PREDICTING THE EFFECT OF HYDROPHOBICITY SURFACE ON BINDING AFFINITY OF PCP-LIKE COMPOUNDS USING MACHINE LEARNING METHODS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MİNE YOLDAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

APRIL 2011

Approval of the thesis:

PREDICTING THE EFFECT OF HYDROPHOBICITY SURFACE ON BINDING AFFINITY OF PCP-LIKE COMPOUNDS USING MACHINE LEARNING METHODS

submitted by **MİNE YOLDAŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Ferda Nur Alpaslan
Supervisor, **Computer Engineering Dept., METU**

Prof. Dr. Erdem Büyükbingöl
Co-supervisor, **Pharmacy Dept., Ankara University**

Examining Committee Members:

Prof. Dr. Erdem Büyükbingöl
Pharmacy, Ankara University

Assoc. Prof. Ferda Nur Alpaslan
Computer Engineering, METU

Assoc. Prof. Nihan Kesim Çiçekli
Computer Engineering, METU

Asst. Prof. Tolga Can
Computer Engineering, METU

Asst. Prof. Sinan Kalkan
Computer Engineering, METU

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MİNE YOLDAŞ

Signature :

ABSTRACT

PREDICTING THE EFFECT OF HYDROPHOBICITY SURFACE ON BINDING AFFINITY OF PCP-LIKE COMPOUNDS USING MACHINE LEARNING METHODS

Yoldaş, Mine
M.Sc., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Ferda Nur Alpaslan
Co-Supervisor : Prof. Dr. Erdem Büyükbingöl

April 2011, 47 pages

This study aims to predict the binding affinity of the PCP-like compounds by means of molecular hydrophobicity. Molecular hydrophobicity is an important property which affects the binding affinity of molecules. The values of molecular hydrophobicity of molecules are obtained on three-dimensional coordinate system. Our aim is to reduce the number of points on the hydrophobicity surface of the molecules. This is modeled by using self organizing maps (SOM) and k-means clustering. The feature sets obtained from SOM and k-means clustering are used in order to predict binding affinity of molecules individually. Support vector regression and partial least squares regression are used for prediction.

Keywords: spherical-self organizing maps, support vector regression, partial least squares regression, k-means clustering, prediction

ÖZ

HİDROFİBİSİTİ YÜZEYLERİN PCP BENZERİ BİLEŞİKLERİN BAĞLANMA EĞİLİMLERİNE ETKİSİNİ MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN KULLANILARAK ÖNGÖRÜLMESİ

Yoldaş, Mine
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ferda Nur Alpaslan
Ortak Tez Yöneticisi : Prof. Dr. Erdem Büyükbingöl

Nisan 2011, 47 sayfa

Bu çalışmada, PCP benzeri bileşiklerin bağlanma eğilimleri moleküler ipofilisiti kullanılarak öngörülmeye çalışılmaktadır. Moleküler hidrofibisiti bir molekülün bağlanma eğilimini etkileyen önemli bir faktördür. Moleküllerin moleküler hidrofobisiti değerleri üç boyutlu koordinat sisteminde elde edilmektedir. Amacımız moleküllerin hidrofobisiti yüzeyindeki noktaların sayısını azaltmaktır. Bu nedenle bu özellik küresel örgütlemeli harita ve k-ortalama kümeleme kullanılarak modellenmektedir. Moleküllerin bağlanma eğilimlerini öngörmek için SOM ve k-ortalama kümeleme kullanılarak elde edilen özellik kümesi ayrı ayrı kullanılacaktır. Bağlanma eğilimlerini öngörmek amacıyla destek vektör regresyonu ve kısmi en küçük kareler regresyonu kullanılacaktır.

Anahtar Kelimeler: Küresel örgütlemeli harita, destek vektör regresyonu, kısmi en küçük kareler regresyonu, k-ortalama kümeleme, tahmin

To my family and my friends

ACKNOWLEDGMENTS

I would like to express my deepest and sincere gratitude to all those who help me to complete this thesis.

Firstly, I would like to thank my supervisor Assoc. Prof. Dr. Ferda Nur Alpaslan and co-supervisor Prof. Dr. Erdem Büyükbingöl. Because they give me opportunity to study this interesting problem, I can learn something about drugs.

I would also like to thank Asst. Prof. Dr. Tolga Can for helping some points about Self-Organizing Maps (SOM).

I am also grateful to my friends especially Özlem, Aslı, Serdar, Ayşegül and Zerrin. They always listened to me and tried to help me.

Lastly, I would like to thank to my parents and brother. They are always with me in sense during this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The aim of drug design is to discover chemical substances interacting with a specific molecule such as proteins. When the chemical substance binds the corresponding protein, the drug cures the patient from a specific illness. These chemical substances are called ligands. However, finding appropriate ligand in a huge chemical space is very difficult. High-throughput screening (HTS) is used for this purpose. In HTS, large numbers of diverse compounds are tested, chemical space is searched randomly and the molecules ensuring some biochemical activity are chosen [21].

In ligand-based drug design, if a ligand binds a target protein, new ligands may be designed which interact with target protein. Also, quantitative structure-activity relationship (QSAR) may be used to find correlation between the properties of the molecules and biological activity.

Neural networks may also be used in order to find appropriate ligands or for feature extraction. Thus, large amount of data may be analyzed. Self organizing maps are sometimes used in classification of large datasets.

Although there are many physiochemical properties affecting the binding affinity of a ligand to the receptor, hydrophobicity is a major and basic property regarding for the optimal interaction features of the appropriate ligands with their target sites. So, molecular hydrophobicity is used for prediction of the binding affinity of 38 Phencyclidine (PCP)-like compounds in this study. Phencyclidine is ”dissociative drug formerly used as an anesthetic agent, exhibiting hallucinogenic and neurotoxic effects” [18]. Molecular hydrophobicity means that a molecule is repelled from water. Molecular lipophilicity means that a molecule loves fat.

Molecular hidrophobicity is often used interchangeably with molecular lipophilicity. Because if a molecule loves fat, it will repel from water. In this study, molecular lipophilicity will be used instead of molecular hydrophobicity in the following chapters.

The organization of thesis is as given below.

- In chapter 2, the background of feature extraction and prediction methods are explained which are used in this thesis. Feature extraction methods are self-organizing maps(SOM) and k-means clustering. Prediction methods are support vector regression and partial least squares(PLS).
- In chapter 3, implementation of SOM to molecular lipophilicity surfaces is discussed.
- In chapter 4, binding affinity of molecules is predicted by molecular lipophilicity.
- The conclusion is discussed in chapter 5.

CHAPTER 2

RELATED WORK

Self-organizing map is an unsupervised learning algorithm. In unsupervised learning, the aim is to classify the data into some classes which are not known apriori. Actually, the class of all training samples are not known. If the distribution of data is not known, it will be hard to choose convenient algorithm and get correct results [24]. In SOM, high-dimensional input space is projected onto a low-dimensional grid. This grid can be used for visualization of surfaces and to show different features of the SOM. These features of SOM is also the features of the data [25].

k-means clustering is used in order to divide the data set into k clusters. The mean of each cluster is calculated and each data sample is assigned to the cluster with minimum mean.

In this chapter, the use of SOM and k-means clustering in molecular surfaces is discussed because we applied SOM and k-means algorithms to the hidrophobicity surface molecules in this study.

Waganer et. al [26] use self-organizing maps in rational drug design and combinatorial chemistry in [26]. Electrostatic potential on the Van der Waals surface of each molecule is mapped onto one self-organizing neural network. These surfaces can be mapped onto a two-dimensional plane by using self-organizing map. The color of each unit is determined by the sample point(s) which is(are) mapped on the corresponding unit. In Figure 2.1, a 3-dimensional model of a molecule is mapped onto a torus that is a special implementation of two dimensional SOM and the colors of the map are similar with the colors of molecule. The smallest negative values of electrostatic potential are shown by red while the smallest positive values are shown by blue or violet. Other values are shown by continuous mixtures of colors. The SOMs of the molecular electrostatic potential of four ligands binded to muscarinic receptors and four ligands binded to nicotinic receptors are generated. When visualization of the maps are compared, ligands binded to muscarinic receptors have common characteristics and these characteristics are not included the maps of the ligands that bind to nicotinic receptors.

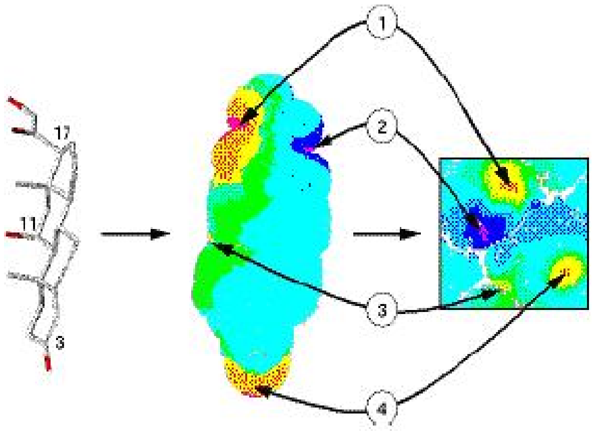


Figure 2.1: Mapping of a 3-dimensional molecule onto a two-dimensional SOM [26]

Gasteiger et. al [27] map molecular electrostatic potential of a molecule onto a two-dimensional plane, as well. Since it is easy to perceive all features of the electrostatic potential of a molecule in a two-dimensional plane. The features of molecules which result from heteroatoms, conformation and chirality are seen in these maps.

Alhoniemi et. al [28] used two-dimensional self-organizing maps in exploratory phase of data mining. Firstly, the input data set is mapped onto a two-dimensional SOM. If the number of units of SOM is large, similar units will be clustered by using hierarchical agglomerative clustering and partitive clustering using k-means clustering in the second phase. Thus, the time of computation is reduced. In agglomerative clustering, two clusters that are closest to each other are merged until there is only one cluster left. In partitive clustering, the data set is partitioned into a some clusters. Here, k-means clustering is used as partitive clustering.

Liu et. al [29] tried to compare 3-dimensional shape of flexible molecules by presenting a new shape descriptor named Diffusion Distance Shape Descriptor (DDSD). m points are selected from the surface of molecule by k-means clustering. The inner distance between each pair of points in S is calculated. And then the diffusion distances between each pair of points in S are calculated. Finally, the descriptor of the shape O is built as the histogram of values of diffusion distances. The flow chart of the method is shown in Figure 2.2.

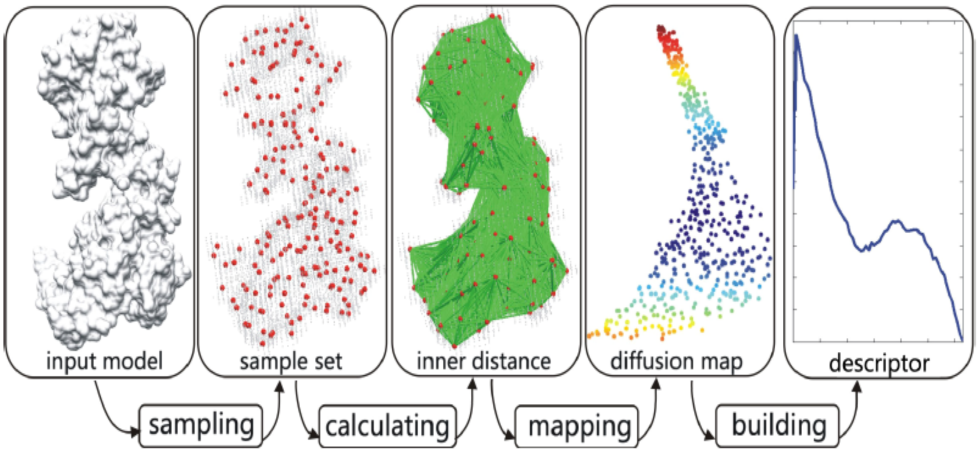


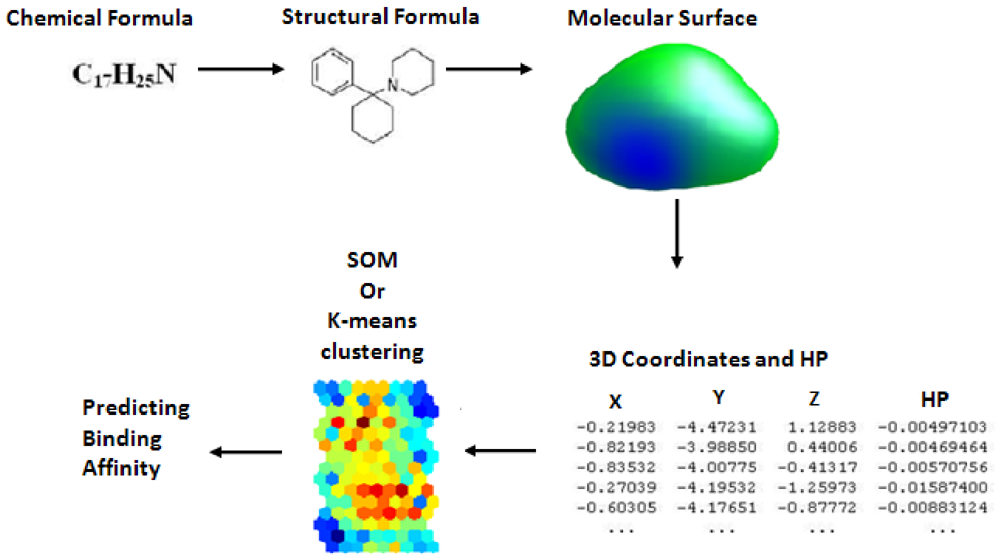
Figure 2.2: Flow chart of the method in [29]

Clustering algorithms are used for gene expression data analysis in molecular biology. Genes are partitioned into similar groups by clustering algorithms such as k-means, hierarchical clustering, SOM. So, functionally related genes are identified. Since there are too many data in molecular biology, faster clustering algorithms must be developed. Lu et. al [30] proposed *Fast Genetic K-means Algorithm (FGKA)*. It is based on *Genetic K-means Algorithm(GKA)* which is proposed by Krishna and Murty in 1999. K-means algorithm may converge to a local optimum while FGKA and GKA always converge to a global optimum. FGKA is faster than GKA. Gene expression data which consists of N genes and each gene is a n-dimensional vector. The aim of FGKA is to divide the N patterns into K clusters. Firstly, FGKA generates an initial population P_0 of Z coded partitions. The selection, the mutation and the k-means operator are used to generate next generation P_{i+1} from the current population P_i.

CHAPTER 3

BACKGROUND

In this study, the dataset in [1] is used and the steps shown in figure 3.1 are followed. As can be seen from Figure 3.1, structural formula of each molecule are obtained on 3D coordinate system. Then, molecular lipophilicity surfaces are obtained by the structural formulas of molecules. The molecular lipophilicity surfaces are acquire by VegaZZ program [15]. Thus, we obtain 3D coordinates of hydrophobicity of each molecule. Since there are a lot of points on the surfaces for prediction, the number of points on the surfaces are reduced by using a Self-Organizing Map (SOM) and k-means clustering. We extract the value of each neuron on SOM as feature sets.



X	Y	Z	HP
-0.21983	-4.47231	1.12883	-0.00497103
-0.82193	-3.98850	0.44006	-0.00469464
-0.83532	-4.00775	-0.41317	-0.00570756
-0.27039	-4.19532	-1.25973	-0.01587400
-0.60305	-4.17651	-0.87772	-0.00883124
...	...	...	...

Figure 3.1: Flow of the thesis

[1]

In this chapter, we will mention methods used for feature extraction, data preparation and prediction.

3.1 Feature Extraction

The methods for feature extraction are Self Organizing Maps (SOM) and k-means clustering.

3.1.1 Self-Organizing Maps(SOM)

There are three categories of artificial neural networks: *feedforward networks, feedback networks* and *competitive learning*. In*feedforward networks*, some outputs are tried to be learned from some inputs by using supervised adjustment. In *feedback networks*, the initial state of the system is tried to find by using the input samples. In *competitive or unsupervised learning*, neighboring cells in a neural network try to win a competition whose aim is to match an input pattern with minimum distance.

Self- organizing map (SOM), which is created by Kohonen in 1981, is a competitive learning based artificial neural networks. In SOM, each neuron has some neighbors because of connection of the network's nodes to each other. There are training vectors and weight vectors of each node which are generally initialized randomly. Training vectors are shown as $\overline{x} = x(t) \in R^n$ where t is time coordinate and weight vectors are shown as $w_i(t) : w_i \in R^n$ where i=1,2,...,k(k is the number of nodes in the network). Training vectors are compared with each node in the network. Euclidean distance is generally used to compute the distance between weight vector of each node and training vectors at each step t, where t=1,2,3,... Some other distance metrics can also be used instead of Euclidean distance. The node with minimum distance between weight vector of the node and training vector is called *winning or best-matching node* and the weights vector of the winning node and its neighbors are updated while all the other weight vectors remain the same.

The major aim of using SOM is to create feature maps having reduced the dimension of the space. SOM is used especially for data visualization. It is widely used in fields such as practical speech recognition, robotics, process control and telecommunications.

3.1.1.1 The Learning Algorithm

Firstly, the SOM size should be determined. SOM size may be small, medium or large depending on the purpose of the purpose according to Ultsch and Simon [7]. Moreover, Versanto offers equation 3.1 for determination the optimal size of the map:

$$msize = 5\sqrt{k} \tag{3.1}$$

where *k* is the product of columns and rows of the dataset [7]. Although optimal map size will minimize the empty units, some empty units are needed to facilitate cluster interpretation [7].

There are three approaches to initialize the weight vectors [6].

- **Random initialization:** The weight vectors are initialized randomly.
- **Sample initialization:** The weight vectors are initialized from the input patterns. If there are *k* weight vectors, first *k* input patterns or randomly chosen *k* input patterns are assigned as weight vectors.
- **Linear initialization:** Two principal eigenvectors of the input data which comprise a two-dimensional linear subspace are found. A rectangular or hexagonal array is defined along this subspace. The centroid of this array coincide with the mean of the input samples. The weight vectors are initialized by using this array.

SOM toolbox [9] includes random and linear initializations. We used both of them in this study.

The basic learning algorithm of SOM is explained below:

1. Assume that $x = x(t) \in \Re^n$ is the vector of samples where t is the number of iterations. t is set as desired.
2. $w_i(t) : w_i \in \Re^n, i = 1, 2, ..., k$ is the weight vector which is initialized by random initialization or sample initialization or linear initialization..
3. *x(t)* is compared with each $w_i(t)$ at each instant of time and best-matching $w_i(t)$ is selected with respect to distance between *x(t)* and $w_i(t)$.

4. The closest $w_i(t)$ to *x(t)* and the neighbours of this weight are updated with respect to time.

5. Repeat steps 3-4 until the number of iterations is t.

The arrangement of cells in the two-dimensional map can be hexagonal, rectangular etc. In figure 3.2, input *x* is connected in parallel with all neurons.

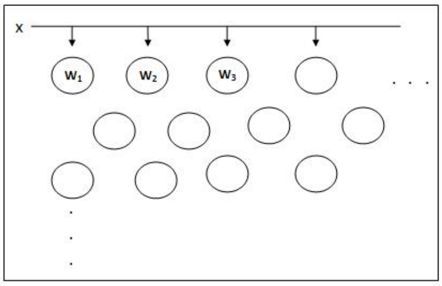


Figure 3.2: Cell-arrangement for the map [16]

The distance between the sample vector and the weight vectors is often calculated by Euclidean distance although dot product can also be used. Euclidean distance gives the distance between two points. The Euclidean distance between a sample vector and a weight vector is calculated by equation 3.2.

$$d(x, w) = \sqrt{(x_1 - w_1)^2 + (x_2 - w_2)^2 + ... + (x_n - w_n)^2} \tag{3.2}$$

where n is the dimension of the input samples and weight vectors. The weight vector with minimum distance to corresponding sample vector is called the ”winner” or ”best-matching” unit.

Not only the weight vector with the minimum distance is updated but also the neighboring units of weight vectors are updated. The width of the neighborhood is indicated by N_c. If N_c is too large, the spatial resolution will be coarse. While the width of N_c is shrinking, the spatial resolution improves. Width of the neighborhood shrinks with time(See Figure 3.3). Finally, only the best-matching unit is updated as in the case of simple competitive learning.

The winning node’s weight is updated as:

$$w_i(t + 1) = w_i(t) + \alpha(t)h_i(t)[x_s(t) - w_i(t)] \tag{3.3}$$

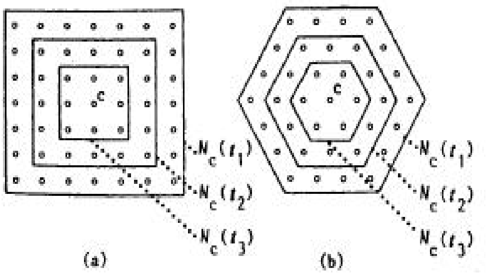


Figure 3.3: Topological neighborhood $N_c(t_1 < t_2 < t_3)$ (a) Rectangular (b) Hexagonal [6]

where $\alpha(t)$ is learning rate in range (0,1] and may be constant or linear function or exponential function decreasing with time or inversely proportional to t. For example, $\alpha(t) = 0.9(1 - t/1000)$ can be used as learning rate function [6]. If $\alpha(t)$ is selected randomly, it should have reasonably high values for the first 1000 iterations and decrease monotonically. Here, $h_i(t)$ is the neighborhood function and effects the convergence. $h_i(t)$ may be equal to $\alpha(t)$ within neighborhood width, N_c. Moreover, Gaussian kernel function in equation 3.4 may be used in some situations.

$$h_i(t) = exp\left(\frac{- \parallel r_i - r_c \parallel}{\sigma^2(t)}\right) \tag{3.4}$$

Here, $\sigma(t)$ is the the width of neighborhood decreasing with time. $\parallel r_i - r_c \parallel$ is the distance between the best-matching unit c and its neighbor i.

Determination of the number of iterations is important for the accuracy of the mapping. The number of iterations should be at least 500 times of the number of network units for good statistical accuracy. This is a rule of thumb [6]. The dimension of the dataset has no effect on the number of iterations. For instance, 10000 or less iterations may be enough in speech recognition for fast learning [6].

Two dimensional self organizing maps have "boundary effect" problem which means that the boundary units have less neighbors than the others. So, boundary units have less chances for updating. Heuristic weighting rule and local-linear smoothing are proposed as mathematical solutions for this problem [1]. Besides these solutions, mapping of data onto toric map is suggested. A cylinder is created by connecting opposite sides of a rectangular grid. A torus is cerated by connecting opposite cycles of this cylinder (See Figure 3.4). However, this solution have some problems, too. Firstly, it is difficult to determine where to cut torus. Secondly, each neuron associates different surface areas with respect to the place of it on the

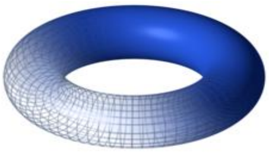


Figure 3.4: Example of a torus [18]

torus. Finally, it is hard to interpret intuitively for most people. In addition to all of these suggestions, Kohonen proposed an easily applied rule named ”One-Half Rule”. According to Kohonen, using asymmetrical SOM shape is better than using symmetrical one. The length of short side should be at least half of the longer side of the SOM with respect to ”One-Half Rule” [7].

3.1.1.2 Types of Self-Organizing Maps

Three kinds of self-organizing maps are shown in Figure 3.5. The architectures and neighborhoods are introduced.

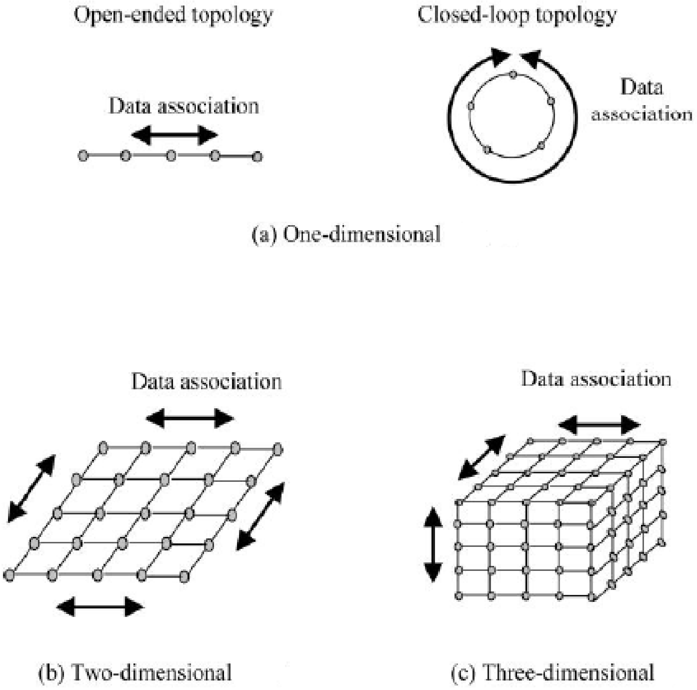


Figure 3.5: Types of Self-Organizing Maps [23]

- **One-dimensional SOM:** N units are arranged as open-ended or closed-loop topology. Although each unit except the first and last units has two neighbors in the open-ended topology, all units have two neighbors in the closed-loop topology.

- **Two-dimensional SOM:** *NxM* units are arranged on a two-dimensional grid. There are two neighborhood types as rectangular and hexagonal neighborhoods. Each unit except boundaries of the grid has 8 neighbors in the rectangular neighborhood and 6 neighbors in the hexagonal neighborhood as can be seen in Figure 3.3(a) and 3.3(b). If the distance between the center unit and its neighbors are equal, hexagonal neighborhood is often preferred.

- **Three-dimensional SOM:** Units are arranged inside a rectangular prism or hypercube. Neighbor units may lie inside a sphere, cube or three-dimensional diamond or star shaped structures.

3.1.2 K-means Clustering

K-means clustering algorithm was discovered by MacQueen in 1967 [18]. It is an unsupervised learning algorithm used to classify the data with respect to the features of the dataset. The k-means clustering algorithm is given below.

1. The number of clusters(k) must be determined and the each centroid vector is initialized randomly.

2. Each cluster has a centroid.

3. Each point in the dataset is assigned to the nearest centroid.

4. After no points remain, new centroids are re-calculated as the mean of the points in the corresponding cluster.

5. Steps 2-4 are repeated until the centroids do not change their location.

In k-means clustering, centroids are placed far away from each other as much as possible. As can be seen, the aim of the algorithm given above is to minimize the equation 3.5. This function is called squared error function.

$$Error = \sum_{j=1}^{k} \sum_{i=1}^{n} \| x_i^{(j)} - c_j \|^2 \tag{3.5}$$

where $\| x_i^{(j)} - c_j \|^2$ is a chosen distance measure between $x_i^{(j)}$ is a data point and c_j is j^{th} cluster center, k is the number of centroids and n id the number of input smaples.

The performance of the algorithm depends on the initial randomly selected centroids.

3.2 Prediction Methods

In this section, support vector regression (SVR) and partial least square regression (PLS) which are used for predicting the value of lipophilicity for each molecule are explained.

3.2.1 Support Vector Machines (SVM)

It will be easier to understand support vector regression if support vector machines are explained. So, firstly support vector machines are explained.

3.2.1.1 Linear Support Vector Machines

Starting with separable support vector machines which are linear machines facilitate to understand SVM. x_i is the i^{th} dimension of the input samples where $x \in R^n$. A separating hyperplane separates positive examples from negative ones where input vectors are in two dimensional form (See Figure 3.6). The data points which satisfy the equation $\mathbf{w}x + b = 0$ are on the hyperplane. Here, **w** is perpendicular to the hyperplane and each data point classifies as y_i = -1,1 because of the decision function. The decision function is:

$$f(x) = sign(w.x + b) \tag{3.6}$$

where w is weight vector, x is the input pattern and b is the threshold.

Sum of the shortest Euclidean distance to the positive example and the shortest Euclidean distance to the negative example is called *m*argin. The aim of the SVM classification is finding

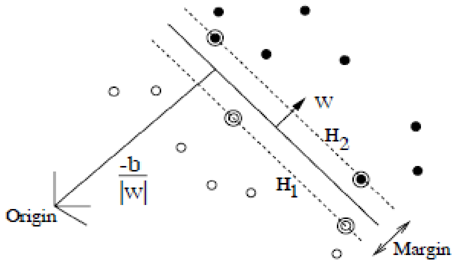


Figure 3.6: Support vector machines for separable case of binary classification [5]

the hyperplane with the largest margin between two classes. This problem is formulated as a quadratic optimization problem:

$$\mathbf{w}.\mathbf{x}_i + b \geq 1, y_i = 1 \tag{3.7}$$

$$\mathbf{w}.\mathbf{x}_i + b \leq -1, y_i = -1 \tag{3.8}$$

These two inequalities can be written in one inequality.

$$y_i(\mathbf{x}_i.\mathbf{w} + b - 1 \geq 0), \forall i \tag{3.9}$$

For equation 3.7, the data points are on the hyperplane H1: $\mathbf{w}.\mathbf{x}_i+b = 1$. And for equation 3.8, the data points are on the hyperplane H2: $\mathbf{w}.\mathbf{x}_i + b = -1$. Margin is between these two hyperplane and equals to $2/\|w\|$. Support vectors are the points which satisfies the equality 3.9 and shown in Figure 3.6 by extra circles. Now, the equation 3.9 is converted to Lagrangian formulation, because it is easier to handle [5]. It uses Lagrange multipliers α_i, where *i=1,2,...,n* for each of inequality constraints. The Lagrangian is:

$$L_1 = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i.w + b) + \sum_{i=1}^{n} \alpha_i \tag{3.10}$$

Here, L_1 is minimized with respect to *w, b* and the derivatives of the L with respect to all α_i. Setting the gradient of L_1 with respect to w and b to zero gives:

$$w = \sum_i \alpha_i y_i x_i \tag{3.11}$$

$$\sum_i \alpha_i y_i = 0 \tag{3.12}$$

If these equations are substituted into 3.10, equation 3.13 is obtained [5]:

$$L_2 = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{3.13}$$

Here, the aim is maximizing L_2 with respect to α_i, subject to constraint 3.12 and $\alpha_i \geq 0$. Support vectors are the points which satisfy $\alpha_i > 0$ and lie on hyperplane H_1 or H_2. Furthermore, the Karush-Kuhn-Tucker(KKT) theorem has an important role for α, w and b in SVM. In other words, we can say that solving the SVM problem is also solving the KKT conditions.

Although w can be found by training procedure, b cannot. If the algorithm for separable data is used for non-separable data, there will be no solution. Positive slack variables ξ_i are introduced in order to solve this problem for equations 3.7 and 3.8 which become:

$$\mathbf{x}_i.\mathbf{w} + b \geq +1 - \xi_i, y_i = +1 \tag{3.14}$$

$$\mathbf{x}_i.\mathbf{w} + b \leq -1 + \xi_i, y_i = -1 \tag{3.15}$$

$$\xi_i \geq 0, \forall i \tag{3.16}$$

$\sum_i \xi_i$ is an upper bound on the number of training errors. Hence the objective function is minimized from $\frac{\|w\|^2}{2}$ to $\frac{\|w\|^2}{2} + C(\sum_i \xi_i)^k$ where C is a parameter determined by the user. Thus the aim of the support vector machine is maximizing the dual problem:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i j_i \mathbf{x}_i \mathbf{x}_j \tag{3.17}$$

subject to:

$$0 \leq \alpha_i \leq C \tag{3.18}$$

$$\sum_i \alpha_i y_i = 0 \tag{3.19}$$

The solution is:

$$w = \sum_{i=1}^{N_S} \alpha_i y_i x_i \tag{3.20}$$

where N_S is the number of support vectors. As can be seen from equation 3.18, α_i has an upper bound C. This is the only difference from separable case. The primal Lagrangian is:

$$L_P = \frac{1}{2} \parallel w \parallel^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i (x_i.w + b) - 1 + \xi_i) - \sum_i \mu_i \xi_i \tag{3.21}$$

where μ_i are the Lagrange multipliers used for positivity of the ξ_i.

3.2.1.2 Nonlinear Suppor Vector Machines

If the hyperplane is non-linear, the data space is mapped into another Euclidean space H, called the *feature space*. Φ is the mapping which is shown in Figure 3.7.

$$\Phi : R^n \mapsto H \tag{3.22}$$

There is a kernel function which can be substituted with $x_i x_j$ in the training algorithm because

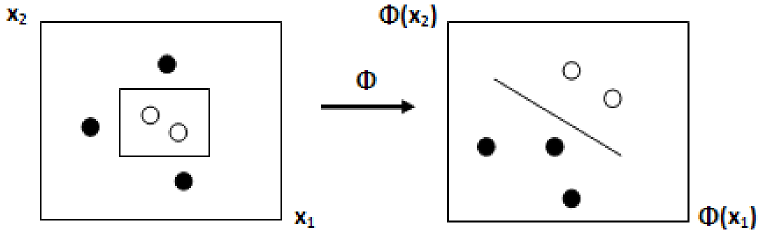


Figure 3.7: Mapping training data into high-dimensional featuring space via Φ

$x_i.x_j$ are in the form of dot products in equations 3.13 and 3.12. All the equations for linear machines are valid for non-linear machines in a different space.

$$K(x_i, x_j) = \Phi(x_i)\Phi(x_j) \tag{3.23}$$

Thus, an SVM is used by computing:

$$f(x) = \sum_{i=1}^{N_S} \alpha_i y_i \Phi(s_i).\Phi(x) + b = \sum_{i=1}^{N_S} \alpha_i y_i K(s_i, x) + b \tag{3.24}$$

where s_i are support vectors, N_S is the number of support vectors and $\Phi(x)$ is replaced by $K(s_i, x)$. Some kernel functions mostly used are:

- Linear : $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)$
- Radial Basis Function : $K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$
- Sigmoid : $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)$
- Polynomial : $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d$

where $\gamma > 0$, r and d are the kernel parameters.

3.2.2 Support Vector Regression (SVR)

In regression, the input and output is in m-dimensional such as $< (x_1, y_1), (x_2, y_2), ..., (x_m, y_m) >$ where $x_i \in \Re^m$ and $y_i \in \Re$. The aim is to find a function f(x) (eq.3.25) having at most ε error that is ε deviation from target y_i.

$$f(x) = w.x + b \tag{3.25}$$

$\frac{1}{2} \parallel w \parallel^2$ is minimized subject to 3.26 and 3.27.

$$y_i - w.x_i - b \leq \varepsilon \tag{3.26}$$

$$w.x_i + b - y_i \leq \varepsilon \tag{3.27}$$

f function does not always find all (x_i, y_i) pairs with ε error. It is sometimes allowed for some errors. So, soft margin loss function found by Bennett and Mangarasian is used in SVMs by Cortes and Vapnik. In this function, slack variables ξ_i, ξ_i^* is used for infeasible constraints. Hence, he goal is minimizing equation 3.28 subject to 3.29, 3.30 and 3.31:

$$\frac{1}{2} \parallel w \parallel^2 + \sum_{i=1} m(\xi_i + \xi_i^*) \tag{3.28}$$

$$y_i - w.x_i - b \leq \varepsilon + \xi_i \tag{3.29}$$

$$w.x_i + b - y_i \leq \varepsilon + \xi_i^* \tag{3.30}$$

$$\xi_i, \xi_i^* \geq 0 \tag{3.31}$$

where constant C is a trade-off between flatness of function f and allows deviations larger than ε. This is called ε-intensive loss function $|\xi|_\varepsilon$ which is defined in 3.32.

$$|\xi|_\varepsilon = \begin{cases} 0 & if\ |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{3.32}$$

An ε-tube which is shown by grey area in Figure 3.8 is defined in equation 3.32. If loss is zero, the predicted value is in the ε-tube. Additionally if the predicted value is not in ε-tube, the difference between predicted value and radius of the ε-tube gives the loss. A Lagrange

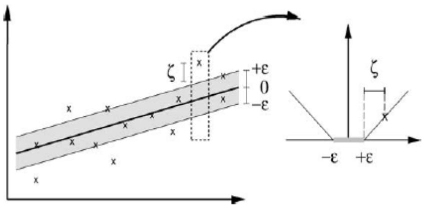


Figure 3.8: The soft margin loss, ε-tube and slack variables for a linear SVM [4]

function having a saddle point is obtained from 3.28 :

$$\begin{aligned} L &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ &\quad -\sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + \mathbf{w}\cdot\mathbf{x}_i + b) \\ &\quad -\sum_{i=1}^{n}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \mathbf{w}\cdot\mathbf{x}_i - b) \end{aligned} \tag{3.33}$$

where α_i,α_i^*,η_i,$\eta_i^* \geq 0$ are Lagrange multipliers. The saddle point is computed by using partial derivatives of L with respect to w, b, ξ_i and ξ_i^*:

$$\frac{dL}{db} = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \tag{3.34}$$

$$\frac{dL}{dw} = w - \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)x_i = 0 \tag{3.35}$$

$$\frac{dL}{d\xi_i} = C - \alpha_i - \eta_i \tag{3.36}$$

$$\frac{dL}{d\xi_i^*} = C - \alpha_i^* - \eta_i^* \tag{3.37}$$

The equation 3.38 is acquired by substituting 3.34, 3.35, 3.36 and 3.37 to 3.33:

$$\begin{aligned} L_d(\alpha_i, \alpha_i^*) &= -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\quad -\varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i \end{aligned} \tag{3.38}$$

subject to $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$ and $0 \leq \alpha_i\alpha_i^* \leq C$. η_i and η_i^* is eliminated by using equation 3.36 and 3.37 and equation 3.36 and 3.37 is rewritten as $\eta_i^{(}*) = C - \alpha_i^{(}*)$. Equation 3.35 transforms into $w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i$. And finally 3.25 becomes:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x \cdot x_i + b \tag{3.39}$$

The number of support vectors are not dependent the dimension of input vector. Moreover, b is computed by using KKT conditions:

$$\alpha_i(w \cdot x_i + b - y_i + \varepsilon + \xi_i) = 0 \tag{3.40}$$

$$\alpha_i^*(-w \cdot x_i - b + y_i + \varepsilon + \xi_i^*) = 0 \tag{3.41}$$

$$\eta_i\xi_i = (C - \alpha_i)\xi_i = 0 \tag{3.42}$$

$$\eta_i^*\xi_i^* = (C - \alpha_i^*)\xi_i^* = 0 \tag{3.43}$$

We can make some conclusions through KKT conditions. If $\alpha_i^* = C$, $(x_i, y_i) isoutsidethe \varepsilon$-tube. Besides that $\alpha_i \alpha_i^* = 0$ where $\alpha_i \neq 0$ and $\alpha_i^* \neq 0$.

For nonlinear case, input vector x_i is mapped onto a feature space by using $\Phi : \Re^m \mapsto \Re^f$. However, this mapping procedure is hard to verify. So kernels are used such as $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. Hence, the goal is maximizing:

$$\begin{aligned} L &= -\frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) \\ &\quad -\varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) \end{aligned} \tag{3.44}$$

subject to $\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$ and $0 \leq \alpha_i, \alpha_i^* \leq C$. We can rewrite equation 3.39 as:

$$w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \Phi(x_i) \tag{3.45}$$

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b \tag{3.46}$$

The difference of non-linear case from linear case is trying to find flattest function in input space in linear case when trying to find in feature space in non-linear case.

3.2.3 Partial Least Squares (PLS)

PLS is used in areas such as chemistry, industrial process control etc. PLS was firstly developed by Herman Wold in the late sixties in order to use in econometrics [11]. It is started to use in chemistry in the late seventies by S. Wold and H. Martin.

PLS is better than classical multiple linear regression (MLR) and principal component regression (PCR) and combines the features of these two regression methods. MLR is efficient when the number of dependent variables is few and these variables are not redundant. If the number of dependent variables is many for predicting and dependent variables are collinear, PLS is efficient. If the number of dependent variables is too many, there may be some latent factors. The aim in PLS is to find these latent factors.

When the number of independent variables are too many, regression is not feasible. One method is to reduce the number of independent variables by using some stepwise methods such as principal component regression. However, when the dimension of X is reduced, it does not guarantee that this reduced X can predict vector Y successfully.

The aim of PLS is to predict dependent variables, vector Y, from independent variables, matrix X. The independent variables are shown as:

$$X = TP^T$$

with $TT^T = I$ where I identifies the identity matrix. T is called *score matrix* and P is called *loading matrix*. Predicted Y is shown as:

$$\hat{Y} = TBC^T$$

where B and the columns of T identify regression weights and latent vectors respectively.

CHAPTER 4

DATASET AND MODELLING MOLECULAR SURFACES

4.1 Dataset and Data Preparation

38 phenncyclidine (PCP) drug-like compounds are used as dataset in the experiments [1]. A drug-like compound has NMDA (N-methyl-D-aspartic) receptor binding affinity value K_i which is shown in Figure 4.1. Compounds are represented on Cartesian coordinate system by using structural formulas of compounds in [1] by using *HyperChem* and *VegaZZ* programs. The steps performed in [1] are:

- The rough two-dimensional structure of molecules are obtained using *HyperChem* program [8].
- Molecular stability is obtained by MM+ molecular mechanics method. Molecular mechanics method is used to model molecular systems by Newtonian mechanics.
- Polak-Ribiere method is used to obtain the geometric optimization of the compounds.

The structure of molecules are obtained in [1] as mentioned above. We use these structures to obtain molecular lipophilicity surfaces (MLP) of each molecules by using *VegaZZ* program [15] in this study. We choose number of points per unit area is 5 for molecular lipophilicity whose average number of points is 411. The number of points on each molecular lipophilicity surfaces varies between 380 and 550. Since the number of points on each molecule surfaces is different from each other, we cannot use these points directly in prediction. Moreover, if the dependent variables are too many, it will be hard to predict independent variables. We decide to reduce the number of points on each molecule surfaces because of these reasons.

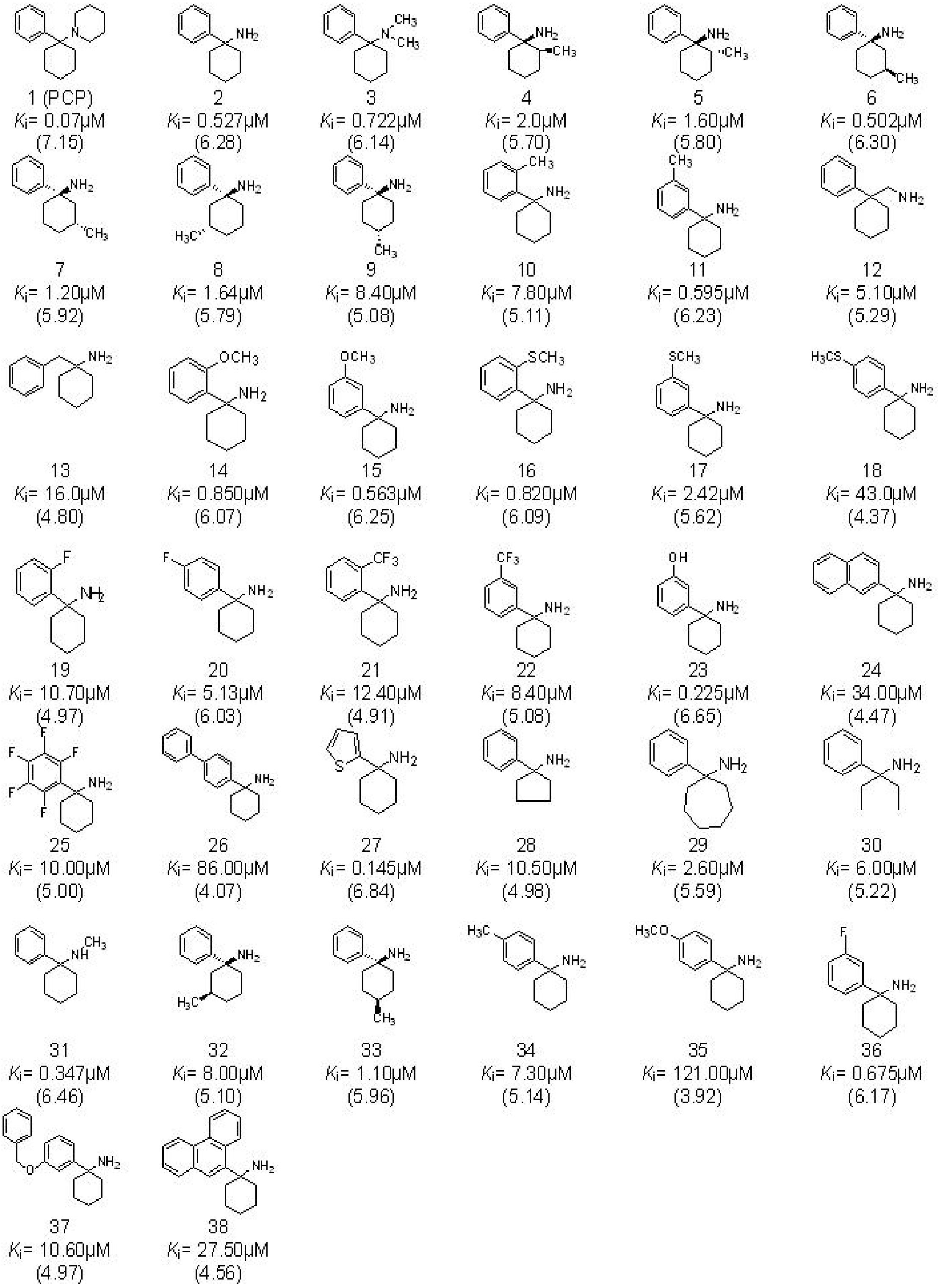


Figure 4.1: The chemical formulas of PCP-like compounds with their NMDA binding values K_i. The values in parenthesis express the $\log(1/K_i)$

4.2 The Modeling Architecture

Self organizing map is a useful method for analysis and visualization of high dimensional data. Each unit on the map has a three dimensional weight vector.

In this study, we used SOM toolbox developed by Vesanto [9].

4.3 The Learning Algorithm

There are two SOM algorithms which are batch and sequential algorithms in the SOM toolbox [9]. We used both of these algorithms for our implementation. And also, we used k-means clustering for modeling our dataset. In this chapter, we will explain batch and sequential SOM algorithms. K-means clustering algorithm that we used is the algorithm explained in section 2.1.2.

4.3.1 The Batch Map

Since Voronoi sets are used in the batch algorithm, we will explain *vector quantization (VQ)*, *Voronoi tessellation* and *Voronoi sets* in this section.

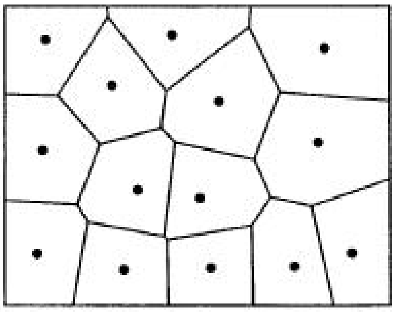


Figure 4.2: Voronoi tessellation partitions in two-dimensional space. The points show the codebooks of each Voronoi tessellation and all vectors in the same tessellation are the nearest vectors to the codebook of the tessellation [6]

Vector quantization ”forms a quantized approximation to the distribution of the input data vectors $x \in \Re^n$, using a finite number of so-called *codebook vectors* $m_i \in \Re^n$, $i=1,2,...,k$” [6].

In *vector quantization*, the aim is to find closest codebook to each input data vector by using Euclidean distance as given in equation 4.1:

$$\| x - m_c \| = \min_i \| x - m_i \| \tag{4.1}$$

And also, optimal selection of m_i minimizes the quantization error in the equation 4.2.

$$E = \int \| x - m_c \|^2 \, p(x) dx \tag{4.2}$$

where *p(x)* is the probability density function of *x*.

Voronoi tessellation is useful for the illustration of the vector quantization methods. In figure 4.2, the codebooks are shown as points in two-dimensional space. This space is partitioned into regions by lines. Each region includes the nearest vectors to the codebook of the corresponding codebook. These lines constitute the Voronoi tessellation.

Voronoi set is all input points assigned to a Voronoi tessellation.

In the light of all the information given above, the learning algorithm of the batch algorithm is given as:

1. The weight of each unit $w^i = [w^i_1, w^i_2, w^i_3]$ is initialized randomly where *i=1,2,...,M*. Here, M is the number of units of the map.

2. At each iteration,

   (a) Calculate the Euclidean distance between each input point ($x^p = [x^p_1, x^p_2, x^p_3]$) and the weights of all neurons by equation 4.3.

   $$d(x^p, w^i) = \sum_{k=1}^{3} (x^p_k - w^i_k)^2 \tag{4.3}$$

   where *i=1,2,...,M*. *M* is the number of units. And the unit with minimum distance is chosen. Thus, all of the input vectors are assigned to the unit with minimum distance and Voronoi sets are obtained.

   (b) Each weight vector and its neighbors are updated by equation 4.4:

   $$w_i(t+1) = \frac{\sum_{j=1}^{n} h_{ic(j)}(t) x_j}{\sum_{j}^{n} h_{ic(j)}(t)} \tag{4.4}$$

   where c(j) is the best match unit of sample vector x_j, $h_{i,c(j)}$ is the neighborhood function, and n is the number of sample vectors in the Voronoi set V_i. In equation 4.4, we used Gaussian function as neighborhood function.

3. Repeat step 2 until maximum number of iterations is reached.

The batch algorithm is faster to calculate than sequential algorithm. Also, the results of batch algorithm are as good as or better than the results of sequential algorithm.

4.3.2 The Sequential Map

The learning algorithm of the sequential map is:

1. Initialize weight vectors $w_i = [w^i_1, w^i_2, w^i_3]$ where *i=1,2,...,M* and *M* is the number of units on the map.
2. At each iterations,
   (a) A point $x^p = [x^p_1, x^p_2, x^p_3]$ is selected randomly from input data set where *p=1,2,...,N* and *N* is the number of input points.
   (b) Calculate Euclidean distance between the selected point and the weights of all units by equation 4.3.
   (c) The best matching unit(BMU), the unit with minimum distance, is chosen.
   (d) The weight of the BMU and its neighbors are updated by equation 4.5:

$$w^i(t+1) = w^i(t) + \alpha(t)h^i(r,s)(x^p - w^i(t)) \tag{4.5}$$

   where $t = 1,2,...T$ and T is the number of iterations. r is the neighborhood radius which shrinks with time. $\alpha(t) = 0.5(1-\frac{t}{T})$ is the learning factor. $h^i(r) = exp(\frac{-d(w^i,w^c)}{2r(t)})$ is the neighborhood function which also shrinks with time.

3. Repeat step 2 until maximum number of iterations is reached.

4.4 Performance

The resulting map changes with respect to training vectors and the learning rate. The optimal map is obtained for the same input data. Quantization error is useful in order to compare maps with the same array structure. So, quantization error and topographic error are used for performance measurement in this thesis.

The best-matching unit is the weight vector with minimum distance to the input vector. The quantization error is the distance between the input vector and the best-matching unit. It is calculated as:

$$qe = \frac{\sum_{p=1}^{N} d(x^p, w^{pc})}{N} \tag{4.6}$$

where N is the number of input points, x^p is the input vector and w^{pc} is the weight vector of the best-matching unit of input x^p.

Self organizing map is used to map the input vector onto a low dimensional map in a topologically ordered manner. Topographic error is used in order to measure topological preservation. It indicates whether closest and second closest weight vector to the input vector are neighbor or not. It is calculated as:

$$te = \frac{\sum_{s=1}^{N} te^s}{N} \tag{4.7}$$

where $te^s = 0$ if these two weight vectors are neighbor, $te^s = 1$, otherwise.

4.5 Implementation

In this study, each molecule is on three-dimensional coordinate system and each point on the molecule has a molecular lipophilicity value. The number of points of each molecular lipophilicty surface varies between 380 and 600. We use two-dimensional SOM and k-means clustering to reduce the number of points in the input data set. We use two-dimensional SOM with 40 neurons and k=40 for k-means clustering. We have 38 molecules and after reducing the number of input data set, we will predict the binding affinity of each molecule by using these selected 40 values. In regression, if the number of independent variables is much greater than the dimension of the dependent variables, the prediction will be inefficient. Since there are 38 molecules to predict the binding affinity, we use SOM with 40 neurons. Thus, the regression will not be inefficient.

According to the ”One-Half Rule” [6], the length of short side should be at least half of the longer side of the SOM. So, we use 8x5 SOM.

In SOM toolbox [9], there are two training phases, rough and fine tuning phases, which are trained respectively. In rough phase, large neighborhood radius and big initial value for learning coefficient are used. In fine tuning phase, small neighborhood radius is used and learning coefficient is small already at the beginning.

In *batch algorithm*, the weights of units are initialized randomly. At each iteration, all of the input points are mapped onto a unit with minimum distance. The number of iterations should be at least 500 times the number of units. So, the number of iterations for *batch algorithm* is 20000 for both rough and fine tuning phases. However, the map converges before this number of iterations is reached. Of course, the points in each Voronoi set change with respect to initialization values of weights of the units.

The performance is measured by using quantization error and topographic error mentioned in section 3.4. In figure 4.3, quantization error and topographic error for molecular lipophilicity surfaces are shown.

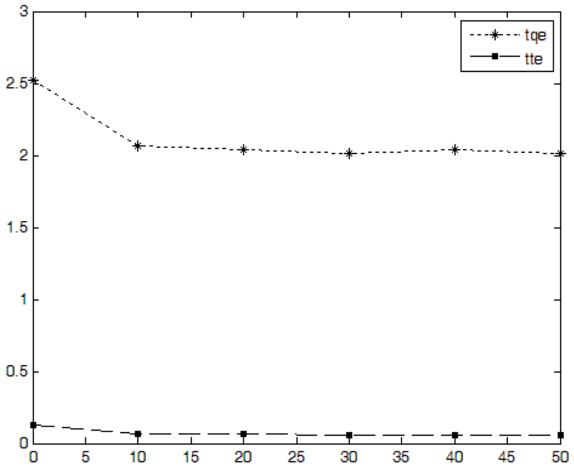


Figure 4.3: Total quantization error(tqe) and total topographic error(tte)

In *sequential algorithm*, each molecule is divided into 40 regions. Each region consists of points that are close to each other. Our aim is to select a point from each region. The weights of units are initialized randomly. However, one more point is selected from some regions and no point is selected from some regions. For instance, no point is selected from 10 regions for molecule 1. This varies between 10 and 15. When we initialize the weights of units by selecting one point from each region, the result does not change. So, we cannot represent our data set by *sequential map* with 40 units and we decided to use a SOM with more units. Thus, the probability of selection a point from each region will be higher. The average number of points of all molecules is 411. We compute the optimal size of SOM by equation 4.8.

$$msize = 5\sqrt{k} \tag{4.8}$$

where *k* is the product of the columns and rows of the dataset. The size is calculated as 170 by equation 4.8. We use a 17x10 SOM with respect to ”One-Half Rule”. We calculated standard deviation of each unit by using the weights of the units. We thought that if the standard deviations of two units were close to each other, the best matching units (BMUs) of these units would be close to each other. So, standard deviations are grouped in fives whose standard deviations are close to each other and a unit is chosen from each group. There are 40 groups. For instance, no point is selected from 3 regions for molecule 1. Of course, this result is better than the SOM with 40 units.

In *k-means clustering*, k=40 and the number of iterations is 100. Actually, *k-means clustering algorithm* is similar to *batch map algorithm*.

CHAPTER 5

EXPERIMENTAL RESULTS

Our aim is to predict binding affinity of molecules which are given in Figure 3.1 by using molecular lipophilicity properties of molecules. We use support vector regression(SVR) and partial least Squares(PLS) for this purpose. We chose Radial Basis Function(RBF) kernel and ϵ-loss function for SVR. We try to approximate the number of components for PLS.

5.1 Data Set

Molecular lipophilicity of each molecule given in Figure 3.1 are mapped onto a two-dimensional SOM and are clustered by *k-means clustering*. Self-organizing map has 40 neurons which are assumed as features of molecular lipophilicity. We only used the molecular lipophilicity values which are mapped onto a neuron. So, a molecule has 40 features for molecular lipophilicity property. Eventually, we have a data set having 38 samples with 40 features for molecular lipophilicity.

5.2 Software

LIBSVM 3.0 [32] which is developed by Chang and Lin is used for Support Vector Regression. It is performed on Linux platform. Unscrambler 9.5 [14] is used for Partial Least Squares(PLS) regression which is performed on Windows XP platform. C++ and MATLAB is used for achieving prediction results of SVR and PLS, and other calculations.

5.3 Performance

Root mean squared error(RMSE) and coefficient of determination(R^2) are used for performance measurement.

RMSE is used to measure difference between the predicted value evaluated by an estimator and observed value. This is used in error measurement of the system. RMSE is calculated by 5.1.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{5.1}$$

where y_i is observed value of i^th variable, $\hat{y}_i$ is the predicted value and N is the number of samples.

R^2 is used to determine variance of a variable which is predicted from model. It shows how much predicted values are close to the observed values. R^2 is between 0 and 1. If R^2 is close to 1, the system will be explained by the model well. Conversely, if R^2 is close to 0, the system can not be explained well by the model. R^2 is calculated by

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y_i})^2} \tag{5.2}$$

where y_i and $\hat{y}_i$ are the same as given above and $\overline{y}_i$ is the mean of the observed values.

5.4 Assigning SVR Parameters

Setting the appropriate SVR parameters is an important issue in order to provide good prediction performance. C, ϵ and kernel parameter γ are the optimal parameters of SVR. C is the trade off between the model complexity and error tolerance. Very large C causes to minimize error with regard to model complexity. ϵ is width of ϵ-intensive zone and affect the number of support vector of the training data. If ϵ is large, there are less support vectors which causes less complex prediction. γ is the width of kernel function which represents the distribution of independent values in the training data.

We try to assign optimal set of SVR parameters using two different methods. In first one which we call it as SVR1 in the following sections, is suggested in [2]. In this approach, C is selected directly obtained from training data and ϵ is selected by using noise in the training data and number of training examples. C is calculated by equation 5.3:

$$C = max(\overline{y} + 3\sigma_y, \overline{y} - 3\sigma_y) \tag{5.3}$$

where $\overline{y}$ and σ_y is the mean and standard deviation of the observed values of the training data respectively. ϵ the proportion of noise level and the number of samples. Since, it depends on the number of samples, if the number of samples is many, ϵ will be small. ϵ is is calculated by equation 5.4 if the number of samples is less than 30:

$$\epsilon = \frac{\sigma}{\sqrt{N}} \tag{5.4}$$

If the number of samples is greater than or equal to 30, ϵ is calculated by equation 5.5:

$$\epsilon = 3\sigma_y \sqrt{\frac{\ln N}{N}} \tag{5.5}$$

where N is the number of the samples. Since, we have 38 samples in the dataset, we use equation 5.5.

According to our data C = 7.8477 and ϵ = 0.7176. Leave-n-out cross validation and these calculated values are used to calculate γ. Firstly, the dataset is divided into 5 subset randomly. One of these five set is used as test set and rest of them are used as training set. This is repeated for each subset and *RMSE* is calculated for each test set. Finally, total *RMSE* values of test sets is calculated. γ is tested for values between 0.01 and 6 with 0.01 interval where C and ϵ are the values given above. γ value which gives minimum *RMSE* value is selected.

In second approach of parameter selection which is called SVR2 in the following sections is grid-search method. Leave-n-out cross validation is used in this approach, either. The aim is assigning set of SVR parameters which gives minimum *RMSE* result. Grid-search method is applied after determining ranges of C, ϵ and γ. 8(γ = [0.01,6], ϵ = [0.05,0.2], C = [1,600]) [10].

5.5 Evaluating Results

We model molecular surfaces by using two-dimensional SOM and *k-means clustering*.

After selection of SVR parameters and the number of components for PLS, training and test sets are defined. Because we have few samples, we generate 1000 training and test sets randomly which have 30 samples for each training set and 8 samples for each test set. We use these 1000 random training and test sets to predict binding affinity of the molecules by using molecular lipophilicity. We choose three of these random sets with minimum RMSE of SVR1, SVR2.

5.5.1 Batch Map Algorithm

For *batch map*, the γ parameter of SVR1 is 0.52. The parameter set of SVR2 is $\gamma = 0.28$, $\epsilon = 0.2$ and C = 1. Before the implementation, the data is scaled between -1 and 1 because some values of features are close to zero. This situation causes the values of features are rounded up during calculation. Training and test results of SVR1 and SVR2 can be seen in Table 5.1.

Table 5.1: $RMSE$ values of SVR1 and SVR2 for batch map algorithm

	$RMSE$			
	SVR1		SVR2	
Selection	Training	Test	Training	Test
Random-177	**0.257556**	**0.395728**	**0.23667**	**0.38883**
Random-712	0.267031	0.39711	0.142961	0.416504
Random-421	0.256832	0.407607	0.196094	0.404565

Random-177 gives lowest results for testing for both SVR1 and SVR2. Moreover, SVR2 training and test results are better than SVR2. Therefore, we choose Random-177 to compare the performance of SVR1, SVR2 and PLS.

PLS values of training and testing are 0.550664 and 0.395946 for Random-177 set, respectively. The overall statistical results can be seen in table 5.2. SVR2 gives better results than both SVR1 and SVR2. SVR2 has the lowest RMSE and highest R^2. As can be seen from Table 5.1, SVR2 has % 62 of predicted values are fitted to the observed values. The R^2 value of SVR2 shows that predicted values does not fit and does not explain variability in binding affinity very well. Also, the difference between SVR1 and SVR2 is not much.

In Figure 5.1, the correlation between observed and predicted values is shown for SVR1,

Table 5.2: $RMSE$ and R^2 values of SVR1, SVR2 and PLS for batch map algorithm

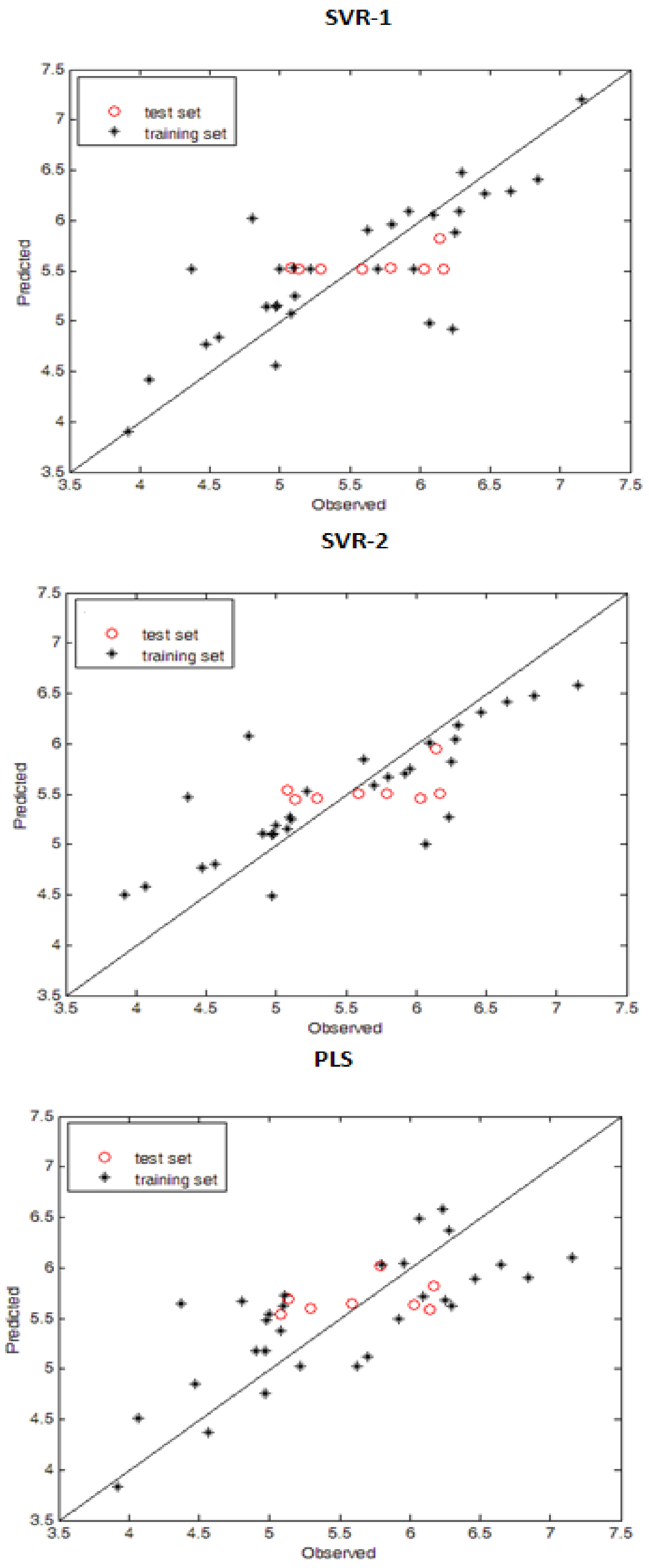
	SVR1	SVR2	PLS
$RMSE$	0.487939	0.468075	0.521917
R^2	0.590853	0.623486	0.531885

SVR2 and PLS. The error between observed and predicted values are represented by the horizantal distance between the points and the line. If the point is close to the line, the error will be small. For SVR1, test and training set values are not too close to the line. The results of SVR2 are very similiar to the results of SVR1. The test results of PLS are worst than SVR1 and SVR2. As a result, RMSE and R^2 values of SVR1, SVR2 and PLS are sometimes misleading. The training and test sets of Random-265 dataset are shown in Table 5.4 and Table 5.3, respectively.

Table 5.3: Observed and predicted values for test set for batch map algorithm

	Observed	Predicted		
No.		SVR1	SVR2	PLS
3	6.14	5.82466	5.94611	5.5906
8	5.79	5.5248	5.50959	6.015
9	5.08	5.53152	5.54403	5.5426
12	5.29	5.51334	5.4539	5.6017
20	6.03	5.51435	5.46146	5.6314
29	5.59	5.51743	5.50923	5.6432
34	5.14	5.51326	5.44979	5.6919
36	6.17	5.51845	5.50003	5.8175

The average RMSE of 1000 random test set is 0.7424 and 0.7431 for SVR1 and SVR2 respectively. The average RMSE of 1000 random test sets are very close to each other and is high for both SVR1 and SVR2 because RMSE values of most test sets are high. Even the RMSE values of RMSE is sometimes greater than 1. The average RMSE of training tests are 0.2756 and 0.1933 for SVR1 and SVR2, respectively. SVR2 is explained our dataset better than SVR1 for training sets. Of course, the average RMSE of the training sets is much better than the average RMSE of the test sets.



SVR-1

SVR-2

PLS

Figure 5.1: Predicted vs. Observed values for Batch Map Algorithm

Table 5.4: Observed and predicted values for training set for batch map algorithm

	Observed	Predicted		
No.		SVR1	SVR2	PLS
1	7.15	7.20432	6.58104	6.1038
2	6.28	6.09016	6.04722	6.3745
4	5.7	5.51439	5.58302	5.1231
5	5.8	5.9646	5.66832	6.0305
6	6.3	6.4758	6.18375	5.6177
7	5.92	6.09144	5.70933	5.4935
10	5.11	5.25079	5.24416	5.7257
11	6.23	4.92052	5.27508	6.583
13	4.8	6.02471	6.07831	5.6717
14	6.07	4.97789	4.99874	6.4831
15	6.25	5.88127	5.82701	5.6772
16	6.09	6.0566	6.01237	5.7158
17	5.62	5.90005	5.85066	5.0261
18	4.37	5.51632	5.46708	5.6455
19	4.97	4.56126	4.49339	4.7623
21	4.91	5.14079	5.10447	5.1779
22	5.08	5.0761	5.16076	5.3719
23	6.65	6.28697	6.41361	6.0266
24	4.47	4.77447	4.76549	4.8502
25	5	5.51636	5.19401	5.538
26	4.07	4.41444	4.5852	4.5146
27	6.84	6.40066	6.47669	5.9018
28	4.98	5.15144	5.11008	5.479
30	5.22	5.52124	5.52406	5.0256
31	6.46	6.2656	6.30831	5.8903
32	5.1	5.53389	5.27415	5.6227
33	5.96	5.5212	5.75243	6.0445
35	3.92	3.89789	4.5012	3.8339
37	4.97	5.14574	5.09933	5.1727
38	4.56	4.83905	4.80983	4.3669

5.5.2 Sequential Map

Since *Sequential map* for SOM with 40 neurons cannot select a point from each region which we divide molecular lipophilicity surface of each molecule into 40 regions, we did not predict for this map. Instead of this map, we made prediction for SOM with 170 neurons. The γ parameter of SVR1 is 0.14. The parameter set of SVR2 is $\gamma = 0.22$, $\epsilon = 0.1$ and C = 1. The results of SVR1 and SVR are shown in table 5.5.

Table 5.5: $RMSE$ values of SVR1 and SVR2 for sequential algorithm

	$RMSE$			
	SVR1		SVR2	
Selection	Training	Test	Training	Test
Random-223	0.270489	0.418183	0.259652	0.41087
Random-265	**0.246853**	**0.403818**	**0.212593**	**0.396236**
Random-712	0.254544	0.404062	0.235578	0.429875

Random-265 has lowest test results for SVR1 and SVR2. SVR2 gives better results than SVR1 for both training and test. Therefore we choose Random-265 to compare the performance of SVR1, SVR2 and PLS.

PLS values of training and test are 0.756899 and 0.626323 for Random-265, respectively. And the overall statistical results can be seen in table 5.6. SVR1 gives better results than SVR2 and PLS. So, SVR1 has the lowest RMSE and the highest R^2 values. The R^2 value of SVR1 is 0.87. So, our dataset is not explained by SVR1 perfectly.

Table 5.6: $RMSE$ and R^2 values of SVR1, SVR2 and PLS for sequential algorithm

	SVR1	SVR2	PLS
$RMSE$	0.268602	0.341272	0.731349
R^2	0.876016	0.799852	0.080822

In Figure 5.2, the correlation between observed and predicted values can be seen for SVR1, SVR2 and PLS. The difference between the training and test points of SVR1 and SVR2 is not too much. The test points of PLS are worst than from SVR1 and SVR2 when the training points are not very different from SVR1 and SVR2. The training and test sets of Random-265 dataset are shown in Table 5.8 and Table 5.7, respectively.

Table 5.7: Observed and predicted values for test set for sequential algorithm

No.	Observed	Predicted		
		SVR1	SVR2	PLS
4	5.7	5.2895	5.43035	5.7694
5	5.8	5.83938	5.70145	6.1946
7	5.92	5.26819	5.45199	5.4231
25	5	5.53152	5.49607	5.6521
28	4.98	5.21735	5.44076	6.2495
29	5.59	5.40958	5.49648	5.6644
34	5.14	5.46646	5.49232	5.9357
36	6.17	5.68869	5.56725	5.9349

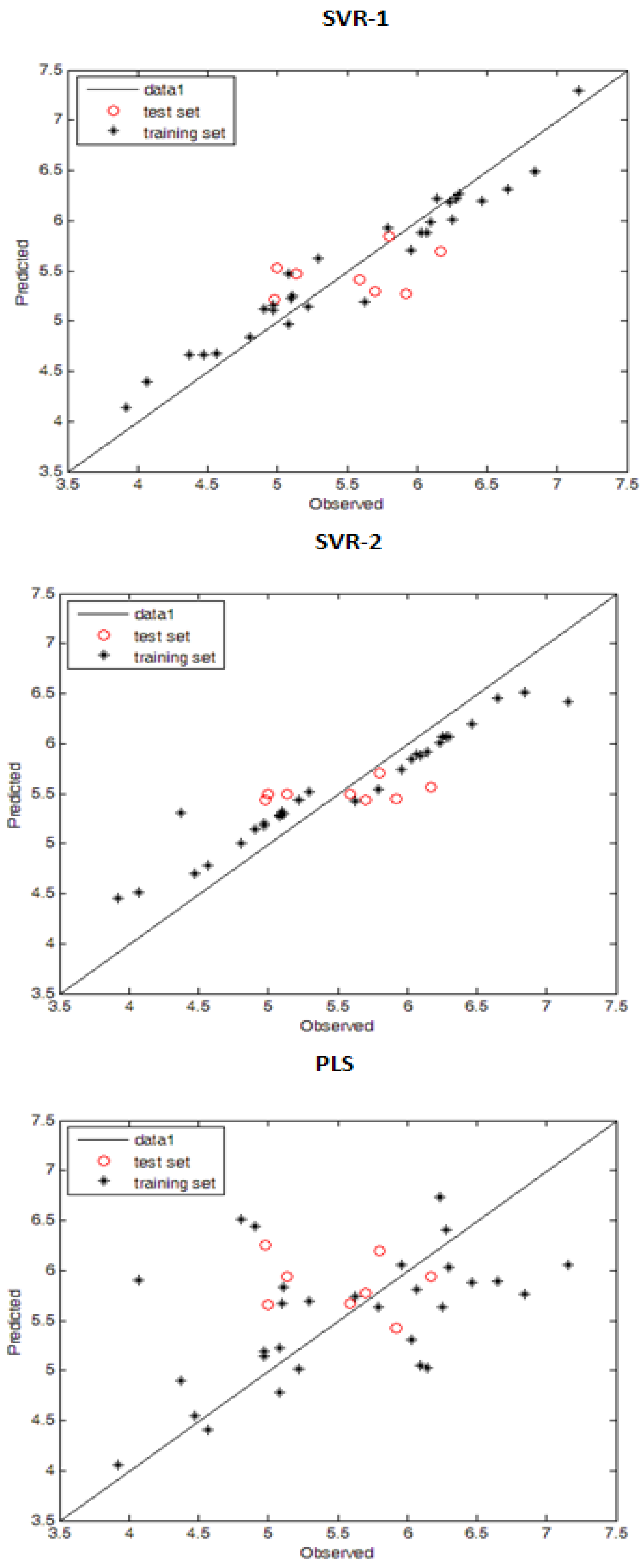
The average RMSE value of 1000 random test sets for SVR1 and SVR2 are 0.7833 and 0.7802, respectively. The results are not good most of the time as well. The average RMSE values of test sets are 0.2611 and 0.06312 for SVR1 and SVR2 respectively. SVR1 is explained our training sets much more better than SVR1 and also the average RMSE of training sets is much more better than the average RMSE of test sets.

5.5.3 K-means Clustering

In *k-means clustering*, the γ parameter of SVR1 is 0.19. The parameter set of SVR2 is γ = 0.18, ϵ = 0.08 and C = 2. The results of SVR1 and SVR2 can be seen from table 5.9.

Random-767 has the lowest test results for SVR1 and SVR2. When the test results of SVR1 are better than the test results of SVR2, the training results of SVR2 is better than SVR1's results. So, Random-767 is used to compare the performance of SVR1, SVR2 and PLS.

PLS values of training and test are 0.864482 and 0.640974 for Random-767, respectively. The overall statistical results can be seen in table 5.10. SVR2 outperforms both SVR2 and PLS. So, SVR1 has the lowest RMSE and the highest R^2. Moreover, our dataset can be explained well with respect to R^2.



SVR-1

SVR-2

PLS

Figure 5.2: Predicted vs. Observed values for Sequential Algorithm

Table 5.8: Observed and predicted values for training set for sequential algorithm

	Observed	Predicted		
No.		SVR1	SVR2	PLS
1	7.15	7.29051	6.41884	6.058
2	6.28	6.21798	6.06217	6.401
3	6.14	6.21702	5.91263	5.0266
6	6.3	6.2605	6.07263	6.0361
8	5.79	5.92612	5.54215	5.6361
9	5.08	4.96728	5.26908	5.2229
10	5.11	5.24813	5.2997	5.8373
11	6.23	6.18375	6.00488	6.7359
12	5.29	5.62817	5.51699	5.6973
13	4.8	4.84368	5.00037	6.5171
14	6.07	5.88437	5.88639	5.8107
15	6.25	6.00458	6.07218	5.6369
16	6.09	5.99067	5.88341	5.0505
17	5.62	5.19395	5.42044	5.7346
18	4.37	4.66239	5.31146	4.8998
19	4.97	5.10612	5.20215	5.1486
20	6.03	5.87598	5.84464	5.3019
21	4.91	5.12182	5.14771	6.4376
22	5.08	5.46623	5.28668	4.7753
23	6.65	6.31566	6.44936	5.8926
24	4.47	4.66954	4.69748	4.5416
26	4.07	4.39638	4.51655	5.9038
27	6.84	6.49023	6.50954	5.7602
30	5.22	5.14575	5.43274	5.0157
31	6.46	6.19365	6.19358	5.8754
32	5.1	5.22842	5.32347	5.6719
33	5.96	5.70537	5.74353	6.0582
35	3.92	4.13387	4.4484	4.0612
37	4.97	5.15026	5.17744	5.1895
38	4.56	4.67353	4.7863	4.4009

Table 5.9: $RMSE$ values of SVR1 and SVR2

	$RMSE$			
	SVR1		SVR2	
Selection	Training	Testing	Training	Testing
Random-712	0.27535	0.372461	0.0357069	0.363565
Random-767	**0.27535**	**0.344911**	**0.0357069**	**0.363565**
Random-265	0.263702	0.378848	0.155701	0.0267476

Table 5.10: $RMSE$ and R^2 values of SVR1, SVR2 and PLS

	SVR1	SVR2	PLS
$RMSE$	0.28731	0.169488	0.822491
R^2	0.858143	0.950634	-0.16255

In Figure 5.3, the correlation between observed and predicted values can be seen for SVR1, SVR2 and PLS. The training points of SVR2 is very close to the line. As can be seen, they are much better than the results of SVR1 and PLS. However, the test points of SVR1 and SVR2 are not very different. The test and training results of PLS are worst than SVR1 and SVR2. The training set of Random-767 is shown in Table 5.12 while the test set is shown in Table 5.11.

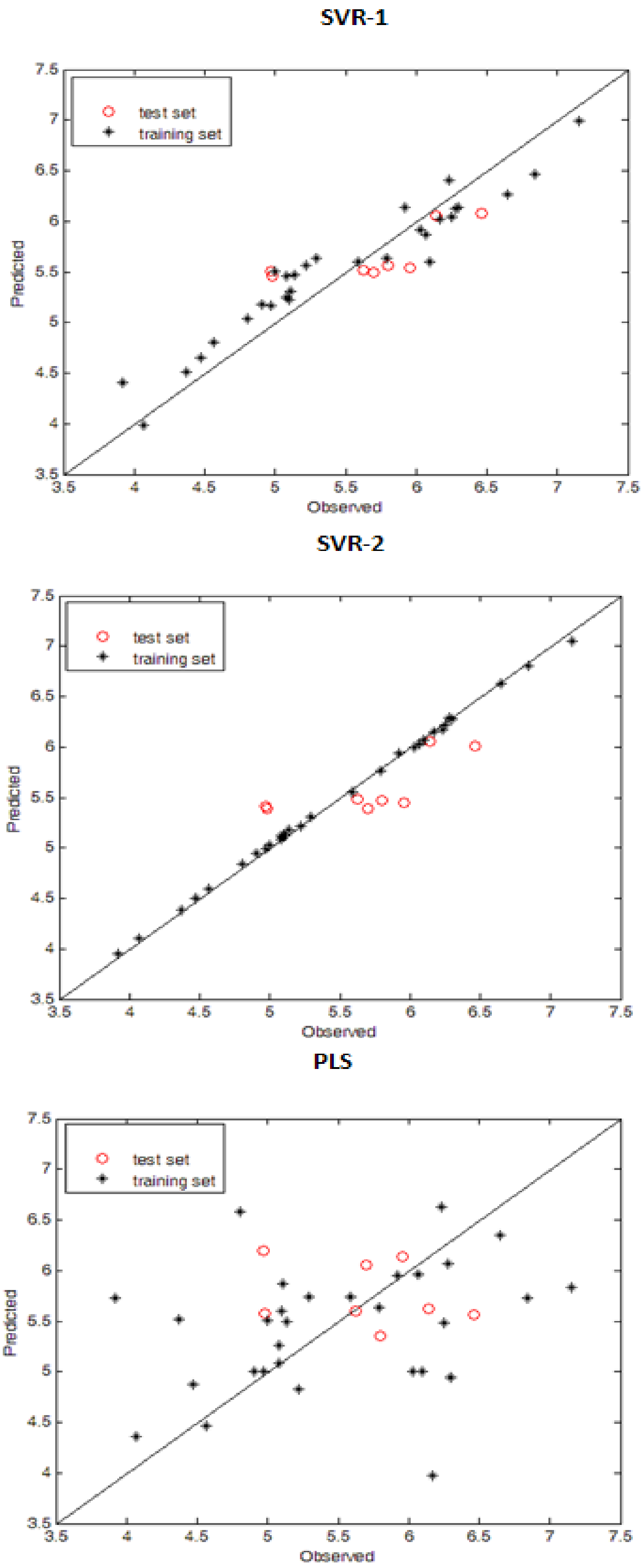
Table 5.11: Observed and predicted values for test set for k-means algorithm

	Observed	Predicted		
No.		SVR1	SVR2	PLS
3	6.14	6.05703	6.05627	5.73223
4	5.7	5.49338	5.3908	5.48782
5	5.8	5.5653	5.46852	5.14772
17	5.62	5.51393	5.47836	5.72177
19	4.97	5.50119	5.418	5.57676
28	4.98	5.46115	5.38369	5.7394
31	6.46	6.08089	6.00962	5.35908
33	5.96	5.53762	5.4517	5.5996

The average RMSE values of 1000 random test sets for SVR1 and SVR2 are 0.7383 and 0.8769, respectively. The average for SVR1 is better than the average for SVR2 for test tests. The average RMSE values of training sets are 0.2707 and 0.3146 for SVR1 and SVR2, respectively. The average of SVR1 and SVR2 are close to each other for training sets.

The average RMSE values of test sets for batch map, sequential and k-means algorithms are not good. Our dataset is too small so it is hard to predict the binding affinity of test molecules. If our training dataset is larger, the RMSE values may be lower.

Moreover, when we compare the SVR1, SVR2 and PLS for batch map, sequential and k-means algorithms, SVR outperforms PLS for all of the algorithms which are used in this



SVR-1

SVR-2

PLS

Figure 5.3: Predicted vs. Observed values for K-means Algorithm

Table 5.12: Observed and predicted values for training set for k-means algorithm

	Observed	Predicted		
No.		SVR1	SVR2	PLS
1	7.15	6.98629	7.05239	5.8387
2	6.28	6.12437	6.29231	6.0632
6	6.3	6.13674	6.27806	4.9427
7	5.92	6.13449	5.93924	5.9498
8	5.79	5.63138	5.76426	5.6379
9	5.08	5.25323	5.08487	5.261
10	5.11	5.31135	5.125	5.8646
11	6.23	6.40963	6.1741	6.6337
12	5.29	5.63481	5.30912	5.7424
13	4.8	5.03688	4.83625	6.5824
14	6.07	5.86961	6.03496	5.9669
15	6.25	6.0486	6.22296	5.4814
16	6.09	5.59905	6.06537	4.9997
18	4.37	4.50832	4.38231	5.5156
20	6.03	5.92001	5.99207	5.0055
21	4.91	5.1756	4.94264	5.0023
22	5.08	5.46039	5.11965	5.0836
23	6.65	6.26892	6.62384	6.3508
24	4.47	4.65051	4.50218	4.8746
25	5	5.50769	5.02547	5.5088
26	4.07	3.97999	4.10614	4.3559
27	6.84	6.45907	6.8096	5.7277
29	5.59	5.59707	5.55447	5.7348
30	5.22	5.5625	5.21107	4.8312
32	5.1	5.22021	5.11444	5.6021
34	5.14	5.4667	5.17907	5.4929
35	3.92	4.40924	3.95028	5.7297
36	6.17	6.01648	6.14407	3.9757
37	4.97	5.17135	4.98995	4.999
38	4.56	4.80376	4.59815	4.465

study. However, overfitting is not significantly observed by neither SVR nor PLS. So, best parameters cannot always be found in SVR.

CHAPTER 6

CONCLUSION

In this thesis, molecular lipophilicity surfaces are obtained in 3D coordinate system which are used to predict binding affinity of molecules. However, the number of points on these surfaces are different from each other and is too many in order to use for prediction directly. So, the number of points is reduced by two dimensional self organizing maps (SOM) and k-means clustering. The surfaces are mapped onto two-dimensional maps. SOM with 40 neurons is used for the purpose of feature extraction. The value of molecular lipophilicity of each neuron is used for prediction of binding affinity of molecules.

When we use two-dimensional SOM with 40 neurons and apply *batch map algorithm*, all points in the input data set are mapped onto a neuron. Thus, we acquire 40 Voronoi sets. We calculate the average molecular lipophilicity for each Voronoi set and use these values as features to predict binding affinity of molecules.

We divided each molecule into 40 regions for *sequential map algorithm.* Our aim is to map a point from each region onto units. Although *batch map algorithm* can model our molecules well, *sequential map algorithm* does not give good results. In *sequential map algorithm*, a point from each region cannot be mapped because of the random structure of the algorithm. So, we decided to use a SOM with 170 units. Although a point from each region cannot be mapped onto units in this approach, as well, the results are better than the first sequential algorithm approach.

In *k-means clustering*, due to the nature of the algorithm each molecule can be divided into 40 regions. We finally see that *batch map algorithm* of SOM and *k-means clustering* are reduced the number of input samples efficiently. The *sequential map algorithm* is not very reliable for our purpose.

When we look at the average RMSE values of test sets for batch map, sequential and k-means algorithm, they are not good. Even RMSE values of some test sets are greater than 1. It is clear that the models using SVR-1, SVR-2 and PLS are not always explained our dataset well. However, when we compare SVR-1, SVR-2 and PLS algorithms with each other, SVR outperforms the PLS.

In study [1], the prediction of binding affinity of molecules are done using molecular electrostatic potential. When we compare the results in [1] with our results, molecular hydrophobicity predicts the binding affinity of molecules better.

The molecular electrostatic potential is another property that affects the binding affinity of molecules. As future work, the effect of the molecular electrostatic potential and the molecular lipophilicity to the binding affinity can be predicted.

In prediction part, if other methods are used for finding SVR parameters, the prediction results may improve. Moreover, different regression methods or machine learning algorithms might be used to obtain better prediction results.

REFERENCES

[1] O. Erdas, ”Modeling and Predicting Binding Affinity of PCP-LIKE Compounds Using Machine Learning Method”, Master’s thesis, Middle East Technical University, 2007.

[2] Vladimir Cherkassky, Yunqian Ma, ”Practical selection of SVM parameters and noise estimation for SVM regression”, *Neural Networks*, vol. 17, pp. 113-126, 2004.

[3] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, B. Scholkopf, ”Support Vector Machines”, *Intelligent Systems and their Applications, IEEE*, vol.13, pp. 18-28, 1998.

[4] A. J. Smola, B. Schoölkorf, ”A tutorial on support vector regression”, *Statistics and Computing*, vol.14, pp. 199-222, 2004.

[5] C. J. C. Burges, ”A Tutorial on Support Vector Machines for Pattern Recognition”, *Data Mining and Knowledge Discovery*, vol.2, pp. 121-167, 1998.

[6] Teuvo Kohonen, *Self Organizing Maps*, New York: Springer, 2001.

[7] J. Wendel, B. P. Buttenfield, ”Formalizing Guidelines for Building Meaningful Self-Organizing Maps”.

[8] *HyperChem version 5.1* http://www.hyper.com, last accessed on 25/05/2011.

[9] *SOM Toolbox* http://www.cis.hut.fi/somtoolbox/, last accessed on 25/05/2011.

[10] B. Üstün, ”A comparison of support vector machines and partial least squares regression on spectral data”, Master’s thesis, Katholieke Universiteit Nijmegen, 2003.

[11] P. Geladi, B. R. Kowalski, ”Partial least squares regression: a tutorial”, *Analytica Chimica Acta*, vol.185, pp. 1-17, 1986.

[12] Jian Yanga, Jing-yu Yanga, D. Zhangb, J. Lua, ”Feature fusion: parallel strategy vs. serial strategy”, *Pattern Recognition*, vol.36, pp. 1369-1381, 2003.

[13] C. Liu, H. Wechsler, ”A Shape- and Texture-Based Enhanced Fisher Classifier for Face Recognition”, *IEEE Transaction on Image Processing*, vol.10, no.4, 2001.

[14] *Unscrambler Version 9.5* http//www.camo.com, last accessed on 25/05/2011.

[15] *VegaZZ Version 2.3.3* http://www.vegazz.net, last accessed on 25/05/2011.

[16] T.Kohonen, ”The Self-Organizing Map”, *Proceedings of the IEEE*, vol.79, no.9, 1990.

[17] H. Ritter, ”Self-Organizing Maps on non-euclidean Spaces”, *Kohonen Maps*, p. 97-108, 1999.

[18] *http://en.wikipedia.org*, last accessed on 25/05/2011.

[19] H. Abdi, ”Partial least squares(pls) regression”, *Encyclopedia of Social Sciences Research Methods* (M. Lewis-Beck, A. Bryman, T. Futing, eds.), pp.792-795, 2003.

[20] A. Neme, P. Miramontes, ”Statistical properties of lattices affect topographic error in self-organizing maps”, *ICANN 2005*, *LNCS 3996*, pp. 427-432, 2005.

[21] G. Schneider, ”Neural networks are useful tools for drug design”, *Neural Networks*, vol.13, pp. 15-16, 2000.

[22] T. Heskes, ”Self-organizing maps, vector quantization and mixture modeling”, *IEEE Transactions on neural networks*, vol. 12, no.6., 2001.

[23] A.Sangole, G. K. Knopf, ”Geometric representations for high-dimensional data using a spherical SOFM”, *Smart Engineering System Design*, vol.5 , pp. 11-20, 2003.

[24] X. Zhang, Y. L, ”Self-organizing map as a new method for clustering and data analysis”, *International Joint Conference on Neural Networks*, 1993.

[25] J. Vesanto, E. Alhoniemi, ”Clustering of the self-organizing map”, *IEEE Transactions on Neural Networks*, vol.11, no.3, 2000.

[26] S. Anzali, J. Gasteiger, U. Holzgrabe, J. Polanski, J. Sadowski, A. Teckentrup, M. Waganer, ”The use of self-organizing neural networks in drug design”, *Perspective in Drug Discovery and Design*, vol.2, pp. 273-299, 1998.

[27] J. Gasteiger, X. Li, C. Rudolph, J. Sadowski, J. Zupan, ”Representation of molecular electrostatic potentials by topological feature maps”, *Journal of the American Chemical Society* ,vol. 116, pp. 4608-4620, 1994.

[28] J. Vesanto, E. Alhoniemi, ”Clustering of the self-organizing maps”, *IEEE Transactions on Neural Networks*, vol. 11, no. 3, 2000.

[29] Y. S. Liu, Q. Li, G. Q. Zheng, K. Ramani, W. Benjamin, ”Using diffusion distances for flexible molecular shape comparison”, *BMC Bioinformatics*, vol. 11, 2010.

[30] Y. Lu, S. Lu, F. Fotouhi, ”FGKA: A fast genetic k-means clustering algorithm”, *ACM Symposium on Applied Computing*, Poster Abstract, 2004.

[31] J. Vesanto, J. Himberg, E. Alhoniemi, J. Parhangankas, ”Self-organizing map in Matlab: the SOM Toolbox”, *Proceedings of the Matlab DSP Conference*, pp. 35-40, 1999.

[32] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm, last accessed on 25/05/2011.