Review on Lazy Learning Regressors and their Applications in QSAR

Abhijit J. Kulkarni[§], Valadi K. Jayaraman and Bhaskar D. Kulkarni[*]

Chemical Engineering and Process Development Division, National Chemical Laboratory, Pune, India

[§]*Current Address: Tata Research Development and Design Centre, 54-B, Hadapsar Industrial Estate, Pune- 411 013, India*

Abstract: Building accurate quantitative structure-activity relationships (QSAR) is important in drug design, environmental modeling, toxicology, and chemical property prediction. QSAR methods can be utilized to solve mainly two types of problems *viz.*, pattern recognition, (or classification) where output is discrete (*i.e.* class information), *e.g.*, active or non-active molecule, binding or non-binding molecule *etc.*, and function approximation, (*i.e.* regression) where the output is continuous (*e.g.*, actual activity prediction). The present review deals with the second type of problem (regression) with specific attention to one of the most effective machine learning procedures, *viz.* lazy learning. The methodologies of the algorithm along with the relevant technical information are discussed in detail. We also present three real life case studies to briefly outline the typical characteristics of the modeling formalism.

Keywords: Quantitative structure activity relationship (QSAR), machine learning, classification, regression, lazy learning.

1. INTRODUCTION

Building accurate quantitative structure-activity relationships (QSAR) is a challenging task and has attracted lot of interest in the recent past [1-5]. In general, QSAR is defined as a process by which chemical structure is quantitatively correlated with well-defined parameters such as biological activity or chemical reactivity [6-41]. Mathematically speaking, this is essentially a regression problem. Several algorithms have been designed to model QSAR, which are generally categorized as global and local learning algorithms. Global learning algorithms estimate the algorithm parameters in a global manner, *i.e.* during training of the algorithm once the parameters are estimated (based on the desired loss function), they do not change for future predictions, *e.g.*, multiple linear regression, regression trees, artificial neural networks, and support vector machines [42-46]. On the other hand, local learning algorithms estimate the algorithm parameters on a query basis. A notion of similarity (*e.g.*, Euclidean distance) with respect to historical database is used in estimation of the algorithm parameters. Therefore the algorithm parameters change from query to query, *e.g.*, instance based learning, lazy learning, and case based reasoning [47-52].

The present review deals with the adaptive memory based local learning paradigm, *viz.* lazy learning. This learning paradigm has many desirable properties compared to their global learning counterparts [47, 48, 50-52]. We present a detailed modeling methodology along with the mathematical details of the algorithm. We bring out its efficacy by presenting three QSAR modeling examples from real life. The article is arranged as follows. In the next section, we explain the details of the lazy learning (memory based local learning) algorithm along-with few important practical considerations. Section 3 deals with three important real world case studies for QSAR modeling with a few relevant important references. Section 4 concludes the article.

2. LAZY LEARNING

Linear and nonlinear regression techniques, developed over the years, are widely applied to obtain solutions for a number of different problems in various fields of science and engineering, *e.g.*, system identification, process monitoring, fault detection and diagnosis, nonlinear modeling, optimization and control, modeling quantitative structure- activity relationship (SAR) and structure-property relationship (SPR) etc [50-52]. The common goal here is to provide an accurate input/output mapping that is adequate for the purpose at hand.

Various regression methods can be broadly classified as parametric and nonparametric. The main disadvantage of the parametric models is that if the wrong functional form is chosen, the model remains highly biased in the prediction task [48]. The statistical complexity and computational cost hamper the performance of the nonparametric methods. Also, they require large amount of data for training. The whole procedure is divided in training, validation and assessment of the generalization of the model by applying it to unseen test data. This procedure is time consuming and can badly affect the performance in the cases where time is a limiting factor (*e.g.*, control related problems, virtual screening of molecules *etc*).

The input/output mapping $y = f(\mathbf{x})$ of a highly nonlinear process may exhibit reasonably smooth behavior over a certain domain of variables and may possess very uneven or peaky surfaces for certain other domain of variables. The global models try to obtain a relationship that approximates the actual mapping with least overall error over the entire domain of variables. It is possible that the model is very accurate in certain regions and not so much in other regions. In other words, the single global model may have different extents of error in different regions of the variables span. It is

*Address correspondence to this author at the Chemical Engineering and Process Development Division, National Chemical Laboratory, Pune, India; E-mail: bdk@ncl.res.in

1386-2073/09 $55.00+.00

more advantageous in such cases to build local models. The idea of using local techniques as an alternative to such methods originated in nonparametric statistics and then it was rediscovered and developed by the machine learning community [53].

There are three aspects of local learning: local representations, local selection, and locally weighted learning. In local representation, each new data point, also called as a query point, affects a small subset of the historical data and thus to answer a query involves only a small subset of the observations. Local selection methods then store all (or most) of the training data and use a distance function to determine the points relevant to a query point. Finally, locally weighted learning stores the training data explicitly and only fits parameters to the relevant subset of the training data when a query point is known. The seemingly global models, *e.g.*, rules, decision trees and parametric models, can be converted to their local equivalents by using a locally weighted training criterion. Nearest neighbor, weighted average and locally weighted regression are some of the local models in common use [47].

Lazy learning is a recently introduced memory based local learning method using local selection of parameters. It defers the computations till a request for prediction is received (*i.e.* query based), and it answers the query point by interpolating locally relevant examples according to a distance measure. Therefore, each prediction of a query point requires local modeling procedure. This local modeling procedure is composed of parametric and structural identification. Given a model structure, parametric identification involves optimization of the parameters of the local approximator. The structural identification, on the other hand, comprises the selection of a family of local approximators, a metric to evaluate relevant examples and *bandwidth* which indicates the size of the region in which the data are correctly modeled by the members of the local approximators chosen earlier. Since it is a memory-based technique, no separate training is required to answer the query points, which greatly improves the speed of implementation. Secondly, it predicts by locally interpolating the relevant points based on a distance measure. This is particularly useful when limited amount of input/output data is available and an accurate prediction is required. Lastly, it is less susceptible to the noise contamination [54]. All these features greatly improve the overall performance of the learning from input/output data as compared to other parametric as well as nonparametric methods. The advantages are of considerable practical relevance. First, we explain the basic algorithm along-with its mathematical details and then cite few important pertinent applications.

2.1. The Algorithm

The version of the lazy learning method employed in this work is a memory based local learning technique requiring storage of training data in the memory. This method is termed as lazy learning method because it delays processing of data until a query is required to be answered. Being a memory based technique, the unknown function is estimated by giving the whole attention to the region surrounding the point (query point) where the estimation is required. The relevant data is measured using a distance function. Lazy learning attempts to fit the training data only in a region around the location of the query point [55].

As it postpones the computations till the query is received for prediction, local modeling procedure is required for each query point. The local modeling procedure consists of parametric and structural identification. Given a model structure, parametric identification involves the optimization of the parameters of the local approximator. The structural identification, on the other hand, comprises the selection of a family of local appproximators, a metric to evaluate relevant examples and *bandwidth* which indicates the size of the region in which the data are correctly modeled by the members of the local approximators chosen earlier. After generating candidate models, the generalizing ability of each of the models needs to be assessed by mean squared error (MSE). It is customary to provide a reliable assessment of MSE employing a cross-validated approach. Normally, such a cross validation approach requires heavy computational requirements. For linear models, however, the PRESS (Prediction Sum of Squares) statistic procedure can rapidly estimate the leave-one-out cross validation estimate and at a very nominal computation load. This work takes advantage of this methodology for identifying the optimal local structure around every query point. The entire procedure is explained below [54]:

Consider $\mathbf{x} \in \Re^m$ and $y \in \Re$ for unknown input-output mapping $f : \Re^m \to \Re$ known through a set of n examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$. Let this mapping be represented as:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i \tag{1}$$

where $\forall i$, ε_i is a random variable such that $E[\varepsilon_i] = 0$ and $E[\varepsilon_i \varepsilon_j] = 0$, $\forall j \neq i$, and such that $E[\varepsilon_i^r] = \mu_r(\mathbf{x}_i), \forall r \geq 2$, where $\mu_r(\cdot)$ is the unknown r^{th} moment of distribution of ε_i and is defined as a function of $\mathbf{x}_i$.

Now given a query point $\mathbf{x}_q$, the parameter β of a local linear approximation of $f(\cdot)$ in a neighborhood of $\mathbf{x}_q$ is obtained by solving the local polynomial regression:

$$\sum_{i=1}^{n} \left\{ (y_i - \mathbf{x}_i' \beta)^2 K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h} \right) \right\} \tag{2}$$

where, given a metric on the space $\Re^m$, $d(\mathbf{x}_i, \mathbf{x}_q)$ is the distance from the query point to the i^{th} example, $K(.)$ is weight function, and h is the bandwidth. To consider a constant term in the regression, a constant value of 1 is appended to each input vector $\mathbf{x}_i$.

The solution of the above weighted least squares problem to find β is

$$\hat{\beta} = (X'W'WX)^{-1} X'W'W\mathbf{y} \tag{3}$$

where X is a matrix whose i^{th} row is $\mathbf{x}_i'$, $\mathbf{y}$ is a vector whose i^{th} element is y_i, W is a diagonal matrix whose i^{th} diagonal element is,

$$w_{ii} = \sqrt{K(d(\mathbf{x}_i, \mathbf{x}_q)/h)} \tag{4}$$

Replacing, $Z = X\,W$ and $v = W\,\mathbf{y}$ in equation (3),

$$\hat{\beta} = (Z'Z)^{-1} Z'v = P\,Z'v \tag{5}$$

It is assumed that matrix P is nonsingular so that its inverse is defined.

Once the local linear polynomial approximation is obtained, a prediction of $y_q = f(\mathbf{x}_q)$ is given by,

$$\hat{y}_q = \mathbf{x}_q'\,\hat{\beta} \tag{6}$$

The cross-validation procedure gives the assessment of the mean-squared-error as,

$$mse = E\left[(y_q - \hat{y}_q)^2\right] \tag{7}$$

Cross-validation requires a large computational effort to be performed due to the series of training steps. Instead, the PRESS statistic, which returns the leave-one-out cross validation error at the reduced computational effort, is used extensively in case of linear models. The PRESS statistic [56] is given by:

$$e_j^{CV} = y_j - \mathbf{x}_j'\,\hat{\beta}_{-j} \tag{8}$$

where, e_j^{CV} is the leave-one-out-error and $\hat{\beta}_{-j}$ is the estimated regression parameter with the j^{th} sample removed from the available set of examples.

The above equation can be simplified as:

$$e_j^{CV} = \frac{y_j - \mathbf{x}_j'\,PZ'v}{1 - \mathbf{z}_j'\,P\,\mathbf{z}_j} = \frac{y_j - \mathbf{x}_j'\,\hat{\beta}}{1 - h_{jj}} \tag{9}$$

where,

$$\mathbf{z}_j = w_{jj}\,\mathbf{x}_j \tag{10}$$

which is nothing but the j^{th} row of Z.

h_{jj} is the j^{th} diagonal element of the *Hat matrix*, H, given as

$$H = Z\,P\,Z' \tag{11}$$

From equation (9), it is evident that it is possible to calculate leave-one-out error without having to explicitly identifying $\hat{\beta}_{-j}$. This simplification greatly reduces the computations making the lazy learning regression algorithm very robust and competitive. Having provided the general procedure, we now proceed to explain how to recursively estimate the parameters, obtain the leave-one-out errors and generate the models.

2.1.1. Recursive Algorithm Associated with Leave-One-Out Estimation

To start with, consider the first-degree linear approximators. The algorithmic extension to generic polynomial approximators of any degree is simple. It is assumed that metric on the space $\Re^m$ is given. Naturally, the problem of bandwidth selection will be the rivet of attention. For the weight function K (.), the following indicator function is adopted [54]:

$$K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) = \begin{cases} 1 & \text{if } d(\mathbf{x}_i, \mathbf{x}_q) \le h \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

By doing this, the optimization of the parameter h, is reduced to the optimization of the k neighbors to which unit weight is assigned in the local regression, *i.e.* the problem of bandwidth selection is reduced to a search in the space of $h(k) = d(\mathbf{x}(k), \mathbf{x}_q)$, where $\mathbf{x}(k)$ is k^{th} nearest neighbor of the query point.

By adopting weight function as given in equation (12), it becomes easier and inexpensive to calculate $\hat{\beta}(k+1)$ by simply updating the parameter $\hat{\beta}(k)$ of the model identified with k nearest neighbors. By using standard steps of the recursive least squares algorithm, it is shown that [57]:

$$\left.\begin{aligned} P(k+1) &= P(k) - \frac{P(k)\,\mathbf{x}(k+1)\,\mathbf{x}'(k+1)\,P(k)}{1 + \mathbf{x}'(k+1)P(k)\,\mathbf{x}(k+1)} \\ \gamma(k+1) &= P(k+1)\,\mathbf{x}(k+1) \\ e(k+1) &= y(k+1) - \mathbf{x}'(k+1)\,\hat{\beta}(k) \\ \hat{\beta}(k+1) &= \hat{\beta}(k) + \gamma(k+1)\,e(k+1) \end{aligned}\right\} \tag{13}$$

where, $P(k) = (Z'Z)^{-1}$ when $h = h(k)$ and $\mathbf{x}(k+1)$ is the $(k+1)^{th}$ nearest neighbor of the query point.

With these recursive equations, local model candidates are generated. Once the matrix $P(k+1)$ is calculated from equation (13), the leave-one-out errors can be calculated directly without any further model identification as:

$$e_j^{cv}(k+1) = \frac{y_j - \mathbf{x}_j'\,\hat{\beta}(k+1)}{1 - \mathbf{x}_j'\,P(k+1)\,\mathbf{x}_j}, \quad \forall j : d(\mathbf{x}_j, \mathbf{x}_q) \le h(k+1) \tag{14}$$

This gives for each k, the $[k \times 1]$ vector $\boldsymbol{e}^{cv}(k)$ that contains all the leave-one-out errors associated with the model $\hat{\beta}(k)$.

2.1.2. Model Selection

Before starting the algorithm, the range (k_{min}, k_{max}) of nearest neighbors to be employed is selected *a priori*. With $k = k_{min}$, $P(k)$ and $\hat{\beta}(k)$ are calculated using equations (3-5). The recursive algorithm returns for a given query point a set of parameters, $\hat{\beta}(k+1)$ and a vector containing associated leave-one-out errors, $e_j^{cv}(k+1)$ respectively (equation (13)). After very estimation, a null hypothesis is formulated to check whether $e_j^{cv}(k)$ and $e_j^{cv}(k+1)$ belong to the same distribution. This hypothesis can be evaluated by a permutation test (For details, see [58]). This test facilitates determination of the maximum number of nearest neighbors to be

employed. The above procedure of estimating the parameters and the leave-one-out-errors can be done with a set of local models, *e.g*., local constant, local linear and local quadratic models. Now the question to be answered is how to make final prediction based on the local models generated? This is nothing but the model selection problem. There are mainly two approaches by which a final prediction is done: *winner-takes-all* (competitive) approach, local combination (cooperative) approach.

Winner-takes-all selects the best approximator based on some given criterion, *mean-squared-error* (MSE) being the most convenient and classical one. So, prediction obtained for each value of *k* is compared based on mean square error and the final prediction is done as:

$$\hat{y}_q = \mathbf{x}_q' \hat{\beta}\left(\hat{k}\right) \tag{15}$$

with,

$$\hat{k} = \arg\min_{k \in S} mse^{cv}(k) \tag{16}$$

where *S* is a range from which the optimal number of neighbors are selected.

In the other approach for model selection, a local combination of *b* best models is chosen based on the weighed average of MSE. Here b is a user-defined parameter. If the predictions $\hat{y}_q(k)$ and the error vectors $\mathbf{e}^{cv}(k)$ are ordered creating a sequence of integers $\{k_i\}$ such that $mse(k_i) \le mse(k_j),\ \forall i < j$. The final prediction is made as:

$$\hat{y}_q = \frac{\sum_{i=1}^{b} \zeta_i \hat{y}_q(k_i)}{\sum_{i=1}^{b} \zeta_i} \tag{17}$$

where, the weights are the inverse of the mean squared errors: $\zeta_i = \frac{1}{mse^{cv}(k_i)}$

Using equation (17), lazy learning combines a number of best models for optimal performance. The complete lazy learning algorithm is explained in a step-by-step algorithmic form below:

1. Input the data $\left\{\left(\mathbf{x}_i, y_i\right)\right\}_{i=1}^{n}$, where **x** represents the attributes of the data and y the corresponding outputs.
2. Input the query points $\mathbf{x}_q$ for which prediction is required.
3. Define the range of the *k*-nearest neighbors (k_{min}, k_{max}) to identify the local models like local constant, local linear, local quadratic or combination of these models.
4. For a query point $\mathbf{x}_q$, based on the nearest neighbor range and the indicator function given by equation (12), calculate the weights by using equation (4).
5. Use these weights to calculate P matrix in equation (5).
6. Use the recursive algorithm to calculate different $\hat{\beta}$ parameters for the given range of the nearest neighbors and associated errors using PRESS statistic.
7. To select the best number of nearest neighbors, form null hypothesis and evaluate it using a permutation test. If the hypothesis identifies $e_j^{cv}(k+1)$ different from the distribution $e_j^{cv}(k)$, stop including more neighbors and go to step 8 else go to step 6.
8. Use the model selection approach (*winner-takes-all* or local combination) based on mean-squared-error to select the best local approximator for the query point, $\mathbf{x}_q$.
9. Make the final prediction with equation (6) using the best approximator given by step (8).
10. Repeat steps 1 to 9 with different query points. Employ appropriate performance metric (mean-absolute-error, root-mean-squared- error etc.) to calculate the final prediction error.

Below we give the form of the local constant model which is later used in the simulations also [54].

$$\hat{y}_{0,q}(k) = \frac{k-1}{k}\hat{y}_{0,q}(k-1) + \frac{1}{k}y(k) \tag{18}$$

and the corresponding MSE is calculated as:

$$mse_0^{cv}(k) = \frac{k(k-2)^2}{(k-1)^3} mse_0^{cv}(k-1) + \frac{1}{k-1}(y(k) - \hat{y}_{0,q}(k-1))^2 \tag{19}$$

where, recursion on MSE is started for *k* = 2. Equation (18) and (19) computes the leave-one-out mean squared error, without explicitly computing each single cross-validation error.

This interesting modeling domain has found many applications in various fields of science and engineering.

We now move on to explain few important practical considerations of the algorithm.

2.2. Important Practical Aspects of the Algorithm

The algorithm explained above has many important aspects when it comes to solving real life problems. We explain them one by one.

2.2.1. Regularization and Parameter Selection

In data based modeling methods, regularization ability of the modeling method plays an important role [43]. In regression estimation problems, the learning bias is typically expressed as a smoothness criterion to optimize. But in case of memory based local learning, smoothness constraint is not explicit. Smoothing or bandwidth parameter can be manipulated to improve generalization and regularization capabilities. Apart from the form of equation used in this work, there exists many other ways to use the bandwidth parameter. These include fixed bandwidth selection, nearest neighbor bandwidth selection, global bandwidth selection, point-based bandwidth selection etc. A detailed discussion on bandwidth selection can be found in [47]. Ridge regression and *PCA*

dimensionality reduction have also been employed previously for the purpose of regularization [43].

2.2.2. Model Interpretation

Since the algorithm models the data locally by finding only the relevant subset from the training data, they can be interpreted easily. Due to similarity metric, relevant points can be analyzed in a more focused manner. The magnitude of the estimated parameters of the respective local models has the same significance as the global regression methods have [43].

2.2.3. Scaling Issues and Real Time Deployment

With new incoming data the size of the database grows. There are several ways to find relevant data so that lazy learning does not slow down while processing queries. A few methods have been described in literature for accelerating the search without discarding data. These include techniques like binning, use of k-d trees binary data structure etc. [47]. Once the relevant data is identified, lazy learning method can be automatically used without retraining or incremental learning.

If computer memory size is a constraint, there are methods wherein dataset which is stored can be forgotten selectively [47]. Few points are discarded based on some heuristics (either from knowledge of the process which is generating the data) or by some systematic quantitative way [47]. These methodologies are particularly important for the problems wherein concept drift is observed (*i.e.* underlying dynamics is non-stationary). Due to its adaptive nature, they can be deployed in real time applications as well.

2.2.4. Limits of Application

As the problem with all regression methods, this method will also not extrapolate beyond the prediction limits. However, due to its adaptive nature and local modeling mechanism, it can adjust to drift in dynamics quite fast as compared to other adaptive methods. Selection of effective similarity metric for the problem at hand is an important issue. There are several similarity metrics available in the literature, *e.g.*, Manhattan distance, Euclidean distance, Correlation information *etc*. The knowledge about the dataset may give some insight in selection of one of the similarity metrics which will explain the underlying mechanism appropriately. Otherwise, one may go for trial and error approach (sort of bruit force method) and select the one which best fits the data. This method is time consuming many times.

2.2.5. Software Availability

There are several versions of the local modeling software, which are available in public domain in variety of programming languages and platforms [59-61]. For the algorithm which is explained above, we particularly recommend a MATLAB library by Bontempi *et al.* [59]. It's a very fast, easy to use library and due to structured programming, one can very easily integrate the models in his own code.

In the next section, we apply this interesting methodology to solve three important real life case studies to model the quantitative structure activity relationship. In light of several practical concerns of the algorithm outlined above, we bring out its merits as compared to other modeling paradigms. We particularly focus on the comparison of these models with two contemporary methods which are quite popular among chemo-informaticians, namely multiple linear regression and support vector regression (a more recent method). Multiple linear regression is a well studied technique in the statistics literature. Hence, we do not present mathematical details of the algorithm here. Support vector regression is a relatively new, well-studied technique [42, 45]. For its algorithmic details, we refer many standard texts in this area [42, 45, 62-64].

3. CASE STUDIES

QSAR analysis is based on the assumption that there exists relationship between the biological activity within a group of molecular compounds with the variation of their respective structural and chemical features. So based on these physicochemical descriptors, analyst tries to predict the molecule's activity.

The benchmark data set under consideration is taken from Selwood *et al.* [65] where they developed a series of analogues of the mammalian electron transport inhibitor antimycin A_1 as potential anitfilarials in order to design new antifilarial drugs. The dataset is explained with 53 descriptors.

The second dataset is steroids data set. It mainly deals with a problem to model the corticosteroid binding globulin (CBG) receptor activity with autocorrelation of molecular surface properties. Wagener *et al.* [66] modified this particular dataset which resulted in 31 steroids with 1248 descriptors defining the corresponding activity.

And finally, the third dataset under consideration is Benzodiazepines (BZD) data available at http://www.iro.umontreal.ca/~lheureup/dataset.html. The original dataset comprises of 245 benzodiazepines compounds with 474 descriptors which act on benzodiazepine receptor. No common substructure is observed for this dataset [67, 68]. Primary analysis showed that two identical molecules have differing biological activity. After removal of these observations, 243 benzodiazepines compounds remain in the dataset for final analysis [69]. Benzodiazepines are mainly used in muscle relaxation, different convulsive disorders, insomnia, anxiety relief, sedation, different types of seizures, ethanol withdrawal etc [67, 69].

3.1. Preprocessing of the Datasets

It is known from literature that QSAR datasets are difficult to model due to fewer observations as compared to the number of descriptors (*i.e.* high dimensionality) [10]. The molecular descriptors are strongly correlated to each other [10, 70, 71]. Also, many times they are nonlinear [4, 5, 10, 70, 71]. To tackle these problems, effective preprocessing of the data is a major step while building the models using various algorithms. To deal with correlations and nonlinearity in the data, we preprocessed all the datasets using principal component analysis (traditional approach), PCA and local linear embedding (recently introduced interesting local technique for manifold learning), LLE.

PCA is a well studied technique in statistics and hence for the sake of brevity, we avoid mathematical details here. However, LLE is a relatively new algorithm and hence provide some mathematical details in Appendix A. For details,

readers may refer to few important articles [72-75]. These algorithms are used to reduce the dimensionality in the data effectively while retaining all the important information in the data. We compare the effect of preprocessing also with all regression algorithms. Prior to dimensionality reduction, all attributes were mean centered.

3.2. Results and Discussion

We followed two approaches in the analysis of the datasets. In the first approach, we provide the dataset as it is to all the learning algorithms and in the second approach we provide the reduced dimensionality dataset obtained by applying PCA and LLE. In the second approach, we selected those many transformed dimensions which retain 99% of the variance in the dataset. By this way, we compare the effect of preprocessing algorithms as well.

Multiple linear regression (MLR) and support vector regression (SVR) being global methods require separate training. So all the datasets were partitioned as training set (randomly selected 80% of the total size of the data) and test set (remaining 20% of the data). To tune the algorithm parameters, we used 5-fold cross validation error statistics on training data. Since lazy learning requires no separate training, randomly selected 80% of the total data forms the historical database and remaining 20% are used as query points. We ensured that these splits (80% training and 20% test) remain same for all the three algorithms in all simulations. To avoid the sample bias during learning, we repeated above procedure 20 times and took the average value for the error statistics. In case of SVR modeling, we took commonly used Gaussian Radial Basis Functions (RBF) as a kernel and selected its width based on cross validation error statistics [62] (Refer Table **6**). In case of Lazy learning, we report results with local constant, local linear and local quadratic models. We report usual QSAR statistics (R & S) for all the datasets and all the algorithms. The formulae for R and S statistics are as follows:

$$R = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sqrt{\left(\sum x^2 - \frac{\left(\sum x\right)^2}{n}\right)\left(\sum y^2 - \frac{\left(\sum y\right)^2}{n}\right)}} \quad (20)$$

$$S = \sqrt{\left(1-R^2\right)\frac{\left(\sum y^2 - \frac{\sum y^2}{n}\right)}{\left(n-k-1\right)}} \quad (21)$$

where, x is observed value, y is predicted value, n is number of data points and k represents number of variables in the dataset.

The results are reported in Tables **1-6**. Brackets below R values in result tables indicate the dimensionality of the data

Table 1. Test Set Results for QSAR Datasets Using Multiple Linear Regression

Dataset	QSAR Statistics					
	R			S		
	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA
Selwood	0.77 (N × 53)	0.76 (N × 9)	0.76 (N × 9)	0.61 (N × 53)	0.60 (N × 9)	0.60 (N × 9)
Steroids	0.839 (N × 1248)	0.821 (N × 6)	0.8013 (N × 28)	1.17 (N × 1248)	1.19 (N × 6)	2.25 (N × 28)
BZD	0.55 (N × 474)	0.75 (N × 32)	0.63 (N × 70)	0.6 (N × 474)	0.7 (N × 32)	0.65 (N × 70)

Table 2. Test Set Results for QSAR Datasets Using Lazy Learning (with Local Constant Model)

Dataset	QSAR Statistics					
	R			S		
	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA
Selwood	0.78 (N × 53)	0.79 (N × 9)	0.74 (N × 9)	0.64 (N × 53)	0.5925 (N × 9)	0.69 (N × 9)
Steroids	0.8385 (N × 1248)	0.8621 (N × 6)	0.7925 (N × 28)	1.1718 (N × 1248)	1.216 (N × 6)	2.315 (N × 28)
BZD	0.68 (N × 474)	0.78 (N × 32)	0.74 (N × 70)	0.56 (N × 474)	0.6 (N × 32)	0.62 (N × 70)

Table 3. Test Set Results for QSAR Datasets Using Lazy Learning with Local Linear Model

Dataset	QSAR Statistics					
	R			S		
	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA
Selwood	0.81 (N × 53)	0.86 (N × 9)	0.84 (N × 9)	0.56 (N × 53)	0.47 (N × 9)	0.52 (N × 9)
Steroids	0.85 (N × 1248)	0.88 (N × 6)	0.84 (N × 28)	1.05 (N × 1248)	0.98 (N × 6)	1.06 (N × 28)
BZD	0.71 (N × 474)	0.83 (N × 32)	0.77 (N × 70)	0.4 (N × 474)	0.30 (N × 32)	0.34 (N × 70)

Table 4. Test Set Results for QSAR Datasets Using Lazy Learning with Local Quadratic Model of Polynomial Degree 2

Dataset	QSAR Statistics					
	R			S		
	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA
Selwood	0.82 (N × 53)	0.87 (N × 9)	0.85 (N × 9)	0.543 (N × 53)	0.46 (N × 9)	0.53 (N × 9)
Steroids	0.85 (N × 1248)	0.88 (N × 6)	0.84 (N × 28)	1.05 (N × 1248)	0.98 (N × 6)	1.06 (N × 28)
BZD	0.72 (N × 474)	0.84 (N × 32)	0.78 (N × 70)	0.39 (N × 474)	0.30 (N × 32)	0.36 (N × 70)

Table 5. Test Set Results for QSAR Datasets Using Support Vector Regression

Dataset	QSAR Statistics					
	R			S		
	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA	Without Dimension Reduction	Dimension Reduction with LLE	Dimension Reduction with PCA
Selwood	0.80 (N × 53)	0.82 (N × 9)	0.80 (N × 9)	0.56 (N × 53)	0.543 (N × 9)	0.56 (N × 9)
Steroids	0.848 (N × 1248)	0.883 (N × 6)	0.81 (N × 28)	1.067 (N × 1248)	1.19 (N × 6)	1.287 (N × 28)
BZD	0.65 (N × 474)	0.8 (N × 32)	0.74 (N × 70)	0.55 (N × 474)	0.56 (N × 32)	0.6 (N × 70)

used in the simulation. As can be seen from the Tables **1-5**, lazy learning method (with Euclidean distance as similarity metric) in general performed remarkably well on all the datasets as compared to MLR and SVR. We cite a common observation that all the algorithms are benefited from dimensionality reduction. LLE is found to be quite helpful in effectively reducing the dimensionality of the data. Other local models like local linear and local quadratic models also performed favorably as compared to MLR and SVR. Next, we give Y-randomization test results for lazy learning models.

Table 6. Optimal SVM Parameters

Dataset	ε*	C⁺	σ#
Selwood	0.001	1.0	2
Steroids	0.001	2.5	1.5
BZD	0.001	10	2.5

* Width of the regression tube, + Regularization parameter, # Gaussian Kernel width parameter.

3.2.1. Y-Randomization Test

We did Y-randomization test for all local models for all the datasets with their original dimensionality. In all the models, we observed that R value decreases when we randomized the outputs. Thus, it is ensured that lazy learning does not model wrong outputs correctly. Results are summarized in Table **7**. Next we compare the regressors using statistical significance test.

Table 7. Test Set Y-Randomization Results for Lazy Learning (Data without Dimension Reduction is Used)

Dataset	Local Constant		Local Linear		Local Quadratic	
	R	R_{rand} [+]	R	R_{rand}	R	R_{rand}
Selwood	0.78	0.45	0.81	0.40	0.82	0.60
Steroids	0.8385	0.38	0.85	0.60	0.85	0.60
BZD	0.68	0.50	0.71	0.45	0.72	0.45

[+] R_{rand} is R value after randomization of the output.

3.2.2. Residual Analysis and Significance Testing

To compare the regressors, we did residual analysis and significance testing using *t-test*. It is observed that in all the three case studies, lazy learning and support vector regression performed statistically significantly different at 95% confidence limit compared to multiple linear regression. However, among lazy learning and SVR, this difference was not significant. We also observed in all the case studies that with dimensionality reduction, all the algorithms performed significantly different as compared to their counterparts in which all the dimensions in the dataset were used which justified the use of dimensionality reduction.

Accurate QSAR modeling is quite important. Sometimes analysis time becomes a constraint due to involved economic conditions. The lazy learning algorithm performs favorably compared to its counterparts (MLR and SVR) as it is quite fast (no separate training), robust (ability to deal with nonlinearity in the data) and easy to scale (due to its inherent adaptive nature) it up to tackle bigger problems. MLR and SVR are global methods. MLR fails in many situations where nonlinearity is a typical characteristic of the data (though it gives a very good first hand judgment about the data). SVR is a promising modeling methodology [76, 77]. But it requires a lot of careful tuning of the algorithm parameters (to avoid over-fitting) and effective optimization solution strategy (as formulation is based on quadratic programming optimization) to scale it up in the real time applications. We observe that support vector machines are not exploited fully for regression problems as compared to its classification counterpart in chemoinformatics.

Finally, we cite a recent QSAR modeling application in which authors used locally weighted regression (*i.e.* lazy learning) in an automated way [70]. They used the technique on several experimental QSAR datasets with competitive results and extended the methodology for virtual screening of chemical databases also [70].

4. CONCLUSIONS

We presented a detailed review on lazy learning (*a.k.a.* memory based local learning) for solving regression problems encountered in the analysis of quantitative structure activity relationships datasets. Three real world case studies justify its simplicity and robustness. It has an advantage over all other methods that it requires no separate training to find the algorithm parameters. It is innately adaptive in nature. Also, with minor modifications in effective mechanism of storing the dataset, it can be easily deployed in real time and can be scaled to tackle problems of bigger size. All these interesting characteristics make them amenable for application in solving problems from the area of combinatorial chemistry and high throughput screening.

APPENDIX A

Locally Linear Embedding Algorithm

The LLE algorithm is based on simple geometrical considerations [72]. Let the data set $\{\vec{X}\}_{i=1,2.....N} \in \Re^D$ be sampled from some smooth underlying manifold. Provided the manifold is well sampled (*i.e.* there is enough data), we expect each data point and its neighbors lie on or close to a locally linear patch of the manifold. Thus we can approximate the non-linear manifold in the vicinity of X_i by the linear hyperplane passing through its nearest neighbors.

Based on this simple idea, the LLE algorithm consists of a three step process:

1. Identify K nearest neighbors for every data point, as measured by Euclidean distance (Other notions of "closeness" are also possible, such as all points within a certain radius, or by using more sophisticated rules based on local metrics).

2. Compute the weights W_{ij} that best linearly reconstruct each X_i from its K neighbors. Here we are required to minimize the reconstruction error as measured by the cost function

$$\varepsilon(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2 \quad (1)$$

subject to two constraints:

a) Each data point $\vec{X}_i$ is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if $\vec{X}_i$ does not belong to this set and

b) $\sum_j W_{ij} = 1$ for every i.

The weights W_{ij} signify the contribution of the j^{th} data point to the i^{th} reconstruction. The optimal weights W_{ij} subject to these constraints are found by solving a least squares problem.

The constrained weights that minimize these reconstruction errors characterize intrinsic geometric properties of each neighborhood, as opposed to

properties that depend on a particular frame of reference. This is due to the fact that for any particular data point, the weights are invariant to rotations, rescalings, and translations of that data point and its neighbors. The invariance to rotations and rescalings results from the form of equation (1); the invariance to translations is imposed by the sum-to-one constraint (b).

Since the data lie on or near a smooth nonlinear manifold of dimensionality $d<<D$, there exists a linear mapping— comprising a translation, rotation, and rescaling—that maps the high dimensional coordinates of each neighborhood to global internal coordinates on the manifold. Thus reconstruction weights W_{ij}, invariant to such transformations, should characterize the local geometry to same extent, both in the original data space and local patches on the manifold. In particular, the same weights W_{ij} that reconstruct the i[th] data point in D dimensions should also reconstruct its embedded manifold coordinates in d dimensions. This forms the basis of third step of algorithm.

3. Find the d-dimensional vectors $\vec{Y}_i$ that best match those reconstruction weights W. Here we are required to minimize the reconstruction errors as measured by embedding cost function:

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2 \quad (2)$$

To ensure the uniqueness of the solution the following two constraints are imposed: translation invariance by requiring the coordinates to be centered on the origin, *i.e.* $\sum_i \vec{Y}_i = 0$ and we constrain the embedding vectors to have unit covariance,

$$\frac{1}{N} \sum_i \vec{Y}_i . \vec{Y}_i^{\mathrm{T}} = \mathrm{I}$$

where I is the $d \times d$ identity matrix.

These constraints do not affect the generality of the solutions as $\Phi(Y)$ is invariant to translation, rotations and homogeneous rescalings. The additional constraint that the covariance is equal to the identity matrix expresses an assumption that reconstruction errors for different coordinates in the embedding space should be measured on the same scale.

The optimal embedding $\vec{Y}_{i=1,2.....N} \in R^d$ is given by eigenvectors associated with the smallest d +1 eigen values of the matrix M [73]:

$$\mathrm{M} = (\mathrm{I} - W)^{\mathrm{T}} (\mathrm{I} - W) \quad (3)$$

The bottom eigenvector of this matrix is discarded, as it is a vector composed of all ones, with zero as eigenvalue. Discarding this eigenvector enforces the constraint that the embeddings have zero mean, as the components of other eigenvectors must sum to zero, by virtue of orthogonality. The remaining d eigenvectors of size N give the final embedding Y.

Although the reconstruction weights for each data point are computed from its local neighborhood independently, the embedding coordinates are computed by an $N \times N$ eigen solver, a global operation that couples all data points in connected components of the graph defined by the weight matrix. The different dimensions in the embedding space can be computed successively; this is done simply by computing the bottom eigenvectors from equation (2) one at a time.

In situations where data is not available in vector form $\vec{X}_i$, but only as the measurements of dissimilarity or pair wise distance between different data points, a simple variation of LLE can be applied. The nearest neighbors are identified by the smallest non-zero elements of each row in the distance matrix. The reconstruction weights calculation for each data point requires computing the local covariance matrix C_{jk} between its nearest neighbors. This is done by exploiting the usual relation between pair wise distances and dot products that form the basis of metric MDS [74]. Therefore, for a particular data point

$$C_{jk} = \frac{1}{2}(D_j + D_k - D_{jk} - D_0) \quad (4)$$

where D_{jk} is the squared distance between the j[th] and i[th] neighbors,

$$D_\ell = \sum_z D_{\ell z} \text{ and } D_0 = \sum_{jk} D_{jk}$$

The rest of the algorithm proceeds as usual.

The procedure as described above leads to nonlinear dimensionality reduction of data.

B.1. Parameters

In order to get a good LLE mapping, two parameters viz. the dimensionality, d, and the number of neighbors, K, need attention. If d is set too high, the mapping will enhance noise (due to the constraint $1/\mathrm{nYY}^{\mathrm{T}} = \mathrm{I}$); if it is set too low, distinct parts of the data set might be mapped on top of each other. If K is set too small, the mapping will not reflect any global properties; if it is too high, the mapping will lose its nonlinear character and behave like traditional PCA, as the entire data set is seen as local neighborhood. In the present work dimensionality d is chosen according to the criteria: d + 1 should be less than or equal to the number of eigen values of M that are close to zero [73, 74].

The nearest neighbor parameter K is a measure of the "quality" of input-output mapping (*i.e.* how well the high-dimensional structure is represented in the embedded space) and is selected based on the residual variance [75]. It is defined as $1 - \rho^2_{D_x D_y}$ where ρ is the standard linear correlation coefficient, computed over all entries of D_x and D_y; D_x and D_y are the matrices of Euclidean distances (between pairs of points) in X and Y (Y was computed in Step 3 above), respectively. The lower the residual variance is, the better the high-

dimensional data are represented in the embedded space. Hence, the optimal value for K, K_{opt} can be determined as

$$K_{opt} = \arg\min_{K}(1-\rho^2_{D_x D_y}) . \quad (5)$$

Although, few techniques are given in the literature such as linear interpolations and training a neural network or RBF network for mapping a new (previously unseen) sample, we have preferred a simple strategy of concatenating the new sample with given samples and repeating the whole LLE procedure. This preference is based on our observation that the LLE algorithm takes only few seconds of time to run, retaining non-linear mapping even for query point. Whereas, the approaches like Neural or RBF networks are hard to train, and linear interpolations will lose the non- linearity of data.

ACKNOWLEDGEMENTS

Abhijit acknowledges Girish Palshikar, Prof Harrick Vin for their constant encouragement and management support. Dr. VKJ wishes to acknowledge in-house project grant (MLP-010126) from National Chemical Laboratory, Pune, India.

REFERENCES

[1] http://joelib.sourceforge.net/wiki/index.php/Cheminformatics_and_mining_quotation
[2] http://wikipedia.org/
[3] Brown, F.K. *Ann. Rep. Med. Chem.*, James, A. Ed. Bristol, **1998**, *33*, 375-384.
[4] Gasteiger, J.; Engel, T. *Chemoinformatics: A Textbook;* John Wiley and Sons, **2004**.
[5] Leach, A.R.; Gillet, V.J. *An Introduction to Chemoinformatics;* Springer, **2003**.
[6] Apostolakis, J; Caflisch, A. *Comb. Chem. High Throughput Screen.*, **1999**, *2*, 91-104.
[7] Kirckpatrick, D.L.; Watson, S.; Ulhaq, S. *Comb. Chem. High Throughput Screen.*, **1999**, *2*, 211-221.
[8] Xue, L.; Bajorath, J. *Comb. Chem. High Throughput Screen.*, **2000**, *3*, 363-372.
[9] Yan, A.; Gasteiger, J. *QSAR Comb. Sci.*, **2003**, *22*, 821-829.
[10] Selassie, C.D. In *Burger's Medicinal Chemistry and Drug Discovery*; Donald J. Abraham, Ed.; John Wiley and Sons, Inc., **2003**, Vol. *1*, pp. 1-48.
[11] Ivanciuc, O. *Internet Electron. J. Mol. Des.*, **2004**, *3*, 426-442, http://biochempress.com/.
[12] Nussinov, R.; Wolfson, H. *Comb. Chem. High Throughput Screen.*, **1999**, *2*, 249-259.
[13] Nussinov, R.; Wolfson, H. *Comb. Chem. High Throughput Screen.*, **1999**, *2*, 261-269.
[14] Xue, L.; Godden, J.W.; Gao, H.; Bajorath, J. *J. Chem. Inf. Comput. Sci.*, **1999**, *39*, 669-704.
[15] Xue, L; Bajorath, J. *J. Chem. Inf. Comput. Sci.*, **2000**, *40*, 801-809.
[16] McGregor, M.J.; Muskal, S.M. *J. Chem. Inf. Comput. Sci.*, **1999**, *39*, 569-574.
[17] McGregor, M.J.; Muskal, S.M. *J. Chem. Inf. Comput. Sci.*, **2000**, *40*, 117-125.
[18] Ivanciuc, T.; Ivanciuc, O. *Internet Electron. J. Mol. Des.*, **2003**, *2*, 403-412, http://biochempress.com/.
[19] Ivanciuc, O. In *QSPR/QSAR Studies by Molecular Descriptors;* Diudea. M.V. Ed.; Nova Science: Huntington, N.Y., **2001**, 233-280.
[20] Hansch, C., Fujita, T. *J. Am. Chem. Soc.*, **1964**, *86*, 1616-1626.
[21] Leo, A.; Hansch, C.; Church. C. *J. Med. Chem.*, **1969**, *12*, 155-156.
[22] Klopman, G. *J. Am. Chem. Soc.*, **1984**, *106*, 7315-7320.
[23] Rogers, D.; Hopfinger, A.J. *J. Chem. Inf. Comput. Sci.*, **1994**, *34*, 854-866.
[24] Brown, R. D.; Martin, Y. C. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 572-584.
[25] Brown, R.D.; Martin, Y.C. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 1-9.
[26] Smith, D.A.; van de Waterbeemd, H. *Curr. Opin. Chem. Biol.*, **1999**, *3*, 373-378.
[27] Prentis, R.A.; Lis, Y.; Walker, S.R. *Br. J. Clin. Pharmacol.*, **1986**, *21*, 437-443.
[28] Prentis, R.A.; Lis, Y.; Walker, S.R. *Br. J. Clin. Pharmacol.*, **1988**, *25*, 387-396.
[29] Delie, F.; Rubas, W.A. *Crit. Rev. Ther. Drug Carrier Syst.*, **1997**, *14*, 221-286.
[30] Eddershaw, P.J.; Dickens, M. *Pharm. Sci. Technol. Today*, **1999**, *2*, 13-19.
[31] Lipinski, C.A.; Lombardo, F.; Dominy, B.W.; Feeney, P.J. *Adv. Drug Deliv. Rev.*, **1997**, *23*, 3-25.
[32] Sadowski, J.; Kubinyi, H. *J. Med. Chem.*, **1998**, *41*, 3325-3329.
[33] Bender, A.; Glen, R.C. *Org. Biomol. Chem.*, **2004**, *2*, 3204-3218.
[34] Vracko, M.; Gasteiger, J. *Internet Electr. J. Mol. Des.*, **2002**, *1*, 527-544, http://biochempress.com/.
[35] Gasteiger, J. *Mini Rev. Med. Chem.*, **2003**,*3*, 789-796.
[36] Teckentrup, A.; Briem, H.; Gasteiger, J. *J. Chem. Inf. Comput. Sci.*, **2004**, *44*, 626-634.
[37] Gasteiger, J. *Comput. Appl. Chem.*, **2005**, *22*, 827-831.
[38] Gasteiger, J. *Anal. Bioanal. Chem.*, **2006**, *384*, 57-64.
[39] Gasteiger, J. *Chemom. Intell. Lab. Syst.*, **2006**, *82*, 200-209.
[40] Gasteiger, J. *J. Med. Chem.*, **2006**, *49*, 6429-6434.
[41] Spycher, S.; Escher, B.I.; Gasteiger, J. *Chem. Res. Toxicol.*, **2005**, *18*, 1858-1867.
[42] Vapnik, V. *Statistical Learning Theory*; Wiley: New York, **1998**.
[43] Hastie, T.; Tibshirani, R.; Friedman, J.H. *The Elements of Statistical Learning*; Springer- Verlag, **2001**.
[44] Bishop, C.M. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, **1996**.
[45] Haykins, S. *Neural Networks: A Comprehensive Foundation*; 1'st Ed, Prentice Hall PTR: Upper Saddle River, NJ, USA, **1994**.
[46] Chapelle, O.; Scholkopf, B.; Zien, A. *Semi-Supervised Learning*; MIT Press: Cambridge, MA, **2006**.
[47] Atkeson, C.G.; Moore, A.W.; Schaal, S. *A.I. Rev.*, **1997**, *11* (1-5), 11-73.
[48] Constans, P.; Hirst, J.D. *J. Chem. Inf. Comput. Sci.*, **2000**, *40*, 452-459.
[49] Duarte, B.P.M.; Saraiva, P.M. *Ind. Eng. Che. Res.*, **2003**, *42*, 99-107.
[50] Kumar, R.; Kulkarni, A.; Jayaraman, V. K.; Kulkarni, B. D. *Internet Electron. J. Mol. Des.*, **2004**, *3*, 118-133, http://biochempress.com/.
[51] Kulkarni, A; Patil, S. V. ; Jayaraman, V. K. ; Kulkarni, B. D. *Int. J. Chem. Reactor Eng.*, **2003**,*1*, A23.
[52] Rajshekhar; Kulkarni, A; Jayaraman, V.K.; Kulkarni, B.D. *Internet Electron. J. Mol. Des.*, **2005**, *4*, 381-392, http://biochempress.com/.
[53] Cleveland, W.S.; Devlin, S.J.; Grosse, E. *J. Econ.*, **1988**, *37*, 87-114.
[54] Bontempi, G.; Birattari, M.; Bersini, H. *A. I. Commun.*, **2000**, *13(1)*, 41-48.
[55] Bontempi, G.; Birattari, M.; Bersini, H. In *Proc. IJCNN' 98;* (Anchorage, Alaska), **1997**, 2102-2106.
[56] Myers, R.H. *Classical and Modern Regression with Applications*; Boston, MA: PWS-KENT Publishing Company, **1994**.
[57] Birattari, M.; Bontempi, G.; Bersini, H. In *Adv. in NIPS*; Kearns, M.S.; Sola, S.A.; Cohn, D.A. Eds.; MIT Press, **1999**.
[58] Bontempi, G.; Birattari, M.; Bersini, H. *Int. J. Control*, **1999**, *72* (7-8), 643-658.
[59] http://iridia.ulb.ac.be/~lazy/
[60] http://cran.r-project.org/
[61] http://www.cs.waikato.ac.nz/ml/weka
[62] Burges, C.J.C. *Data Min. Knowl. Discov.*, **1998**, *2*(2), 121-167.
[63] Cristianini, N.; Shawe-Taylor, J. *An introduction to Support Vector Machines*; Cambridge University Press: Cambridge, UK, **2000**.
[64] Gunn, S. *ISIS Tech. Rep.*, **1997**.
[65] Selwood, D. L.; Livingstone, D. J. ; Comley, J. C. W.; O'Dowd, A. B.; Hudson, A. T.; Jackson, P.; Jandu, K. S.; Rose, V. S.; Stables, J. N. *J. Med. Chem.* **1990**, *33*, 136-142.
[66] Wagener, M.; Sadowski, J.; Gasteiger, J. *J. Am. Chem. Soc.*, **1995**, *117*, 7769-7775.
[67] Harrison, P.W.; Barlin, G.B. ; Davies, L.P.; Ireland, S.J.; Matyus, P.; Wong, M.G. *Eur. J. Med. Chem.*, **1996,** *31*, 651-662.
[68] Burden, F.R.; Ford, M.G.; Whitley, D.C.; Winkler, D.A. *J. Chem. Inf. Comput. Sci.,*, **2000**, *40*, 1423-1430.
[69] L'Heureux, P.J.; Carreau, J.; Bengio, Y.; Delalleau, O.; Yue, S.Y. *J. Comput. Aided Mol. Des.*, **2004**, *18*, 475-482.

[70] Zhang, S.; Golbraikh, A.; Oloff, S.; Kohn, H.; Tropsha, A. *J. Chem. Inf. Model.*, **2006**, *46* (5), 1984 -1995.
[71] Armengol, E.; Plaza, E. In *Artificial Intelligence Methods and Tools for Systems Biology*; Springer Netherlands, **2004**, Vol. 5, pp. 1-18.
[72] Roweis, S.T.; Saul, L.K. *Science*, **2000**, *290*, 2323-2326.
[73] Horn, R. A.; Johnson, C. R. *Matrix Analysis*; Cambridge University Press: Cambridge, **1990**.
[74] Bai, Z.; Demmel, J.; Dongarra, J.; Ruhe, A.; Van Der Vorst, H. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*; Society for Industrial and Applied Mathematics: Philadelphia, **2000**.
[75] Kouropteva, O.; Okun, O.; Pietikainen, M. *Proc. First Inter. Conf. Fuzzy Sys. Know. Discov.;* Singapore, **2002**, 359-363.
[76] Ivanciuc, O. *Internet Electron. J. Mol. Des.*, **2002**, *1*, 157-172, http://biochempress.com/.
[77] Ivanciuc, O. *Internet Electron. J. Mol. Des.*, **2002**, *1*, 203-218, http://biochempress.com/.

Received: April 27, 2008 **Revised: June 30, 2008** **Accepted: August 29, 2008**