Predicting activities without computing descriptors: graph machines for QSAR§

Article *in* SAR and QSAR in Environmental Research · January 2007

DOI: 10.1080/10629360601054313 · Source: PubMed

4 authors, including:

Aurélie Goulon
Lavoix
12 PUBLICATIONS **90** CITATIONS
SEE PROFILE

Gérard Dreyfus
École Supérieure de Physique et de Chimie Industrielles
226 PUBLICATIONS **5,019** CITATIONS
SEE PROFILE

SAR and QSAR in Environmental Research
vol. 18, pp. 141 – 153 (2007)

Predicting activities without computing descriptors: graph machines for QSAR

A. GOULON†, T. PICOT†, A. DUPRAT‡†, and G. DREYFUS†*

†Laboratoire d'Électronique, École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris, (ESPCI-ParisTech), 10 rue Vauquelin, 75005 PARIS, France

‡Laboratoire de Chimie Organique, ESPCI-ParisTech, 10 rue Vauquelin, 75005 PARIS, France (CNRS UMR 7084)

Running title: Graph machines for QSAR

*Corresponding author. Email: Gerard.Dreyfus@espci.fr

We describe graph machines, an alternative approach to traditional machine-learning-based QSAR, which circumvents the problem of designing, computing and selecting molecular descriptors. In that approach, which is similar in spirit to recursive networks, molecules are considered as structured data, represented as graphs. For each example of the data set, a mathematical function (graph machine) is built, whose structure reflects the structure of the molecule under consideration; it is the combination of identical parameterized functions, called "node functions" (e.g. a feedforward neural network). The parameters of the node functions, shared both within and across the graph machines, are adjusted during training with the "shared weights" technique. Model selection is then performed by traditional cross-validation. Therefore, the designer's main task consists in finding the optimal complexity for the node function. The efficiency of this new approach has been demonstrated in many QSAR or QSPR tasks, as well as in modeling the activities of complex chemicals (e.g. the toxicity of a family of phenols or the anti-HIV activities of HEPT derivatives), generally outperforming traditional techniques without requiring the selection and computation of descriptors.

Keywords: Graph; Graph machine; Structured data; Machine learning; Phenol toxicity; Anti-HIV activity; Carcinogenicity

1. Introduction

In recent years, the development of machine-learning techniques focused mainly on methods for learning from *vector* data. For a given problem, finding a vector representation of the data may be straightforward; in order to model a physical process for instance, the variables that have an influence on the quantity of interest are usually known from prior knowledge, and, after some preprocessing (normalization) they can be grouped into a vector that is input to a multilinear model, a polynomial, a kernel machine, a neural network, etc. However, there are problems for which the computation of the variables of the model is a difficult and time-consuming task; such is the case for QSAR/QSPR problems, where relevant variables (called descriptors) must be designed and computed. For each activity to be predicted, a set of potentially relevant descriptors must be selected from a pool of conventional descriptors, and/or designed specifically; appropriate methods for computing those descriptors must be found, and descriptor selection must be performed in order to keep only truly relevant descriptors.

Alternatively, one may consider that a wealth of information is present in the graph structure of the molecule itself; this spurred the recent development of machine-learning methods that learn from graphs instead of learning from vectors. The first part of the present paper is devoted to the description of graph machines, which exempt the model designer from designing or computing any descriptor. The second part analyzes results obtained for the prediction of two activities: the toxicity of phenols and the anti-HIV activity of a family of molecules; the results of a classification task on the carcinogenicity of molecules are also described. Comparisons with results obtained by other authors on the same databases show that graph machines perform at least as well, and generally better, than traditional methods, without requiring the design, computation and selection of descriptors.

2. Graph machines

The idea of learning from graphs (and generally structured data) can be traced back to the early days of machine learning, when Recursive Auto-Associative Memories (RAAMs) were designed for providing compact representations of trees [1]. It evolved subsequently to Labeled RAAMs, recursive networks, and graph machines (for a review of the development of numerical machine-learning from structured data, see [2]).

2.1. Definition

Graph regression (resp. graph classification) consists in performing a mapping between a set G of graphs and a set of real (resp. binary) numbers.

2.1.1. Acyclic graphs

For each acyclic graph G_i of G, a function $g^i\ \mathbb{R}^n \to \mathbb{R}$ ("graph machine") is constructed as a combination of identical parameterized functions (node functions), in such a way that g^i has the same structure as graph G_i. Function g^i is intended (i) to encode the structure of the graph [3], and (ii) to provide a prediction of the quantity of interest, e.g. a property or an activity of the molecule, from its graph structure. If the node functions are neural networks, the g^i's are termed, in some cases, recursive neural networks [4].

As an example, Figure 1 shows a set of three graphs; the functions are obtained by replacing each node by a parameterized node function f_θ, where $\boldsymbol{\theta}$ is the vector of parameters of the function; the function $F_{\boldsymbol{\Theta}}$ that replaces the root node may be different from the function implemented at the non-root nodes:

Graph G_1: $g^1_{\boldsymbol{\theta},\boldsymbol{\Theta}}\left(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4\right) = F_{\boldsymbol{\Theta}}\left(\mathbf{x}_4, f_{\boldsymbol{\theta}}\left(\mathbf{z}_1\right), f_{\boldsymbol{\theta}}\left(\mathbf{z}_2\right), f_{\boldsymbol{\theta}}\left(\mathbf{z}_3\right)\right)$, where

- $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$ are optional label vectors of size X_1, which convey information about the nodes. If no specific information is necessary to perform the task at hand, one has $X_1 = 0$; therefore, the value of $g^1_{\boldsymbol{\theta},\boldsymbol{\Theta}}$ depends on the graph only.
- $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$ are of size D_1+1; denoting by d_k the in-degree of node k, and defining $M_1 = \arg\max_k d_k$, one has $D_1 = M_1 + X_1$: in this example, $M_1 = 3$, hence $D_1 \geq 3$; those vectors are constructed as follows: for all nodes, the first component z_0 is equal to 1; for node k, of in-degree d_k, components 2 to d_k+1 of $\mathbf{z}_k$ are the values of $f_{\boldsymbol{\theta}}$ computed by the parent nodes of node k; if $d_k < M_k$, components d_k+2 to M_1+1 are equal to zero; if $X_1 \neq 0$, components M_1+2 to M_1+1+X_1 are the components of vector $\mathbf{x}_k$.

In QSAR/QSPR problems for instance, the information conveyed by the labels may be the chemical nature of the atom present at each node in the graph of the molecule. It may also be molecular descriptors, but it has been shown in [5] that molecular descriptors can be learnt by graph machines, so that inputting descriptors to graph machines is often redundant.

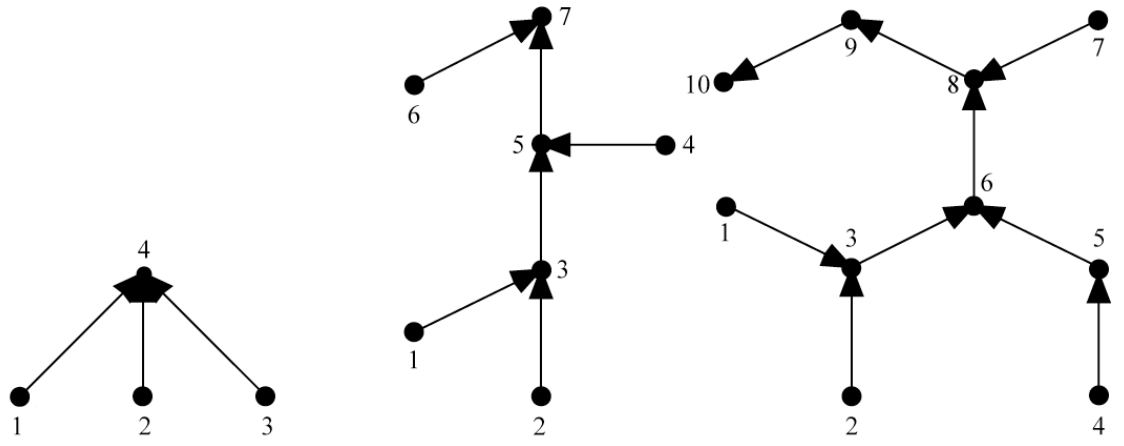

Figure 1

Three directed acyclic graphs

In the present example, if no information about the nodes is necessary ($X_1 = 0$), one has: $D_1 = 3$, $\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{z}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$.

Graph G_2: $g^2_{\boldsymbol{\theta},\Theta}\left(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4,\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7\right) = F_{\Theta}\left(\mathbf{x}_7, f_{\boldsymbol{\theta}}\left(\mathbf{z}_6\right), f_{\boldsymbol{\theta}}\left(\mathbf{x}_5, f_{\boldsymbol{\theta}}\left(\mathbf{z}_4\right), f_{\boldsymbol{\theta}}\left(\mathbf{x}_3, f_{\boldsymbol{\theta}}\left(\mathbf{z}_2\right), f_{\boldsymbol{\theta}}\left(\mathbf{z}_1\right)\right)\right)\right)\right)$,

where vectors $\mathbf{x}_1$ to $\mathbf{x}_7$ and $\mathbf{z}_1$ to $\mathbf{z}_6$ are constructed similarly to those of graph G_1, with $M_2 = 2$. If no label is necessary ($X_2 = 0$), one has:

$D_2 = 2$, $\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{z}_4 = \mathbf{z}_6 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$, $\mathbf{z}_3 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_1\right) & f_{\boldsymbol{\theta}}\left(\mathbf{z}_2\right) \end{pmatrix}^{\mathrm{T}}$, $\mathbf{z}_5 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_3\right) & f_{\boldsymbol{\theta}}\left(\mathbf{z}_4\right) \end{pmatrix}^{\mathrm{T}}$, $\mathbf{z}_7 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_5\right) & f_{\boldsymbol{\theta}}\left(\mathbf{z}_6\right) \end{pmatrix}^{\mathrm{T}}$.

Graph G_3:

$$g^3_{\boldsymbol{\theta},\Theta}\left(\mathbf{x}_1,\ldots,\mathbf{x}_{10}\right) = F_{\Theta}\left(\mathbf{x}_{10}, f_{\boldsymbol{\theta}}\left(\mathbf{x}_9, f_{\boldsymbol{\theta}}\left(\mathbf{x}_8, f_{\boldsymbol{\theta}}\left(\mathbf{z}_7\right), f_{\boldsymbol{\theta}}\left(\mathbf{x}_6, f_{\boldsymbol{\theta}}\left(\mathbf{x}_5, f_{\boldsymbol{\theta}}\left(\mathbf{z}_4\right)\right), f_{\boldsymbol{\theta}}\left(\mathbf{x}_3, f_{\boldsymbol{\theta}}\left(\mathbf{z}_2\right), f_{\boldsymbol{\theta}}\left(\mathbf{z}_1\right)\right)\right)\right)\right)\right)\right)$$

where vectors $\mathbf{x}_1$ to $\mathbf{x}_{10}$ and $\mathbf{z}_1$ to $\mathbf{z}_9$ are constructed as explained above, with $M_3 = 2$.

If the three graphs are to be used as a training set, as described in section 2.2, the three graph machines have the same node functions, so that the number of variables of the node functions is $D = \max_i D_i$, $i = 1$ to 3.

2.1.2. Cyclic graphs

Graph machines handle cyclic graphs and parallel edges, so that, for QSAR/QSPR applications, cycles and multiple bonds can be handled. To that effect, the initial graph is preprocessed by deleting a number of edges equal to the number of cycles, and all parallel edges but one, and by assigning to each node a label that is equal to its degree, thereby retaining the information about the original graph structure. Finally, a root node is chosen and the edges are assigned orientations, according to an algorithm described in [6].

2.2. The training of graph machines

Graph machines are trained in the usual framework of empirical risk minimization. A cost function $J(\Theta,\boldsymbol{\theta})$ is defined, and its minimum with respect to the parameters is sought, given the available training data. The cost function takes into account the discrepancy between the predictions of the models and the observations present in the training set, and may include regularization terms, e.g.

$$J\left(\boldsymbol{\theta},\Theta\right) = \sum_{i=1}^{N}\left(y^i - g^i_{\boldsymbol{\theta},\Theta}\right)^2 + \lambda_1\left\|\boldsymbol{\theta}\right\| + \lambda_2\left\|\Theta\right\| \qquad (1)$$

where N is the size of the training set, y^i is the value of the i-th observation of the quantity to be modeled, and λ_1 and λ_2 are suitably chosen regularization constants.

Since the parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ must be identical within each function g^i and across all those functions, one must resort to the so-called *shared weight* trick; the k-th component of the gradient of the cost function can be written as:

$$\frac{\partial J}{\partial \theta_k} = \sum_{i=1}^{N} \frac{\partial J^i}{\partial \theta_k} \tag{2}$$

where J^i is the contribution of example i to the cost function. We denote by $n_{\theta_k}^i$ the number of occurrences of parameter θ_k in acyclic graph G_i; if the root is assigned the same parameterized function as the other nodes, then $n_{\theta_k}^i$ is equal to the number of nodes in graph G_i. The shared weight trick consists in setting

$$\frac{\partial J^i}{\partial \theta_k} = \sum_{j=1}^{n_{\theta_k}^i} \frac{\partial J^i}{\partial \theta_{k_j}} \tag{3}$$

so that finally one has:

$$\frac{\partial J}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{j=1}^{n_{\theta_k}^i} \frac{\partial J^i}{\partial \theta_{k_j}} \tag{4}$$

Relation (4) is subsequently used for minimizing cost function (1) by any suitable gradient descent algorithm (Levenberg-Marquardt, BFGS, conjugate gradient, …).

If functions $f_{\boldsymbol{\theta}}$ and $F_{\boldsymbol{\Theta}}$ are neural networks, the usual backpropagation algorithm may be conveniently used for computing the gradient, otherwise one resorts to numerical estimations thereof.

2.3. Two didactic examples

In order to clarify the method, we consider here two toy examples. First, assume that one wishes to learn, from examples, the number of nodes of a graph. Assume that the training set consists of the three graphs shown on Figure 1; then the desired mapping is: $G_1 \rightarrow 4$; $G_2 \rightarrow 7$; $G_3 \rightarrow 10$. The node function f_θ is sought in the family of affine functions $f_{\boldsymbol{\theta}}(\mathbf{z}) = \sum_{j=0}^{D} \theta_j z_j$, and $F_{\boldsymbol{\Theta}}$ is taken identical to $f_{\boldsymbol{\theta}}$. Since no specific information about the nodes is required $X_1 = X_2 = X_3 = 0$. Since the node functions are the same for all graphs of the training set, we take $D = \max_i D_i = 3$, and the $\mathbf{x}_k$'s are non-existent. Since all edges are equivalent, one has $\theta_1 = \theta_2 = \theta_3 = \theta$. Therefore, there are actually 2 independent parameters.

The obvious solution is $\theta_0 = \theta = 1$. For graph G_1 for instance, one has:

$$g^1_{\boldsymbol{\theta},\boldsymbol{\Theta}}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = f_{\boldsymbol{\theta}}\left(1, f_{\boldsymbol{\theta}}(\mathbf{z}_1), f_{\boldsymbol{\theta}}(\mathbf{z}_2), f_{\boldsymbol{\theta}}(\mathbf{z}_3)\right) = \theta_0 + 3\theta_1\theta_0 = 4,$$

where $\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{z}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$.

Similarly, consider learning, from examples, the number of edges in a graph. Assume that, in addition to the previous 3 graphs, the training set contains example G_4, as shown on Figure 2. Graph G_4 is cyclic, so that it must first be preprocessed, e.g. by deleting the edge

between nodes 1 and 2; since it is mandatory, given the task at hand, to retain the information about the edges of the original graph, the degree of each node in the original graph is input as a label, i.e. X_k = 1 for all nodes k of all graphs of the training set: we take $D = 1 + \max_i M_i = 4$, and, for each node k of each graph, $\mathbf{x}_k$ is a scalar, equal to the degree of node k. No additional information is necessary, so that one has, for the nodes of G_4:

$g_{\boldsymbol{\theta}}^4(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) = f_{\boldsymbol{\theta}}(\mathbf{z}_5)$

with $\mathbf{x}_1 = 2$, $\mathbf{x}_2 = 2$, $\mathbf{x}_3 = 3$, $\mathbf{x}_4 = 1$, $\mathbf{x}_5 = 2$, $\mathbf{z}_1 = \mathbf{z}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \end{pmatrix}^{\mathrm{T}}$,

$\mathbf{z}_3 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}(\mathbf{z}_1) & f_{\boldsymbol{\theta}}(\mathbf{z}_2) & 0 & 3 \end{pmatrix}^{\mathrm{T}}$, $\mathbf{z}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}$, $\mathbf{z}_5 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}(\mathbf{z}_3) & f_{\boldsymbol{\theta}}(\mathbf{z}_4) & 0 & 2 \end{pmatrix}^{\mathrm{T}}$.

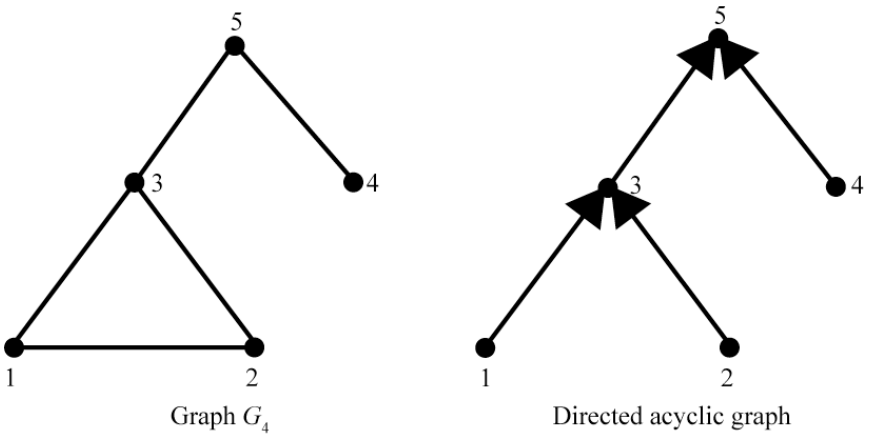

Figure 2
A cyclic graph

If a solution is sought within the family of affine functions $f_{\boldsymbol{\theta}}(\mathbf{z}) = \sum_{j=0}^{D} \theta_j z_j$ with z_0 = 1, the obvious solution is θ_0 = 0, $\theta_1 = \theta_2 = \theta_3 = \theta$ = 1, θ_4 = ½. This gives, for graph G_4 for instance:

$g_{\boldsymbol{\theta}}^4(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) = 1 + 2\theta + 2\theta^2 = 5$.

In the absence of any obvious solution, the training procedure described in section 2.2 should be used.

2.4. Model selection

Similarly to vector machines, usual model selection techniques such as hold-out, *K*-fold cross-validation, leave-one-out, virtual leave-one-out [7], bagging, can be applied to recursive networks and to graph machines. In the following applications to QSAR, hold-out and cross-validation were used.

3. Graph machines for QSAR

In the following, we describe applications where graph machines were applied to the prediction of activities. For comparison purposes, the databases were identical to those investigated previously by other authors.

3.1. Encoding the molecules

In the applications described here, the node functions f_{θ} are feedforward neural networks with a single hidden layer. The functions g_{θ}^{i} are generated from the SMILES files of the molecules by the following procedure: the molecules, described by these files, are converted into labelled graphs by the association of each non-hydrogen atom to a node, and each bond to an edge. The nodes are also assigned labels describing the atoms they are related to. The heteroatoms are encoded in a one-out-of-*N* code: if the molecules of the database feature *N* different types of heteroatoms, each node has *N* labels, each of which codes for one type of heteroatom. The label corresponding to the heteroatom present at the node is equal to one, while all others are equal to zero; if the node is a carbon atom, all labels are equal to zero. In addition, other labels encode the degree of the node (also in a one-out-of-*n* code), when the database includes molecules with multiple bonds or cycles, and its stereoisomery if necessary. The presence of hydrogen atoms is not encoded.

Then, the adjacency matrices associated to these labelled graphs are generated. In the subsequent step, the matrices are cast into a canonical form, by an algorithm that ranks the nodes according to criteria such as their degree, the fact that they belong to a cycle, the nature of the atom… [6]. Essentially, this algorithm chooses the root node of the acyclic graph to be constructed, and deletes the edges that are most distant from the root node. Figure 3 illustrates the processing of a molecule from its SMILES representation into a directed acyclic graph.

Graph machines are then built for each graph of the data set, by replacing each node of the graph by a feedforward neural network, whose complexity (i.e. the number of neurons in the hidden layer) is controlled by cross-validation. The graph machines are then trained, with the shared weight condition, using the software package NeuroOne™[1], which computes the gradient of the cost function by backpropagation and minimizes the cost function by the Levenberg-Marquardt algorithm.

[1] NeuroOne is a product of Netral S.A. (http://www.netral.com)

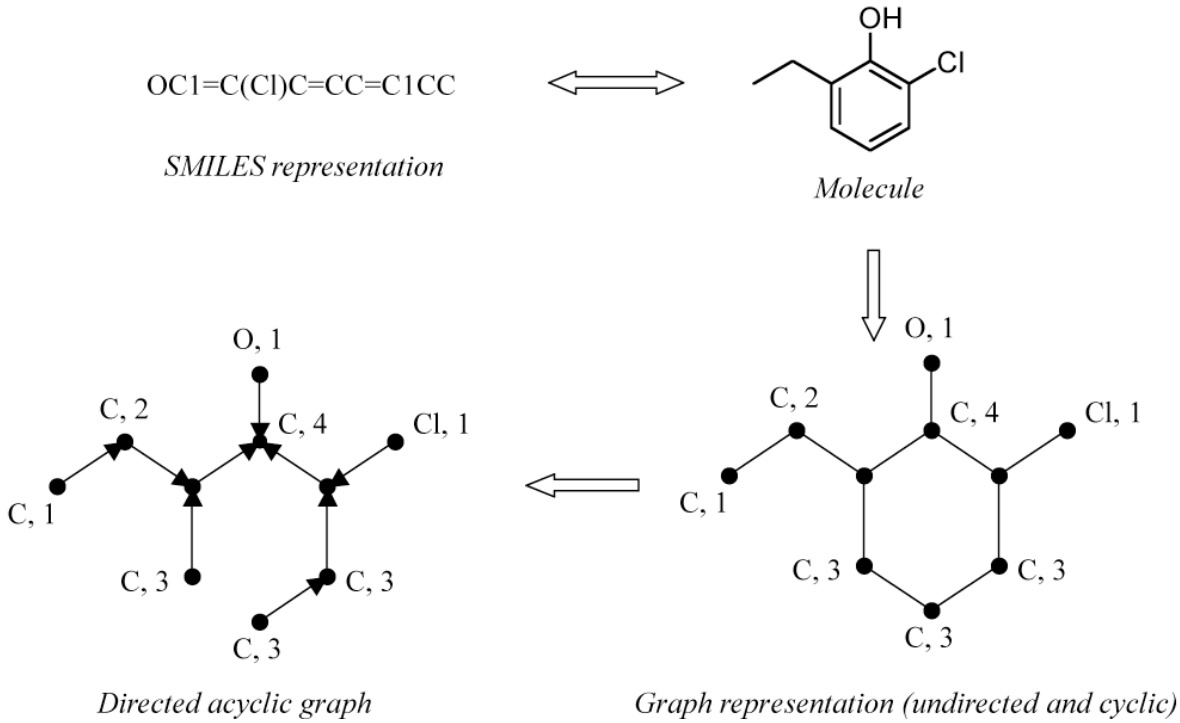

Figure 3
Encoding a molecule into a graph machine

For comparison purposes, the databases investigated in the following sections were those used previously by other researchers, so that the proportions of the different heteroatoms or stereoisomery in the training and test sets were not to be chosen freely. As usual in machine learning techniques, the database should represent as well as possible the types of molecules under investigation. In standard learning machines, influential examples can be detected by their *leverages* [8]; the generalization of leverages to graph machines is described in [7], and will be demonstrated in detail in a forthcoming article.

3.2. Predicting the toxicity of phenols

Phenol derivatives are widely used in the industry and in agriculture, for their activities as biocides or antiseptics. However, many of them also have a toxic activity and are considered as dangerous pollutants.

Several QSAR investigations tried to model this activity in the past. In particular, the toxicity of a family of phenols to an aquatic organism, the ciliate Tetrahymena pyriformis, was modelled with methods such as Multi Linear Regression (MLR), Radial Basis Function Neural Networks (RBFNN) and Support Vector Machines (SVM) [9, 10]. This toxicity arises from several modes of action, which can be categorized, according to structural criteria, into four groups: weak acid respiratory uncouplers, pro-electrophiles, soft electrophiles and polar narcotics [11]. In order to model the toxicity of phenols, two approaches can be taken [12]: the most straightforward consists in predicting the toxicity without taking into account the mechanism of action of the studied compound. However, the prediction may be inaccurate

since a single model may not fit correctly the four different mechanisms. Therefore we chose the alternative approach, and focused on one particular category, the polar narcotics. The studied data set features 153 phenol derivatives whose toxicity is expressed as $\log(1/IG_{50})$, where IG_{50} is the concentration in mM causing 50% inhibition of growth, after 40 hours, to T. pyriformis.

To provide a comparison with results obtained with the other methods, we used the same training and test sets, featuring 131 and 22 examples respectively, as in the referenced studies [9]. All predictions are based on computed descriptors such as the hydrophobicity, the acidity constant, the frontier orbital energies and the hydrogen bond donor/acceptor counts.

Graph machines were built for each example of the whole set; the node functions f_θ were implemented by feedforward neural networks with a single hidden layer, and the complexity was selected by 6-fold cross-validation. The best results were obtained with 3 hidden neurons, overfitting appearing with more complex models, as shown on Table 1.

Table 1. Model selection for the prediction of the toxicity of phenols : RMS prediction error; figures between parentheses are the ratio of the RMS error to the variation range of $\log(1/IG_{50})$ in the database.

Model complexity	*6-fold cross-validation RMSE*
GM-3N	0.27 (6.7 %)
GM-4N	0.30 (7.5 %)
GM-5N	0.34 (8.5 %)

The results on the test set are shown on Figure 4 (GM with 3 hidden neurons) and Table 2, where they are compared to those of other methods. The quality of the fitting is estimated by the Root Mean Square Error (RMSE) and by the ratio of the RMSE to the variation range of the predicted quantities in the database. For comparison purposes, the RMS error when training on the whole data set was computed (last column), although its statistical significance is not clear.

The results show that graph machines provide the best results on the test set, without requiring the computation and the selection of descriptors, as opposed to alternative methods.

Table 2. Prediction of the toxicity of phenols by different methods. RMS prediction error; figures between parentheses are the ratio of the RMS error to the variation range of $\log(1/IG_{50})$ in the database.

Method [6, 7]	*Training set*	*Test set*	*Whole set*
GM 3N	0.19	**0.27**	0.21
MLR	0.30	0.46	0.33
RBFNN	0.19	0.29	0.21
SVM	0.22	0.36	0.24

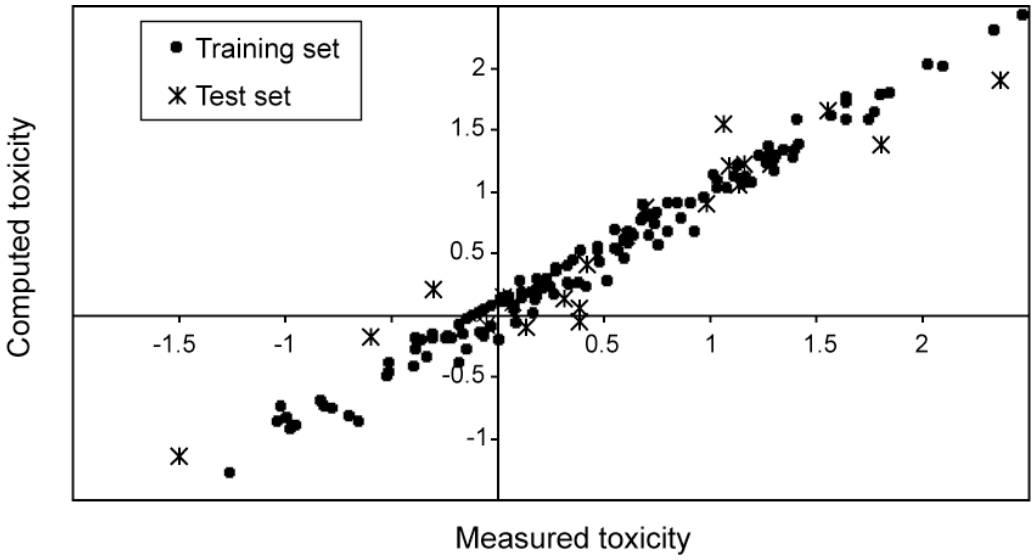

Figure 4

Scatter plot for the prediction of the toxicity of phenols by graph machines

3.3. Predicting anti-HIV activity (HEPT Data Set)

The HIV-1 virus is a retrovirus, i.e. a RNA virus that needs an enzyme, the reverse transcriptase (RT), to transcribe its single-stranded RNA genome into single-stranded DNA. A DNA double helix, able to integrate into host cell chromosomes, can then be formed. Therefore this enzyme is necessary for the virus to replicate, and inhibitors of reverse transcriptase are potential therapeutic agents in the battle against HIV. For example, the well-known anti-AIDS drug, AZT, targets this enzyme: it belongs to the class of nucleoside reverse transcriptase inhibitors (NRTI), which cause chain termination of the polymerase reaction. However, these compounds are often associated to undesirable side effects. Another class of molecules, non-nucleoside reverse transcriptase inhibitors (NNRTI), which include for example HEPT or TIBO derivatives, also inhibits the replication of the virus, but has low side effects.

The anti-HIV activity of HEPT (1-[2-Hydroxyethoxy)methyl]-6-(phenylthio)-thymines) analogues was studied with several QSAR methods, including 3D methods - e.g. Comparative Molecular Field/Surface Analysis (CoMFA/CoMSA) [13-16].

To compare the performances of graph machines to those of methods based on molecular surface calculation, we studied a data set of 107 HEPT derivatives whose activity was modelled previously by 4D-QSAR [16]. This method is based on molecular dynamics simulations, which investigate the conformational space of molecular objects, and find the likelihood of a formation of a common 3D pattern for the series of molecules. The anti-HIV activity is expressed as $\log(1/EC_{50})$, EC_{50} being the molar concentration of drug required to achieve 50% protection of MT-4 cells against the cytopathic effects of HIV-1 virus. The experimental inaccuracy in EC_{50} varies from 1.1 to 45%, which corresponds to an error in $\log(1/EC_{50})$ from 0.004 to 0.20. Unfortunately, no estimation of the uncertainty of each individual measurement is available.

The data set was divided twice into a training and a test set of respectively 80 and 27 compounds, identical to the sets studied with 4D-QSAR. The first test set gathers the molecules for which the only available information is an upper limit of their activities. Therefore the results obtained on this set are not very relevant, and the second partition was used to assess the generalisation abilities of the models.

Table 3 shows the results obtained on the two test sets by graph machines built with node functions featuring two, three and four hidden neurons. They are compared to the performances of 4D-QSAR using Partial Least Squares (PLS) and self-organising neural networks (SOM). Figure 5 shows the scatter plot of the results obtained on test set 2 by a graph machine with 3 hidden neurons.

These results show that graph machines outperform 4D-QSAR methods on the prediction of the anti-HIV activity of HEPT derivatives on both sets. Moreover, 4D-QSAR requires the optimisation of the geometry of each molecule, the sampling of one thousand conformations, and the computation of 4D descriptors, whereas Graph Machines only use the SMILES of each molecule.

To further assess the performances of GM, we compared their ability to predict the same activity to the results obtained with a method based on substructural molecular fragments (SMF) [13]: each molecular graph is split into fragments, whose contribution to the modelled activity is computed. The original data set features 84 molecules, and contains no compound for which only the upper limit of $\log(1/EC_{50})$ was known. The generalisation capabilities were tested on two different sets, named Test set 3 and Test set 4, of 8 examples.

Graph Machines with 2, 3 and 4 hidden neurons were built and trained on the two training sets. The error (RMSE) on the corresponding test sets are reported in Table 4, and compared to the errors obtained with SMF models.

Table 3. Prediction of anti-HIV activities by different methods. RMS prediction error; figures between parentheses are the ratio of the RMS error to the variation range of $\log(1/EC_{50})$ in the database.

Method [13]	*Test set 1*	*Test set 2*
PLS 4D-QSAR	1.41	0.79
SOM 4D-QSAR	1.39	0.67
GM 2N	1.17	0.62
GM 3N	1.15	**0.51**
GM 4N	**0.98**	0.54

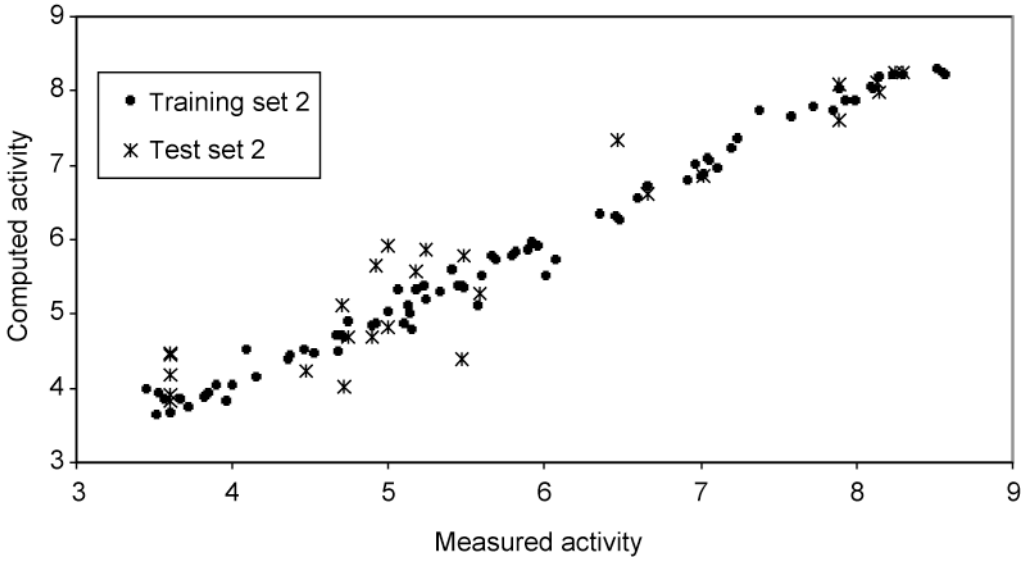

Figure 5
Scatter plot of the prediction of anti-HIV properties by graph machines

Table 4. Prediction of anti-HIV activities by different methods. RMS prediction error; figures between parentheses are the ratio of the RMS error to the variation range of $\log(1/EC_{50})$ in the database.

Method [10]	*Test set 3*	*Test set 4*
SMF	0.41	0.37
GM 2N	0.39	0.36
GM 3N	**0.27**	**0.31**
GM 4N	0.30	0.36

In both test sets, Graph Machines with three hidden neurons lead to the best prediction of the activity. In view of the small number of examples in the training set, the higher error obtained with four hidden neurons is a symptom of overfitting. These results are comparable to the errors obtained on Test set 2, which confirms the good performances of GMs on this QSAR task.

The prediction of the anti-HIV activity of HEPT derivatives highlights several specificities of graph machines. Firstly, as opposed to 4D-QSAR, but also to traditional methods such as MLR, PLS [14] or conventional neural networks [15], they do not require the computation of any descriptor, nor the optimisation of the molecular geometry. Furthermore, several molecules had to be removed from the set studied with the SMF method because of the presence of unique fragments, which was not necessary for graph machines. Finally, the models provided by graph machines were satisfactory, since the errors obtained on the test sets, even if higher than the experimental error, compared favourably with the errors obtained with other QSAR methods. As mentioned above, it would be crucially important to know reliable estimates of the experimental error in order to make a more accurate and relevant assessment of the accuracy of the predictions.

3.4. Predicting carcinogenicity

Other methods were developed to build QSPR/QSAR models without the use of descriptors, by exploiting the structures of the molecules themselves [17, 18]. These methods, based on the computation of graph kernels, consist in considering the molecules as labelled graphs, and computing a kernel function between them to measure their similarities. The graph is generally decomposed into substructures, and the count of these substructures forms a feature vector, from which the kernel function can be computed. In that context, the main issue is to choose the most efficient way to decompose the graphs. To that effect, several types of kernels can be defined. For example, 1D kernels are based on the SMILES representation of the molecules and the substructures are sequences of letters, whereas 2D kernels require the definition of labelled paths, i.e. successions of vertices and edges. Molecules can otherwise be described by the spatial coordinates of each atom. They are then assigned a set a histogram, one histogram per pair of atom labels representing the distances between all pairs of such atoms. Similarities between molecules are then measured via the similarities between their histograms, leading to the computation of the 3D kernel.

We compared graph machines to graph kernels on a classification task. The studied chemical compounds come from the Predictive Toxicology Challenge (PTC) dataset [19], which consists of several hundred molecules. Their carcinogenicity to Male Mice, Female Mice, Male Rats and Female Rats is known. The classification of compounds into carcinogenic and non-carcinogenic to female rats was performed.

Table 5 reports the results of 9-fold cross-validation of graph machines, which are compared to the best results described in [20], obtained by graph kernel classification methods.

Table 5. Classification accuracy for the carcinogenicity of chemical compounds of the PTC dataset on female rats.

Method [17]		*Accuracy (%)*
Graph Kernels	1D	67.0 (LOO)
	2D	66.9 (LOO)
Graph Machines (4 hidden neurons)		71.0 (9 f-CV)

4. Conclusions

The present paper describes QSAR/QSPR applications of a promising machine-learning technique for learning from structured data. It exempts the model designer from the design, computation and selection of descriptors, which are currently a major burden in QSAR applications. The method consists essentially in building a parameterized function that is intended (i) to encode the graph of the molecule, (ii) to provide a prediction of the quantity of interest. The parameters are estimated by training from examples, i.e. by minimizing a suitable cost function that expresses the discrepancy between the measurements and the corresponding predictions. Model selection is performed by usual methods such as hold-out, cross-validation or leave-one-out.

Several applications of graph machines to QSAR/QSPR are described. In the cases reported here, as well as in several others, graph machines outperform conventional methods, albeit sometimes slightly, with the additional advantage that they do not require molecular descriptors. However, if the graph structure is not sufficient for predicting the property, descriptors may be used as additional inputs to the graph machine; this was not deemed necessary for the problems described here. Nevertheless, the fact that the vast majority of databases do not indicate the accuracy of the reported measurements is a major problem for assessing the accuracy of the predictions.

Scalability issues must be further investigated in the future; the maximum data set size investigated thus far is 1200 molecules, without serious computation time problems. Experimental planning for graph-machine based models is also an open problem at present.

References

[1] J. B. Pollack. *Artificial Intell.*, **46**, 77, (1990).
[2] A. Goulon-Sigwalt-Abram, A. Duprat, G. Dreyfus. *Th. Comp. Sci.*, **344**, 298, (2005).
[3] B. Hammer. *Cognitive Systems Res.*, **3**, 145, (2002).
[4] P. Frasconi, M. Gori, A. Sperduti. *IEEE Trans. on Neural Networks*, **9**, 768, (1998).
[5] A. Goulon, A. Duprat, G. Dreyfus, in *Applied Statistical Modeling and Data Analysis*, Brest, 2005.
[6] C. Jochum, J. Gasteiger. *J. Chem. Inf. Comput. Sci.*, **17**, 113, (1977).
[7] A. Goulon-Sigwalt-Abram, A. Duprat, G. Dreyfus. *Lecture Notes in Computer Science (Springer)*, **4135**, 1, (2006).
[8] G. Monari, G. Dreyfus. *Neural Computation*, **14**, 1481, (2002).
[9] X. J. Yao, A. Panaye, J. P. Doucet, R. S. Zhang, H. F. Chen, M. C. Liu, Z. D. Hu, B. T. Fan. *J. Chem. Inf. Comput. Sci.*, **44**, 1257, (2004).
[10] M. T. D. Cronin, A. O. Aptula, J. C. Duffy, T. I. Netzeva, P. H. Rowe, I. V. Valkova, T. W. Schultz. *Chemosphere*, **49**, 1201–1221, (2002).
[11] A. O. Aptula, T. I. Netzeva, I. V. Valkova, M. T. D. Cronin, T. W. Schultz, R. Kühne, G. Schüürmann. *Quant. Struct.-Act. Relat.*, **21**, 12, (2002).
[12] S. Ren. *Chemosphere*, **53**, 1053, (2003).
[13] V. P. Solov'ev, A. Varnek. *J. Chem. Inf. Comput. Sci.*, **43**, 1703, (2003).
[14] J. M. Luco, F. H. Ferretti. *J. Chem. Inf. Comput. Sci.*, **37**, 392, (1997).
[15] M. Jalali-Heravi, F. Parastar. *J. Chem. Inf. Comput. Sci.*, **40**, 147, (2000).
[16] A. Bak, J. Polanski. *Bioorg. Med. Chem.*, **14**, 273–279, (2006).
[17] H. Kashima, K. Tsuda, A. Inokuchi. In *Twentieth International Conference on Machine Learning*, pp. 321-328, Washington DC (2003).
[18] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, J.-P. Vert. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 552-559, Banff, Alberta, Canada (2004).
[19] C. Helma, King,R.D., Kramer,S. and Srinivasan,A. . *Bioinformatics*, **17**, 107, (2001).
[20] S. J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, P. Baldi. *Bioinformatics*, **21**, i359, (2005).