1
Current Modeling Methods Used in QSAR/QSPR

Liew Chin Yee and Yap Chun Wei

1.1
Introduction

A drug company has to ensure the quality, safety, and efficacy of a marketed drug by subjecting the drug to a variety of tests [1]. Therefore, drug development is a time-consuming and expensive process. From the initial stage of target discovery, development often takes an average of 12 years [2] and was estimated to cost USD868 million per marketed drug [3]. This high cost and lengthy process is due to the high risk of drug development failure. It was estimated that only 11% of the drugs that completed developmental stage were approved by the US or European regulators [4]. In year 2000, it was found that 10% of attrition during drug development was contributed by poor pharmacokinetic and bioavailability, while in the clinical stage, 30% of attrition was due to lack of efficacy and another 30% was caused by toxicity or clinical safety issues [4]. Thus, it will be useful to predict these failures prior to the clinical stage in order to reduce drug development costs. It was claimed that savings of USD100 million in development costs per drug could be attained with 10% prediction improvement [5]. Therefore, various methods, such as *in vitro, in vivo*, or *in silico* methods, are being used early in the drug development stage to filter out potential failures. An example of an *in silico* method is quantitative structure–activity relationship (QSAR) models, which can be used to understand drug action, design new compounds, and screen chemical libraries [6–9]. Recently, the European Chemicals Legislation, Registration, Evaluation and Authorisation of Chemicals (REACH) suggested the use of *in silico* methods as reliable toxicological risk assessment [10, 11].

QSARs, or quantitative structure–property relationships (QSPRs), are mathematical models that attempt to relate the structure-derived features of a compound to its biological or physicochemical activity. Similarly, quantitative structure–toxicity relationship (QSTR) or quantitative structure–pharmacokinetic relationship (QSPkR) is used when the modeling applies on toxicological or pharmacokinetic systems. QSAR (also QSPR, QSTR, and QSPkR) works on the assumption that structurally similar compounds have similar activities. Therefore, these methods have predictive and

Statistical Modelling of Molecular Descriptors in QSAR/QSPR. First Edition.
Edited by M. Dehmer, K. Varmuza, and D. Bonchev

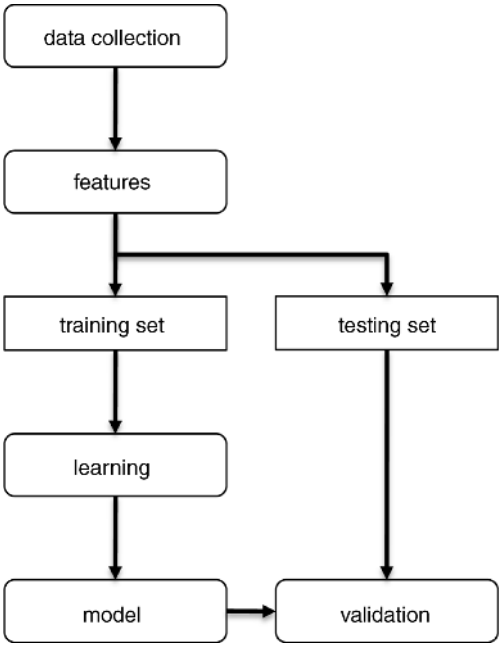

Figure 1.1 General workflow of developing a QSAR model.

diagnostic abilities. They can be used to predict the biological activity (e.g., IC_{50}) or class (e.g., inhibitor versus noninhibitors) of compounds before the actual biological testing. They can also be used in the analysis of structural characteristics that can give rise to the properties of interest.

As illustrated in Figure 1.1, developing QSAR models starts with the collection of data for the property of interest while taking into consideration the quality of the data. It is necessary to exclude low-quality data as they will lower the quality of the model. Following that, representation of the collected molecules is done through the use of features, namely molecular descriptors, which describes important information of the molecules. There are many types of molecular descriptors but not all will be useful for a particular modeling task. Thus, uninformative or redundant molecular descriptors should be removed before the modeling process. Subsequently, for tuning and validation of the QSAR model, the full data set is divided into a training set and a testing set prior to learning.

During the learning process, various modeling methods like multiple linear regression, logistic regression, and machine learning methods are used to build models that describe the empirical relationship between the structure and property of interest. The optimal model is obtained by searching for the optimal modeling parameters and feature subset simultaneously. This finalized model built from the optimal parameters will then undergo validation with a testing set to ensure that the model is appropriate and useful.

This chapter gives an introduction to the algorithm of the various modeling methods that have been commonly used in constructing QSAR models. We have used most of these methods in developing QSAR models for various pharmacodynamic, pharmacokinetic, and toxicological properties [12–16]. Even though our research have found that models developed using more complex modeling methods like the newer machine learning methods frequently outperform those developed using traditional statistical methods, it is essential to have a good foundation of all these methods. This is because no method is useful for all QSAR problems and the principle of parsimony states that we should use the simplest method that provides the desired performance level. This is to prevent overfitting of the data, which can lead to a loss in generalizability. Data collection, data processing, computation and selection of features, and model validation have been thoroughly reviewed elsewhere [17–22], so they are not described here. Software that is available for QSARs development will be discussed.

1.2 Modeling Methods

In general, methods for constructing QSAR can be divided into two groups: methods for regression problems or classification problems. The methods are organized into the two groups in the following section.

1.2.1 Methods for Regression Problems

1.2.1.1 Multiple Linear Regression

Multiple linear regression (MLR) is one of the most fundamental and common modeling method for regression QSAR. Recent application of MLR in QSAR or QSPR includes prediction for luteinizing hormone-releasing hormone antagonists [23], 5-HT6 receptor ligands [24], interleukin-1 receptor-associated kinase 4 inhibitors [25], potencies of endocrine disruptors [26], and chlorine demand by organic molecules [27]. MLR is favored for its simplicity and ease of interpretation as the model assumes a linear relationship between the compound's property, $\hat{y}$, and its feature vector, denoted $\mathbf{X}$, which is usually the molecular descriptors. Thus, with the notion of X, the property of an unknown compound can be predicted by the fitted model. The following equation represents a general expression of a MLR model:

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

where β_0 is the model constant, $X_1, \ldots, X_k$ are molecular descriptors with their corresponding coefficients $\beta_1, \ldots, \beta_k$ (for molecular descriptors 1 through k). These coefficients can be obtained through the use of estimators like least-squares method which minimizes the sum of squared residuals.

The size of the coefficients may reveal the degree of influence of the corresponding molecular descriptors on the target property. In addition, a positive coefficient suggests that the corresponding molecular descriptor contributes positively to the

target property, while a negative coefficient suggests negative contribution. However, these interpretations may not be accurate as collinear descriptors have the potential to influence the coefficients such that erroneous values may be assigned. Thus, the molecular descriptors in the model should be independent of each other and the number of instances for model building should be at least five times the number of descriptors used [28]. In addition, the assumption of a linear relationship makes MLR less suitable to model complex problems like toxicity, where multiple mechanisms may interplay to elicit a toxic response. Nonetheless, MLR has been used for modeling toxicity systems [29–33].

To date, MLR remains in use with enhancements or in combination with feature selection to improve its performance. Examples of enhancements are: the use of independent component analysis – MLR in QPSR of aqueous solubility [34], local lazy regression [35], retro-regression applied on boiling points of nonanes [36], ensemble feature selection [37], and other feature selection methods like genetic algorithm, ridge regression, partial least-squares method, pair-correlation method, forward selection, and best subset selection in the application of MLR [38–42].

1.2.1.2 **Partial Least Squares**

Partial least squares (PLSs) assumes a linear relationship between feature vector, $\mathbf{X}$, and target property, $\hat{y}$, but unlike MLR, PLS is more appropriate when the number of features greatly exceed the number of samples and when features are highly collinear [43]. It is to note that advancement has brought about methods like quadratic-PLS and kernel-PLS for nonlinear systems, multiway-PLS, unfolding-PLS, hierarchical-PLS, three-block bifocal PLS, and so on. These will not be discussed here and interested readers can refer to the review by Hasegawa *et al*. [44].

PLS works on the assumption that the examined system is subjected to the influence of just a few causal factors, termed *latent factors* or *latent variables* [43, 45]. PLS avoids the problem of collinear features by extracting these latent factors that can explain the variations of the molecular descriptors while simultaneously models the response of the target property.

PLS was also interpreted as the initialism for "Projection to Latent Structure" [45]. As illustrated in Figure 1.2, the latent factors can be estimated through X-scores and Y-scores, which are extracted from the molecular descriptors and desired compound properties, respectively. Subsequently, the X-scores are used to predict the Y-scores, which in turn can be used to predict the compound properties. The number of latent factors used in PLS is an important consideration for QSAR modeling, and it is usually obtained through the use of cross-validation methods like *n*-fold cross-validation and leave-one-out methods, where a portion of the samples is used as training set, while the other portion is set aside as testing set to validate the model that was built from the training set.

PLS has been applied on various QSAR studies like toxicity of quaternary ammonium compounds on *Chlorella vulgaris* [46], angiotensin II AT1 receptor antagonists [47], CYP11B2 binding affinity and CYP11B2/CYP11B1 selectivity [48], toxicity to *Daphnia magna* [49], and nonpeptide HIV-1 protease inhibitors prediction [50]. PLS is also used as an analysis method in the popular 3D-QSAR technique,

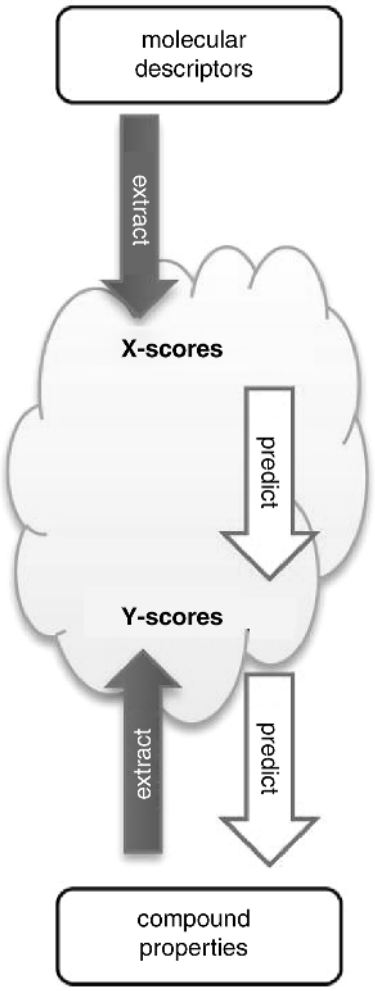

Figure 1.2 Extraction of latent factors from molecular descriptors and compound properties.

Comparative Molecular Field Analysis (CoMFA) that is available in the SYBYL software [51]. In CoMFA, the molecular descriptors are obtained from the magnitude of the steric and electrostatic field of molecules, which are sampled at regular interval when the molecules are aligned to a common substructure. As a result, a large number and correlated descriptors may be produced from a small training sample. Hence, PLS is applied to reduce the number of descriptors to make them more suitable for further analysis.

1.2.1.3 **Feedforward Backpropagation Neural Network**

Artificial neural network (ANN) attempts to imitate a biological neural network and is inspired from the structure, processing, and learning method of a biological brain.

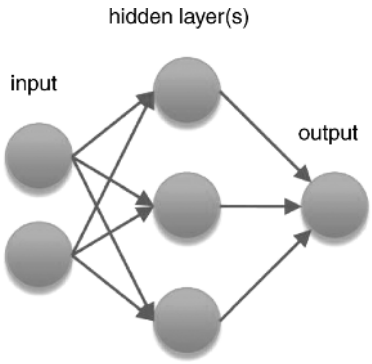

Figure 1.3 A simple structure showing the three layers of an artificial neural network.

It is a network of processing elements (akin to neurons) with weighted connections between them. A typical artificial neural network consists of three or more layers: an input layer, hidden layer(s), and an output layer as shown in Figure 1.3. In training, ANN adapts the weight of the connections until it approximates the input–output relationship of the training data. For model building, the number of hidden layers and the number of elements in the hidden layers is commonly optimized [52], although one hidden layer with large number of elements is generally sufficient to approximate most functions [53]. Nonetheless, it is not trivial to find an optimal topology that can generalize the data well; many rounds of training are usually required, which makes building an ANN model a time-consuming process.

A feedforward neural network was the first and possibly the simplest type of ANN. The input layer of the network represents molecular descriptors and the output layer represents the target properties of compounds. The network is feedforward because the elements in one layer are only connected to the elements in the next layer and the information moves forward from one layer to another toward the output layer. In the case of fully connected neural network, each element in the hidden layer receives information from all elements in the previous layer. Subsequently, an activation function, which is commonly linear or sigmoidal, will transform the input before forwarding the information to the next layer (if any) and eventually to the output layer.

The learning of a feedforward neural network can be done through a variety of techniques and one of the most popular methods is through backpropagating errors [54]. In backpropagation, the output of the network (from forward phase) is compared with the actual compound property to calculate a predefined error-function. This calculated value is then feedback (backward phase) into the network, allowing the algorithm to re-adjust the weights of the connections, which were randomly assigned initially, with the aim to minimize the error. The approximation improves as the errors converge after numerous training cycles. However, the model may run into the problem of overfitting and thus incapable of predicting the property of unknown compounds that are sufficiently different from the training set. Therefore, there are various methods such as the use of a validation set or techniques like early stopping, pruning, and weight decay to minimize the risk of overfitting [55–58].

Examples of application of feedforward backpropagation neural network in QSAR studies are toxic effect on fathead minnows [59, 60], calcium channel antagonist activity [61], alpha adrenoreceptors agonists [62], air to water partitioning for organic pesticides [63], aldose reductase inhibitors [64], antinociceptive activity [65], and boiling points [66]. Neural network is used in QSAR because of its ability to approximate any target function and also it can handle redundant descriptors well, as their weights can be learned and reduced to insignificant levels [67]. However, it is not easy to optimize the best network topology. Furthermore, parameters like learning rate and momentum needs to be defined by the user, thus the lack of automation makes the process rather time consuming. In addition, if the network is not optimized, undersized network may not approximate the relationship between the descriptors and target property well. Conversely if the network is oversized, overfitting may occur. Other disadvantages of neural network includes its susceptibility to noisy data which can be overcome with the use of a validation set during training, and also hard to interpret connection weights which makes optimization of compound structures difficult for medicinal chemists. Nonetheless, it is to note that a few studies have attempted to interpret neural network for QSAR studies with success [68–70], indicating that neural network is still a useful tool for QSAR studies.

1.2.1.4 General Regression Neural Network

One of the difficulty of building a neural network is the lack of automation in the selection of parameters and network topology. Coupled with many iterations that may be needed by the backpropagation method to converge to an acceptable error, model building is usually a time-consuming process [71]. To overcome these disadvantages, Specht introduced a one-pass neural network learning algorithm as an alternative to increase the training speed. With the implementation of the one-pass algorithm, user-defined parameters have been reduced to a minimal, of which optimization of the network mainly involves adjusting the sigma, σ, of the estimation kernel [71].

The one-pass algorithm was first implemented in probabilistic neural network for use in classification problems. Following that, the general regression neural network (GRNN) was introduced for estimation of regression problems [72]. It is noted that GRNN was rediscovered by Schløler [73, 74] a year later, and it is related to the kernel regression invented by Nadaraya [75] and Watson [76] independently [71]. GRNN was used in QSAR studies of CCR2 inhibitors [77], HIV-1 reverse transcriptase inhibitors [78], drug total clearance [79], estrogenic activity [80], phytoestrogen binding to estrogen receptors [81], aqueous solubility of nitrogen- and oxygen-containing small organic molecules [82], and QSAR on effect of substitution on the phenyl ring of orally active β-lactam inhibitor [83].

For target property, y, let X represents a value of molecular descriptor x, and $f(x,y)$ represents the joint probability density function (PDF) of x and y. The prediction of the target property can then be obtained by the conditional expected value of y given by X [71, 72, 82]:

$$E[y|\mathrm{X}] = \frac{\int_{-\infty}^{-\infty} y f(\mathrm{X}, y)\mathrm{d}y}{\int_{-\infty}^{-\infty} f(\mathrm{X}, y)\mathrm{d}y}$$

The joint probability density function, $f(x,y)$, symbolizes the relationship between the target property and molecular descriptors, and it is usually not known [72]. Therefore, the value is commonly estimated from the training data using techniques like Parzen's nonparametric estimator [84]:

$$g(x) = \frac{1}{n\sigma}\sum_{i=1}^{n} W\left(\frac{x-x_i}{\sigma}\right)$$

where n is the set cardinality, σ is a smoothing parameter that defines the kernel width, W is a weight function, and $(x - x_i)$ is the distance between a given instance and an instance in the training data. Cacoullos has expanded Parzen's nonparametric estimator for the multivariate case [85] and becomes

$$g(x_1, \ldots, x_p) = \frac{1}{n\sigma_1 \ldots \sigma_p}\sum_{i=1}^{n} W\left(\frac{x_1-x_{1,i}}{\sigma_1}, \ldots, \frac{x_p-x_{p,i}}{\sigma_p}\right)$$

For the weight function, W, a commonly used function is the Gaussian kernel. Updating the equation for PDF gives the following:

$$g(x) = \frac{1}{n}\sum_{i=1}^{n} \exp\left(-\sum_{j=1}^{p}\left(\frac{x_j-x_{j,i}}{\sigma_j}\right)^2\right)$$

There are two types of models defined by the number of sigma used: single-sigma models and multisigma models where an individual σ for each molecular descriptor is used. Multisigma models are suitable for cases which the descriptors are of different nature and importance. In general, these models are able to perform considerably better than the corresponding single-sigma models [82]. In single-sigma model, a single σ value is used to simplify the equation. Single-sigma model is preferred for cases where the descriptors have similar importance because reasonable models can still be obtained with faster computation speed [86].

$$\hat{y} = \frac{\sum_{i=1}^{n} y_i \exp(-D(\mathrm{x}, \mathrm{x}_i))}{\sum_{i=1}^{n} \exp(-D(\mathrm{x}, \mathrm{x}_i))}$$

The above is the basic equation of GRNN obtained from substituting Parzen's nonparametric estimator for $f(x,y)$, where the distance function, $D(x,x_i)$, can be the squared weighted Euclidean distance:

$$D(\mathrm{x}, \mathrm{x}_i) = \sum_{j=1}^{p}\left(\frac{x_j-x_{j,i}}{\sigma_j}\right)^2$$

GRNN can be visualized as a network with four layers as shown in Figure 1.4. The first layer is the input layer where each molecular descriptor forms an element in the layer. From the input layer, scaled information is fed to the pattern layer. The pattern layer contains elements that represent each training compounds. In this layer, each of these pattern elements calculates a distance measure between the input compound and the training compound that it represents, and then further processes it with

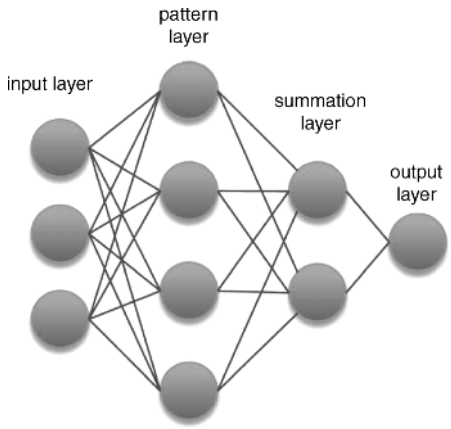

Figure 1.4 GRNN is a neural network with four layers.

Parzen's nonparametric estimator. Each element of the pattern layer is connected to precisely two elements in the summation layer where the numerator and denominator of the basic GRNN equation are calculated. At the final step, the prediction for the target property is obtained by division in the sole node of the output layer.

1.2.1.5 **Gaussian Processes**

Gaussian process (GP) is a generalization of the Gaussian probability distribution to an infinitely large pool of possible functions [8, 87, 88] and it is formally defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [87]. GP is one of the kernel methods for model development and have been applied in QSAR modeling of ADMET and aqueous solubility [89–91], and hERG inhibitors [92].

Consider target property y with molecular descriptors x, in modeling, we would like to find a function $f(x)$ that can relate the observations of x with the observed y. One can assume a linear basis function (or any other basis functions) so that the parameters of the function $f(x)$ can be determined from the observed data set. However, just like multiple linear regression, the restriction to a linear basis function will result in a poorly modeled function if the underlying relationship between the target property and molecular descriptors is nonlinear. Therefore, with Bayesian setting as an alternative, one may include all possible types of functions that may be derived, and assign a prior probability to each of them such that more plausible functions will have higher probabilities, and to use this knowledge in making a weighted decision. However, the number of possible functions may be countless and thus impractical to compute. Therefore, a Gaussian process can be introduce to "generalize" them – an inference is made from these functions such that "hyper"-parameters, that is, a collection of random variables, may be derived or refined and made closer to the observations as more data points are introduced. Since GP is a nonparametric model, we need not worry about the linear or nonlinear underlying relationship [8, 87, 88].

A Gaussian process is completely specified by its mean function $m(\mathrm{x})$ and covariance function $\mathrm{cov}(f(\mathrm{x}), f(\mathrm{x}')) = k(\mathrm{x}, \mathrm{x}')$ [87], thus written as

$$f(\mathrm{x}) \sim GP(m(\mathrm{x}), k(\mathrm{x}, \mathrm{x}'))$$

where the mean function is usually taken as zero. The covariance function is sometimes called the kernel trick, and similar to that used in support vector machines. The learning process involves optimizing the properties for this covariance function. Some covariance functions that may be used are:

$$\text{squared exponential}: \qquad k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left|x_i - x_j\right|^2}{2l^2}\right)$$

$$\gamma\text{-exponential}: \qquad k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\left(\frac{\left|x_i - x_j\right|^\gamma}{l}\right)\right)$$

where l is the characteristic length scale. In the case of squared exponential, sometimes known as radial basis function (RBF), the inverse of l determines the degree of influence of the molecular descriptors. When l had a large value, the covariance will become independent of the molecular descriptor, thus rendering the descriptor irrelevant [87].

To predict the property of an unknown compound x_*, the GP model assigns probabilistic outcomes, that is, a predicted mean:

$$\bar{f}(\mathrm{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathrm{x}_i, \mathrm{x}_*)$$

and a standard deviation for the unknown compound [8]:

$$\mathrm{std} f(\mathrm{x}_*) = \left(k(\mathrm{x}_*, \mathrm{x}_*) - \sum_{i=1}^{N}\sum_{j=1}^{N} k(\mathrm{x}_i, \mathrm{x}_*) k(\mathrm{x}_j, \mathrm{x}_*) H_{ij}\right)^{\frac{1}{2}}$$

where the coefficient $\alpha = (\alpha_1, \ldots, \alpha_N)$ is obtained by solving a system of linear equations

$$(K + \sigma^2 I)\alpha = y$$

where I is the unit matrix, $y = (y_1, \ldots, y_N)$, and H_{ij} are the elements of the matrix $H = (K + \sigma^2 I)^{-1}$. For the mathematical details, please refer to the chapter by Rasmussen and Williams [93] and also Schwaighofer *et al.* [89].

1.2.2 Methods for Classification Problems

1.2.2.1 Logistic Regression

Logistic regression (LR) is similar to linear regression in many ways. LR is used to model the probability of the occurrence of some event as a linear function of a set of

predictors. For example, the relationship between categorical target property (usually a property with binary outcomes like inhibitor/noninhibitor) and a set of molecular descriptors. The following equation calculates the probability:

$$\hat{y} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)}}$$

where β_0 is the model's intercept, $X_1, \ldots, X_k$ are molecular descriptors with their corresponding regression coefficients $\beta_1, \ldots, \beta_k$ (for molecular descriptors 1 through k).

Given an unknown compound, LR calculates the probability that the compound belongs to a certain target property. For example, in predicting whether an unknown compound is toxic or nontoxic, LR tries to estimate the probability of the compound being a toxic substance. If the calculated $\hat{y}$ is >0.5, then it is more probable that the compound is toxic. Conversely, if $\hat{y} < 0.5$, then the compound is more probable to be nontoxic.

Similar to multiple linear regression, the regression coefficients in LR can describe the influence of a molecular descriptor on the outcome of the prediction. When the coefficient has a large value, it shows that the molecular descriptor strongly affect the probability of the outcome, whereas a zero value coefficient shows that the molecular descriptor has no influence on the outcome probability. Likewise, the sign of the coefficients affects the probability as well, that is, a positive coefficient increases the probability of an outcome, while a negative coefficient will result in the opposite. From our experience, LR usually cannot handle high dimensional data well, especially if the data set is large (e.g., more than 60 000 compounds) with large class imbalance (active: nonactive = 1: 68), which can be common in a training set meant for virtual screening. Such LR models commonly have increased computational time and do not perform well [15].

Applications of LR in QSAR studies includes modeling of nucleosides against amastigotes of *Leishmania donovani* [94], skin sensitization prediction [95–97], *Tetrahymena pyriformis* toxicity [14], classification of antibacterial activity [98], and sediment toxicity prediction [99].

1.2.2.2 **Linear Discriminant Analysis**

Linear discriminant analysis (LDA) is commonly used for classification problems and also for dimensionality reduction. It works on data that has categorical target properties and molecular descriptors that are continuous variables. LDA tries to find a hyperplane that can best separate different classes of a target property. The hyperplane is defined by a linear discriminant function which is a linear combination of molecular descriptors:

$$L = \sum_{i=1}^{k} w_i x_i$$

where L is the discriminant score, molecular descriptors $x_i, \ldots, x_k$ and their corresponding weights, $w_i, \ldots, w_k$. As depicted in Figure 1.5, the discriminant function, L, represents the function where a set of data points is projected onto. This function is

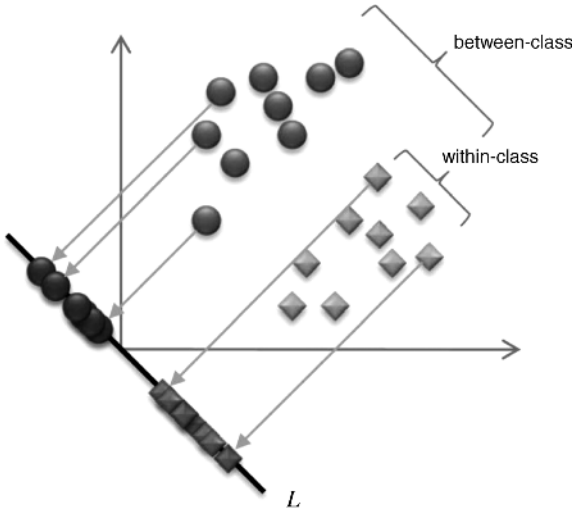

Figure 1.5 Distribution of a two-class samples on a transformed axis, *L*.

obtained through optimizing a set of weights, w_i, that maximizes the ratio of the between-class to the within-class variance to obtain the greatest class separation.

The class of an unknown compound is assigned based on its *L* discriminant score falling below or above a threshold *C*. When two classes are of equal cardinality and have similar distributions, a possible value for *C* is the average of the mean discriminant scores from both classes:

$$C_{\text{threshold}} = \frac{\mu_{L1} + \mu_{L2}}{2}$$

Application of LDA in QSAR includes: prediction of passive blood-brain partitioning [100], mutagenicity [101], prediction of antiparasitic drugs against different parasite species [102], prediction of antitrypanosomal compounds [103], aquatic toxicity of pesticide [104], telomerase inhibitory activity [105], trichomonacidals prediction [106], and QSAR for compounds with antiproliferative activity [107].

1.2.2.3 **Decision Tree and Random Forest**

A decision tree (DT) is a structure with hierarchical arrangement of nodes and branches. A DT has three types of nodes: a root node, internal nodes, and leaf nodes. A root node does not have any incoming branches, while an internal node has one incoming branch and two or more outgoing branches. Lastly, the leaf nodes, also known as terminal nodes, have one incoming branch and no outgoing branches. Each leaf node is assigned with a target property, while a nonleaf node (root or internal node) is assigned with a molecular descriptor that becomes a test condition which branches out into groups of differing characteristics.

The classification of an unknown compound is based on the leaf node that it reaches after going through a series of questions (nodes) and answers (deciding

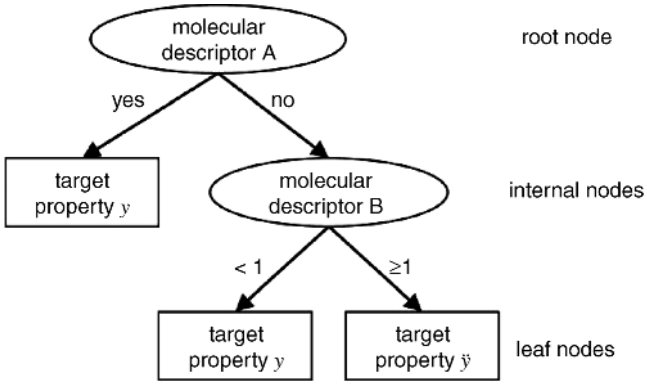

Figure 1.6 Decision tree has three types of nodes.

which branches to take), starting with the first question from the root node. In the example in Figure 1.6, an unknown compound will be classified with target property *y*, if it fulfilled a certain condition for molecular descriptor *A*. Otherwise, molecular descriptor *B* of the unknown compound is checked at the next step. If the value is less than 1, the unknown compound will be labeled with target property *y*. If not, the unknown will be given the label of target property ÿ.

A decision tree is constructed by systematically subdividing the information within a training set with rules and relationships. With a given set of descriptors, many possible variations of trees may be constructed and they may have varying accuracies. Nonetheless, there are algorithms such as the Hunt's algorithm that can be used to induce a decision tree [67]. The algorithms frequently use a recursive greedy heuristic to select which descriptors to split the training data. In the first step of Hunt's algorithm, it examines if all instances in the training set belong to one class *y*. If so, a leaf node *y* will be created and all these training cases will be associated with this node. If not, a molecular descriptor with a certain threshold is chosen to split the samples into smaller subsets, where each of these subsets form a new child node, and the process repeats from the first step until all training cases are associated with a leaf node. The threshold of molecular descriptors that specify the best split can be determined using measures like misclassification error, entropy, and Gini index that enables comparison of "impurities" in the parent node and child nodes; the child nodes should have less impurity than the parent node, therefore, the greater is the impurity difference, the better is the selected threshold for splitting the samples.

Decision trees have the advantage of easy interpretation especially if they are small, and the performance of the decision tree is not so easily affected by unnecessary descriptors. It has been applied on QSAR of cytochrome P450 activities [108], peptide–protein-binding affinity [109], catalysts discovery [110], and in a study of substrates, inhibitors, and inducers of P-glycoprotein [111]. However, a potential drawback of decision tree is its susceptibility to model overfitting due to lack of data or the presence of mislabeled training instances. To overcome the problem of overfitting, methods such as pruning, cross-validation or random forest may be used.

Pruning works by preventing the construction of an excessively complicated tree that flawlessly fits the whole data set, of which mislabeled data may be present. On the other hand, random forest (RF) uses consensus classification to reduce the problem of overfitting while improving the accuracy [112–114]. The algorithm works by growing many decision trees, thus, collectively known as a "forest" that makes a final prediction based on the majority prediction from each of the trees. To construct each tree, a training sample of size *D* is selected at random with replacement from an original data with size *D*. Using the new training sample, a tree is grown with randomly selected descriptors and it is not pruned. RF is easy to use as the user only need to fix two parameters: the number of trees in the forest and the number of descriptors in each trees. It was recommended that a large number of trees should be grown and the number of descriptors to be taken from the square root of the total descriptors [115].

RF can handle large number of training data and descriptors. Besides classifying an unknown compound, it can be extended for unsupervised clustering and outlier detection [114]. RF can also be used to infer the influence of the descriptors in a classification task and also to estimate missing data. It was found that RF is less affected by noisy data or data with many weak inputs [114]. Although it is claimed that RF does not overfit, it was shown that the performance of RF can be influenced by imbalanced data set or small sample size and also by the number of trees and features selected [116, 117]. Therefore, the parameters for RF should be carefully selected and the optimization may be carried out through the use of cross-validation [117].

1.2.2.4 *k*-Nearest Neighbor

k-nearest neighbor (kNN) is a type of lazy learner whereby it delays the learning of the training data until it is needed to classify an unknown sample. It is useful for QSAR studies because QSAR works on the assumption of compounds with similar structure should have similar activities [118]. kNN has been applied on QSAR studies of binding affinity and receptor subtype selectivity of human 5HT1E and 5HT1F receptor-ligands [119], anti-HIV activity of Isatin analogs [120], inhibitors of γ-amino butyric acid transaminase [121], T-helper-type-2 cells receptor antagonist [122], selective cyclooxygenase-2 inhibitors [123], and geranylgeranyltransferase-I inhibitors [124].

kNN works by measuring the distance between the unknown compound and every compound in the training set and then classifies a test compound by searching for the training compounds that are similar in characteristics to the unknown compound. There are various types of distance measures that may be used; two of the common ones are the Euclidean distance

$$d(\mathrm{p},\mathrm{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

and the Manhattan distance

$$d(p,q) = \sqrt{\sum_{i=1}^{n} |p_i - q_i|}$$

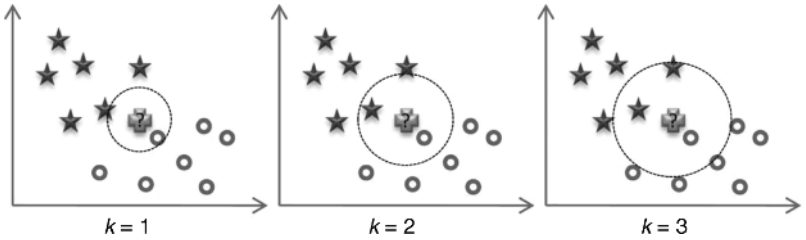

Figure 1.7 Classification of the unknown compound changes when k is different.

where n is the number of molecular descriptors, and p_i and q_i is the ith descriptor for compounds p and q, respectively. The class of the unknown compound is then determined by the majority of the class of its k neighbor(s). The number of neighbors, k, is a user-defined integer that needs to be optimized as it will affect the performance of the model (Figure 1.7). Misclassification can occur if the k is too small or too large. When dealing with binary classification problems, an odd number k is usually chosen to reduce the ambiguity of the predictions. We find that k of 3 is usually sufficient to produce good results and it is frequently used in kNN-related research. Also, since kNN relies on measuring distances between compounds, the chemical descriptors should be scaled to avoid descriptors with large magnitudes from adversely influencing the kNN model.

1.2.2.5 **Probabilistic Neural Network**

Probabilistic neural network (PNN) is similar to the general regression neural network, but it is used in classification problems [125, 126]. PNN is effective in nonlinear mapping, pattern recognition, estimation of target property and likelihood ratios [127]. PNN has been applied on prediction of acute toxicity to *Daphnia magna* and fathead minnow [128, 129], *Tetrahymena pyriformis* toxicity [14], anti-HIV activity of compounds [130], and QSAR of soluble epoxide hydrolase inhibitors [82]. PNN works on the basis of nonparametric estimators of conditional probability density functions and the Bayes strategy to minimize expected risk [127]. Similar to GRNN, the probability density function for each target property for a univariate case can be estimated by the Parzen's nonparametric estimator.

In a binary classification problem where a compound belongs to either class θ_i or θ_j, PNN predicts the class of unknown compounds based on a set of molecular descriptors, x, through the use of Bayes decision rule:

$$\begin{aligned} d(x) &= \theta_i \quad \text{if} \quad h_i\, c_i\, f_i(x) > h_j\, c_j\, f_j(x) \\ d(x) &= \theta_j \quad \text{if} \quad h_i\, c_i\, f_i(x) < h_j\, c_j\, f_j(x) \end{aligned}$$

where class θ_i and θ_j have the probability density function of $f_i(x)$and $f_j(x)$, prior probabilities of h_i and h_j, and costs of misclassification of c_i and c_j, respectively. The prior probabilities and misclassification cost are treated as being equal in most applications. In classification tasks, an unknown compound is predicted with target property θ_j if the product of all three variables is greater for target property θ_j than for any other data with class θ_i not equal to θ_j.

The network architecture for PNN is similar to GRNN, but unlike GRNN which has two elements in the summation layer, PNN will have the same number of elements as the number of target properties. Within each summation element, the estimated probability density function for the corresponding target property is obtained by summing all the acquired inputs from the pattern layer. This information is then passed on to the single element in the output layer where the probability density functions will be evaluated and the class of an unknown compound is assigned with the target property with the highest value.

Like GRNN, PNN has the advantage of fast training for classification problems with small sample size and target properties because it uses the one-pass neural network learning algorithm. Also, PNN is able to handle sparse samples and learns only the necessary information from outliers, thus making it robust and less susceptible to outliers [131]. Memory requirement increases with the increase of data size [131], which may cause a problem if computational resource is limited. Therefore, PNN is usually recommended for problems of small to moderately large sample sizes. Alternatively, clustering methods should be used to reduce the number of elements required in the network before modeling [127].

1.2.2.6 **Support Vector Machine**

Support vector machine (SVM) is based on the structural risk minimization principle from statistical learning theory [132] and it is probably one of the most well-known kernel methods for model development [133]. It is a classifier that is less affected by duplicated data and has lower risk of model overfitting [67]. SVM has become very popular in recent years with its applications in various pattern recognition fields like bioinformatics, medical, economics, and cheminformatics [15, 16, 134–140].

In binary classification of linearly separable data, SVM tries to build a maximal margin hyperplane to separate one class of compounds from the other class as illustrated in Figure 1.8. The hyperplane, also known as the decision boundary, is

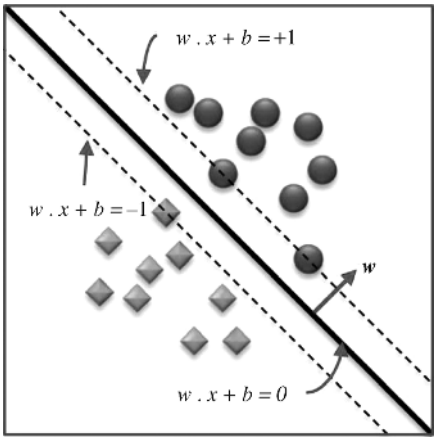

Figure 1.8 Margin and decision boundary of SVM in linearly separable case.

built on the basis of the data points called support vectors and can be represented by the following:

$$w \cdot x + b = 0$$

The parameters w and b are estimated during learning and they must satisfy the following conditions:

$$\begin{array}{ll} \text{Class } 1: & w \cdot x_i + b \geq +1, \quad \text{if } y_i = +1 \\ \text{Class } -1: & w \cdot x_i + b \leq -1, \quad \text{if } y_i = -1 \end{array}$$

and at the same time maximizing the margin by minimizing the following function:

$$f(w) = \frac{\|w\|^2}{2}$$

where y_i is the class label and x_i is a vector of molecular descriptors for compound i, w is a vector perpendicular to the hyperplane, and $\|w\|^2$ is the Euclidean norm of w. With optimized parameters w and b, an unknown compound with vector x can be classified by

$$\hat{y} = \text{sign}[(w \cdot x) + b]$$

The unknown compound is classified as Class 1 if $\hat{y} > 0$ and classified as Class -1 when $\hat{y} < 0$.

Optimization of the parameters of the hyperplane can also be viewed as a convex optimization problem where the margins should be constructed furthest from each of the convex hulls as shown in Figure 1.9. Using the Lagrange multiplier, the function for the optimization problem becomes

$$L = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \, \alpha_j \, y_i \, y_j (\mathrm{x}_i \cdot \mathrm{x}_j)$$

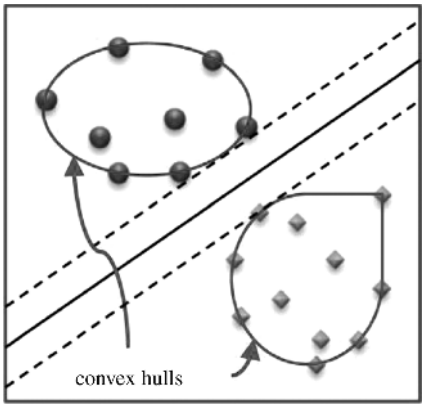

Figure 1.9 Most optimal boundary separating closest points in the convex hulls.

where the parameters α_i are called the Lagrange multipliers and it is obtained by maximizing the expression. Subsequently, feasible solutions for w and b can be obtained once α_i are found. In nonlinearly separable data, SVM uses a kernel function, $k(x_i, x_j)$, to map the vectors into a higher dimensional feature space to make them linearly separable. Classification of an unknown compound, x_j, is performed using

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{l} \alpha_i^0 \, y_i \, k(x_i, x_j) + b\right)$$

where l is the number of support vectors. The unknown compound is classified as a positive or negative compound if $\hat{y} > 0$ or $\hat{y} < 0$, respectively. α_i^0 and b are estimated by maximizing the Lagrangian expression

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \, \alpha_j \, y_i \, y_j \, k(x_i, x_j)$$

under the following conditions:

$$0 \leq \alpha_i \leq C$$
$$\sum_{i=1}^{l} \alpha_i \, y_i = 0$$

where C is a penalty for training errors. Some common kernel function, $k(x_i, x_j)$, that may be used are

$$\text{Polynomial kernel:} \qquad k(x_i, x_j) = (x_i \cdot x_j)^d$$
$$\text{Gaussian radial basis function (rbf):} \qquad k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

We have used SVM in many of our QSAR models and it has been shown to perform well on many problems and is robust even when there is redundant and overlapping data [14–16]. Another advantage of SVM is that it is relatively simple to use as there are only a few user-defined parameters. For example, if the Gaussian rbf kernel is selected, the user will only need to fine-tune the parameters for C and σ. Furthermore, the final results of SVM are reproducible and stable, unlike those of methods like neural networks, which may change from run to run because of the random initialization of the weights [133].

1.3 Software for QSAR Development

There are many commercial or free software available for QSAR development. These include specialized software for drawing chemical structures, interconverting chemical file formats, generating 3D structures, calculating chemical descriptors,

developing QSAR models, and general-purpose software that have all the necessary components for QSAR development. A good website for QSAR resources is the Cheminformatics and QSAR Society website (http://www.qsar.org/). There are lists of software, data sets, and resources pertaining to QSAR in the website.

1.3.1 Structure Drawing or File Conversion

ChemDraw (http://www.cambridgesoft.com/software/ChemDraw/)

ChemDraw is a commercial software for chemical structure drawing and editing. It may be packaged with other programs such as ChemDraw ActiveX/Plugin Viewer, Chem3D, ChemBioFinder, and ChemNMR which enhance the functionality of the program. Besides drawing and editing of chemical structures, the program offers integration of the drawn structure into Microsoft Office documents, conversion of structure from name (or name from structure), 13C and 1H NMR prediction, query of online databases, and many other features [141].

ACD/ChemSketch (http://www.acdlabs.com/resources/freeware/chemsketch/)

ACD/ChemSketch is a software for drawing of chemical structures that comes with other functionalities such as calculation of molecular properties, 2D and 3D structure cleaning, structure naming, and prediction of log*P*. The software is available in two versions: the commercial and freeware version. The freeware version does not include ACD/Dictionary, technical support, ACD/Lab extension for ChemDraw, and the function to search files by structure [142].

Open Babel (http://openbabel.org/)

Conversion of files at different stages of QSAR development may be necessary to satisfy the input requirements of various software. The file conversion can be easily done by using software like Open Babel. Open Babel is an open-source program that enables users to search, convert files, analyze or store data from molecular modeling projects [143]. Open Babel can convert over 90 chemical file formats, and it also has compounds preprocessing functionality like "adding hydrogen bond," "convert dative bonds," and "generate 3D coordinates."

1.3.2 3D Structure Generation

CORINA (http://www.molecular-networks.com/products/corina)

CORINA is one of the commercial software offered by Molecular Networks. It is used for generating three-dimensional structure of small- and medium-sized compounds, necessary as a preprocessing step prior to calculation of 3D molecular descriptors or structure-based docking studies. CORINA can be used as a component in Accelrys Pipeline Pilot or on its own through a Java-based graphical user interface or command line interface which supports batch processing [144].

Concord (http://tripos.com/index.php?family = modules,SimplePage,,,&page = sybyl_concord)

Concord is available as one of SYBYL applications. It is a commercial software that converts 2D inputs into 3D structures rapidly. The main benefits of Concord includes the variety of built-in geometry optimization options and its capability of handling inputs and outputs of common industry-standard formats [145].

Frog (http://bioserv.rpbs.jussieu.fr/Help/Frog-Help.html)

Frog is an online tool for 3D conformation generation from 1D or 2D information using Merck molecular force field [146]. It is accessible as one of the web services in the Mobyle@RPBS (http://mobyle.rpbs.univ-paris-diderot.fr/cgi-bin/portal.py) website. Frog accepts compound structures in the form of SMILES or SDF restricted to 5000 compounds per submission. A newer version, Frog 2, is able to accept 3D information as input to generate multiconformations. Frog is able to process structures with common atoms only and ions in the input file must be removed first [147].

smi23d (http://www.chembiogrid.org/projects/proj_ws_all.html)

smi23d is an open-source program that can be downloaded and compiled for use in Windows or Linux [148, 149]. The program generates 3D structures from SMILES string. It is also accessible via a web service called REST, hosted by the Indiana University. To use the service, one would simply need to append the SMILES string at the end of this URL: "http://cheminfov.informatics.indiana.edu/rest/thread/d3.py/SMILES/" for conversion into 3D structure.

1.3.3 Descriptor Calculation

ADRIANA.Code (http://www.molecular-networks.com/products/adrianacode)

ADRIANA.Code is one of the commercial software offered by Molecular Networks for computing molecular descriptors. Similar to CORINA, ADRIANA.Code can be used as a component in Accelrys Pipeline Pilot or on its own through a graphical user interface or command line interface. The descriptors calculated include physicochemical property descriptors, shape- and size-related descriptors, autocorrelation of 2D interatomic distance distributions, autocorrelation or radial distribution functions of 3D interatomic distance distributions weighted and autocorrelation of distances between surface points [150].

Dragon (http://www.talete.mi.it/products/dragon_description.htm)

Dragon is a commercial software for the computation of molecular descriptors [151]. It also has a free-for-use web-based version called the E-Dragon (http://www.vcclab.org/lab/edragon/) which uses Dragon version 5.4 [152]. It is to note that the features of E-Dragon is more restricted compared to the commercial version as every job submission can only handle up to a maximum of 149 compounds with a maximum of 150 atoms per compound. Currently, Dragon version 5.5 can compute 3224 molecular descriptors which are divided into 22 blocks. These blocks include constitutional or topological descriptors,

walk and path counts, connectivity or information indices, 2D autocorrelations, BCUT descriptors, topological charges indices, 3D-MoRSE descriptors, WHIM descriptors, GETAWAY descriptors, functional group counts, 2D frequency fingerprints and so on. Dragon can work in both Windows and Linux, and it also has simple functions for conducting preliminary graphical and statistical analysis of descriptors, for example, histograms, Pareto plots, and 2D and 3D scatter plots.

Molconn-Z (http://www.edusoft-lc.com/molconn/)

Molconn-Z is a commercial software for molecular descriptor calculation that works on multiple platforms, for example, Windows, Mac OS X, and Linux. It calculates molecular connectivity chi indices, kappa shape indices, electrotopological state indices, topological indices, counts of subgraphs, and vertex eccentricities [153].

PaDEL-Descriptor (http://padel.nus.edu.sg/software/padeldescriptor/)

We released the first version of PaDEL-Descriptor in 2008 and have recently updated it to version 2.0. PaDEL-Descriptor is an open-source Java-based software developed using the Chemistry Development Kit for the calculation of molecular descriptors and fingerprints. Currently, it can calculate 797 descriptors and 10 types of fingerprints which includes 1D, 2D, and 3D descriptors, for example, atom-type electrotopological state descriptors, McGowan volume, molecular linear free energy relation descriptors, ring counts, WHIM, Petitjean shape index, count of chemical substructures identified by Laggner, and binary fingerprints and count of chemical substructures identified by Klekota and Roth [154]. PaDEL-Descriptor works as a standalone program and also available as a Java Web Start version. It has a graphical user interface, a command line interface, and can also be used as an extension to RapidMiner. The program also has some compound preprocessing capabilities like "remove salt," "add hydrogen," and "convert to 3D" [155].

1.3.4 Modeling

KNIME (http://www.knime.org/)

Konstanz Information Miner (KNIME) is an open-source platform with pipelining ability for data integration, processing, analysis, and exploration [156]. Modules for data preprocessing, modeling, visualization, and others, are organized as "nodes" which allows the users to create data flow by connecting these nodes. There are more than 100 processing nodes in the KNIME base version. It also integrates modules from WEKA and has a plugin that allow execution of R scripts. In addition, it has chemistry nodes based on the Chemistry Development Kit (CDK) which enables the calculation of molecular properties and fingerprints. User customized nodes can be implemented in KNIME easily. This enables organizations such as Tripos [157], ChemAxon [158], and Schrödinger [159] to offer their commercial tools as KNIME extensions (nodes).

RapidMiner (http://rapid-i.com/)

RapidMiner is an open-source system with a large collection of algorithms for data analysis and model development. There are more than 500 operators for data processing, model development, evaluation, and visualization, and it also integrates another modeling library, WEKA [160]. It has the "Optimize Parameters" operator which allows semiautomation of parameters searching. The software is able to run on major platforms like Windows, Linux, and Mac OS X. Users are able to visualize the modeling workflow in the form of an intuitive process interface and users also have the option of adding their own algorithms in the form of extensions, written in Java, into RapidMiner easily.

WEKA (http://www.cs.waikato.ac.nz/ml/weka/)

WEKA has a rich compilation of modeling methods and tools for data preprocessing, classification, regression, clustering, and visualization, which are organized into different sections in the WEKA Explorer [161]. It is an open-source software that can run on major platforms like Windows, Mac OS X, and Linux. The WEKA Explorer is used for most modeling tasks. Alternatively, the WEKA KnowledgeFlow which is the graphical front of the software can be used to allow the user to see the flow of the data processing or modeling. WEKA is a flexible software as new analysis methods can be added easily with users own implementation of algorithms or downloads from the "WEKA-related Projects" in the WEKA website.

Orange (http://www.ailab.si/orange/)

Orange is a free program that offers tools for some simple data preparation, evaluation, visualization, classification, regression, and clustering. Each of these functions are available as widgets and the user will need to connect these widgets in a flow for data analyses, similar to WEKA KnowledgeFlow and KNIME. It has the "Select Data" widgets which allow easy manipulation and filtering of data. Although Orange does not have widgets for automated parameter optimization and has fewer operators compared to programs like WEKA or RapidMiner, the variety is suffice for many modeling tasks [162].

TANAGRA (http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html)

TANAGRA is an open-source software containing tools for data analysis, statistics, modeling, and database exploration [163]. Some of these tools are applications for feature selection, feature construction, for example, principal component analysis, descriptive statistics, *t*-test, various clustering algorithms, and modeling methods like multiple linear regression, regression trees, SVM, random forest, naive Bayes classifier, and decision trees. The website of TANAGRA contains a collection of comprehensive tutorials describing the usage of these applications.

MATLAB (http://www.mathworks.com/products/matlab/)

MATLAB is a commercial software that provides an interactive system for algorithm development, data visualization, data analysis, and numeric computation with wide application in image processing, financial analysis, computational biology, and so on. Data can be analyzed easily with ready-to-use functions, but users are also allowed to customize some of these tools or add their own

algorithms for use. It also has functions to integrate MATLAB-based algorithms with external applications and languages such as Microsoft Excel, Java, and C++. This enables developed QSAR models to be easily distributed as stand-alone programs or software modules.

R (http://www.r-project.org/)

R is a free software environment for graphical and statistical analysis that can run on Windows, Linux, and Mac OS X. It has a variety of statistical tools like linear or nonlinear modeling, classical statistical tests, time-series analysis, classification, and clustering. R is extensible as user can add more utilities by creating their own functions or via add-on "packages" that comes with the distribution of R or downloadable through the CRAN websites. Some packages of interests are *kernlab* which provides algorithms for kernel-based machine learning methods such as kernel PCA; *klar* which provides various visualization and classification functions like support vector machine; *RWeka* which is the R interface to WEKA; *nnet* and *tree* for their application in single-layer neural network and classification and regression trees modeling. More information on packages related to modeling is available in the R website under Task View for Machine Learning and Statistical Learning [164].

1.3.5 General purpose

SYBYL (http://www.tripos.com/) SYBYL, as a basic program, provides a wide range of molecular modeling tools which includes tools in structure building, optimization, and comparison (and visualization) of structures and related data. It also comes with a broad selection of force fields that can be used in compound analysis [51]. One of the widely used QSAR techniques, Comparative Molecular Field Analysis (CoMFA), can be found as an integrated module in SYBYL. Besides ligand-based design, users may choose to integrate other SYBYL applications for receptor-based design, structural biology, library design or cheminformatics.

Discovery Studio (http://accelrys.com/products/discovery-studio/)Accelrys Discovery Studio contains a suite of applications for optimizing the drug discovery process. It has applications for properties examination, leads identification and candidate's optimization. For example, other than structure-based design, simulation, QSAR and library design tools, it also contains the predictive ADME and toxicology application which help in identifying undesirable compounds early in the discovery process. It also has the Protein Modeling and Sequence Analysis, and Biopolymer Building tools to allow better understanding of the biological function of the targets. Automation of routine tasks can be done through the use of Pipeline Pilot or scripting in Perl. Discovery Studio is also integrated with web resources like the protein data bank (PDB) and PubChem.

MOE: Molecular Operating Environment (http://www.chemcomp.com/software.htm) MOE provides drug discovery software suites for structured-based

design, pharmacophore discovery, protein and antibody modeling, molecular modeling and simulations, cheminformatics and (HTS) QSAR, medicinal chemistry applications, and methods developments and deployment. The Cheminformatics and (HTS) QSAR suite comes with pipeline tools to process SD files and calculation of over 600 molecular descriptors, model building, similarity searching and combinatorial library design. Customizable functions can be added through their built-in scripting language, Scientific Vector Language (SVL), that comes with over 1000 specific functions for chemical structure manipulation and analyses [165]. MOE is operating systems independent and is also adapted into MOE/batch and MOE/web that enables usage in batch mode (nongraphical interface) and web interface.

CODESSA™ (http://www.codessa-pro.com/index.htm)CODESSA is a commercial software that combines various mathematical and computational systems to build QSAR models. CODESSA is capable of calculating a range of molecular descriptors based on the chemical compounds' three-dimensional structure and/or quantum-chemical wave function. Some of the molecular descriptors calculated are constitutional, topological, geometrical, electrostatic, charged surface area, quantum-chemical, molecular-orbital related, and thermodynamic descriptors. CODESSA is also used for developing models, and cluster analysis of molecular descriptors (or data). It also has tools for model interpretation and compound property prediction from its chemical structure [166]. The use of CODESSA can be integrated with AMPAC™ where the quantum mechanical information produced by AMPAC can be used to calculate molecular descriptor [167].

1.4 Conclusion

Table 1.1 shows the number of published studies from two abstract and citation databases for the period of June 2005 to June 2010. These studies were found by PubMed and Scopus when the different data analysis methods were used as keywords together with "qsar" in the searches limited to title, abstract, and keywords. Take note that the list is nonexhaustive and the results from the two databases are not exclusive from each other.

From Table 1.1, it is clear that classical (simple) modeling methods like MLR and PLS remained as popular choices for QSAR studies. SVM is an increasingly popular method and this may be attributed to its ease of use and generally superior performance in comparative studies [168–171]. However, its usage is still lower than those of MLR and PLS. This may be due to the fact that MLR or PLS models are simple to interpret. There are also many new modeling methods introduced in recent years and they may potentially be used for QSAR studies. These new methods include local lazy regression [35], retro-regression [36], rotated general regression neural network [71], kernel partial least squares and unfolding partial least squares [44].

It is important to note that there is no particular method that is ideal for all problems [91, 169, 172]. The choice of an algorithm should be based on the nature of

Table 1.1 Number of published studies in PubMed and Scopus with different algorithms as keywords.

Data analysis methods	No. of published studies		
	PubMed	Scopus	Rank
Gaussian process	7	16	8
k-Nearest neighbor	38	56	5
Linear discriminant analysis	51	77	4
Logistic regression	9	15	9
Multiple linear regression	218	447	1
Neural network: feedforward backpropagation	3	10	11
Neural network: general regression	2	13	10
Neural network: probabilistic	4	6	12
Partial least squares	241	409	2
Support vector machines	69	125	3
Trees: decision tree	20	37	6
Trees: random forest	14	26	7

the data, and also whether the goal is to build a predictive or diagnostic model. Classification methods are suitable for the prediction of compound class (active versus nonactive) when the availability of activity information is limited. On the other hand, regression methods are suitable for quantitative (activity values e.g., IC_{50} or EC_{50}) prediction when sufficient activity information of compounds possessing the same property is available.

Some potential obstacles in model development include insufficient data size (e.g., lack of report of weak or nonactive compounds, or proprietary compounds), lack of curation of data, lack of standardized procedures that facilitate sharing of data or models, toxicological modeling which is inherently a complicated problem [9, 19, 173–176]. Although these issues present challenges in model development and much research is needed to address them, our research and those of other QSAR modelers have shown that both classification and regression methods have been effective in the predictions of a diverse set of compounds and of various physicochemical, pharmacodynamic, pharmacokinetic, and toxicological properties so far [174, 177].

Despite all the successful stories in QSAR modeling, in our opinion, computational methods may not and should not completely replace conventional *in vitro* or *in vivo* testing methods, but should be further developed as an important and essential complementary tool in the drug development process. Moreover, there are still many research opportunities in the areas of data harmonization, model applicability domain, model validation, inclusion of nonstructural information in modeling, similarity or pharmacophore research, consensus modeling and many more [19, 22, 175, 178–183].

Lastly, we recommend current and future modelers to subscribe to the "Organization for Economic Cooperation and Development (OECD) Principles for

the Validation of QSAR in developing models for scientific or regulatory use. All QSAR models developed should be presented with the following information [184]:

- a defined endpoint;
- an unambiguous algorithm;
- a defined domain of applicability;
- appropriate measures of goodness-of-fit, robustness and predictivity; and
- a mechanistic interpretation if possible.

In essence, the guidelines demonstrate the importance of a transparent validation process and reliable QSAR for its acceptance in regulatory context. The readers are to refer to the guideline document, available on the OECD website (http://www.oecd.org), for a comprehensive explanation and interpretation of the guidelines.

References

1 Snodin, D.J. (2002) *Toxicology Letters*, **127**, 161–168.
2 Kraljevic, S., Stambrook, P.J., and Pavelic, K. (2004) *EMBO Reports*, **5**, 837–842.
3 Adams, C.P. and Brantner, V.V. (2006) *Health Aff (Millwood)*, **25**, 420–428.
4 Kola, I. and Landis, J. (2004) *Nature Reviews. Drug Discovery*, **3**, 711–716.
5 Critical Path Opportunities Reports > Challenges and Opportunities Report – March (2004) http://www.fda.gov/ScienceResearch/SpecialTopics/CriticalPathInitiative/CriticalPathOpportunitiesReports/ucm077262.htm#f15.
6 Yap, C.W., Xue, Y., Li, Z.R., and Chen, Y.Z. (2006) *Current Topics in Medicinal Chemistry*, **6**, 1593–1607.
7 Guido, R.V., Oliva, G., and Andricopulo, A.D. (2008) *Current Medicinal Chemistry*, **15**, 37–46.
8 Schwaighofer, A., Schroeter, T., Mika, S., and Blanchard, G. (2009) *Combinatorial Chemistry and High Throughput Screening*, **12**, 453–468.
9 Valerio, L.G. Jr. (2009) *Toxicology and Applied Pharmacology*, **241**, 356–370.
10 Worth, A.P., Bassan, A., De Bruijn, J., Gallegos Saliner, A., Netzeva, T., Patlewicz, G., Pavan, M., Tsakovska, I., and Eisenreich, S. (2007) *SAR and QSAR in Environmental Research*, **18**, 111–125.
11 Lilienblum, W., Dekant, W., Foth, H., Gebel, T., Hengstler, J., Kahl, R., Kramer, P.J., Schweinfurth, H., and Wollin, K.M. (2008) *Archives of Toxicology*, **82**, 211–236.
12 Yap, C.W. and Chen, Y.Z. (2005) *Journal of Chemical Information and Modeling*, **45**, 982–992.
13 Yap, C.W. and Chen, Y.Z. (2005) *Journal of Pharmaceutical Sciences*, **94**, 153–168.
14 Xue, Y., Li, H., Ung, C.Y., Yap, C.W., and Chen, Y.Z. (2006) *Chemical Research in Toxicology*, **19**, 1030–1039.
15 Liew, C.Y., Ma, X.H., Liu, X., and Yap, C.W. (2009) *Journal of Chemical Information and Modeling*, **49**, 877–885.
16 Liew, C.Y., Ma, X.H., and Yap, C.W. (2010) *Journal of Computer-Aided Molecular Design*, **24**, 131–141.
17 Todeschini, R. and Consonni, V. (2000) *Handbook of Molecular Descriptors*, vol. 11, Wiley-VCH, Weinheim.
18 Guyon, I. and Elisseeff, A. (2003) *The Journal of Machine Learning Research*, **3**, 1157–1182.
19 Tropsha, A., Gramatica, P., and Gombar, Vijay K. (2003) *The QSAR & Combinatorial Science*, **22**, 69–77.
20 Schultz, T.W., Netzeva, T.I., and Cronin, M.T. (2003) *SAR and QSAR in Environmental Research*, **14**, 59–81.
21 Scior, T., Medina-Franco, J.L., Do, Q.T., Martinez-Mayorga, K., Yunes Rojas, J.A.,

and Bernard, P. (2009) *Current Medicinal Chemistry*, **16**, 4297–4313.
22 Sprous, D.G., Palmer, R.K., Swanson, J.T., and Lawless, M. (2010) *Current Topics in Medicinal Chemistry*, **10**, 619–637.
23 Fernández, M. and Caballero, J. (2007) *Journal of Molecular Modeling*, **13**, 465–476.
24 Goodarzi, M., Freitas, M.P., and Ghasemi, N. (2010) *European Journal of Medicinal Chemistry*, **45**, 3911–3915.
25 Pourbasheer, E., Riahi, S., Ganjali, M.R., and Norouzi, P. (2010) *Journal of Enzyme Inhibition and Medicinal Chemistry*, vol 25, 844–853.
26 Papa, E., Kovarich, S., and Gramatica, P. (2010) *Chemical Research in Toxicology*, **23**, 946–954.
27 Luilo, G.B. and Cabaniss, S.E. (2010) *Environmental Science & Technology*, **44**, 2503–2508.
28 Topliss, J.G. and Edwards, R.P. (1979) *Journal of Medicinal Chemistry*, **22**, 1238–1244.
29 Benigni, R., Giuliani, A., Franke, R., and Gruska, A. (2000) *Chemical Reviews*, **100**, 3697–3714.
30 Cash, G.G. (2001) *Mutation Research*, **491**, 31–37.
31 Abraham, M.H., Hassanisadi, M., Jalali-Heravi, M., Ghafourian, T., Cain, W.S., and Cometto-Muniz, J.E. (2003) *Toxicological Sciences*, **76**, 384–391.
32 Patlewicz, G., Basketter, D.A., Smith Pease, C.K., Wilson, K., Wright, Z.M., Roberts, D.W., Bernard, G., Gimenez Arnau, E., and Lepoittevin, J.-P. (2004) *Contact Dermatitis*, **50**, 91–97.
33 Papa, E., Villa, F., and Gramatica, P. (2005) *Journal of Chemical Information and Modeling*, **45**, 1256–1266.
34 Hiromasa, K., Masamoto, A., and Kimito, F. (2008) *Journal of Chemical Information and Modeling*, **48**, 534–541.
35 Guha, R., Dutta, D., Jurs, P.C., and Chen, T. (2006) *Journal of Chemical Information and Modeling*, **46**, 1836–1847.
36 Randić, M. (2001) *Journal of Chemical Information and Computer Sciences*, **41**, 602–606.
37 Dutta, D., Guha, R., Wild, D., and Chen, T. (2007) *Journal of Chemical Information and Modeling*, **47**, 989–997.
38 Farkas, O. and Heberger, K. (2005) *Journal of Chemical Information and Modeling*, **45**, 339–346.
39 Rebehmed, J., Barbault, F., Teixeira, C., and Maurel, F. (2008) *Journal of Computer-Aided Molecular Design*, **22**, 831–841.
40 Riahi, S., Ganjali, M., Pourbasheer, E., and Norouzi, P. (2008) *Chromatographia*, **67**, 917–922.
41 Gharagheizi, F. (2008) *The QSAR and Combinatorial Science*, **27**, 165–170.
42 Yuan, Y., Zhang, R., Hu, R., and Ruan, X. (2009) *European Journal of Medicinal Chemistry*, **44**, 25–34.
43 An Introduction to Partial Least Squares Regression. http://www.ats.ucla.edu/stat/sas/library/pls.pdf.
44 Hasegawa, K. and Funatsu, K. (2010) *Current Computer-Aided Drug Design*, **6**, 103–127.
45 Wold, S., Sjöström, M., and Eriksson, L. (2001) *Chemometrics and Intelligent Laboratory Systems*, **58**, 109–130.
46 Zhu, M., Ge, F., Zhu, R., Wang, X., and Zheng, X. (2010) *Chemosphere*, **80**, 46–52.
47 Paliwal, S.K., Pal, M., and Siddiqui, A.A. (2010) *Medicinal Chemistry Research*, **19**, 475–489.
48 Roy, P.P. and Roy, K. (2009) *Journal of Enzyme Inhibition and Medicinal Chemistry*, **25**, 354–369.
49 Kar, S. and Roy, K. (2010) *Journal of Hazardous Materials*, **177**, 344–351.
50 Deeb, O. and Goodarzi, M. (2010) *Chemical Biology and Drug Design*, **75**, 506–514.
51 Tripos:: SYBYL-X. http://www.tripos.com/.
52 Ritchie, M.D., White, B.C., Parker, J.S., Hahn, L.W., and Moore, J.H. (2003) *BMC Bioinformatics*, **4**, 28.
53 Hornik, K., Stinchcombe, M., and White, H. (1989) *Neural Networks*, **2**, 359–366.
54 Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) *Nature*, **323**, 533–536.
55 Mundie, D.B. and Massengill, L.W. (1991) *IEEE Transactions on Neural Networks*, **2**, 168–170.
56 Krogh, A. and Hertz, J.A. (1992) *Journal of Physics A –Mathematical and General*, **25**, 1135–1147.

57 Reed, R. (1993) *IEEE Transactions on Neural Networks*, **4**, 740–747.

58 Prechelt, L. (1998) *Neural Networks: Tricks of the Trade*, Springer, Berlin/Heidelberg, pp. 553.

59 Kaiser, K.L.E., Niculescu, S.P., and Schüürmann, G. (1997) *Water Quality Research Journal of Canada*, **32**, 637–657.

60 Cui, X., Wang, Z., Zhang, Z., Yuan, X., and Harrington, P.D.B. (2008) Proceedings – 4th International Conference on Natural Computation, ICNC 2008, 2: pp. 134–138.

61 Hemmateenejad, B., Akhond, M., Miri, R., and Shamsipur, M. (2003) *Journal of Chemical Information and Computer Sciences*, **43**, 1328–1334.

62 González-Arjona, D., López-Pérez, G., and Gustavo González, A. (2002) *Talanta*, **56**, 79–90.

63 Goodarzi, M., Ortiz, E.V., Coelho, L.d.S., and Duchowicz, P.R. (2010) *Atmospheric Environment*, **44**, 3179–3186.

64 Patra, J.C. and Singh, O. (2009) *Journal of Computational Chemistry*, **30**, 2494–2508.

65 Ramírez-Galicia, G., Garduño-Juárez, R., Hemmateenejad, B., Deeb, O., Deciga-Campos, M., and Moctezuma-Eugenio, J.C. (2007) *Chemical Biology and Drug Design*, **70**, 53–64.

66 Nohair, M., Mallouk, N., Benmarzouk, M., and Mohssine, E.M. (2009) *Chemical Product and Process Modeling*, **4**, 1–22.

67 Tan, P.-N., Steinbach, M., and Kumar, V. (2005) *Introduction to Data Mining*, Addison-Wesley, New York.

68 Baskin, I.I., Ait, A.O., Halberstam, N.M., Palyulin, V.A., and Zefirov, N.S. (2002) *SAR and QSAR in Environmental Research*, **13**, 35–41.

69 Guha, R. and Jurs, P.C. (2005) *Journal of Chemical Information and Modeling*, **45**, 800–806.

70 Guha, R. (2008) *Journal of Computer-Aided Molecular Design*, **22**, 857–871.

71 Gholamrezaei, M. and Ghorbanian, K. (2007) International Joint Conference on Neural Networks, 2007 IJCNN 2007, pp. 1959–1964.

72 Specht, D.F. (1991) *IEEE Transactions on Neural Networks*, **2**, 568–576.

73 Schløler, H. and Hartmann, U. (1992) *Neural Networks*, **5**, 903–909.

74 Specht, D.F. (1993) *Neural Networks*, **6**, 1033–1034.

75 Nadaraya, E.A. (1964) *Theory of Probability and its Applications*, **9**, 141–142.

76 Watson, G.S. (1964) *Sankhyā: The Indian Journal of Statistics. Series A*, **26**, 359–372.

77 Arkan, E., Shahlaei, M., Pourhossein, A., Fakhri, K., and Fassihi, A. (2010) *European Journal of Medicinal Chemistry*, **45**, 3394–3406.

78 Hu, R., Doucet, J.P., Delamar, M., and Zhang, R. (2009) *European Journal of Medicinal Chemistry*, **44**, 2158–2171.

79 Yap, C.W., Li, Z.R., and Chen, Y.Z. (2006) *Journal of Molecular Graphics and Modelling*, **24**, 383–395.

80 Ji, L., Wang, X., Luo, S., Qin, L., Yang, X., Liu, S., and Wang, L. (2008) *Science in China. Series B, Chemistry, Life Sciences & Earth Sciences*, **51**, 677–683.

81 Agatonovic-Kustrin, S. and Turner, J.V. (2006) *Letters in Drug Design and Discovery*, **3**, 436–442.

82 Mosier, P.D. and Jurs, P.C. (2002) *Journal of Chemical Information and Computer Sciences*, **42**, 1460–1470.

83 Mager, P.P. and Reinhardt, R. (2002) *Molecular Simulation*, **28**, 287–294.

84 Parzen, E. (1962) *The Annals of Mathematical Statistics*, **33**, 1065–1076.

85 Cacoullos, T. (1966) *AnISM*, **18**, 179–189.

86 Masters, T. (1995) *Advanced Algorithms for Neural Networks: A C^{++} Sourcebook*, John Wiley & Sons, Inc, New York.

87 Rasmussen, C.E. and Williams, C.K.I. (2006) *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA

88 Seeger, M. (2004) *International Journal of Neural Systems*, **14**, 69–106.

89 Schwaighofer, A., Schroeter, T., Mika, S., Laub, J., ter Laak, A., Sülzle, D., Ganzer, U., Heinrich, N., and Müller, K.-R. (2007) *Journal of Chemical Information and Modeling*, **47**, 407–424.

90 Obrezanova, O., Gola, J.M.R., Champness, E.J., and Segall, M.D. (2008) *Journal of Computer-Aided Molecular Design*, **22**, 431–440.

91 Obrezanova, O. and Segall, M.D. (2010) *Journal of Chemical Information and Modeling*, **50**, 1053–1061.

92 Hansen, K., Rathke, F., Schroeter, T., Rast, G., Fox, T., Kriegl, J.M., and Mika, S.

(2009) *Journal of Chemical Information and Modeling*, **49**, 1486–1496.
93 Rasmussen, C.E. and Williams, C.K.I. (2006) *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA.
94 Oliveira, K.M.G. and Takahata, Y. (2008) *The QSAR & Combinatorial Science*, **27**, 1020–1027.
95 Fedorowicz, A., Zheng, L., Singh, H., and Demchuk, E. (2004) *International Journal of Molecular Sciences*, **5**, 56–66.
96 Li, Y., Pan, D., Liu, J., Kern, P.S., Gerberick, G.F., Hopfinger, A.J., and Tseng, Y.J. (2007) *Toxicological Sciences*, **99**, 532–544.
97 Liu, J., Kern, P.S., Gerberick, G.F., Santos-Filho, O.A., Esposito, E.X., Hopfinger, A.J., and Tseng, Y.J. (2008) *Journal of Computer-Aided Molecular Design*, **22**, 345–366.
98 Cronin, M.T.D., Aptula, A.O., Dearden, J.C., Duffy, J.C., Netzeva, T.I., Patel, H., Rowe, P.H., Schultz, T.W., Worth, A.P., Voutzoulidis, K., and Schüürmann, G. (2002) *Journal of Chemical Information and Computer Sciences*, **42**, 869–878.
99 Lee, J.H., Landrum, P.F., Field, L.J., and Koh, C.H. (2001) *Environmental Toxicology and Chemistry/SETAC*, **20**, 2102–2113.
100 Vilar, S., Chakrabarti, M., and Costanzi, S. (2010) *Journal of Molecular Graphics and Modelling*, **28**, 899–903.
101 Pérez-Garrido, A., Helguera, A.M., Rodríguez, F.G., and Cordeiro, M.N.D.S. (2010) *Dental Materials*, **26**, 397–415.
102 Prado-Prado, F.J., García-Mera, X., and González-Díaz, H. (2010) *Bioorganic and Medicinal Chemistry*, **18**, 2225–2231.
103 Castillo-Garit, J.A., Vega, M.C., Rolon, M., Marrero-Ponce, Y., Kouznetsov, V.V., Torres, D.F.A., Gómez-Barrio, A., Bello, A.A., Montero, A., Torrens, F., and Pérez-Giménez, F. (2010) *European Journal of Pharmaceutical Sciences*, **39**, 30–36.
104 Wang, G., Li, Y., Liu, X., and Wang, Y. (2009) *The QSAR and Combinatorial Science*, **28**, 1418–1431.
105 Castillo-González, D., Cabrera-Pérez, M.A., Pérez-González, M., Helguera, A.M., and Durán-Martínez, A. (2009) *European Journal of Medicinal Chemistry*, **44**, 4826–4840.
106 Rivera-Barroto, O.M., Marrero-Ponce, Y., Meneses-Marcel, A., Escario, J.A., Barrio, A.G., Arán, V.J., Alho, M.A.M., Pereira, D.M., Nogal, J.J., Torrens, F., Ibarra-Velarde, F., Montenegro, Y.V., Huesca-Guillén, A., Rivera, N., and Vogel, C. (2009) *The QSAR and Combinatorial Science*, **28**, 9–26.
107 Saíz-Urra, L., Pérez-Castillo, Y., González, M.P., Ruiz, R.M., Cordeiro, M.N.D.S., Rodríguez-Borges, J.E., and García-Mera, X. (2009) *The QSAR and Combinatorial Science*, **28**, 98–110.
108 Hammann, F., Gutmann, H., Baumann, U., Helma, C., and Drewe, J. (2009) *Molecular Pharmacology*, **6**, 1920–1926.
109 Ivanciuc, O. (2009) *Current Proteomics*, **6**, 289–302.
110 Wang, X.Z., Perston, B., Yang, Y., Lin, T., and Darr, J.A. (2009) *Chemical Engineering Research and Design*, **87**, 1420–1429.
111 Hammann, F., Gutmann, H., Jecklin, U., Maunz, A., Helma, C., and Drewe, J. (2009) *Current Drug Metabolism*, **10**, 339–346.
112 Amit, Y. and Geman, D. (1997) *Neural Computation*, **9**, 1545–1588.
113 Ho, T.K. (1998) *ITPAM*, **20**, 832–844.
114 Breiman, L. (2001) *MLear*, **45**, 5–32.
115 Larivière, B. and Van Den Poel, D. (2005) *Expert Systems with Applications*, **29**, 472–484.
116 Machine Learning Benchmarks and Random Forest Regression. http://escholarship.org/uc/item/35x3v9t4.
117 Statnikov, A., Wang, L., and Aliferis, C.F. (2008) *BMC Bioinformatics*, **9**, 319.
118 Kubinyi, H. (1998) *Perspectives in Drug Discovery and Design*, **9–11**, 225–252.
119 Wang, X.S., Tang, H., Golbraikh, A., and Tropsha, A. (2008) *Journal of Chemical Information and Modeling*, **48**, 997–1013.
120 Pawar, V., Lokwani, D., Bhandari, S., Mitra, D., Sabde, S., Bothara, K., and Madgulkar, A. (2010) *Bioorganic and Medicinal Chemistry*, **18**, 3198–3211.
121 Bansal, S.K., Sinha, B.N., and Khosa, R.L. (2010) *Medicinal Chemistry Research*, vol 20, 549–553.
122 Jain, A. and Agrawal, R.K. (2009) 2nd International Conference on Biomedical and Pharmaceutical Engineering, ICBPE 2009, 1–4.

123 Bhandari, S., Bothara, K., Pawar, V., Lokwani, D., and Devale, T. (2009) *Internet Electronic Journal of Molecular Design*, **8**, 14–28.

124 Peterson, Y.K., Wang, X.S., Casey, P.J., and Tropsha, A. (2009) *Journal of Medicinal Chemistry*, **52**, 4210–4220.

125 Specht, D.F. (1990) *Neural Networks*, **3**, 109–118.

126 Specht, D.F. (1988) *IEEE International Conference on Neural Networks*, **1**, 525–532.

127 Specht, D.F. and Romsdahl, H. (1994) IEEE International Conference on Neural Networks, 1994 IEEE World Congress on Computational Intelligence, 2, pp. 1203–1208.

128 Niculescu, S.P., Atkinson, A., Hammond, G., and Lewis, M. (2004) *SAR and QSAR in Environmental Research*, **15**, 293–309.

129 Niculescu, S.P., Lewis, M.A., and Tigner, J. (2008) *SAR and QSAR in Environmental Research*, **19**, 735–750.

130 Vilar, S., Santana, L., and Uriarte, E. (2006) *Journal of Medicinal Chemistry*, **49**, 1118–1124.

131 Niculescu, S.P. (2003) *Journal of Molecular Structure: THEOCHEM*, **622**, 71–83.

132 Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer, New York, London.

133 Bennett, K.P. and Campbell, C. (2000) *SIGKDD Explorations Newsletter*, **2**, 1–13.

134 Kim, H.S. and Sohn, S.Y. (2010) *European Journal of Operational Research*, **201**, 838–846.

135 Zuluaga, M.A., Magnin, I.E., Hernández Hoyos, M., Delgado Leyton, E.J.F., Lozano, F., and Orkisz, M. (2010) *International Journal of Computer Assisted Radiology and Surgery*, vol 6, 163–174.

136 Fernandez, M., Ahmad, S., and Sarai, A. (2010) *Journal of Chemical Information and Modeling*, **50**, 1179–1188.

137 Shen, J., Cheng, F., Xu, Y., Li, W., and Tang, Y. (2010) *Journal of Chemical Information and Modeling*, **50**, 1034–1041.

138 Conforti, D. and Guido, R. (2010) *Computers and Operations Research*, **37**, 1389–1394.

139 Basu, S., Das, N., Sarkar, R., Kundu, M., Nasipuri, M., and Kumar Basu, D. (2010) *Pattern Recognition*, **43**, 3507–3521.

140 Khorrami, H. and Moavenian, M. (2010) *Expert Systems with Applications*, **37**, 5751–5757.

141 CambridgeSoft Desktop Software – ChemDraw (Windows/Mac). http://www.cambridgesoft.com/.

142 ACD/Labs.com:: Freeware:: ACD/ChemSketch. http://www.acdlabs.com/resources/freeware/chemsketch/.

143 Open Babel: About. http://openbabel.org/wiki/Open_Babel:About.

144 CORINA: Generation of 3D coordinates. http://www.molecular-networks.com/software/corina/index.html.

145 Tripos:: Concord. http://tripos.com/index.php?family=modules,SimplePage,,,&page=sybyl_concord.

146 Leite, T.B., Gomes, D., Miteva, M.A., Chomilier, J., Villoutreix, B.O., and Tuffery, P. (2007) *Nucleic Acids Research*, **35**, W568–572.

147 Frog: FRee Online druG conformation generation. http://bioserv.rpbs.jussieu.fr/Help/Frog-Help.html.

148 3D Conformer Generation – Metabolomics Fiehn Lab. http://fiehnlab.ucdavis.edu/staff/kind/ChemoInformatics/Concepts/3D-conformer.

149 Simple 3D Conformer Generation with Smi23D. http://depth-first.com/articles/2007/12/12/simple-3d-conformer-generation-with-smi23d.

150 ADRIANA.Code. http://www.molecular-networks.com/products/adrianacode.

151 Talete – Dragon. http://www.talete.mi.it/products/dragon_description.htm.

152 E-Dragon Software. http://www.vcclab.org/lab/edragon/.

153 Molconn-Z. http://www.edusoft-lc.com/molconn/.

154 Klekota, J. and Roth, F.P. (2008) *Bioinformatics (Oxford, England)*, **24**, 2518–2525.

155 PaDEL-Descriptor. http://padel.nus.edu.sg/software/padeldescriptor/index.html.

156 KNIME | Konstanz Information Miner. http://www.knime.org/.

157 Tripos:: Tripos Chemistry Extension for KNIME. http://tripos.com/index.php?

family=modules,SimplePage, TCE_Knime.
158 Integration nodes≪ ChemAxon – toolkits and desktop applications for cheminformatics. http://www.chemaxon.com/about/our-partners/integration-nodes/.
159 Schrödinger – Products Guides – KNIME http://www.schrodinger.com/products/14/8/.
160 Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T. (2006) KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–940.
161 Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H. (2009) *SIGKDD Explorations Newsletter*, **11**, 10–18.
162 Orange – Data Mining Fruitful & Fun. http://www.ailab.si/orange/.
163 Rakotomalala, R. (2005) Actes de EGC'2005, RNTI-E-3, 2, 697–702.
164 The R Project for Statistical Computing. http://www.r-project.org/.
165 MOE – Methods Development and Deployment. http://www.chemcomp.com/software-mdd.htm.
166 Codessa. http://www.codessa-pro.com/index.htm.
167 AMPAC, Semichem Inc. http://www.semichem.com/ampac/default.php.
168 Czermiński, R., Yasri, A., and Hartsough, D. (2001) *Quantitative Structure–Activity Relationships*, **20**, 227–240.
169 Brace, C.L., Melville, J.L., Pickett, S.D., and Hirst, J.D. (2007) *Journal of Chemical Information and Modeling*, **47**, 219–227.
170 Xue, Y., Yang, X.G., Chen, D., Wang, M., and Chen, Y.Z. (2009) *Journal of Computational Chemistry*, **30**, 1202–1211.
171 Darnag, R., Mostapha Mazouz, E.L., Schmitzer, A., Villemin, D., Jarid, A., and Cherqaoui, D. (2010) *European Journal of Medicinal Chemistry*, **45**, 1590–1597.
172 Plewczynski, D., Spieser, S.A.H., and Koch, U. (2006) *Journal of Chemical Information and Modeling*, **46**, 1098–1106.
173 Benigni, R. and Giuliani, A. (2003) *Bioinformatics (Oxford, England)*, **19**, 1194–1200.
174 Yap, C.W., Xue, Y., Li, H., Li, Z.R., Ung, C.Y., Han, L.Y., Zheng, C.J., Cao, Z.W., and Chen, Y.Z. (2006) *Mini Reviews in Medicinal Chemistry*, **6**, 449–459.
175 Tropsha, A. and Golbraikh, A. (2007) *Current Pharmaceutical Design*, **13**, 3494–3504.
176 Yang, S.Y. (2010) *Drug Discovery Today*, **15**, 444–450.
177 Yap, C.W., Li, H., Ji, Z.L., and Chen, Y.Z. (2007) *Mini Reviews in Medicinal Chemistry*, **7**, 1097–1107.
178 Sedykh, A., Zhu, H., Tang, H., Zhang, L., Richard, A., Rusyn, I., and Tropsha, A. (2010) *Environmental Health Perspectives*, vol 119, 364–370.
179 Lessigiarska, I., Worth, A.P., Netzeva, T.I., Dearden, J.C., and Cronin, M.T. (2006) *Chemosphere*, **65**, 1878–1887.
180 Verma, J., Khedkar, V.M., and Coutinho, E.C. (2010) *Current Topics in Medicinal Chemistry*, **10**, 95–115.
181 Kruhlak, N.L., Contrera, J.F., Benz, R.D., and Matthews, E.J. (2007) *Advanced Drug Delivery Reviews*, **59**, 43–55.
182 Dearden, J.C. (2003) *Journal of Computer-Aided Molecular Design*, **17**, 119–127.
183 Wawer, M., Lounkine, E., Wassermann, A.M., and Bajorath, J. (2010) *Drug Discovery Today*, **15**, 630–639.
184 OECD, *Guidance Document on the Validation of (Quantitative) Structure–Activity Relationships Models*, Organisation for Economic Co-operation and Development (2007).