0} R_k I(p \subseteq g_k)}.$$

The proof of this theorem can be found in the Appendix.

Suppose the search tree is generated up to the pattern i and the best $\hat{d}$ patterns at this point are recorded. Let g^* be the gain of the $\hat{d}$-th best pattern. If the right hand size of (3.6) is smaller than g^*, then there are no possibility that any supergraph p' enters the list of best patterns. Therefore, the search tree can be pruned at p.

4 Lasso-type Linear Regression with Graphs

In this section we summarize the Lasso-type approach to graph regression proposed in [11] that we use for comparison with our approach.

Consider the regression problem $y = Xw + \epsilon$, $\epsilon \sim \mathcal{N}(0, \lambda I)$, in which we assume that each w_i follows a Laplace distribution $p(w_i|b) \propto e^{-b|w_i|}$. The joint distribution $p(y, w|\lambda, b)$ is given by

$$p(y, w|\lambda, b) \propto e^{-\frac{\lambda}{2} \sum_{j=1}^{n} \left(y_j - \sum_{i=1}^{d} X_{ji} w_i\right)^2 - b \sum_{i=1}^{d} |w_i|}.$$

Traditional Lasso [7] finds the optimal value of w by

Regression	GRAPHS	ATOMS	BONDS
EDKB-ES	59	18.2	19.7
EDKB-ER	131	19.2	20.7
EDKB-AR	146	19.5	21.1
Classification	GRAPHS	ATOMS	BONDS
CPDB	684	14.1	14.6
AIDS (CAvsCM)	1503	58.9	61.4
CAS	4337	29.9	30.9

Table 1: Dataset summary. EDKB-ES, EDKB-ER and EDKB-AR are regression datasets, while CPDB, AIDS and CAS are classification datasets. The number of graphs (GRAPHS) and the average number of atoms (ATOMS) and bonds (BONDS) are shown for each dataset.

maximizing $p(y, w|\lambda, b)$, or equivalently by setting

$$(4.7) \quad w \leftarrow \operatorname{argmin} \sum_{j=1}^{n} \left(y_j - \sum_{i=1}^{d} X_{ji} w_i \right)^2 + C \sum_{i=1}^{d} |w_i|,$$

where $C = 2b/\lambda$ is found by validation. The estimate of an unknown target variable y^* given a new input variable x^* is obtained by computing $(x^*)^\mathsf{T} w$.

For the case of graph data where d is very large and the problem becomes intractable, [11] proposed a column generation approach by rewriting (4.7) as an optimization problem with a small number of variables but a large number of constraints, following a similar procedure as the one described in Section 3.1. Notice that this approach can be seen as a special case of our Bayesian approach in which z is set to 1 (since (4.7) is equivalent to (2.3) with $z = 1$).

5 Experiments

In this section we apply our Bayesian model to six molecular graph datasets. We first compare the performance obtained by using the predictive mean as an estimate of the target variable with the Lasso-type performance, which was shown to be competitive with respect to other approaches on this data [11]. We then analyze whether this performance can be improved by disregarding test samples that show high predictive variance.

The analyzed data include three datasets from Endocrine Disruptors Knowledge Base (EDKB)[5]: ES, ER and AR, two mutagenicity datasets: CPDB[6] and CAS[7], and one antiviral screen dataset: AIDS (CAvsCM)[8].

The first three are regression while the remaining three are classification datasets. The statistics of the datasets are summarized in Table 1.

5.1 Bayesian vs Lasso-type Linear Regression

Experimental Setup Each molecular graph dataset was randomly split into five equally-sized subsets. Four subsets were used for training, while half of the filth subset was used for validation and the other half for testing and viceversa. The five subsets were circularly shifted five times, such that the experiments were performed using five different parts of the dataset for training, validation and testing. The validation set was used to choose the noise variance λ in the Bayesian approach and the hyperparameter C in the Lasso-type approach (see (4.7)) in the range $\{0.01, 0.05, 0.1, 0.5, 1\}$, the maximum subgraph pattern size[9] in the range $\{10, 15\}$, and the number of iterations K. The number $\hat{d}$ of most violated constrains to be selected at each iteration was set to 5, except for the Bayesian approach applied to the regression data where $\hat{d}$ was set to 10, and no threshold was imposed on the frequency of the subgraphs. In the Bayesian model, instead of iterating steps 7-11 in Algorithm 2 until convergence, we fixed a maximum number of iterations (25 for the regression data and 5 for the classification data).

Results The performance given by the two models is summarized in Table 2 (the best performance is highlighted using bold characters). For the regression datasets, we report the average of the root mean squared error (RMSE) $\pm$ the standard deviation (std) over the different train-validation-test configurations described above. For the classification datasets, we report the average classification accuracy (CA) $\pm$ std. As we can see, the two approaches give similar performance.

In the table, we also report the average number of selected relevant subgraphs (SG) and the average computation time (in seconds), decomposed into mining time (MT) and optimization time (OT). The former indicates the time for expanding and traversing the pattern space (step 3 in Algorithm 2), and the latter indicates the time for solving the series of restricted dual problems and variable updates (steps 4-12 and step 8 with $z = 1$ in Algorithm 2 for the Bayesian and Lasso-type methods respectively).

The selected maximum pattern size omitted in the table was 15 for the EDKB-ES and EDKB-AR datasets in the Bayesian model and 10 otherwise.

The top 20 active subgraphs found by our Bayesian

Table 2: Performance of Lasso-type and Bayesian Linear Regression on six molecular datasets. The values in parentheses indicate the hyperparameters selected by validation (K is the number of iterations). We report the average RMSE $\pm$ the standard deviation (std) for the first three regression datasets and the average classification accuracy CA $\pm$ std for the last three classification datasets. SG indicates the average number of selected relevant subgraphs, while MT and OT indicate (in seconds) the average mining time required for selecting the most violated constraints and for optimization respectively.

	Lasso-type Linear Regression					Bayesian Linear Regression				
	(C,K)	SG	MT(s)	OT(s)	RMSE/CA	(λ,K)	SG	MT(s)	OT(s)	RMSE/CA
ES	(0.5,10.6)	39.3	41	25	**0.33 ± 0.14**	(0.05,4.9)	32.7	348	142	0.34 ± 0.15
ER	(0.5,15.1)	52.8	60	103	**0.36 ± 0.08**	(0.1,8.6)	47.0	33	836	0.38 ± 0.08
AR	(0.5,16.7)	60.4	158	171	**0.28 ± 0.07**	(0.1,8.1)	40.8	1737	961	**0.28 ± 0.11**
CPDB	(0.5,19.6)	83.1	78	1621	77.0 ± 1.82	(0.1,13.6)	59.8	86	3442	**77.6 ± 2.22**
AIDS	(0.5,17.0)	68.6	2511	3830	81.8 ± 1.06	(1,16.5)	46.7	2293	6244	**82.8 ± 1.05**
CAS	(0.01,19.5)	74.0	2733	22067	81.5 ± 1.79	(0.05,16.5)	71.0	3662	42099	**82.1 ± 1.81**

model for the EDKB-AR and CPDB datasets are plotted in Fig. 3. The subgraphs mined from the CPDB dataset are especially interesting, since a toxicophore known as *aromatic amine* is detected as a whole with relevance 0.1274, and a part of it is detected many times with high relevance. Furthermore, Cl and Br atoms are known to form the toxicophore *aliphatic halide*, and our model gave high relevance to them. On the other hand, the subgraphs mined from the EDKB-AR dataset are rather difficult to interpret due to the smaller number of edge and atom types. It is possible to increase the interpretability, e.g., by setting the maximum pattern size larger, and/or by enriching the atom/bond labels by incorporating neighboring atom/bond information.

5.2 Bayesian Test Sample Rejection In this section we investigate whether by using the predictive variance in (2.2) in addition to the mean we can improve the performance of the Bayesian model.

The predictive variance gives us an indication of uncertainty in the target estimate. As pointed out in Section 2, due to model assumptions, this variance is expected to be of limited expressivity. Nevertheless, from the predictive means and standard deviations of the test samples plotted in Fig. 4 (blue circles and bars respectively), it would seem that means which are further from the true values (magenta stars) have higher standard deviation, and therefore that the predictive variance in this data gives useful information about the target estimate. To further analyze this point, we investigate whether the information contained in the predictive variance can improve the performance of our model, by rejecting test samples with high variance.

We performed the analysis on the three regression datasets EDKB-ES, EDKB-ER and EDKB-AR. More specifically, we computed the performance of the model using the selected hyperparameters given in Table 2 on the restricted test datasets containing only samples whose predictive standard deviation was below a certain threshold. In each test dataset, we varied the threshold in a range between the minimum and maximum predictive standard deviation. For each threshold, we compared the performance obtained with this approach with the one obtained by randomly rejecting the same portion of test data.

In Fig. 5 we report the obtained average mean squared error (MSE) as a function of the average standard deviation threshold. The performance of the random and Bayesian rejection is plotted in green and blue respectively. The original performance obtained by keeping all test samples is given by the first value on the X-axis for each plot (where the errors obtained by random and Bayesian rejection coincide). As expected, the error obtained by random rejection oscillates when varying the threshold but on average remains close to the value obtained by keeping all test samples. On the other hand, by decreasing the standard deviation threshold, the error obtained by performing rejection using our Bayesian approach is reduced in the EDKB-ES and EDKB-AR datasets. In the EDKB-ER dataset, on average the use of the predictive variance does not improve the performance. However, it is interesting to note that, when using different hyperparameters than those selected by validation (that give similar test performance), we can often observe improvement. The reason why different types of predictive variance are learned and why this does not affect the original performance is a point that needs to be investigated.
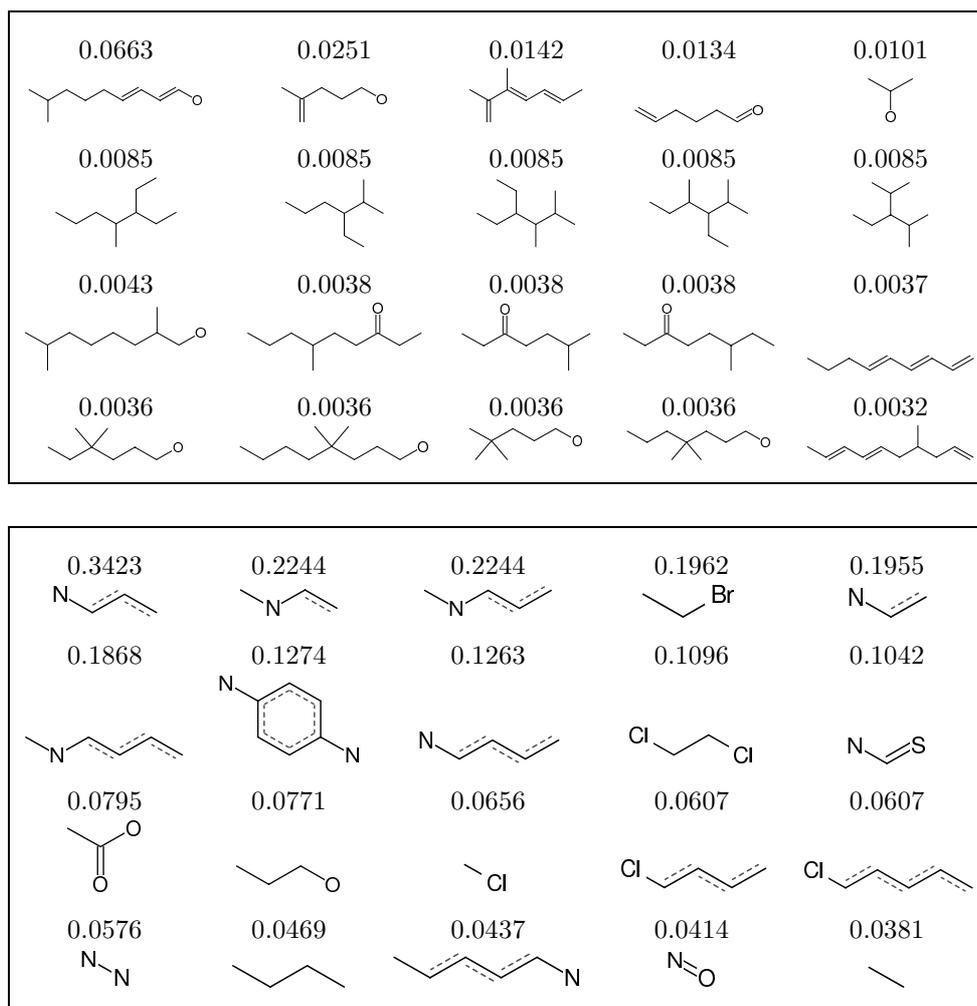
Figure 3: Top 20 relevant subgraphs selected by our Bayesian model for the EDKB-AR (top) and CPDB (bottom) datasets. Each subgraph is shown with the corresponding γ_i which gives a measure of its relevance. H atom is omitted, and C atom is represented as a dot for simplicity. Aromatic bonds appeared in an open form are displayed by the combination of dashed and solid lines.

6 Conclusions

In this paper we introduced a Bayesian approach for solving regression problems with graphs that require the identification of relevant subgraphs. In our approach, the objective function optimization with a large number of variables is reformulated as a dual mathematical programming problem with a small number of variables but a large number of constraints. The dual problem is then solved by column generation, where the subgraphs corresponding to the most violated constraints are found by weighted subgraph mining. We have demonstrated that, on molecular graph data, this approach is competitive with others previously presented in the literature. In addition, we have shown that the target posterior

distribution variance provides useful additional information to the target estimate.

It would be interesting to analyze the use of this model for experimental design in applications such as chemoinformatics, where active selection of the experiments to perform could reduce the costs. Another interesting area of research would be to investigate how to use a column generation approach with other Bayesian regression models that can provide a richer predictive variance.

Appendix

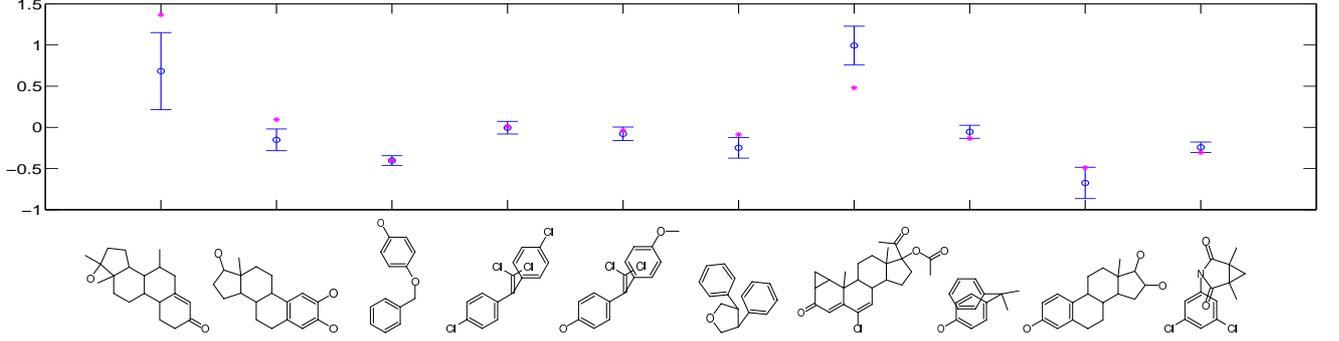Here we proof Theorem 1 introduced to perform tree pruning in the mining algorithm (Section 3.3).

Figure 4: Ten test samples from the EDKB-AR dataset. The blues circles and their bars indicate the means and standard deviations given by the predictive distribution. The magenta starts indicate the true target values.
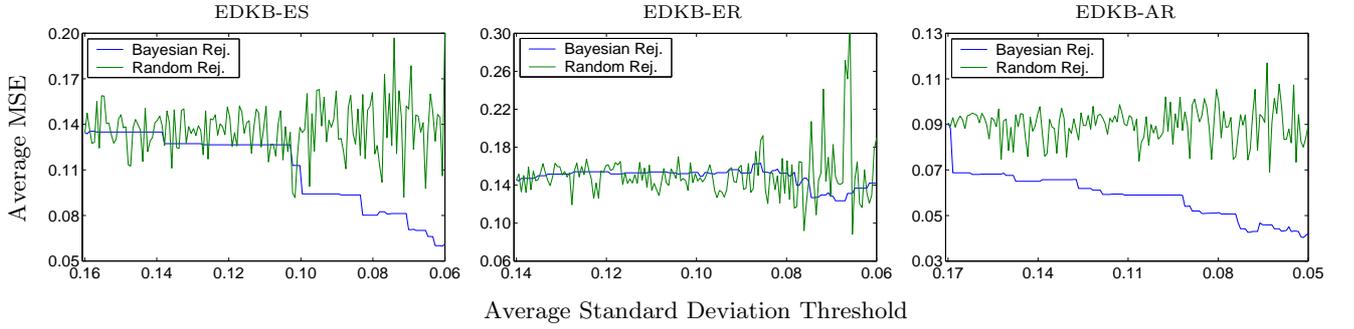

Figure 5: Average mean squared error (MSE) obtained in the datasets EDKB-ES, EDKB-ER and EDKB-AR by rejecting test samples. In blue we show the error obtained by our Bayesian model keeping only the samples whose predictive distribution has standard deviation below a certain threshold (X-axis). The first value on the X-axis indicates the error obtained by keeping all test samples. In green we show for each threshold value the error obtained by randomly rejecting the same amount of samples as in the Bayesian model.

Proof of Theorem 1

Let us bound the first term of $g(p')$ from above first

$$
\begin{aligned}
\sum_{j=1}^{n} u_j X_{jp'} &= 2 \sum_{\{j|p' \subseteq G_j\}} u_j - \sum_{j=1}^{n} u_j \\
&\leq 2 \sum_{\{j|u_j \geq 0, p' \subseteq G_j\}} u_j - \sum_{j=1}^{n} u_j \\
&\leq 2 \sum_{\{j|u_j \geq 0, p \subseteq G_j\}} u_j - \sum_{j=1}^{n} u_j = s^+(p),
\end{aligned}
$$

where the second inequality follows from

(6.8) $\{j|u_j \geq 0, p' \subseteq G_j\} \subseteq \{j|u_j \geq 0, p \subseteq G_j\}$.

Similarly the inequality $\sum_{j=1}^{n} u_j X_{jp'} \geq -s^-(p)$ holds, hence the first term is bounded by the maximum of $s^+(p)$ and $s^-(p)$. Regarding the second term of $g(p')$,

the quadratic term inside the square root is written as

$$
\begin{aligned}
L &= \sum_{j=1}^{n} \sum_{k=1}^{n} \Sigma_{jk}^{-1}(2I(p' \subseteq g_j) - 1)(2I(p' \subseteq g_k) - 1) \\
&= 4 \sum_{j,k} \Sigma_{jk}^{-1} I(p' \subseteq g_j) I(p' \subseteq g_k) - 4 \sum_{k} R_k I(p' \subseteq g_k) \\
&\quad + \sum_{k} R_k.
\end{aligned}
$$

Due to the positive definiteness of Σ^{-1}, the first term is positive, therefore

$$
\begin{aligned}
L &\geq -4 \sum_{k} R_k I(p' \subseteq g_k) + \sum_{k} R_k \\
&\geq -4 \sum_{R_k > 0} R_k I(p' \subseteq g_k) + \sum_{k} R_k \\
&\geq -4 \sum_{R_k > 0} R_k I(p \subseteq g_k) + \sum_{k} R_k,
\end{aligned}
$$

where the second inequality follows from (6.8).

References

[1] C. Helma, T. Cramer, S. Kramer, and L.D. Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *J. Chem. Inf. Comput. Sci.*, 44:1402–1411, 2004.

[2] J. Kazius, S. Nijssen, J. Kok, and T. Bäck A.P. Ijzerman. Substructure mining using elaborate chemical representation. *J. Chem. Inf. Model.*, 46:597–605, 2006.

[3] A. Inokuchi. Mining generalized substructures from a set of labeled graphs. In *Proceedings of the 4th IEEE Internatinal Conference on Data Mining*, pages 415–418. IEEE Computer Society, 2005.

[4] X. Yan and J. Han. gSpan: graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 721–724. IEEE Computer Society, 2002.

[5] S. Nijssen and J.N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 647–652. ACM Press, 2004.

[6] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[7] R. Tibshrani. Regression shrinkage and selection via the LASSO. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[8] S. Morishita and J. Sese. Traversing itemset lattices with statistical metric learning. In *Proceedings of ACM SIGACT-SIGMOD-SIGART Symposium on Database Systems (PODS)*, pages 226–236. ACM Press, 2000.

[9] K. Takabayashi, P.C. Nguyen, K. Ohara, H. Motoda, and T. Washio. Mining discriminative patterns from graph structured data with constrained search. In T. Gärtner, G.C. Garriga, and T. Meinl, editors, *Proceedings of the International Workshop on Mining and Learning with Graphs (MLG)*, pages 205–212, 2006.

[10] B. Bringmann, A. Zimmermann, L. D. Raedt, and S. Nijssen. Don't be afraid of simpler patterns. In *10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 55–66. Sprinter, 2006.

[11] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. Gboost: A mathematical programming approach to graph classification and regression. *Machine Learning*, 2008.

[12] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

[13] G. Rätsch, A. Demiriz, and K.P. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3):189–218, 2002.

[14] F. Steinke, M. Seeger, and K. Tsuda. Experimental design for efficient identification of gene regulatory networks using sparse Bayesian models. *BMC Systems Biology*, 1:51, 2007.

[15] A. Schwaighofer, T. Schroeter, S. Mika, J. Laub, A. ter Laak, D. Sülzle, U. Ganzer, N. Heinrich, and K.-R. Müller. Accurate solubility predictions with error bars for electrolytes: A machine learning approach. *J. Chem. Inf. Model.*, 47:407–424, 2007.

[16] D. Wipf and S. Nagarajan. A new view of automatic relevance determination. In *Advances in Neural Information Processing Systems 21*, 2007.

[17] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[18] D. Wipf, J. Palmer, and B. Rao. Perspectives on sparse Bayesian learning. In *Advances in Neural Information Processing Systems 16*, 2004.

[19] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. Bakir. Weighted substructure mining for image analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2007.