TOWARD BIG DATA IN QSAR/QSPR

A. Duprat[1]*, J.L. Ploix*[1]*, F. Dioury*[2]*, G. Dreyfus*[1] *FIEEE*

[1]SIGnal processing and MAchine learning (SIGMA) lab, ESPCI ParisTech, Paris, France
[2]Equipe Chimie Moléculaire, Laboratoire CMGPCM, Cnam, Paris France

ABSTRACT

We investigate a prospective path to processing "big data" in the field of computer-aided drug design, motivated by the expected increase of the size of available databases. We argue that graph machines, which exempt the designer of a predictive model from handcrafting, selecting and computing *ad hoc* molecular descriptors, may open a way toward efficient model design procedures. We recall the principle of graph machines, which perform predictions directly from the molecular structure described as a graph, without resorting to descriptors. We discuss scalability issues in the present implementation of graph machines, and we describe an application to the prediction of an important thermodynamic property of contrast agents for MRI imaging.

Index Terms— QSAR/QSPR, graph machine, stability, chelate, scale

1. INTRODUCTION

QSAR/QSPR (Quantitative Structure – Activity/Property Relationships) are two major areas of application of Machine Learning in computer-aided drug design. The purpose is the prediction of the biological activity, or of the physico-chemical properties, of hitherto not synthesized molecules, in order to avoid costly chemical syntheses and tests of molecules that turn out not to have the desired activity/property, or to have undesirable side effects. The traditional approach consists in designing and measuring (or computing *ab initio*) selected descriptors, that are input to a machine learning tool (typically neural nets or Support Vector Machines). In order to avoid the computationally costly step of designing and selecting appropriate descriptors, an alternative method, termed Graph Machines or Graph Neural Networks, has been proposed: it predicts the activity/property of interest from the structure of the molecule, described as a graph. As a result, the whole design procedure can be automated, from the encoding of the molecular structure into a graph, to the estimation of the generalization error of the predictive model after training. This opens the way to very fast model design for processing thousands of molecules.

In section 2, we recall the principle of graph machine design, training and performance evaluation. Section 3 describes the scalable structure that is advocated, and section 4 describes the promising results of a real-life (although not yet that "big") application.

2. GRAPH MACHINES FOR QSAR/QSPR

In order to avoid costly and time consuming chemical syntheses of molecules that turn out not to have the desired activity/property, and/or to have undesirable side effects, the prediction of biological activities and of physico-chemical properties has become a major issue for speeding up the development of new drugs. The first QSAR/QSPR methods were developed in the 1970s, and machine learning provided a number of efficient computational tools that served efficiently the purposes of QSAR/QSPR. In the traditional approaches, descriptors of the molecules that were expected to be relevant for predicting the quantity of interest (e.g. the anti-HIV activity measured by a suitable index) or for performing the classification task (e.g. toxic *vs.* non-toxic), were handcrafted; they were subsequently computed *ab initio*, and used as variables of models that were trained from the available data. The usual steps of variable selection and model selection were carried out as usual.

In order to circumvent the problems related to descriptor design and selection, methods for performing prediction or classification directly from a graph representation of the molecular structure were developed ([1]-[4]). In the present paper, we focus on graph machines.

A graph machine is a composition of parameterized functions whose structure reflects the structure of the graph, so that the value taken by the function, after training, depends on the graph structure, and possibly on exogenous data. In the present application, each node of the graph is a non-H atom, and each edge is a bond between atoms. In order to take into account multiple bonds, the leaves of the graph contain the degree of each atom (i.e. the number of chemical bonds that bind it to the adjacent atoms); the leaves also contain a label that indicates the nature of the atom (i.e. carbon, oxygen, nitrogen…), in one-out-of-*N* code and possibly additional data such as stereochemical information.

978-1-4799-3694-6/14/$31.00

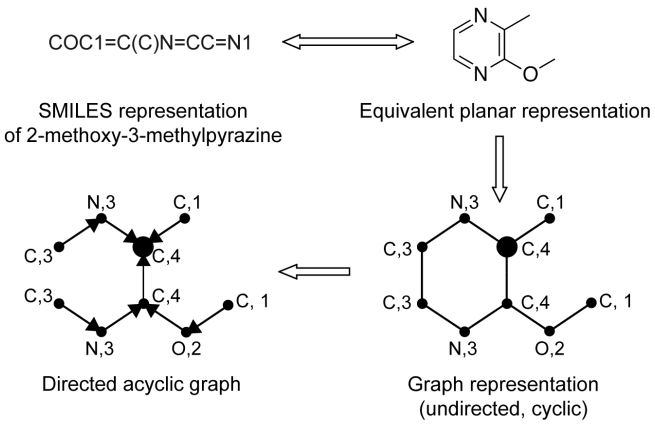

Figure 1

From a planar representation to a directed acyclic graph, with deletion of an edge and choice of a central node (large dot)

In order to handle acyclic graphs only, a minimum number of bonds are deleted if necessary, but the information about the existence of the deleted bonds is retained in the degrees of the adjacent nodes, present in the labels. In addition, a central node is selected [5].

Figure 1 illustrates the steps that create the structure of a graph machine. The starting point is the description of the molecule of interest in a standard text format for describing molecular structures called SMILES (Simplified Molecular Input Line Entry Specification). It provides the planar representation, which is turned into an undirected, cyclic or acyclic, graph with appropriately labeled nodes. After deletion of edges and selection of the central node, a directed acyclic graph is constructed, where all paths in the graph end at the central node.

The final step of the construction of the model consists in postulating a parameterized function (termed *node function*), and implementing it at each node of the graph. The output of a node function is one of the inputs of the functions of the nodes to which it is linked by an edge of the graph; since all paths of the graph terminate at the central node, the output of that node (also termed output node) is the output of the graph machine. Node functions may be polynomials, neural networks, radial basis functions, etc. All non-output node functions are identical within a graph machine and in all graph machines of the database; the output node function may be different from the other node functions, but it is the same for all graph machines of the database. Therefore, the number of parameters to be estimated during training is the number of parameters of the postulated node function and of the output node function. Figure 2 shows the final graph machine of the molecule of Figure 1; the triangles are the symbols of the node functions. The variables of each node function are (i) the labels (nature and degree of the atom of the node, and possibly additional information) and (ii) the outputs of the node functions that are connected to it. Because all node functions are identical, they must have the same number of variables, but all nodes do not have the same number of incoming edges; therefore, variables that are not used by a node are set to zero. After training, the output of the graph machine of a molecule provides a prediction of the value of the quantity of interest for that molecule, or, in the case of a classification problem, the label of the class of the molecule.

Training is performed by minimizing the sum of the squared prediction errors with respect to the parameters of the node functions. We denote by $\boldsymbol{\theta}$ the vector of parameters of the postulated node function, by Θ the vector of parameters of the postulated output node function, and by $g^i_{\boldsymbol{\theta},\Theta}$ the output of the graph machine pertaining to molecule i of the training set. We denote by y^i the measured value of the quantity of interest for molecule i. Then the cost function to be minimized with respect to the parameters is:

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

where λ_1 and λ_2 are regularization constants, and N is the size of the training set.

Optimization is performed by any suitable optimization method (Levenberg-Marquardt, BFGS, conjugate gradient, etc.). If the postulated node function is a neural network, the gradient of the cost function can be computed by backpropagation; the usual *weight sharing* method guarantees that all node functions are identical.

Complexity selection can be performed by standard hold-out, cross-validation, leave-one-out, or virtual leave-one-out (a powerful, computationally efficient nonlinear extension of the PRESS statistic [6], [7]).

Despite the similarity of the training and model selection methods of graph machines to those of standard machine learning regression methods such as neural networks or support vector regression, the concept is different in important respects. Standard regression methods use a single parameterized function, whose parameters are estimated from N input-output pairs. By contrast, the training set of graph machines comprises N different parameterized functions, with shared parameters that are estimated from N structure-output pairs. Therefore, the computational structures required for graph machines are different from those of standard nonlinear regression.

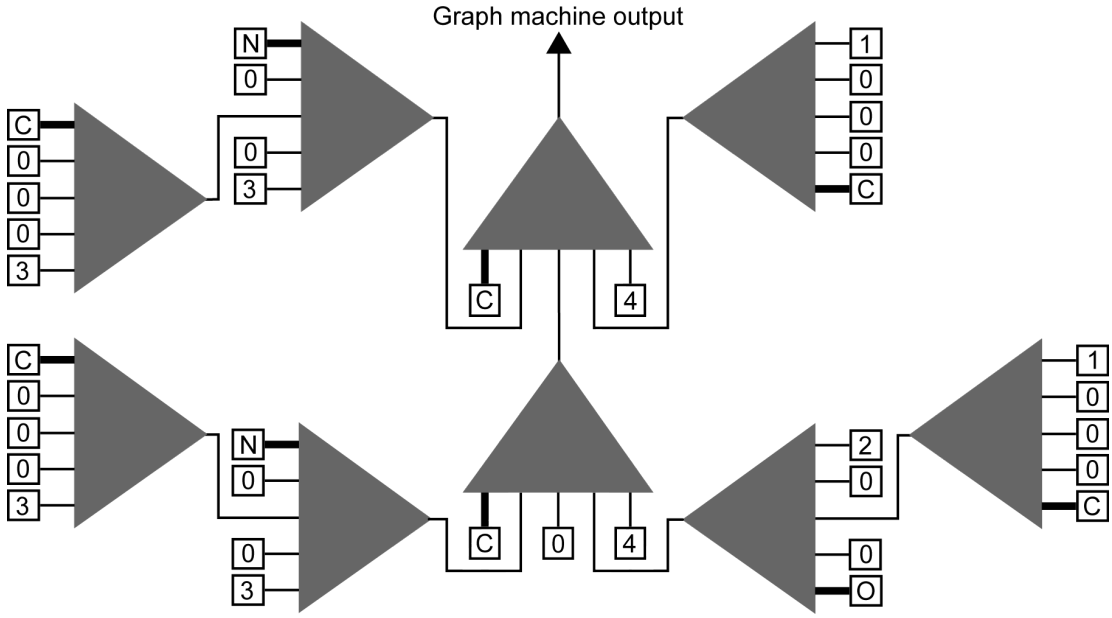

Figure 2

Graph machine based on the directed acyclic graph shown on Figure 1. The triangles depict the node function (e.g. a neural network). The nature of the non-H atoms (here C, N and O) is encoded in a 1-out-of-3 code; the thick line on one of the inputs expresses the fact that there are actually three (binary) inputs. The maximum number of incoming edges to a node of the acyclic graph of Figure 1 is three (at the central node), there are three different atoms, and one label is necessary for the degree, so that each node function has seven variables. No additional stereochemical descriptor is required for that molecule.

3. COMPUTATIONAL STRUCTURE FOR GRAPH MACHINES

One of the key issues in the implementation of graph machines is the complexity. Each molecule in the database is encoded into a composition of nonlinear functions whose numbers is equal to the number of non-H atoms of the molecule; the number of parameters depends on the number of labels and on the complexity of the node function, typically a few tens of parameters. Since the parameters are shared within each graph machine and across all graph machines, the number of parameters is much smaller than might be expected in view of the number of elementary functions (monomials for polynomial nodes, hidden neurons in the case of neural network node). As a result, the computational complexity is not, as is usually the case, in the optimization algorithm, but actually in the computation of the outputs of the functions and of the first order derivatives of these outputs with respect to the parameters. Therefore, the scalability of the graph machine technique is highly dependent on the technology used for internal computations.

Our choice is based on the "just in time" computation (JIT) and its implementation in the recent versions of the Python language. As mentioned above, a graph machine is a composition of node functions. For each of them, we create "on the fly" a set of optimized C-coded files describing the computation of the functions of interest for training (computation of the graph machine output and computation of its derivatives with respect to the parameters); these files are compiled, also "on the fly", into object files (Figure 3).

This results in a set of object files, one for each graph of the database. These files are linked together (with a negligible computational overhead) to create a binary library that can compute a vector of outputs and a jacobian matrix for a given set of parameters (Figure 4). The key factor in the reduction of computation time is the fact that the code generated from the model structure, and compiled “on the fly”, is fully dedicated to the model of interest, hence very simple. By contrast, conventional implementations use general-purpose codes, which are more flexible but more computationally demanding – a situation similar to custom hardware *vs*. general-purpose circuits. Our approach leads to a decrease of computation time by two to three orders of magnitude, as illustrated in the next section.

4. ILLUSTRATIONS

The above methods are illustrated by the prediction of two physico-chemical properties: the stability constant of chelates used as contrast agent in Magnetic Resonance Imaging, and the octanol-water partition coefficient of organic molecules.

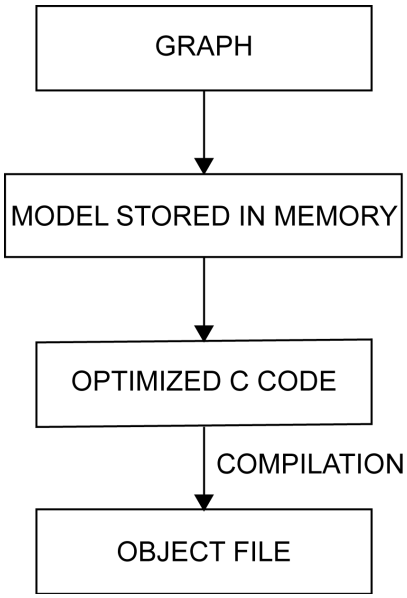

Figure 3
From graph to object file, "just in time" computation.

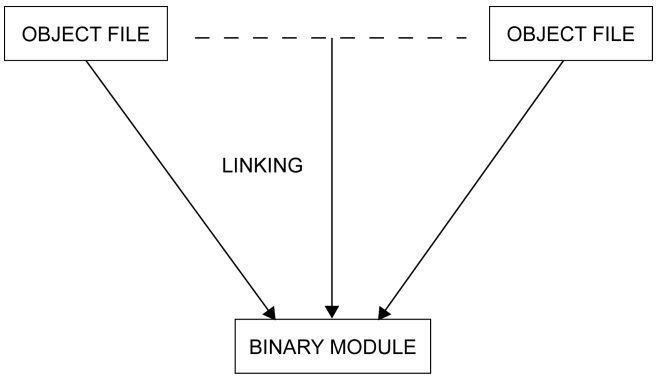

Figure 4
Linking object files to build a binary module.

4.1. Prediction of the stability constants of gadolinium chelates.

Magnetic Resonance Imaging (MRI) is a widely used, non-invasive technique in medical imaging and biomedical research. One of its main advantages over X-ray imaging is its applicability to the study of soft tissues, while X-ray are useful for examining bone conditions, as X-rays are strongly absorbed by bones. In order to improve the contrast between normal and diseased tissues in MRI images, it is customary to inject paramagnetic substances. Among them, gadolinium (Gd) in its ionic form Gd^{3+} is the most frequently used [8]. Unfortunately, free Gd^{3+} is toxic, so that it is necessary to sequester it into a molecular cage (the *ligand*) by forming strong bonds between the active cation and the ligand; the metal-ligand complex thus formed is a *chelate*, which must remain stable in the body and be excreted intact. Figure 5

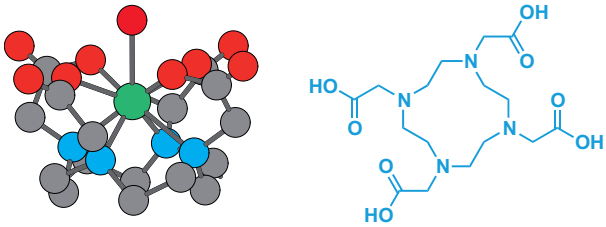

Figure 5
3-D (left) and planar (right) structures of a typical chelate; red spheres: oxygen atoms; blue: nitrogen atoms; grey: carbon atoms; green: chelated Gd^{3+} ion.

shows the 3-D structure of a chelate and the 2-D structure of its ligand; the non covalent bonds between the Gd^{3+} ion (green sphere) and the N and O atoms (blue and red respectively) of the ligand are clearly visible, together with the O atom of a water molecule (top red sphere).

The higher the thermodynamic stability of a metal complex used as a pharmaceutical drug, the lower its toxicity; the thermodynamic stability constant of a gadolinium(III) complex (K_{therm} or K_{GdL}) is useful to assess the amount of free Gd^{3+} or free ligand in a water solution. As the values of the stability constants span several orders of magnitude, they are almost always expressed in log K units in publications, databases and handbooks (the same holds true for the values of the octanol-water partition coefficient log$P$, discussed in section 4.2).

A lot of effort is devoted to finding new gadolinium-based contrast agents with improved performance. Therefore, it is crucial to develop new contrast agents with very high stability constants. The rationale for using machine learning to estimate the stability constant of Gd^{3+} chelates with yet non synthesized ligands stems from the fact that the experimental determination of these thermodynamic constants is long and tedious, so that the development of a computational predictive method is very likely to speed up the design of new, efficient ligands.

In an investigation of the prediction of stability constants of contrast agents for MRI, an exhaustive database of 158 Gd^{3+} chelates was built [9]. 109 of them were used as a training/validation set, 12 of them built up the test set, and 37 ligands were used as an application set of molecules with questionable or yet non-existent experimental values of log K_{GdL}. Figure 6 shows that the scatter plot of the virtual leave-one-out predictions on the training set (dots) and on the test set (squares), together with ±10% prediction RMS error lines. The prediction error being on the order of the experimental measurement error, the results are very satisfactory. A detailed discussion of the chemical significance of the result is provided in [9].

Graph machine predictions were performed both with a conventional implementation of graph machines, and with the implementation described in section 3, on the same

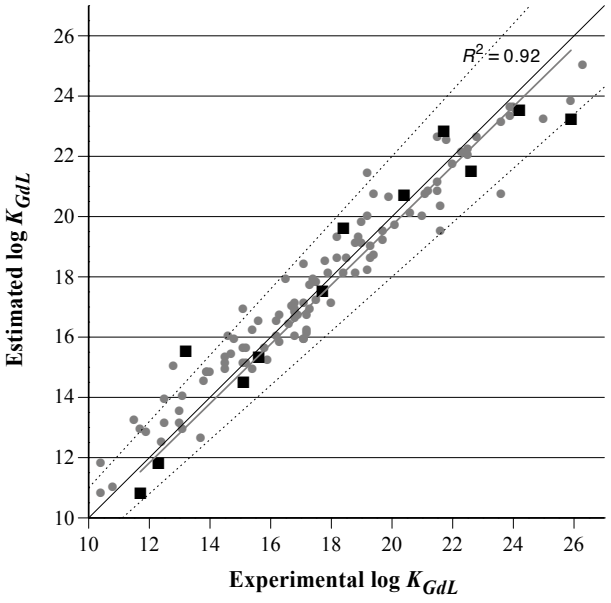

Figure 6

Virtual leave-one-out estimates of log K_{GdL} for the 109 compounds of the training/validation set (●), and estimates for the 12 compounds of the test set (■), *vs.* experimental values of log K_{GdL}. The coefficient of determination R^2 has the same value (within three digits) for both sets. The regression line for the VLOO estimates and the bisector are not distinguishable.

machines. The new implementation resulted in an average reduction of computation time by a factor of 300.

4.2. Prediction of the octanol-water partition coefficient of organic molecules.

The octanol-water partition coefficient of a molecule (log*P*) is the log of the ratio of its equilibrium concentration in octanol to its equilibrium concentration in water (water and octanol are immiscible). Therefore, it describes how hydrophilic (or hydrophobic) a molecule is: the higher its hydrophilicity, the lower its log*P* value; the higher its hydrophobicity, the higher its log*P* value. This is a key figure for pharmaceutical applications, because it is important in estimating how a drug is distributed in the body.

A drug that is administered orally must enter the blood circulation system by passing through the intestinal epithelium, which requires that the molecule is hydrophobic enough. In addition, hydrophobic effects are important for the binding of drugs to their targets. For all the above reasons, the prediction of log*P* is of major interest for computer-aided drug design, and traditional, descriptor-based machine learning methods have been applied extensively (for a review, see [10] and references therein).

A database of 1,800 molecules (from 2 to 52 non-H atoms per molecule) with known values of log*P* [11] was used for investigating the scaling of computation times with the size of the training set, for different numbers of parameters corresponding to neural network node functions, with three to seven hidden neurons. All experiments were performed on a quad-core i7-2600 @3.6 GHz. Figure 7 shows that the computation time necessary for the construction of the graph machines increases linearly with the size of the training set in the range investigated. Similarly, the computation time increases linearly with the number of parameters, for a given size of the training set. It must be noted that this computation is performed only once for a given molecule: if another property of the same molecule is to be predicted, this step is not repeated. In addition, if a multicore machine is used, each core may be assigned a separate set of molecules, as a graph machine is constructed independently from the others.

By contrast, the graph machines must be trained specifically for each property. However, the training time is much smaller than the time required for building the graph machines. Figure 8 shows the computation time for training the graph machines by 250 epochs of the Levenberg-Marquardt algorithm. Linear scaling is observed again.

For molecules of the size and complexity indicated above, the new implementation divides the construction time by 1.5 and the training time (node function with 5 hidden neurons, 250 epochs of Levenberg-Marquardt optimization) by 1,200.

5. CONCLUSION

The results presented here show that graph machines can be conveniently implemented in an efficient way, so as to be used on large databases without resorting to high-performance hardware or grid computing, given the present size of available databases. However, due to the current interest in capitalizing as much as possible on publicly available databases in chemistry will grow more and more rapidly. Therefore, it is important to find scalable implementations QSAR/QSPR methods, as illustrated in the present article.

Interestingly, there is room for a lot of improvement in computational efficiency, especially with the use of multicore machines for the construction of the graphs of molecules, which has become the most time-consuming step for graph machine prediction if a single property is to be predicted. If several properties of the same molecule are to be predicted, the construction of the graphs is performed only once, which is a particularly attractive feature of graph machines.

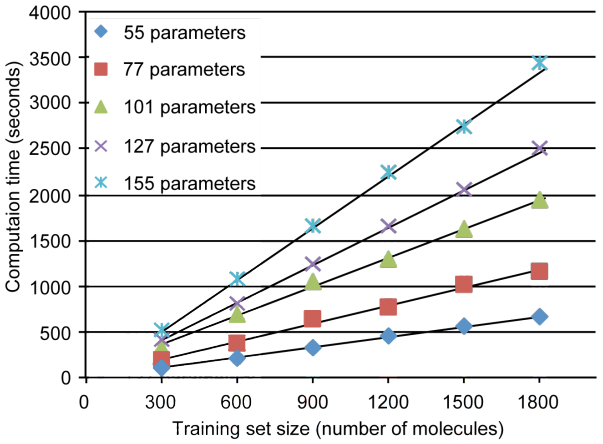

Figure 7
Computation time *vs.* number of molecules in the training set for the construction of graph machines of different node function complexities (number of parameters).

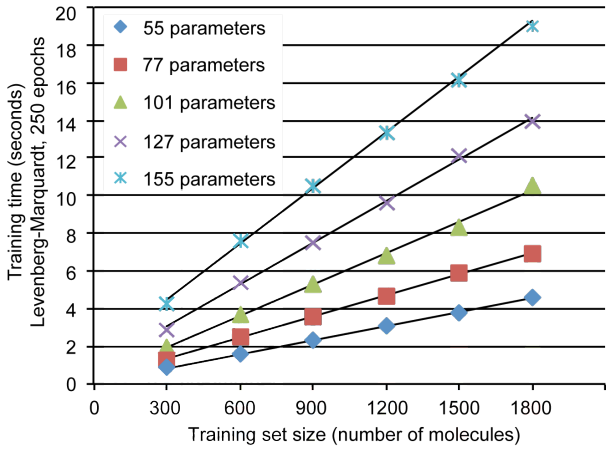

Figure 8
Training time (Levenberg-Marquardt optimization, 250 epochs) *vs.* number of molecules in the training set.

6. REFERENCES

[1] A. Micheli, F. Portera, A. Sperduti, "A preliminary experimental comparison of recursive neural networks and a tree kernel method for QSAR/QSPR regression tasks", Proc. ESANN 2004, pp. 293-298, 2004.

[2] P. Baldi, G. Pollastri, "The Principled Design of Large-Scale Recursive Neural Network Architectures–DAG-RNNs and the Protein Structure Prediction Problem", Journal of Machine Learning Research, pp. 576-602, 2003.

[3] A. Goulon, T. Picot, A. Duprat, G. Dreyfus, "Predicting activities without computing descriptors: graph machines for QSAR", *SAR QSAR Environ. Res.*, pp. 141-153 (2007).

[4] A. Goulon-Sigwalt-Abram, A. Duprat, G. Dreyfus, "From Hopfield nets to recursive networks to graph machines: numerical machine learning for structured data", *Theoretical Computer Science*, pp. 298 - 334, 2005.

[5] C. Jochum, J. Gasteiger, "Canonical Numbering and Constitutional Symmetry", *J. Chem. Inf. Comput. Sci.*, pp. 113-117, 1977.

[6] G. Monari, G. Dreyfus, "Local Overfitting Control via Leverages", *Neural Computation*, pp. 1481-1506, 2002

[7] A. Goulon-Sigwalt-Abram, A. Duprat, G. Dreyfus, *Unconventional Computation*, Springer, 2006.

[8] G.-P. Yan, L. Robinson, P. Hogg, "Magnetic resonance imaging contrast agents: Overview and perspectives", *Radiography*, pp. e5-e19, 2007.

[9] F. Dioury, A. Duprat, G. Dreyfus, C. Ferroud, J. Cossy, "QSPR Prediction of the Stability Constants of Gadolinium(III) Complexes for MRI", *J. Chem. Inf. Model.*, accepted for publication, doi: 10.1021/ci500346w, 2014.

[10] J. Taskinen, J. Yliruusi, "Prediction of physicochemical properties based on neural network modelling", *Advanced Drug Delivery Reviews,* pp. 1163 – 1183, 2003.

[11] C. Hansch, A. Leo, D. Hoekman, "Exploring QSAR, Hydrophobic, Electronic, and steric Constants; American Chemical Society: Washington, DC, 1995; Vol. 2.