A Linear Programming Approach for Molecular QSAR analysis

Hiroto Saigo[1], Tadashi Kadowaki[2], and Koji Tsuda[1]

[1] Max Planck Institute for Biological Cybernetics,
Spemannstr. 38, 72076 Tübingen, Germany
{hiroto.saigo,koji.tsuda}@tuebingen.mpg.de,
[2] Bioinformatics Center, ICR, Kyoto University
Uji, Kyoto 611-0011, Japan
tadakado@gmail.com

Abstract. Small molecules in chemistry can be represented as graphs. In a quantitative structure-activity relationship (QSAR) analysis, the central task is to find a regression function that predicts the activity of the molecule in high accuracy. Setting a QSAR as a primal target, we propose a new linear programming approach to the graph-based regression problem. Our method extends the graph classification algorithm by Kudo et al. (NIPS 2004), which is a combination of boosting and graph mining. Instead of sequential multiplicative updates, we employ the linear programming boosting (LP) for regression. The LP approach allows to include inequality constraints for the parameter vector, which turns out to be particularly useful in QSAR tasks where activity values are sometimes unavailable. Furthermore, the efficiency is improved significantly by employing multiple pricing.

1 Introduction

Nowadays we are facing a problem of screening a huge number of molecules in order to testify, e.g., if it is toxic to human or it has an effect on HIV virus, etc. Such bioactivities or chemical reactivities are measured by laborious experiments, so selecting small number of good candidates for the later synthesis is important. A quantitative structure-activity relationship (QSAR) analysis, a process to relate a series of molecular features with biological activities or chemical reactivities, is expected to decrease a number of expensive experiments. The conventional QSAR analysis manipulates chemical data in a table in which molecules are defined by individual rows and molecular properties (descriptors) in binary or real values are defined by the associated columns. The prediction model is then built to be consistent to the structure-activity relationship. Our approach use subgraphs as molecular properties instead of conventional descriptors. Since we know that simple subgraphs are already included in conventional descriptors, and believe that enriching subgraph features would contribute to make a better prediction model.

A graph is a powerful mathematical framework which can deal with many real-world objects. Several approaches based on kernel methods have been proposed [1–6], which all consider molecules as graphs and tried defining distance measures between molecules. However, these kernel methods lack interpretability, because the feature space is implicitly defined and it is difficult to figure out which features played an important role in prediction.

We take the boosting approach, which works in a feature space explicitly defined by substructure indicators [7]. Therefore, it is rather easy to show the substructures contributed to activity predictions, which may lead to new findings by chemists. Though the number of possible subgraphs in graph database are exponentially large, the recent advance of graph mining algorithms [8–12] suggests a way to handle them. The graph boosting method by Kudo et al. [7] has to be modified in several ways for QSAR applications. First of all, the original classification algorithm should be modified to a regression algorithm, because the activity is real-valued. Very recently, Kadowaki et al.[13] used the graph boosting in a SAR task, where the problem is to predict a chemical compound is active or not. However, the activity values are continuous and they are not obviously separated into active/non-active categories. Second, in publicly available databases, the activity values are not always available, because, if some compounds are obviously inactive, they do not bother to measure the activity. Therefore, for many compounds, one knows that their activities are low, but the actual values are not available. We need a mechanism to take those unusual data into account. Finally, in AdaBoost, only one substructure is found by the search of the whole pattern space. So, if one needs d substructures for good accuracy, the graph mining has to be done d times. For more efficiency, it is desirable that multiple substructures are derived by one mining call.

Among several boosting algorithms for regression [14], we found the linear programming (LP) boosting proposed by Demiriz et al. [15] is most appropriate for our task. One reason is that the linear programming allows to include inequality constraints which are useful for incorporating the compounds with low activities. Another reason is that it is possible to obtain multiple structures by one graph mining call, because the LP boost always updates all the parameters whereas AdaBoost updates one parameter at a time. Finally, in the LP boost, the optimality of the solution can be evaluated by the duality gap, whereas, in AdaBoost, it is not obvious when to stop the iteration and it is hard to figure out the distance from the current solution to the optimal one.

In this paper, we will describe how the LP boost can be combined with the graph mining algorithm to yield an efficient graph regression algorithm. In experiments using Endocrine Disruptors Knowledge Base (EDKB), our method is favorably compared with the marginalized graph kernels [1] that are successfully applied to chemical data recently [5]. We also illustrate the speed up achieved by reducing the number of mining calls.

2 Graph Preliminaries

In this paper, we deal with undirected, labeled and connected graphs. To be more precise, we define the graph and its subgraph as follows:

Definition 1 (Labeled Connected Graph). *A labeled graph is represented in 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, $\mathcal{L} \in \mathcal{R}$ is a set of labels, and $l : V \cup E \rightarrow \mathcal{L}$ is 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 2 (Subgraph). *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)$\mathcal{L}' \subseteq \mathcal{L}$, and (4)$l' \subseteq l$. If G' is a subgraph of G, then G is a supergraph of G'.*

To apply a machine learning method to graphs, one has to represent a graph as a feature vector. One idea is to represent a graph of a set of paths as in marginalized graph kernels (MGK) [1]. MGK and similar methods are recently applied to the classification of chemical compounds [4, 5]. Although the computation of kernels (i.e., the dot product of feature vectors) can be done in polynomial time using the path representations, paths cannot represent structural features such as loops and often end up with poor results (e.g., [16]). Therefore, we employ the substructure representation (Figure 1), where the feature vector consists of binary indicators of *patterns* (i.e., small graphs). Now our central issue is how to select patterns informative for regression. We will adopt the boosting approach to solve the problem as described in the next section. In chemoinformatics, it is common that a set of small graphs (i.e., fingerprints) is determined a priori, and the feature vector is constructed based on them [17]. However, we do not rely on ready-made fingerprints to search for unexplored features and to make our method applicable to any graph regression problem in other areas of science.

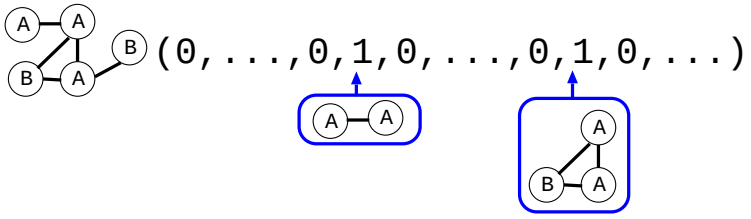

Fig. 1. Substructure representation of a graph. Each substructure is restored in the corresponding position in a vector.

3 Graph Regression by Linear Programming

When a graph is represented as a vector of binary indicators of all possible substructures, the dimensionality becomes too large for conventional regression methods such as ridge regression. In this work, we employ a regression method based on the LP (linear programming) boosting [15, 18], because it greedily selects features in learning and can avoid computational problems in a systematic way. In feature selection, we need to search for the best feature in the whole pattern space. To perform the search efficiently, we adopted a data structure called DFS code tree [12] as will be described in the next section. A QSAR problem is basically considered as a graph regression problem, where graph-activity pairs are given as the training data, and the activities of test graphs are predicted by the learned regression function. However, one problem is that the activity is measured only for the chemicals that are suspected to be active. For apparently inactive chemicals, nobody bothers to measure their activities. Thus, in the database, we have a set of chemicals with real-valued activities and a set of chemicals known to be inactive but the actual activity values are not available. The latter examples are called "clearly negative data". In the following formulation, those examples appear as inequality constraints of the weight vector.

Let $\mathbf{x} \in \mathcal{R}^d$ be a feature vector. Given the training examples $\{\mathbf{x}_i, y_i\}_{i=1}^m$ and clearly negative examples $\{\bar{\mathbf{x}}_k\}_{k=1}^l$, our objective is to learn the regression function

$$f(\mathbf{x}) = \sum_{j=1}^{d} \alpha_j x_j.$$

where α_j is a weight parameter. The objective function to minimize is as follows:

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{m} |\boldsymbol{\alpha}^\top \mathbf{x}_i - y_i|_\epsilon + \sum_{k=1}^{l} |\boldsymbol{\alpha}^\top \bar{\mathbf{x}}_k - z|_+ + \frac{1}{C} \sum_{j=1}^{n} |\alpha_j|$$

where z is a predetermined negative constant, C is the regularization parameter and $|\cdot|_\epsilon$ is the ϵ-insensitive loss [19], and $|\cdots|_+$ is the hinge loss, namely $|t|_+ = t\ (t \geq 0),\ 0\ (t < 0)$. We used the L1-regularizer to force most of the weights to be exactly zero in solution. Even if the dimensionality is large, the number of non-zero weights is kept small by this regularizer. The solution is obtained by solving the following linear programming.

$$\min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \ ||\boldsymbol{\alpha}||_1 + C \sum_{i=1}^{m} \xi_i^+ + \xi_i^- + C \sum_{k=1}^{l} \xi'_k \tag{1}$$

$$s.t. \ \ \boldsymbol{\alpha}^\top \mathbf{x}_i - y_i \leq \epsilon + \xi_i^+, \quad \xi_i^+ \geq 0, \quad i = 1, \ldots, m \tag{2}$$

$$y_i - \boldsymbol{\alpha}^\top \mathbf{x}_i \leq \epsilon + \xi_i^-, \quad \xi_i^- \geq 0, \quad i = 1, \ldots, m \tag{3}$$

$$\boldsymbol{\alpha}^\top \bar{\mathbf{x}}_k \leq z + \xi'_k, \quad k = 1, \ldots, l \tag{4}$$

where ξ_i^+, ξ_i^- are slack variables for over-estimation and under-estimation, respectively, and ξ'_k is for clearly negative example. Let u_i^+, u_i^-, v_k be the Lagrange multipliers for the constraints (2), (3) and (4), respectively. Setting $u_i = u_i^+ - u_i^-$, the dual of the above problem is written as

$$\min_{\mathbf{u},\mathbf{v}} \; z\sum_{k=1}^{l} v_k - \sum_{i=1}^{m} y_i u_i + \epsilon \sum_{i=1}^{m} |u_i| \tag{5}$$

$$s.t. \; -1 \leq \sum_{i=1}^{m} u_i x_{ij} - \sum_{k=1}^{l} v_k \bar{x}_{kj} \leq 1, \; j = 1, \cdots, d \tag{6}$$

$$-C \leq u_i \leq C, \; i = 1, \ldots, m \tag{7}$$

$$0 \leq v_k \leq C, \; k = 1, \ldots, l \tag{8}$$

Instead of the primal problem, we will solve the dual problem and recover the solution for $\boldsymbol{\alpha}$ from the Lagrange multipliers of the dual problem [15, 18].

When the number of features d is extremely large, it is computationally prohibitive to solve the dual problem directly. Such a large scale problem is typically solved by the column generation (CG) algorithm [15, 18], where one starts from a restricted problem with a small number of constraints and necessary constraints are added one by one. At each step, the restricted LP problem is solved, and the solution at step t is used to select the constraint added in step $t+1$. The procedure continues until the convergence, or we can trade the accuracy with the computational time by stopping it before convergence. In our case, the problematic part is (6), so the column generation is performed with respect to the constraints in (6).

The efficiency of the CG algorithm depends crucially on the *pricing* step, where the importance of each constraint is evaluated using an intermediate solution. Here we select the constraint which is violated the most.

$$j^* = \arg\max_j \left| \sum_{i=1}^{m} u_i x_{ij} - \sum_{k=1}^{l} v_k \bar{x}_{kj} \right|. \tag{9}$$

If the maximum value is below or equal to 1, the column generation is stopped. It is also possible to add multiple constraints at a time. For example, one can sort the constraints based on the score (9), and take the top t constraints. This technique is called *multiple pricing* [20] and we will actually adopt it for reducing the number of searches.

When the substructure representation of a graph is employed, the number of all constraints is extremely large, thus we need a specialized machinery to obtain the maximally violated constraint. Since each constraint corresponds to a pattern, the search (9) is formulated as a graph mining problem as explained below.

4 Weighted Substructure Mining

Graph mining algorithms such as gspan efficiently enumerate the set of patterns that satisfy a predetermined condition [12]. Denote by $\mathcal{G} = \{G_i\}_{i=1}^n$ a graph database including l clearly negative examples ($n = m + l$), and let $\mathcal{T} = \{T_j\}_{j=1}^d$ be the set of all patterns, i.e., the set of all subgraphs included in at least one graph in $\mathcal{G}$. There are many variations of graph mining, but the most common one is the frequent substructure mining, where the task is to enumerate all patterns whose support is more than s,

$$S_{freq} = \{j \mid \sum_{i=1}^{n} I(T_j \subseteq G_i) \geq s\}. \tag{10}$$

On the other hand, what we need now is the *weighted substructure mining* to search for the best pattern in terms of the score

$$j^* = \arg\max_j \left| \sum_{i=1}^{n} w_i (2x_{ij} - 1) \right|, \tag{11}$$

where x_{ij} is defined as $x_{ij} := I(T_j \subseteq G_i)$, the weight for a training example is $w_i = u_i - \frac{1}{m}\sum_{k=1}^{m} u_k$, and the weight for a clearly negative example is $w_i = -v_i + \frac{1}{l}\sum_{k=1}^{l} v_k$.

The key idea of efficient graph mining is to exploit the *anti-monotonicity*, namely the frequency of a pattern is always smaller than or equal to that of its subgraph. In frequent substructure mining (10), one constructs a tree-shaped search space (i.e., DFS code tree) where each node corresponds to a pattern (Figure 2). The tree is generated from the root with an empty graph, and the pattern of a child node is made by adding one edge. As the pattern gets larger, the frequency decreases monotonically. If the frequency of the generated pattern T_j is s, it is guaranteed that the frequency of any supergraph of T_j is less than s. Therefore, the exploration is stopped there (i.e., *tree pruning*). By repeating node generation until all possibilities are checked, all frequent subgraphs are enumerated.

In the tree expansion process, it often happens that the generated pattern is isomorphic to one of the patterns that have already been generated. It leads to significant loss of efficiency because the same pattern is checked multiple times. The gspan algorithm solves this problem by the minimum DFS code approach, and we also adopted it for pruning isomorphic patterns.

In weighted substructure mining (11), the search tree is pruned by a different condition. Let us rewrite the weight as $w_i = y_i d_i$ where $d_i = |w_i|$ and $y_i = \mathrm{sign}(w_i)$. Then, the following bound is obtained: For any $T_j \subseteq T_k$,

$$\left| \sum_{i=1}^{n} w_i (2x_{ik} - 1) \right| \leq \gamma,$$

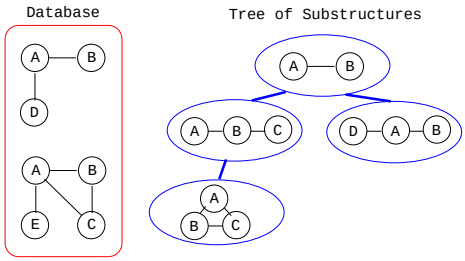

Fig. 2. Schematic figure of the tree-shaped search space of patterns (i.e., substructures)

where $\gamma = \max(\gamma^+, \gamma^-)$ and

$$\gamma^+ = 2 \sum_{\{i|y_i=+1, T_j \subseteq G_i\}} d_i - \sum_{i=1}^{n} d_i y_i,$$

$$\gamma^- = 2 \sum_{\{i|y_i=-1, T_j \subseteq G_i\}} d_i + \sum_{i=1}^{n} d_i y_i.$$

See [7] for the proof. When a pattern T_j is generated, the scores of its supergraphs T_k are upperbounded as above. Thus, if the upperbound is less than the current best value, we can safely quit further exploration.

4.1 Use of multiple constraints

In practice we found that the computation time of our algorithm is dominated by graph mining algorithm, so we propose to use multiple subgraphs for each iteration (multiple pricing[20]) in order to decrease the number of graph mining. This can be performed by mining top k subgraphs at each graph mining. In order to implement this change, we use a band of τ to maintain top k subgraphs which maximizes $|\sum_{i=1}^{m} u_i x_{ij} - \sum_{k=1}^{l} v_k \bar{x}_{kj}|$. It should be noted that no matter how many subgraphs we add for each iteration, the solution is kept optimal by solving linear programming [18].

5 QSAR Experiments

We used Endocrine Disruptors Knowledge Base (EDKB) data provided by the National Center for Toxicological Research[3] for measuring performance of our algorithms. Endocrine disruption is caused by the interference of the endocrine system by environmental or exogenous chemicals. The E-SCREEN assay of the EDKB consists of 59 molecules with activities provided in real number (logRPP).

[3] http://edkb.fda.gov/databasedoor.html

Table 1. 5-fold cross validation results on 59 molecules. For MGK, stopping probability 0.5 is chosen from $\{0.1 \dots 0.9\}$. $k = 3$ is chosen for kNN, and a ridge $1e-5$ is used for ridge regression. Proposed algorithm is stopped at 50 iteration.

Methods	l_1 error	l_2 error	time[s]	iterations	subgraphs
MGK + kNN	0.338±0.0326	0.240±0.0485	10.1	-	-
MGK + ridge	0.343±0.0282	0.213±0.0470	10.3	-	-
Proposed ($\epsilon = 0.01$)	0.291±0.0185	0.157±0.00701	30.7	10.3	9.6
Proposed ($\epsilon = 0.1$)	0.239±0.0136	0.107±0.00481	46.4	15.2	14.2
Proposed ($\epsilon = 0.2$)	**0.227±0.0124**	**0.101±0.00340**	139	37.6	14.4
Proposed ($\epsilon = 0.3$)	0.237±0.0124	0.123±0.00424	200	50	6.2
Proposed ($\epsilon = 0.5$)	0.282±0.0200	0.170±0.00887	178	50	2

We also used 61 clearly negative data, and set z to the lowest active level in the active data. The parameter C, which controls a generalization error, was set to 0.2. The performance was measured by 5 fold cross validation on Linux with Pentium4 2.4Ghz processor.

We compared our method with marginalized graph kernel (MGK) [1] in combination with ridge regression or kNN regression. MGK-based regression, however, cannot correctly include the information of clearly negative data, thus we just added them with labels as same value as the lowest active level in the active data. For comparison, we tried the same experimental setting on our graph regression algorithm, and denoted it as Proposed* in Table 2.

The results in the EDKB dataset are shown in Tables 1 and 2. For MGK with kNN regression or ridge regression, we can observe that inclusion of negative data degrades the performance. Performance of our our method was constantly better than MGK indifferent to regression algorithms suggests that our method better extracted the structurally characteristic patterns. Also good performances of "Proposed" over "Proposed*" in Table 2 validates our way of incorporating clearly negative data.

All the extracted subgraphs from 120 molecules are illustrated in Figure 3. Subgraphs are ordered from the top left to the bottom right according to their weights.

There are pros and cons both for classification and regression, i.e., classification involves a problem of discretization of activities while regression does not, but regression does not take into account clearly negative data. In this sense, our method is located between classification and regression.

While we built a regression model and found its component subgraphs, however, those extracted subgraphs are sometimes not so easy to interpret. For example, if a simple carbohydrate chain with fixed length is extracted, there are many ways of superimposing it on a molecule. Enriching atom and bond labels would be a better way to overcome this difficulty, and we are investigating this direction.

We can see from Figure 4 that the decrease in the number of iterations until convergence is almost in proportion to k, and we can see a similar curve for

Table 2. 5-fold cross validation results on 120 molecules. For MGK, stopping probability 0.5 is chosen from $\{0.1 \dots 0.9\}$. $k = 3$ is chosen for kNN, and a ridge $1e-5$ is used for ridge regression. Proposed algorithm is stopped at 50 iteration. Proposed* method regards 61 negative data labeled as lowest active value in the active data.

Methods	l_1 error	l_2 error	time[s]	iterations	subgraphs
MGK + kNN	0.474±0.0644	0.361±0.0469	29.0	-	-
MGK + ridge	0.454±0.0577	0.335±0.0431	29.1	-	-
Proposed ($\epsilon = 0.01$)	0.234±0.0114	0.100±0.00262	57.0	11.6	10.6
Proposed ($\epsilon = 0.1$)	**0.232±0.0108**	**0.087±0.00207**	112	21.6	20.6
Proposed ($\epsilon = 0.2$)	0.233±0.0132	0.101±0.00298	296	50	20.8
Proposed ($\epsilon = 0.3$)	0.249±0.0154	0.126±0.00420	289	50	15.6
Proposed ($\epsilon = 0.5$)	0.288±0.0201	0.163±0.00563	272	50	11.4
Proposed* ($\epsilon = 0.01$)	0.277±0.0191	0.153±0.00997	51.2	10	9
Proposed* ($\epsilon = 0.1$)	0.267±0.0182	0.128±0.00731	88.8	16.6	17.6
Proposed* ($\epsilon = 0.2$)	0.250±0.0139	0.104±0.00430	173	29.8	25.6
Proposed* ($\epsilon = 0.3$)	0.238±0.0124	0.106±0.00299	322	50	22.2
Proposed* ($\epsilon = 0.5$)	0.278±0.0205	0.147±0.00691	329	50	11.8

the time until convergence. The l_1 error and the number of subgraphs which contributed to the final ensemble almost did not change over different k, which guarantees the appropriate termination of the algorithm. The optimal number of k which let the learning algorithm converge the fastest depends on the data, but the LP theory gives validity of the final ensemble independent of the selection of k, and we observed no practical disadvantages just by setting k large.

6 Discussions

We have presented a graph regression algorithm using multiple subgraphs weighted by a linear programming. Experiments are carried out to show the usefulness of the algorithm in regression problems.

A method that first discovers all the subgraphs satisfying a certain condition, then classifies graphs by SVMs exists [21]. Our method, however, can discover subgraphs and add them to an ensemble simultaneously, therefore can save the cost of discovering subgraphs which satisfies some condition but do not contribute to the final ensemble. Also, knowledge based SVMs [22, 23] can take into account inequality constraints as well, but our algorithm is more efficient by the same reason above.

The importance of aligning functional groups in QSAR/QSPR is discussed in [4] and [5]. Alignment of pharmacophore was used to be done manually, but our algorithm is alignment-free, and might be useful for this problem.

Our algorithm automatically selects sparse features due to sparse regularizer. This has an advantage in interpretability, and we are not required to define or pre-register key structures [24], or to find a pre-image [25]. However, combining classical features such as partial charges, logP etc. might be useful to build a

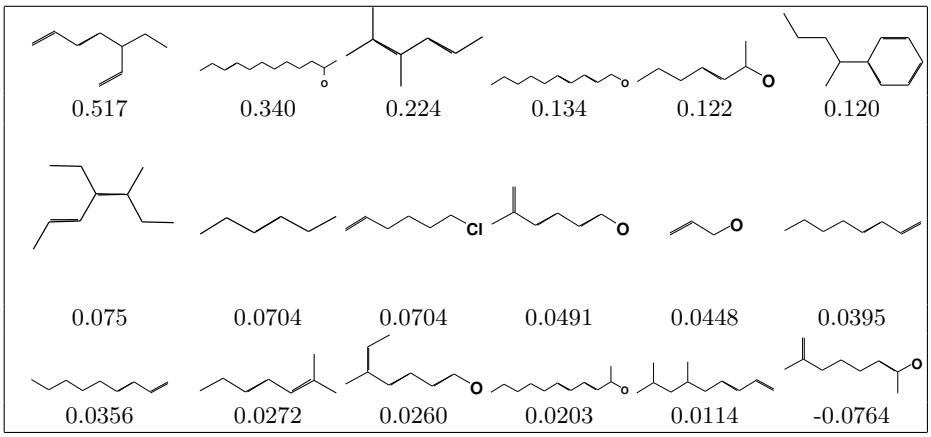

Fig. 3. Extracted subgraphs from 120 molecules. Subgraphs are ordered from the top left to the bottom right according to their weights. H atom is omitted, and C atom is represented as a dot for visual interpretability.

more precise model, and we are investigating this direction. Finally, we focused on mentioning properties and applications of our method in chemistry, but the framework of graph classification and regression is general, and can be applied to any data which consists of graphs.

Acknowledgments Computational resources were provided by the Bioinformatics Center, Institute for Chemical Research, Kyoto University, and the Supercomputer Laboratory, Kyoto University.

References

1. H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the twenty-first International Conference on Machine Learning*, pages 321–328. AAAI Press, 2003.
2. T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the sixteenth Annual Conference on Computational Learning Theory and seventh Kernel Workshop*, pages 129–143. Springer Verlag, 2003.
3. L. Ralaivola, S.J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Netw.*, 18(8):1093–1110, 2005.
4. H. Fröhrich, J. Wegner, F. Sieker, and Z. Zell. Kernel functions for attributed molecular graphs - a new similarity based approach to adme prediction in classification and regression. *QSAR & Combinatorial Science*, 25(4):317–326, 2006.
5. P. Mahé, L. Ralaivola, V. Stoven, and J.P. Vert. The pharmacophore kernel for virtual screening with support vector machines. Technical report, 2006. Technical Report HAL:ccsd-00020066.

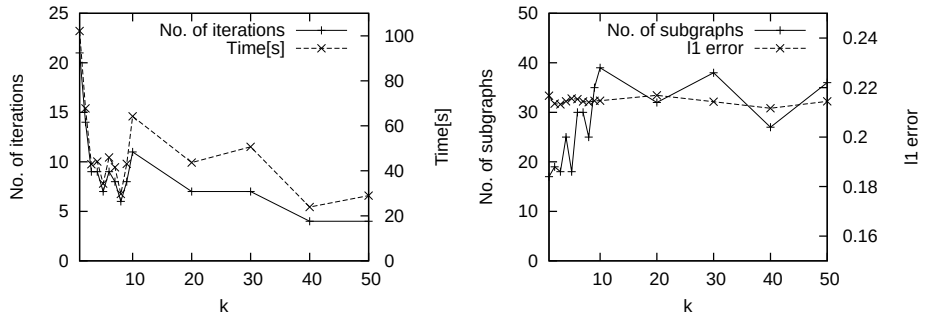

Fig. 4. Speed up of the algorithm by multiple pricing. We can see the number of boosting iterations and the time until convergence decreases in proportion to k (left), while keeping the number of subgraphs and l_1 error almost constant (right).

6. T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data minig*, pages 158–167. ACM Press, 2004.
7. T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *Advances in Neural Information Processing Systems 17*, pages 729–736. MIT Press, 2005.
8. S. Kramer, L.D. Raedt, and C. Helma. Molecular feature mining in hiv data. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2001.
9. L.D. Raedt and S. Kramer. The level-wise version space algorithm and its application to molecular fragment finding. In *Proceedings of the seventeenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 2001.
10. S. Kramer and L.D. Raedt. Feature construction with version spaces for biochemical applications. In *Proceedings of the eighteenth International Conference on Machine Learning*. AAAI Press, 2001.
11. A. Inokuchi. Mining generalized substructures from a set of labeled graphs. In *Proceedings of the fourth IEEE Internatinal Conference on Data Mining*, pages 415–418. IEEE Computer Society, 2005.
12. 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.
13. T. Kadowaki, T. Adachi, T. Kudo, S. Okamoto, N. Tanaka, C.E. Wheelock, K. Tonomura, K. Tsujimoto, H. Mamitsuka, S. Goto, and M. Kanehisa. Chemical genomic study in endocrine disruptors on metabolic pathways. submitted, 2006.
14. R. Meir and G. Rätsch. An introduction to boosting and leveraging. In *Lecture Notes in Computer Science: Advanced lectures on machine learning*, pages 118–183, Heidelberg, 2003. Springer-Verlag.
15. A. Demiriz, K.P. Bennet, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.
16. K. Tsuda and T. Kudo. Clustering graphs by weighted substructure mining. In *Proceedings of the twenty-third International Conference on Machine Learning*, pages 953–960. ACM Press, 2006.

17. J. Gasteiger and T. Engel. *Chemoinformatics: a textbook*. Wiley-VCH, 2003.
18. 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.
19. B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
20. D. G. Luenberger. *Optimization by Vector Space Methods*. Wiley, 1969.
21. S. Kramer, E. Frank, and C. Helm. Fragment generation and support vector machines for including sars. *SAR and QSAR in Environmental Research*, 13(5):509–523, 2002.
22. O.L. Mangasarian and E.W. Wild. Knowledge-based kernel approximation. *Journal of Machine Learning Research*, 5:1127–1141, 2004.
23. Q.V. Le, A.J. Smola, and T. Gärtner. Simpler knowledge-based support vector machines. In *Proceedings of the twenty-third International Conference on Machine Learning*. ACM Press, 2006.
24. C.A. James, D. Weininger, and J. Delany. Daylight theory manual, 2004.
25. T. Akutsu and D. Fukagawa. Inferring a graph from path frequency. In *Proceedings of the sixteenth Annual Symposium on Combinatorial Pattern Matching*, pages 371–382. Springer, 2005.