MODELLING AND PREDICTING 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

ÖZLEM ERDAŞ

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

SEPTEMBER 2007

Approval of the thesis

"**MODELLING AND PREDICTING BINDING AFFINITY OF PCP-LIKE COMPOUNDS USING MACHINE LEARNING METHODS**"

submitted by **Özlem Erdaş** in partial fullfillment of the requirements for the degree of **Master of Science in Computer Engineering, Middle East Technical University** by,

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

Prof. Dr. Volkan Atalay _____
Head of Department, **Computer Engineering**

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

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

Examining Committee Members:

Prof. Dr. Volkan Atalay _____
Computer Engineering, METU

Assoc. Prof. Dr. Ferda Nur Alpaslan _____
Computer Engineering, METU

Prof. Dr. Erdem Büyükbingöl _____
Pharmacy, Ankara University

Assoc. Prof. Dr. Halit Oğuztüzün _____
Computer Engineering, METU

Asst. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics 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 : Özlem Erdaş

Signature :

ABSTRACT

MODELLING AND PREDICTING BINDING AFFINITY OF PCP-LIKE COMPOUNDS USING MACHINE LEARNING METHODS

Erdaş, Özlem

M.S., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ferda Nur Alpaslan

Co-Supervisor: Prof. Dr. Erdem Büyükbingöl

September 2007, 72 pages

Machine learning methods have been promising tools in science and engineering fields. The use of these methods in chemistry and drug design has advanced after 1990s. In this study, molecular electrostatic potential (MEP) surfaces of PCP-like compounds are modelled and visualized in order to extract features which will be used in predicting binding affinity. In modelling, Cartesian coordinates of MEP surface points are mapped onto a spherical self-organizing map. Resulting maps are visualized by using values of electrostatic potential. These values also provide features for prediction system. Support vector machines and partial least squares method are used for predicting binding affinity of compounds, and results are compared.

Keywords: Modelling, spherical self-organizing maps, prediction, support vector regression, partial least squares regression

ÖZ

MAKİNE ÖĞRENİMİ YÖNTEMLERİNİ KULLANARAK PCP BENZERİ BİLEŞİKLERİN MODELLENMESİ VE BAĞLANMA EĞİLİMLERİNİN TAHMİNİ

Erdaş, Özlem

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Ferda Nur Alpaslan

Ortak Tez Yöneticisi: Prof. Dr. Erdem Büyükbingöl

Eylül 2007, 72 sayfa

Makine öğrenimi yöntemleri bilim ve mühendislik alanlarında ümit vaat eden araçlar olmuşlardır. Bu yöntemlerin kimya ve ilaç tasarımı alanında kullanımı 1990lardan sonra gelişmiştir. Bu çalışmada, PCP benzeri bileşiklerin moleküler elektrostatik potansiyel (MEP) yüzeyleri bağlanma eğilimlerinin tahmininde kullanılacak özellikleri çıkartmak amacıyla modellenecek ve görselleştirilecektir. Modellemede MEP yüzey noktalarının Kartezyen koordinatları küresel özörgütlemeli haritaya eşlenecektir. Sonuçta oluşan haritalar elektrostatik potansiyel değerleri kullanılarak görselleştirilecektir. Bu değerler aynı zamanda tahmin sisteminde kullanılan özellikleri sağlayacaklardır. Destek vektör makineleri ve kısmi en küçük kareler bileşiklerin bağlanma eğiliminin tahmininde kullanılacak ve sonuçlar karşılaştırılacaktır.

Anahtar Kelimeler: Modelleme, küresel özörgütlemeli harita, tahmin, destek vektor regresyonu, kısmi en küçük kareler regresyonu

To my parents, my friends and to him.

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 Prof. Dr. Ferda Nur Alpaslan for giving the opportunity to work on this interesting problem. Her trust and motivation guided me in the most hopeless moments. Secondly, I would like to thank my co-supervisor Prof. Dr. Erdem Büyükbingol for spending hours with me to make brainstorm and solve problems. His enthusiasm and inspiration enlightened my way.

I am also grateful to my friends especially Yasemin, Funda, Zerrin, Cuneyt, Ozge and Oral who listened my complaints, gave encouragement and suggestions, and helped me overcome technical problems such as writing with LATEX.

Lastly, and most importantly, I would like to thank my parents for their endless love and support. I am indebted to them for believing in me without a doubt.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

One of the most important objectives of drug design is to reveal and optimize new chemical compounds which unite with target proteins, and as a result, recompense or heal the illness. However, it is hard to find such molecules in a huge chemical space which contains hundred thousands of chemical databases or billions of small molecules. The traditional approach is to produce and test a large number of diverse compounds, and then, focus on compounds having some desired biological activity. This approach is often referred to as "high-throughput screening" (HTS) or "irrational design" to emphasize the fact that some kind of random search is performed in chemical space. Current progress of chemical technologies allows large number of molecular substances to be synthesized rapidly. Recent combinatorial chemistry makes it possible to produce hundreds of molecules in several days. Moreover, biological experiments can be carried out with the help of these new molecular agents. Consequently, some properties of molecules like binding affinity are calculated easily [10].

On the other hand, efficient search techniques are still essential. Recently, machine learning techniques are used in drug discovery process not only for searching the active drugs but also building structure-activity models and exploring relationship between similar drugs. Machine learning methods such as neural networks, support vector machines and genetic algorithms are applied to the following drug design problems:

- Classification of large data sets [10, 11]
- Feature extraction and nonlinear modeling of QSAR[1] (Quantitative Structure-Activity Relationship) [12, 13]

[1]QSAR includes a continuous activity modeling for quantitative prediction of the activity of unseen compounds. QSAR analysis is a regression problem [11].

- Prediction of molecular properties [14, 15]
- Similarity searching [16, 17]

In this study, we attempt to model molecular surfaces to extract features and predict binding affinity[2] of the molecules by using this model. Basically, there are two main properties controlling a drug-like molecule to bind a receptor. The first one is the topological shape of the molecule which allows molecule fit geometrically to the binding site of the receptor. The second property is the electrostatic potential[3] or charge on the molecular surface. The electrostatic potential is very crucial because it influences whether the molecule attracts another molecule or not. Although, the geometry of the molecule fits perfectly to the binding site, it cannot bind to the receptor if their electrostatic potentials are not well-matched [19]. However, it is difficult to visualize or directly use the molecular electrostatic potential (MEP) because of its three-dimensional nature [18]. Then, we decide to map MEP surface onto a simpler but topology preserving structure which is the spherical self-organizing map. By using this method, we aim to obtain a predictive model which carries topological and electrical information of the molecules in order to estimate their binding affinities.

1.2 Organization of the Thesis

In our attempt to predict binding affinity of PCP-like compounds, we followed the steps demonstrated in Figure 1.1. The formulas of 38 molecules are taken from [8]. In data preprocessing part, structural formulas are obtained and optimizations are performed by using HyperChem program[20]. Then, molecular surfaces are constructed by VegaZZ program[21]. 3D coordinates of surface points and their electrostatic potential (ESP) values are extracted from the surface. Using this information, molecular surfaces are mapped onto spherical self-organizing maps. An "internally ordered" representation of ESP values are obtained as the result of mapping. Support vector regression and partial least squares regression are fed by these values in order to predict the binding affinity values of the molecules.

The thesis is organized as follows:

- In Chapter 2, background of the method which is self-organizing maps is explained. The original learning algorithm is told briefly, and information about types of self-

[2]Binding affinity is the tendency of a small molecule to bind a large molecule, generally an enzyme.

[3]The electrostatic potential at a given point on the molecular surface is defined as "the work needed to bring a unit positive charge from infinity to that given point"[18].

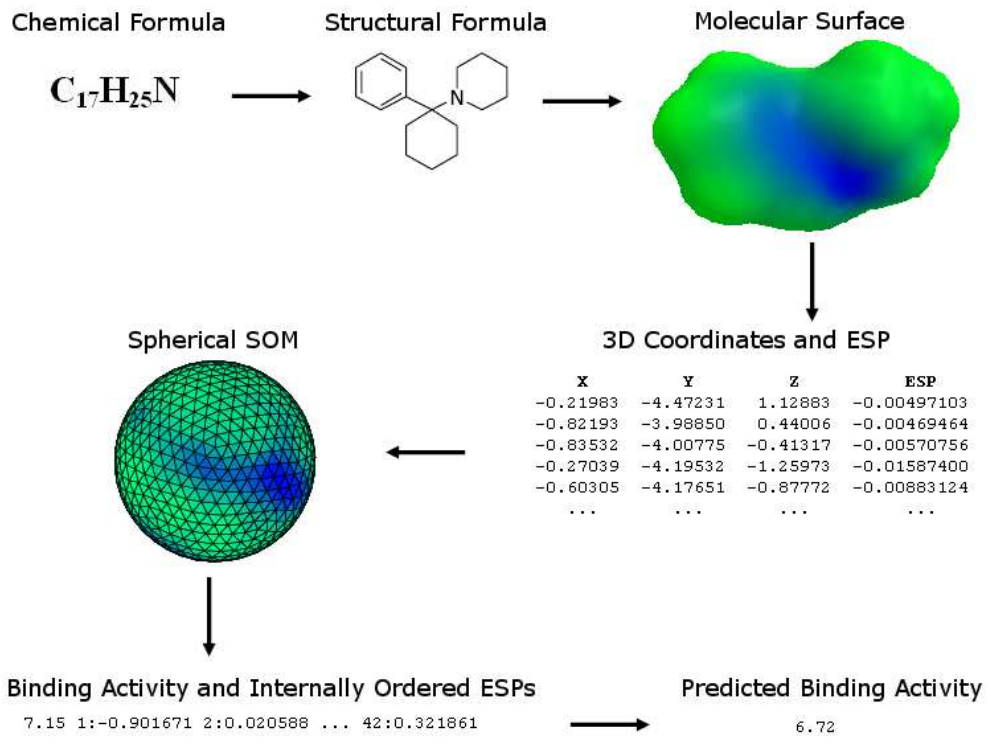


Figure 1.1: Flow of the thesis

organizing maps is given. Further, some applications of self-organizing maps especially in modeling and feature extraction are discussed.

- In Chapter 3, the implementation details of modeling molecular surfaces are given. The steps of data preparation and spherical self-organizing map algorithm are mentioned. The problems which arisen during the implementation and our solutions are discussed. Finally, results of the modeling are demonstrated.

- In Chapter 4, regression methods which are used for prediction in the study are explained.

- The experiments which are performed for predicting binding affinity of molecules are mentioned, and their results are discussed in Chapter 5.

- Finally, the conclusion and future work are presented in Chapter 6.

CHAPTER 2

SELF-ORGANIZING MAPS

2.1 Basic Concepts

Neural networks can be classified into three groups based on their architectural structures [22]. *Feedforward networks* take input signals and turn them into output signals by altering the system parameters with supervision. The system's initial state of activity is determined by input samples in the second group called *feedback networks*. The asymptotic final state is reached after some state transitions and determines the output. In the last group, neurons in a neighborhood try to win a competition by matching an input pattern with minimum distance and update themselves adaptively in order to minimize error. This type of learning is called *unsupervised competitive learning*.

The idea of Self-Organizing Map (SOM) is based on competitive learning which is an adaptive process as follows: Suppose that there are sample vectors $x(t) \in R^n$ and some weight vectors $w_i(t) \in R^n$, where $i = 1, ..., k$, which are initialized in an appropriate way, randomly in general. Here, t is the time coordinate. $x(t)$ is compared to each $w_i(t)$ at each step $t = 1, 2, 3, ...$, and then the best-matching vector $w_c(t)$ is found. After that, the distance between x and w_c is decreased by updating the coordinates of w_c but the coordinates of other vectors where $i \neq c$ remains unchanged [22, 23].

SOM is generally used for visualizing high-dimensional data. Basically, it creates a "similarity graph of input data". Nonlinear statistical relationships of input data are transformed into simple geometric relationships of output units which are represented in a low-dimensional space. Also, it preserves the crucial topological and metric information of input data while making the data compact [2]. Visualization and abstraction features of SOM make it widely used in fields of science and engineering such as robotics, medicine, process control and telecommunication [2, 22].

2.2 The Learning Algorithm

SOM algorithm basically maps a set of input vectors of a high-dimensional space into weight vectors, also called *reference* or *model vectors* [2], of a low-dimensional space. Let $X = \{x^s | x^s = [x_1^s, x_2^s, ..., x_n^s] \in R^n\}$ be the set of input vectors where x_n^s is the n^{th} feature of the s^{th} vector, $s = 1, 2, ..., N$ and N is the number of vectors. $w^i = [w_1^i, w_2^i, ..., w_n^i] \in R^n$ is a weight vector corresponding to a cell in map where $i = 1, 2, ..., M$ and M is the number of cells. In Figure 2.1, two-dimensional array of cells which forms a SOM is shown.

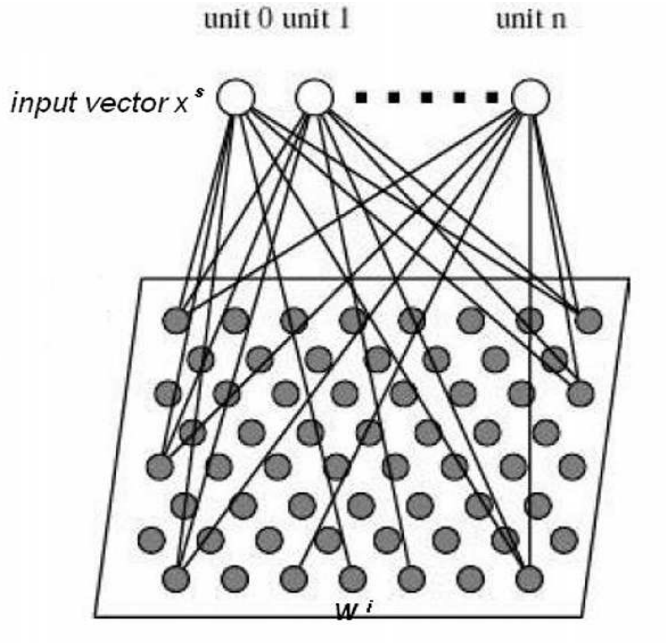


Figure 2.1: Two-dimensional hexagonal array of cells [1]

The basic SOM algorithm is given in Algorithm 1. Some important details about the algorithm should be mentioned. For instance, there are several approaches for initializing weight vectors. Those are:

Random initialization Weight vectors are randomly assigned. Small values are often preferred. The produced weights are unordered.

Initializing with input data Weight vectors are chosen among input patterns. If there are k weight vectors, the first k input patterns or k randomly chosen points can be assigned as initial weight vectors.

Algorithm 1 Self-Organizing Map Algorithm

1: Initialize weight vectors and set the number of iterations as desired.
2: At each iteration, choose input vectors and for each vector apply the steps 3-5.
3: Compute the distance between the selected input vector and all weight vectors.
4: Find the winner cell with minimum distance.
5: Update the weights of the winning cells and its neighbors in order to minimize the distance.
6: Check out the stopping condition. Go back to Step 2 unless the condition is satisfied.

Linear initialization Using ordered weight vectors sometimes gives better results. In this approach, two eigenvectors of input patterns which span a two-dimensional linear subspace is found. After that, a rectangular array whose center is equal to the mean of input patterns is produced from this subspace. At last, the weight vectors are initialized using the points of this array [2].

After initialization, the training part of the algorithm takes place. At each iteration (or epoch), all input patterns should be selected one by one. This selection can be either randomly or iteratively. The distance between the selected input vector and each weight vector is computed. Although "dot" product of vectors can be used for the distance calculation, Euclidean distance $\parallel x^s - w^i \parallel$ is preferred most of the time [2]. The map unit with the minimum distance is identified as the winner unit or the best-matching unit(BMU). Weight vectors of the BMU (w_c) and the units which resides in its neighborhood N_c are updated according to the Equation 2.1:

$$w^i(t+1) = w^i(t) + \alpha(t)h^i(t)[x^s(t) - w^i(t)] \tag{2.1}$$

where t is time coordinate. $\alpha(t)$ is the learning rate factor which is a decreasing function of time. Below, several $\alpha(t)$ functions are shown:

$$\alpha(t) = \frac{A}{t+B} \qquad \alpha(t) = Ce^{\frac{-t}{N}} \qquad \alpha(t) = D(1 - \frac{t}{N}) \tag{2.2}$$

where A, B, C and D are constants. In order to protect $0 < \alpha(t) < 1$ inequality, $A \leq B$ and $0 < C, D < 1$.

$h^i(t)$ in Equation 2.1 is called the neighborhood kernel and has an important role in convergence. $\lim_{t\to\infty} h^i(t) = 0$ should be satisfied to make sure that the algorithm converges. There are two alternatives for $h^i(t)$ according to [2]. The first one allows every unit i in

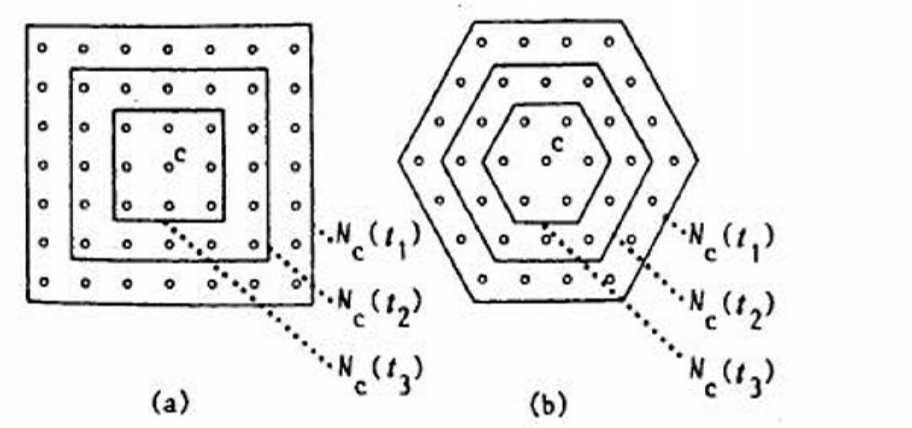


Figure 2.2: Two types of topological neighborhoods: (a) Rectangular (b) Hexagonal ($t_1 < t_2 < t_3$) [2]

the neighborhood N_c of the winning unit c to be updated equally by Equation 2.3. Indices of units in N_c is kept in an array in this approach. Also, radius of N_c shrinks in time as presented in Figure 2.2.

$$h^i(t) = \begin{cases} 1 & if\ i \in N_c \\ 0 & otherwise \end{cases} \tag{2.3}$$

The other choice is to use Gaussian kernel function,

$$h^i(t) = \exp\left(-\frac{\| r^c - r^i \|^2}{2\sigma^2(t)} \right) \tag{2.4}$$

where $\sigma(t)$ which decreases monotonically in time denotes the width of the neighborhood. Gaussian kernel is widely used in literature [2]. However, computational difficulties cause some researchers to use modified versions of Equation 2.4 [3, 6, 24]. Moreover, $\| r^c - r^i \|$ is the distance between the best-matching unit c and its neighbor i. The choice of initial radius of N_c is very important. If N_c is not large enough at the beginning, the ordering of the map will remain local. On the other hand, the radius of N_c should not be more than the half radii of the map initially[2].

2.3 Types of Self-Organizing Maps

In this section, self-organizing map architectures and neighborhoods are introduced. The architectures can be classified into four groups according to data association as illustrated in Figure 2.3.

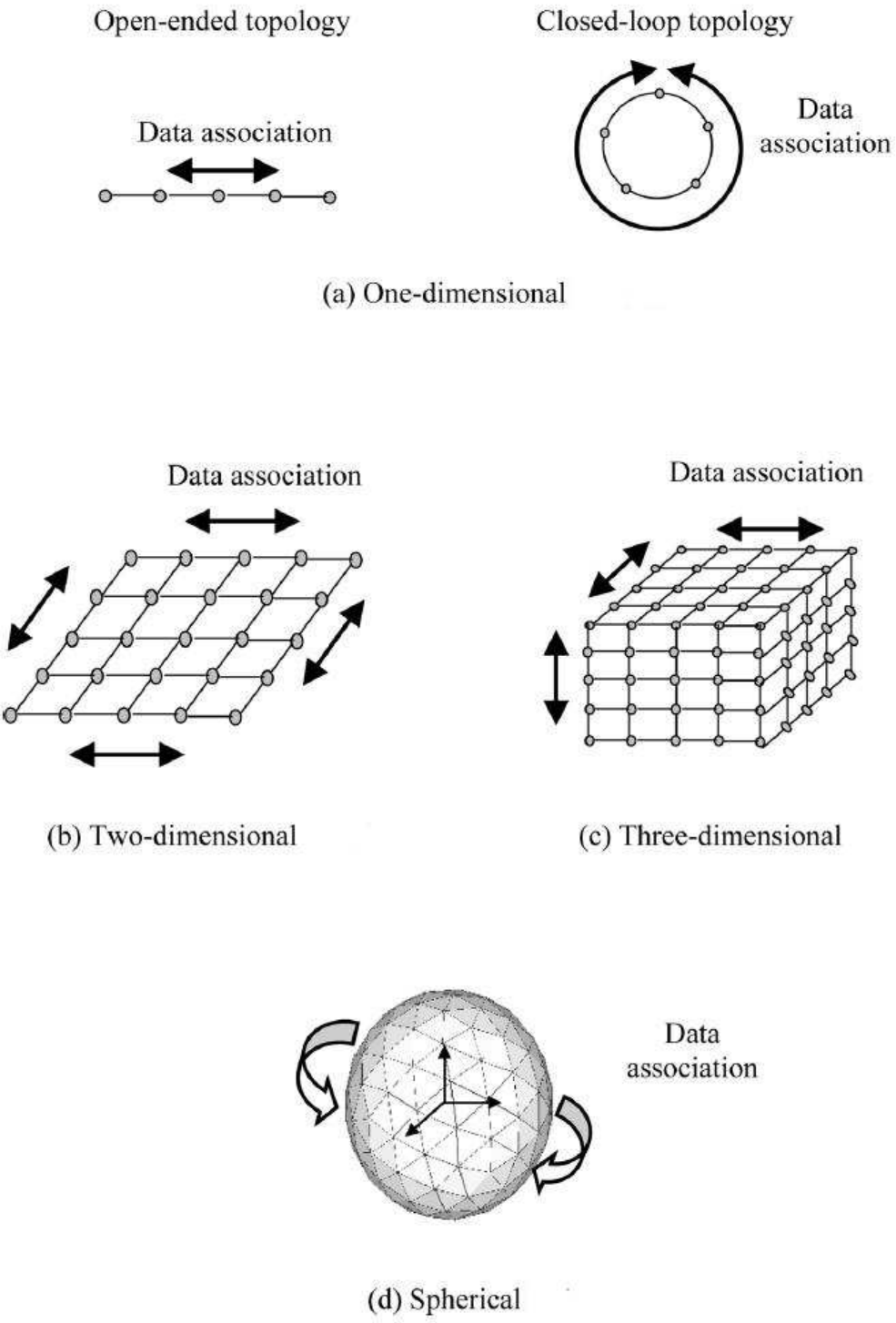


Figure 2.3: Types of SOM adapted from [3]

2.3.1 One-dimensional Case

Architecture

The number of N units are arranged in a one-dimensional array which is called open-ended topology or units may reside on a circle which is resulted in closed-loop topology.

Neighborhood

In open-ended topology of one-dimensional SOM, each unit except the first and last elements of the array has two neighbors, one on the left and one on the right. However, all units have exactly two neighbors in circular form of the map.

2.3.2 Two-dimensional Case

Architecture

$N \times N$ neurons are arranged on a two-dimensional array, grid or lattice in two-dimensional SOM. Each unit is fully connected to the input vectors. Moreover, all units are indexed with discrete (i, j) indices denoting their locations on the map.

Neighborhood

There are two neighborhood types popularly used in literature. These are rectangular and hexagonal neighborhoods.

Rectangular In this form, neighboring units are placed at the borders of a rectangle where the winner unit is at the center as in Figure 2.2.a. A unit has 8 nearest neighbors unless it is on the boundaries of the two-dimensional grid.

Hexagonal Neighbors of the winner cell are located on the sides of a hexagon as can be seen in Figure 2.2.b. A unit has 6 nearest neighbors in this case. Hexagonal neighborhood is preferred rather than rectangular one since the distances between the center unit and its neighbors of the same $N_c(t)$ are equal [1].

2.3.3 Three-dimensional Case

Architecture

Units are placed inside a rectangular prism as in Figure 2.3.c or generally inside a hypercube. Also, two-dimensional grids forming a three-layered map are mentioned as three-dimensional

SOM in [25].

Neighborhood

There are several approaches for three-dimensional neighborhood most of which are generated from two-dimensional neighborhoods. According to these approaches, neighbor units may lie inside a sphere, cube or three-dimensional diamond or star shaped structures [25].

2.3.4 Spherical Case

Architecture

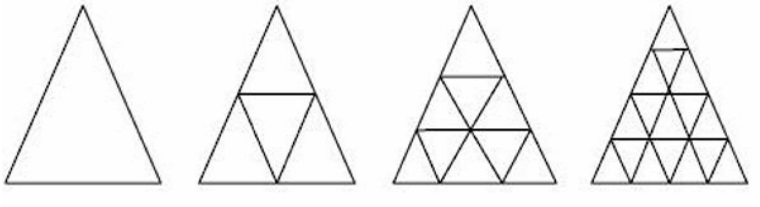


Figure 2.4: Tessellation of a triangle. From left to right: one-frequency, two-frequency, three-frequency and four-frequency tessellation. [4]

Neurons are arranged on "a tessellated unit sphere with uniform triangular elements" [3] in spherical self-organizing map architecture. Here, *tessellation* means dividing a unit two dimensional figure into smaller figures of the same kind such that there are no gaps left. An example of triangle tessellation is demonstrated in Figure 2.4. In order to generate a two-frequency triangle, points in the middle of each side is calculated at the beginning. After then, these points are connected by lines and, at the end, the triangle is divided into 4 triangles which is called a polyhedron . "The tessellated unit sphere with uniform triangular elements" in the definition of spherical SOM architecture is called an icosahedron. In the simplest case, an icosahedron is a polyhedron having 20 faces and 12 vertices. Three levels of an icosahedron can be seen in Figure 2.5. It should be pointed out that an icosahedron can be tessellated at most 5 times because after that it loses its uniformity meaning that the triangles begin to have different edge lengths. It means that the triangles begin to have different edge lengths at the 6^{th} tessellation [4]. In spherical SOM architecture, each vertex corresponds to a neuron [5].

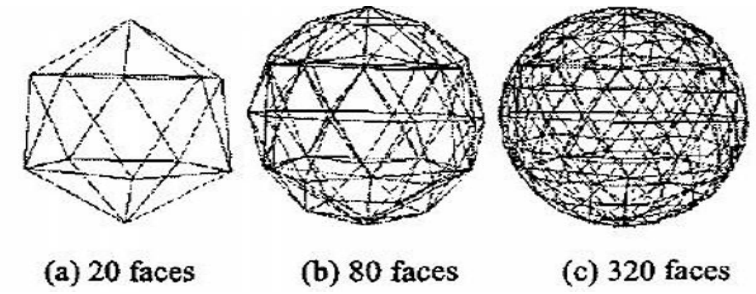


Figure 2.5: (a) Basic icosahedron (b) After first tessellation (c) After second tessellation. [5]

Neighborhood

Neighborhood of spherical SOM is similar to two-dimensional SOM's hexagonal neighborhood since a unit has 6 neighbors in the nearest neighborhood. Moreover, the distance between the unit and its nearest neighbors is the same. In Figure 2.6, neighborhood of a unit in spherical SOM is illustrated.

However, finding the neighbors of a unit is a complicated process because of the complexity of the structure. Some studies [3, 5, 26] work on this problem. A comparison of three approaches is made in [26]. Sangole and Knopf [3] collects indexes of neighboring units at each radius while tessellating the sphere by consuming $O(N^2)$ space where N is the number of neurons. So, it takes $O(1)$ time to find neighbors at training phase. On the other hand, Boudjemaï et al. [5] keeps 6 of the nearest neighbors at hand with $O(N)$ space complexity. Then, other neighbors are found level by level using a method like a Breadth-First Search in $O(n)$ time. Here, n is the number of neighbors at each radius. As the last and novel approach, Wu and Takatsuka [26] developed a data structure to keep neighbors with $O(N^{1/2})$ space, and their search is similar to the one in [5] which takes $O(n)$ time.

2.4 Previous Work

This section discusses studies which use self-organizing map algorithm. Majority of the works are chosen among the ones which model surfaces, map high-dimensional data onto a low-dimensional space, and deal with data visualization. Furthermore, studies mentioned use two and three-dimensional, generally spherical map structures, since one-dimensional maps are mostly used to define and prove some features of the algorithm as in [2].

First, the use of SOM in mapping molecular surfaces will be discussed since it is the

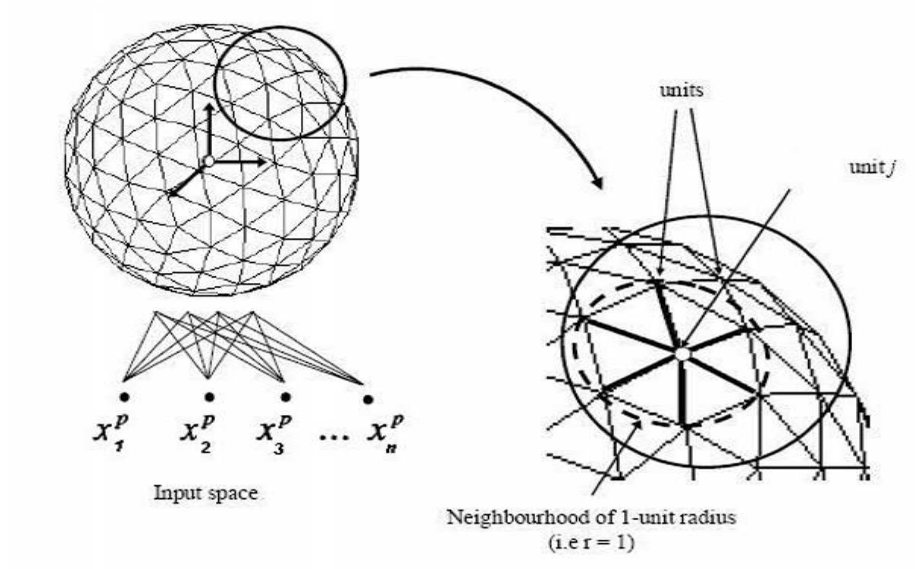


Figure 2.6: The spherical self-organizing map architecture and neighborhood adapted from [6]

starting point of this study. Gasteiger and Li [17, 18] used toric map which is a special version of two-dimensional SOM for mapping the electrostatic potentials of muscarinic and nicotinic agonists in order to compare their structures. In order to generate a toric map, the opposite sides of rectangular grid are connected to form a cylinder. After that, the opposite cycles of the cylinder are connected and, a torus is constructed as a result. Toric map is preferred instead of two-dimensional grid because neurons near the borders of the grid suffer from the lack of neighbors. After generating the torus, training phase starts. Input data includes Cartesian coordinates of randomly chosen points on the molecules' Van der Waals surface. The neuron having weights with the smallest distance to Cartesian coordinates of the input point is selected as the winner unit and weights are updated with respect to the winner. After training, input points are sent to the network again and the neurons which are excited by the data points take their electrostatic potential values. While the size of network is smaller than the size of input, one neuron can be excited by more than one point. In this case, the neuron takes the average value of electrostatic potentials of the points. The projection of the MEP is made onto the surface of a torus and that the planar map is obtained by the reverse process of generating the torus [18]. Mapping of a molecule can be viewed in Figure 2.7. The magnitude of the electrostatic potential is translated into a color code: Strongly negative values of the electrostatic potential are represented by red. Strongly

positive values are marked in blue or violet. The intermediate values are represented by continuous blends of color. However, Zell et. al [27] reported that there are some topological

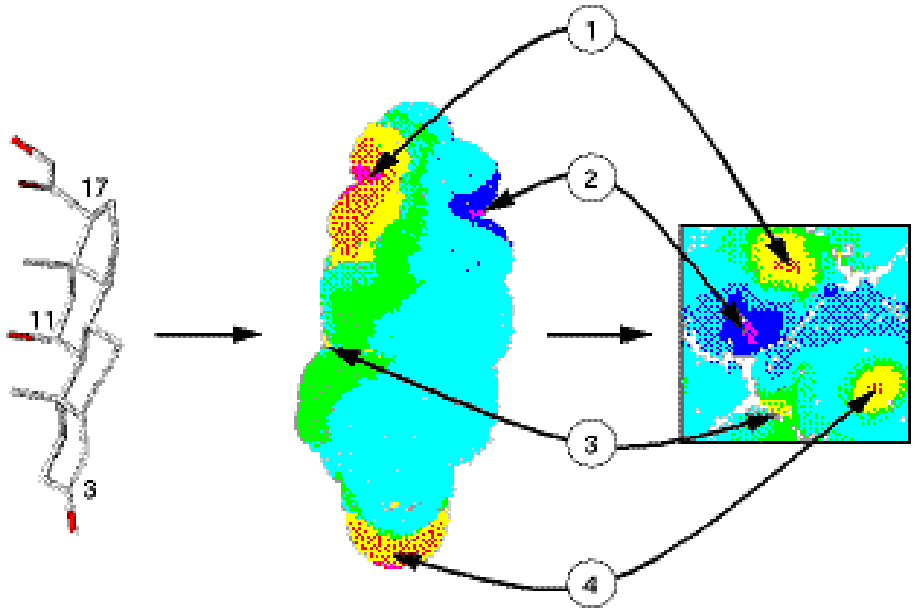


Figure 2.7: Kohonen mapping of three-dimensional model of a molecule [7]

problems with the above approach as follows:

- At the grid boundaries, two nearby points on the 3D surface may be mapped to maximally distant points on the grid. Following figure is a grid obtained from opening a torus. Shaded neurons are distant to each other. However, they may represent closer points on the surface.

- Another topological defect is twist of map in the middle of the torus. This causes correct mapping of half the molecule and inside-out mapping of the other half.

To solve the first problem, the map is rotated until the most interesting regions are at desired positions. Furthermore, torus is cut up again and copies of resulting grid are placed side by side, tiling a plane. For the solution of both topological defects, they extend the self-organizing maps to self-organizing surfaces. In the extension, points are mapped on an arbitrary surface rather than a grid. The learning algorithm is nearly identical to SOM algorithm, except the distance calculation which is done by a parallel minimum search on a linear array of processors.

Seiffert and Michaelis [25] introduced on three-dimensional SOM and two-dimensional data for analyzing moving images.

Knopf and Sangole [28] also used three-dimensional SOM for clustering features of images obtained from three cameras. In their model, neurons which have weight matrix of two-dimensions are arranged on a hypercube. They observed that performance of their method depends on the chosen feature set.

Boudjemaï et. al [6] studied surface reconstruction from unorganized data points by using toric and spherical SOM. Their goal was to construct a connectivity between a cloud of points and transform them into the original topological surface which obeys design rules. They achieved to rebuild the surfaces of art works in a short training time.

Wu and Takatsuka [1] compared two-dimensional hexagonal SOM to their spherical SOM model which is called Geodesic SOM, namely GeoSOM. They also introduced a novel performance metric named Error Entropy. This metric shows whether the error is distributed equivalently on the map. Therefore, a map with high Error Entropy is smooth representation of input. It is observed that GeoSOM has higher Error Entropy than two-dimensional SOM. Also, spherical SOM eliminates the border problem of the other map.

Some other studies of spherical SOM for data visualization belong to Sangole and Knopf [3, 24]. Their aim is to convert unorganized group of scientific data into a three-dimensional space. A deformable spherical SOM is used in the experiments. As a result, they obtain regular and repeatable color-coded geometric surfaces which help scientists and engineers to analyze data.

CHAPTER 3

MODELING MOLECULAR SURFACES BY USING SPHERICAL SELF-ORGANIZING MAPS

3.1 Data Preparation

A data set of 38 drug-like compounds which are phencyclidine (PCP) derivatives are used in the experiments [8]. Each compound has NMDA (N-methyl-D-aspartic acid) receptor binding affinity values, K_i, as shown in Figure 3.1.

Structural formulas of the compounds are processed in order to obtain Cartesian coordinates of points on the surface and corresponding electrostatic potential values. Data is prepared as follows:

- Molecules are drawn using *HyperChem* program [20], and their rough two-dimensional structure is obtained.
- MM+ molecular mechanics method based on the MM2 force field is selected for obtaining molecular stability.
- The geometric optimization of compounds are obtained by Polak-Ribiere method based on conjugate gradient algorithm.
- Atomic charges are calculated by PM3 (MOPAC 7.0) method which uses MNDO-PM3 Hamiltonian. The calculations are done using *VegaZZ* program [21].
- After optimization, molecular electrostatic potential (MEP) surfaces are built with *VegaZZ*. Three surfaces for each molecule are constructed with respect to the number of points per unit area (Table 3.1).

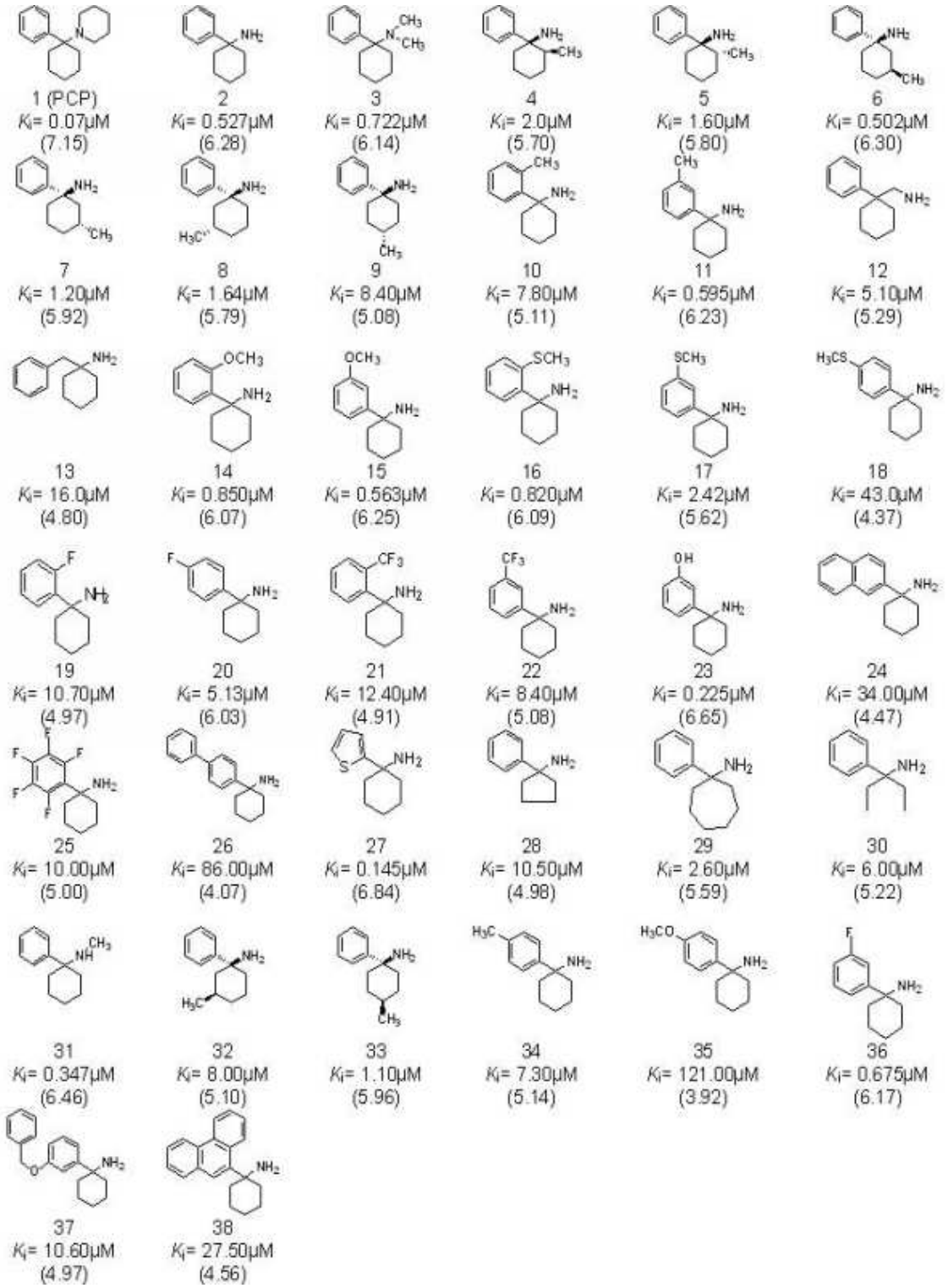


Figure 3.1: The chemical formulas of PCP-like compounds with their NMDA binding values K_i. The values in parenthesis express the log $(1/K_i)$ [8]

Table 3.1: Number of points per unit area and average number of points located on surface.

Density	Avg. Number of Points
10	1,764
8	1,340
3	447

- Cartesian coordinates of surface points and corresponding electrostatic potential values are saved into CSV files.

3.2 The Modeling Architecture

A spherical self-organizing map (SSOM) architecture is used to model the molecular surfaces. Each neuron has a three-dimensional weight vector since the aim of modeling is to map the coordinates of molecular surfaces onto a sphere. This architecture is chosen rather than two-dimensional SOM because of the topological defects of the latter, like neighborhood restriction at the borders of the two-dimensional map. The cells near the sides and the corners of the map have fewer neighbors. As a solution to this problem, a toric map is generated from two-dimensional map as explained in Chapter 2. However, this solution results in other defects. The first one is uncertainty of where to cut torus. The second defect is the twist of map which causes the half of the input to be mapped inside out. On the other hand, SSOM produces an "internally ordered" representation of data points while reducing the size of the data. As a result, SSOM is chosen as the architecture of the model in order to protect the topological information of the molecular surfaces.

SSOFM Toolbox which is developed in MATLAB by Sangole and Leontitsis [3, 6, 24] is used by modifying some parts of the code. The toolbox generates tessellated spheres and neighboring relations before applying the learning algorithm.

3.3 The Learning Algorithm

The learning algorithm used in this study is as follows:

1. Initialize weight vectors $w^i = [w_1^i, w_2^i, w_3^i]$, where $i = 1, 2, ..., M$, such that each vector is the coordinates of the neuron it belongs. Here, M is the number of neurons on the map. Then, weight vectors are scaled with respect to the maximum and minimum

values of input vectors. As a result, the input points are assumed to reside inside the sphere.

2. At each epoch, do the following steps:

   (a) An input point, $x^p = [x_1^p, x_2^p, x_3^p]$, is chosen at random. Here, $p = 1, 2, ..., N$ where N is the number of input points. Apply the steps (b)-(d) to each chosen unit.

   (b) Calculate the distance between the input point and the weight vectors of all neurons by using Equation 3.1

   $$d(x^p, w^i) = \sum_{k=1}^{3} (x_k^p - w_k^i)^2 \tag{3.1}$$

   where $i = 1, 2, ..., M$ and M is the number of neurons.

   (c) Choose the neuron with minimum distance as "winner unit".

   (d) Update weight vectors of winner unit with index c and its neighbors in the neighborhood $C\{c, r\}$ with the following rule:

   $$w^i(t+1) = w^i(t) + \alpha(t) h^i(r, s)(x^p - w^i(t)) \tag{3.2}$$

   where $t = 1, 2, ..., T$, T is the total number of epochs specified at the beginning, and $r = r(t)$ is the neighborhood radius which shrinks in time as follows:

   $$r(t) = \begin{cases} R & if\ 0 < t \leq \frac{T}{4} \\ \frac{R}{2} & if\ \frac{T}{4} < t \leq \frac{T}{2} \\ 1 & if\ \frac{T}{2} < t \leq \frac{3T}{4} \\ 0 & if\ \frac{3T}{4} < t \leq T \end{cases} \tag{3.3}$$

   In Equation 3.2, $\alpha(t) = 0.9(1 - \frac{t}{T})$ is the learning factor which is used in the implementation. As neighborhood kernel, $h^i(r, s) = \exp(-d(w^i, w^c)/s.r(t))$ is used where s is the neighborhood parameter. This neighborhood kernel is used instead of Gaussian kernel because of the computational complexity of the system. It should be noted that both $\alpha(t)$ and $h^i(r, s)$ are functions decreasing over time.

3. Stop if $t = T$. Go back to step 2, otherwise.

The map is trained by using above algorithm for determined number of epochs. At the end, each point of the molecular surface is matched to a neuron.

3.4 Problems Faced with during the Implementation and Their Solutions

Some problems are detected during the first experiments due to finding the optimal value of neighborhood parameter s and presence of dead neurons. We discuss these problems and our solution methods below.

3.4.1 Finding Optimum Neighborhood Parameter

Neighborhood parameter s defines the width of the neighborhood. It regularizes the neighborhood. Finding optimum neighborhood size is very important for SSOM since there is a trade-off between smoothness and preciseness. For instance, a wide neighborhood causes the map to be smooth which is desirable in visualization. However, it neglects some details of data points. On the other hand, a very narrow neighborhood deals with every detail but causes a coarse visualization of the map. As a solution, Sangole and Leonitsis [6] proposed the L-curve approach.

L-curve Approach

L-curve is generally used in linear algebra for solving linear systems when the number of unknown variables exceeds the number of equations. The algorithm is explained below:

- Calculate accuracy (Equation3.4) and smoothness (Equation3.5) for every parameter value in a range:

$$accuracy = \sum_{s=1}^{N} d(x^p, w^i) \tag{3.4}$$

  where w^i's are weight vectors of the matching neurons for x^s.

$$smoothness = \sum_{i=1}^{M} d(w^i, W^i) \tag{3.5}$$

  where W^i is the mean of the nearest neighbors of w^i.

- Plot the results on log-log scale where accuracy lies on the vertical line and smoothness is on the horizontal line. The plot should be L-shaped. In Figure 3.2, the curve is plotted in linear scale. The original L-Curve is not as sharp as in this figure.

- The optimum neighborhood size can be found at the corner of the L-shape. It supplies higher values for both accuracy and smoothness.

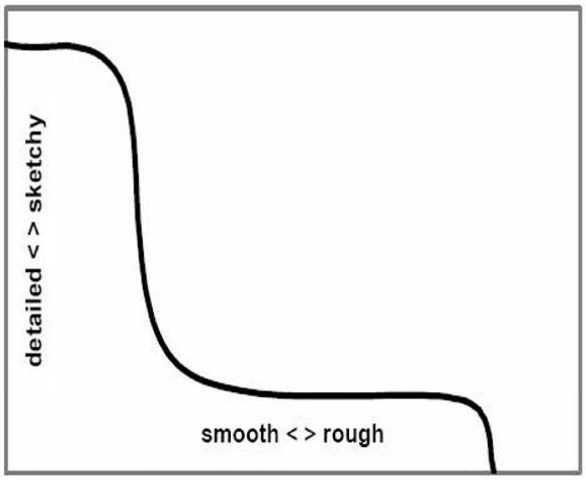


Figure 3.2: L-curve plot [6]

In the study[6], it is observed that the optimum neighborhood parameter depends only on the characteristic features of data. The same values are obtained for both small and large data sets, and for both less and more number of neurons.

3.4.2 Dead Neurons

All neurons do not have equal chance in winning the competition because of the initial configuration of the map. Some neurons left under-utilized after training. It means that they do not match any input point. Neurons in this situation are called *dead neurons*.

There are several approaches proposed to solve this problem. Some of them are using frequency parameter[24, 29], adding conscience factor[30] and maximizing mutual information[31]. In this study, the first two of them are implemented.

Frequency Parameter

Frequency parameter, u^i counts how frequently a neuron i wins competition. $F(u^i)$ is a count-dependent nondecreasing function which determines the distortion measure for distance calculation. By applying $F(u^i)$, distance calculation in Equation 3.1 becomes

$$d'(x^p, w^i) = F(u^i)d(x^p, w^i) = \sum_{k=1}^{3} F(u^i)(x_k^p - w_k^i)^2 \tag{3.6}$$

The neuron with minimum distance is selected as the winner unit. Weights of the winner and its neighbors are updated as in Equation 3.2. After the weight update, $F(u^i)$ is increased by

summing up with $h^i(r,s)$. As frequency function increases the modified distance increases, so the winning chance of the winner neuron and its neighbors decreases. This will result in increasing other neurons' chance to win [24, 29].

Conscience Parameter

The aim of conscience mechanism is to help every neuron to win the competition with almost $1/M$ probability where M is the number of neurons. Conscience algorithm includes two phases: output generation and weight update [30].

In the first phase, an input point x^p is fed to the network. An output y^i of i^{th} neuron is generated as in Equation 3.7

$$y^i = \begin{cases} 1 & if\ d(x^p, w^i) \leq d(x^p, w^j), \qquad \forall j \neq i \\ 0 & otherwise \end{cases} \tag{3.7}$$

A bias is introduced for each neuron depending on how many times the neuron becomes the winner. p^i shows the slice of time when the i^{th} neuron wins.

$$p^i(t+1) = p^i(t) + B(y^i - p^i(t)) \tag{3.8}$$

where B is a very small constant. In this study, $B = 0.0001$ is used. A modified output z^i is produced with bias term b^i.

$$z^i = \begin{cases} 1 & if\ d(x^p, w^i) - b^i \leq d(x^p, w^j) - b^j, \qquad \forall j \neq i \\ 0 & otherwise \end{cases} \tag{3.9}$$

where $b^i = C(1/M - p^i)$. Here, the bias factor C represents the distance a losing unit can win the race. C is 50 in our implementation.

In the second phase, the winner unit is determined based on the generated output z^i. The weights of the winner and its neighbors are adjusted according to Equation 3.2.

3.5 Performance

The quality of the SOM is generally depends on the training data. There are typically two measures which are quantization error and topographic error [2].

Quantization error measures the resolution of the map which is the average distance between the input points and their best-matching units. It is calculated by Equation 3.10

$$qe = \frac{\sum_{p=1}^{N} d(x^p, w^{pc})}{N} \tag{3.10}$$

where N is the number of input points and w^{pc} is the best-matching unit of p^{th} input point.

Topographic error measures the topology preservation. It shows whether the first and second best-matching units of an input unit are neighbors.

$$te = \frac{\sum_{s=1}^{N} te^s}{N} \tag{3.11}$$

where $te^s = 1$ if the first and second best-matching units of s^{th} input point are neighbors, and $te^s = 0$ otherwise.

3.6 Implementation and Results

The aim of the experiments is to map molecular electrostatic potential (MEP) surfaces of molecules onto a tessellated sphere. The experiments were carried out on AMD Athlon 1.83 GHz computer with 512 MB memory using SOFM Toolbox developed by Sangole and Leonitsis[3, 6, 24]. Spherical self-organizing maps (SSOMs) with 42, 162 and 642 neurons are used. At first, SSOMs are trained by applying the learning algorithm in Section 3.3 to each of 38 MEP surfaces in order to find which surface point matches to which neuron. After then, points of the surface are sent to SSOM again for modeling and visualization. The neuron which is excited by each point takes the electrostatic potential value at that point. Since the number of points is greater than the number of neurons, a neuron might be excited by two or more points. In this case, the average electrostatic potentials of the points is assigned to that neuron. In another study [18], minimum and maximum electrostatic potentials are also assigned rather than the average but no important difference is observed.

At the beginning of the implementation, L-Curve algorithm is used to select the optimum neighborhood parameter s. The algorithm is implemented on SSOM with 162 neurons and data set with average 447 points. The number of epochs is chosen as 50. Smoothness and accuracy are computed for s values in range of $2^{-5}, 2^{-4}, ..., 2^2$. The optimum results are obtained when $s = 0.25$ as in Figure 3.3.

After s is determined, experiments are carried out to eliminate dead neurons. The issue is important for visualization and modeling. It affects the interpretation of the molecular surface if the model is used for similarity search. Also, it is not desirable to have zero-valued attributes in prediction meaning that the charge is neutral at that point. The spherical map with 162 neurons is trained by using original distance measure, and by adding frequency and conscience mechanisms separately. The resulting maps who have dead neurons are counted.

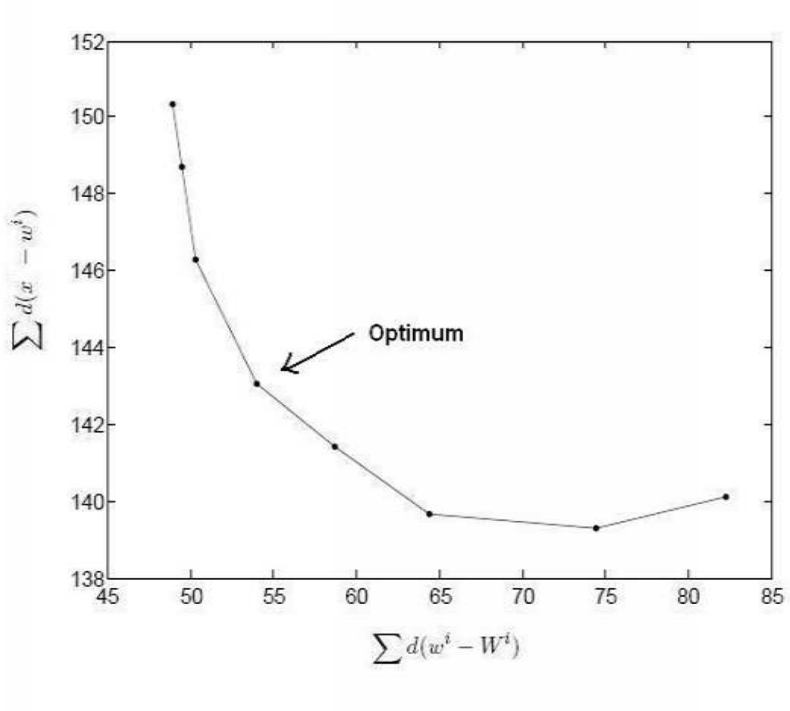


Figure 3.3: L-curve for optimum neighborhood parameter. Accuracy is on vertical axis and smoothness is on horizontal axis.

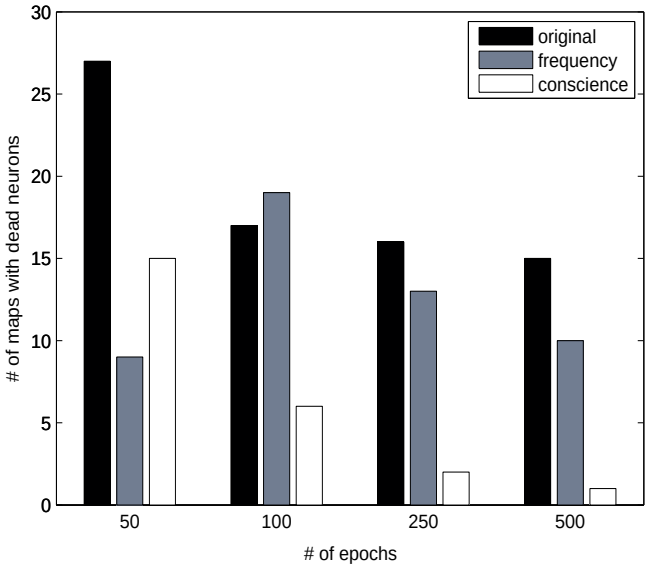


Figure 3.4: Number of maps with dead neurons vs. number of epochs

The results are available in Figure 3.4. It can be seen that conscience mechanism is more effective than the other two methods. Both original algorithm and conscience algorithm decrease the number of dead neurons as the number of epochs increases. However, frequency mechanism can also be preferred if the aim is to train the self-organizing map with a few number of iterations. Conscience mechanism will be used in the rest of the implementation.

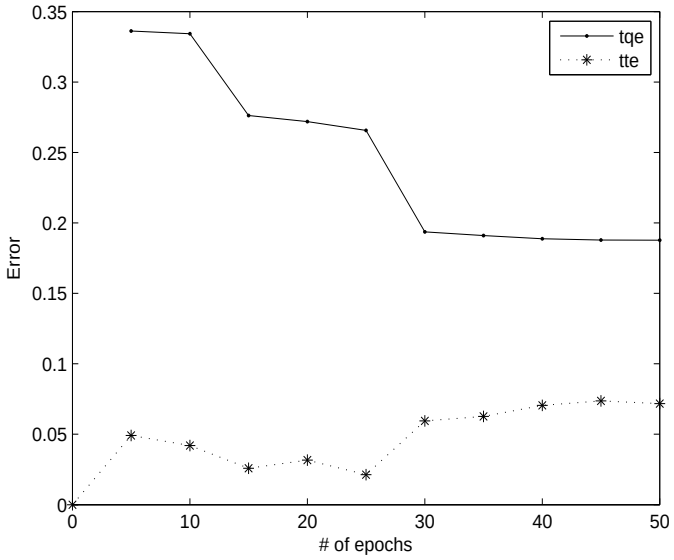
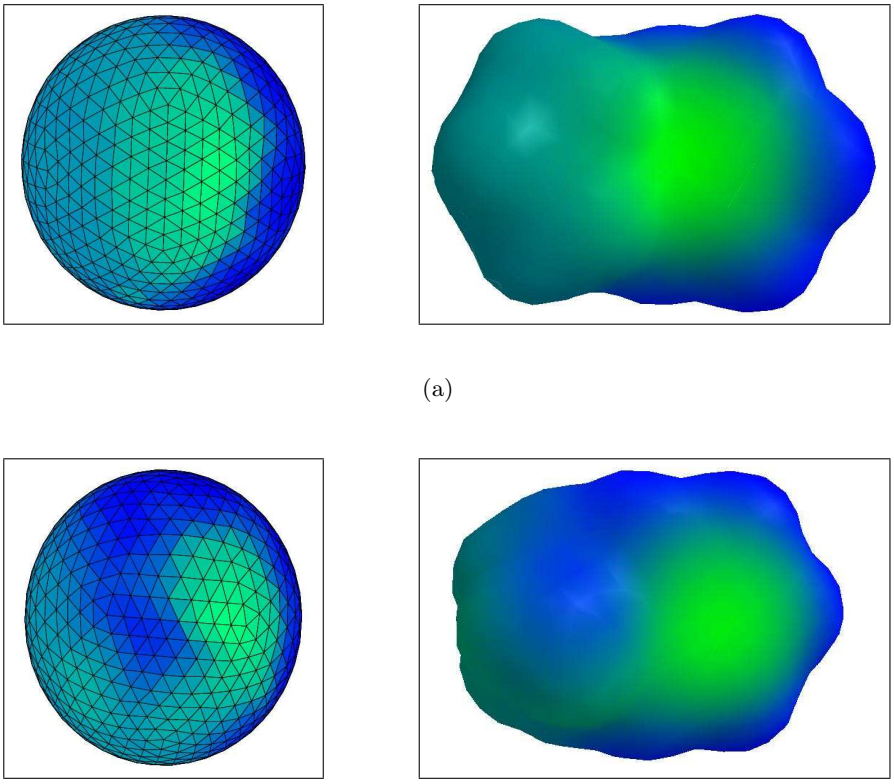


Figure 3.5: Total quantization error (tqe) and topographic error (tte) as the number of epochs increases.

In Figure 3.5, topographic error and quantization error of map with 162 neurons trained by conscience mechanism up to 50 epochs. It is observed that topographic error increases while quantization error decreases. As the number of iterations increases, the winner unit for each point and its neighbors are updated again and again. This causes weight vectors to get closer to the points and quantization error to get smaller. However, topographic error increases in time because updated neighborhood radius shrinks in time. Using larger neighborhood in final iterations preserves the topology better but it consumes time. Moreover, it is easily seen that there are only slight changes in error values after the 30^{th} epoch. There are two reasons. The first one is the shrinking radius. The other reason is that the map is starting to stabilized its weights.

For visualization, data set which consists of 38 molecular surfaces with average of 1,764 points is chosen. Spherical self-organized map with 642 neurons is trained by using conscience

algorithm during 500 epochs. After training, the data points are sent to the map again in order to find corresponding electrostatic potential values. Coloration is based on the values of electrostatic potential. Minimum and maximum values define the color range. MATLAB colormap named "winter" which ranges from blue to green is used to obtain images similar to VegaZZ. An example visualization of the 25^{th} molecule can be seen in Figure A.25. The maps of 38 molecules can be found in Appendix A.



(a)

(b)

Figure 3.6: Visualization of the 25^{th} molecule (a) front view (b) back view.

The color ranges from dark blue to light green indicating the electrostatic potential values. The dark blue parts represent the most negative sites while the light green parts are the most positive sites. At the end of the implementation, we obtain a compact model which is aimed to be used for predicting binding affinity of the molecules. The output of each neuron which carries topological and electrical properties is used instead of 3 Cartesian coordinates and electrostatic potential values corresponding to each point. Also, the number of points is reduced from thousands to hundreds with reasonable topographic and quantization errors.

CHAPTER 4

PREDICTION METHODS

In this section, we describe two regression methods used for prediction in our study. These methods are Support Vector Machines and Partial Least Squares.

4.1 Support Vector Machines

Support Vector Machines (SVMs), which were invented and developed by Vladimir Vapnik and his co-workers, are a very specific class of algorithms are characterized by the use of kernels. Some of their advantages among other architectures are the fast execution speed, the absence of local minima, the sparseness of the solution and the capacity control obtained by acting on the margin, or on other "dimension independent" quantities such as the number of support vectors [32].

The basic principal of the SVMs is the risk minimization built on the motivation of statistical learning theory [33].

Two main advantages of the SVMs are:

1. The dimension of the space is not important in determining the upper bound of the error of generalization.

2. As the margin is maximized, the error is minimized [34].

Support vector machines are generally used in classification problems as in [33, 35, 36] but there are also regression applications as in [9, 37]. In order to understand Support Vector Regression (SVR), it would be better to describe Support Vector Classification (SVC). So, we will start by mentioning SVC before proceeding with SVR.

4.1.1 Support Vector Classification

Considering a binary classification case which can be seen in Figure 4.1 is a good way of understanding the SVM approach in classification problems. Here, we have input vector $\mathbf{x}$ and a weight vector $\mathbf{w}$. The data points $\mathbf{x}_i$ (where $i = 1, \ldots, n$) belong to either one of the classes which has the labels $y_i = \{-1, +1\}$. The decision function will be $f(\mathbf{x}) = sgn(\mathbf{w}.\mathbf{x}+b)$ where b is the threshold [33, 34].

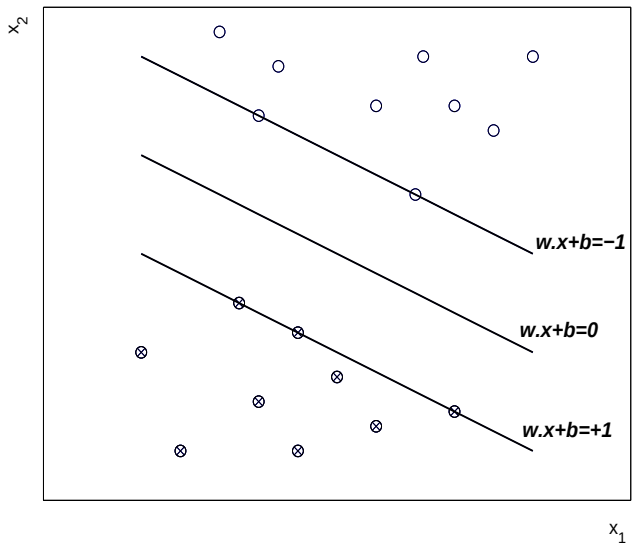


Figure 4.1: Support Vectors for binary classification data.

For separable data points, the data can be classified when $y_i(\mathbf{w}.\mathbf{x} + b) \geq 1$ for all i. The aim is to find an optimal hyperplane which separates data into two classes while maximizing the margin. The margin is the maximum Euclidean distance between the hyperplane and the closest data points to the hyperplane [33, 36]. In the binary case, the margin is $\frac{1}{\|\mathbf{w}\|}$. Therefore, our problem becomes:

minimizing $g(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

subject to $y_i(\mathbf{w}.\mathbf{x} + b) \geq 1$

Support vectors are the data points that satisfy the equality above.

However, it is difficult to solve the above optimization problem numerically. The problem is reduced to equivalent problem when Lagrange multipliers α_i and a Lagrangian [32]:

$$L = L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 \sum_{i=1}^{n} \alpha_i(y_i(x_i.\mathbf{w} + b) - 1) \tag{4.1}$$

The Lagrangian L should be minimized with respect to $\mathbf{w}$ and b. It means that partial derivatives $\frac{\partial L}{\partial \mathbf{w}}$ and $\frac{\partial L}{\partial b}$ should be zero at saddle points which leads to:

$$\sum_{i}^{n} \alpha_i y_i = 0 \tag{4.2}$$

and

$$\mathbf{w} = \sum_{i}^{n} \alpha_i y_i x_i \tag{4.3}$$

with complementary conditions

$$\alpha_i (y_i (x_i . \mathbf{w} + b) - 1) = 0 \tag{4.4}$$

Note that α_i is non-zero for i^{th} support vector. According to The Karush-Kuhn-Tucker (KKT) theorem [38], a separating hyperplane should satisfy the condition in Equation 4.4 in order to be an optimal hyperplane. By using Equations 4.2 and 4.3, $\mathbf{w}$ and b are eliminated. We obtain a new problem of finding α_i's which are
minimizing

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{4.5}$$

subject to $\sum_i^n \alpha_i y_i = 0$ and $\alpha_i \geq 0$

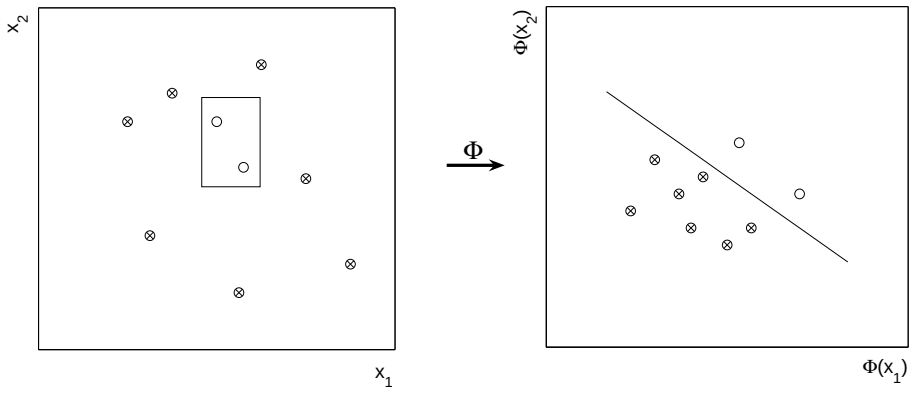


Figure 4.2: Mapping of input space onto feature space.

When hyperplane is non-linear, the input space is mapped onto a high-dimensional space by using kernels. Let $\Phi(\mathbf{x})$ be the mapping which maps input space into a better representation which is called feature space. The mapping is demonstrated in Figure 4.2. So, we can replace $\mathbf{x}_i \cdot \mathbf{x}_j$ by $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ where $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function. Different

kernels can also be chosen for different problems. The suitable kernel makes the data points separable on the feature space when the points are non-separable in the original input space [36]. The mostly used kernels are:

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

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

Soft margin SVMs are used in the cases where the data is noisy. In such cases, slack variables $\xi_i > 0$ are introduced where $\xi_i \geq 1$ if the data point $\mathbf{x}_i$ falls on the wrong side of the hyperplane [36, 33]. In non-seperable case, the problem is minimizing

$$V(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\sum_{i=1}^{n} \xi_i \tag{4.6}$$

subject to $y_i(\mathbf{w}.\mathbf{x} + b) \geq 1 - \xi_i$ and $\xi_i > 0$.

Here, $\sum_{i=1}^{n} \xi_i$ is the bound on the number of the training errors since $\xi_i \geq 1$ for the data points which lie on the wrong side of the hyperplane. C is trade-off parameter for the training error [36].

In this case, the decision function becomes

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{n} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{4.7}$$

In multi-class problems, one-to-one strategy is suitable where there are k(k-1)/2 classifiers to be constructed if there are k classes. Each classifier is used for binary classification of data points from two different categories. For instance, we have data point x_t,and i^{th} and j^{th} classes. We should solve the problem of minimizing

$$V(\mathbf{w}_{ij}, \xi) = \frac{1}{2}\mathbf{w}_{ij} \cdot \mathbf{w}_{ij} + C\sum_{t=1}^{n} (\xi_{ij})_t \tag{4.8}$$

subject to

$\mathbf{w}_{ij} \cdot \mathbf{x}_t + b_{ij} \geq 1 - (\xi_{ij})_t$, if $\mathbf{x}_t$ is in the i^{th} class,

$\mathbf{w}_{ij} \cdot \mathbf{x}_t + b_{ij} \leq -1 + (\xi_{ij})_t$, if $\mathbf{x}_t$ is in the j^{th} class, and $(\xi_{ij})_t > 0$.

The voting strategy is used for classification. Each binary classifier corresponds to a vote. The data point is supposed to be in the class which has the maximum number of votes. If two or more classes have the same number of votes, the class with the smallest index is chosen [39].

4.1.2 Support Vector Regression

In regression, the problem is to guess the functional dependence of dependent variable $y \in \Re$ on an m-dimensional independent variable $\mathbf{x}$. So, there is a mapping from $\Re^m$ to $\Re$ which will lead to approximate a real valued function [40].

Initially, there is a data set of n input-output pairs such as $D = \{[\mathbf{x}_i, y_i] | \mathbf{x}_i \in \Re^m, y_i \in \Re, i = 1, ..., n\}$. In the simplest case, Support Vector Regression (SVR) algorithm tries to approximate the function

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{4.9}$$

In the SVR, we calculate the approximation error rather than the margin as in SVC. The most crucial distinction between SVR and classical regression is that a loss function is used in SVR. Vapnik's ε-insensitive loss function is defined as follows

$$E(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_\varepsilon = \begin{cases} 0 & if\ |y - f(\mathbf{x})| \leq \varepsilon \\ |y - f(\mathbf{x})| - \varepsilon & otherwise \end{cases} \tag{4.10}$$

An ε-tube is described by Equation 4.10. It means that the loss is zero when the predicted value is inside the tube. On the other hand, if the predicted value is outside the tube, the loss becomes the difference between the predicted value and tube's radius ε.

The other mostly used loss functions (Figure 4.3) are:

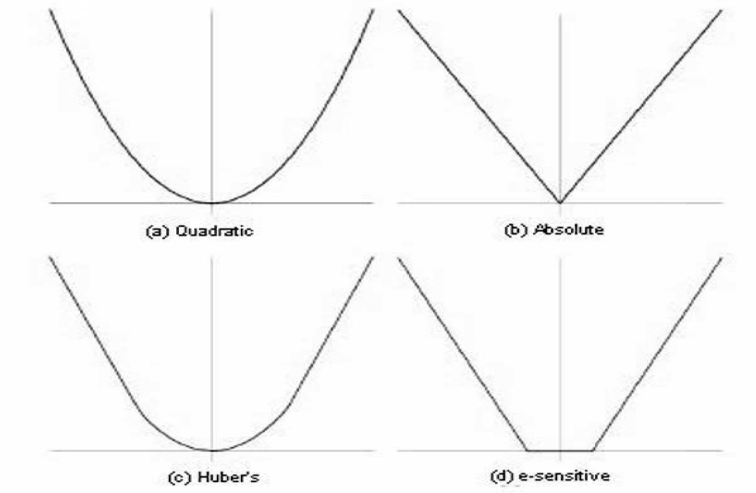


Figure 4.3: Loss functions.

- Absolute error $|y - f(\mathbf{x})|$ which is known as least modulus or L_1 norm

- Quadratic error $(y - f(\mathbf{x}))^2$ which is known as L_2 norm
- Huber's error function [41] in Equation 4.11 which is the most efficient when we know nothing about the model of a noise.

$$E_{Huber}(\mathbf{x}, y, f) = \begin{cases} \frac{1}{2}(y - f(\mathbf{x}))^2 & if\ |y - f(\mathbf{x})| \leq \mu \\ \mu|y - f(\mathbf{x})| - \frac{\mu^2}{2} & otherwise \end{cases} \tag{4.11}$$

By using the loss function, the regression problem becomes minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}|y - f(\mathbf{x})|_\varepsilon \tag{4.12}$$

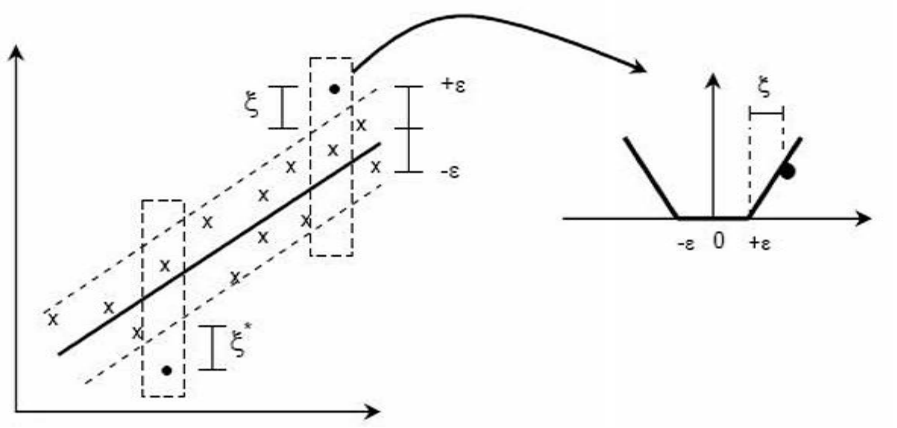


Figure 4.4: ε-tube, slack variables ξ, ξ^* and ε-insensitive loss function are illustrated [9].

If we introduce new slack variables ξ_i and ξ_i^* as in Figure 4.4, the risk function in Equation 4.12 is equal to minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{4.13}$$

subject to

$$\begin{aligned} &y_i - f(\mathbf{x}) \leq \varepsilon + \xi_i \\ &f(\mathbf{x}) - y_i \leq \varepsilon + \xi_i^* \\ &\xi_i, \xi_i^* \geq 0 \end{aligned} \tag{4.14}$$

where $i = 1, ..., n$.

At least, one of the Lagrange multipliers α_i or α_i^* will be zero for input points which are below and above the ε-tube respectively since a point cannot be below and above the ε-tube

at the same time. Furthermore, both α_i and α_i^* are zero for the training points inside the tube. Therefore, $\alpha_i \alpha_i^* = 0$ [40].

Moreover, C in Equation 4.12 and 4.13 is a constant which is a trade-off between error of approximation and model complexity. A large C forces slack variables to be smaller and decreases error. On the other hand, ε controls the radius of the ε-tube. It also determines the number of support vectors which are located on or outside the ε-tube. As ε increases, the number of support vectors decrease. If ε becomes very large, there will be no support vectors. In this case, predictions cannot be valid [9].

As in SVC, the following Lagrangian function is formed to solve the optimization problem:

$$\begin{aligned} L(\mathbf{w}, b, \xi, \alpha) &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}(\beta_i \xi_i + \beta_i^* \xi_i^*) \\ &\quad - \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + \mathbf{w}\cdot\mathbf{x}_i + b) \\ &\quad - \sum_{i=1}^{n}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \mathbf{w}\cdot\mathbf{x}_i - b) \end{aligned} \tag{4.15}$$

where $\alpha_i, \alpha_i^*, \beta_i, \beta_i^* \geq 0$. Lagrangian $L(\mathbf{w}, b, \xi, \alpha)$ should be minimized in order to find saddle points:

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\mathbf{x}_i = 0 \tag{4.16}$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial b} = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \tag{4.17}$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{4.18}$$

$$\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \xi_i^*} = C - \alpha_i^* - \beta_i^* = 0 \tag{4.19}$$

By substituting the KKT conditions into Lagrangian in Equation 4.15, the problem becomes maximizing

$$\begin{aligned} L_d(\alpha_i, \alpha_i^*) &= -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\quad -\varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i \end{aligned} \tag{4.20}$$

subject to

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \tag{4.21}$$

$$0 \leq \alpha_i, \alpha_i^* \leq C \tag{4.22}$$

where $i = 1, ..., n$. The dual variables β_i and β_i^* are eliminated by using Equations 4.18 and 4.19.

At the end of the learning phase, there are n pairs of Lagrangian multipliers (α_i, α_i^*), and the number of nonzero multipliers defines the number of support vectors. It should be noted that the number of support vectors does not depend on the dimensions of the input space.

For the optimal solution, the following KKT conditions should be satisfied:

$$\alpha_i(\mathbf{w} \cdot \mathbf{x}_i + b - y_i + \varepsilon + \xi_i) = 0 \tag{4.23}$$

$$\alpha_i^*(-\mathbf{w} \cdot \mathbf{x}_i - b + y_i + \varepsilon + \xi_i^*) = 0 \tag{4.24}$$

$$\beta_i \xi_i = (C - \alpha_i)\xi_i = 0 \tag{4.25}$$

$$\beta_i^* \xi_i^* = (C - \alpha_i^*)\xi_i^* = 0 \tag{4.26}$$

It can be seen that $\xi_i = 0$ and $\xi_i^* = 0$ hold when $0 < \alpha_i, \alpha_i^* < C$. Also, we can combine it with the first two condition as the following

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i - \varepsilon \quad for \quad 0 < \alpha_i < C \tag{4.27}$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i + \varepsilon \quad for \quad 0 < \alpha_i^* < C \tag{4.28}$$

The above equations allow us to compute b. However, it is stated in [40] that b should be calculated by averaging over the *free support vectors* since the computation of b is sensitive. Note that the *free support vectors* are the points where Lagrange multipliers are nonzero and less than C. On the other hand, if $\alpha_i = C$ or $\alpha_i^* = C$ for the points below and above the ε-tube respectively, these points are called *bounded support vectors*.

The optimal weight vectors are obtained after calculating α_i and α_i^*:

$$\mathbf{w} = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\mathbf{x}_i \tag{4.29}$$

Also the best regression hyperplane can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\mathbf{x}_i \cdot \mathbf{x} + b \tag{4.30}$$

In the non-linear case, the same idea in non-linear SVC is used. The space is mapped into a high-dimensional space by using $\Phi : \Re^m \rightarrow \Re^f$ where $f > m$. For simplicity, kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ is preferred. As a result, the regression function is obtained as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}_j) + b \tag{4.31}$$

$$f(\mathbf{z}) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{z}) + b \tag{4.32}$$

4.1.3 Advantages of SVR

The advantages of SVR are as follows[9]:

- It handles sparse and high-dimensional data.
- It produces unique solutions.
- It works even there is non-linear relationship between the variables.
- It avoids overtraining.

4.2 Partial Least Squares

Partial Least Squares (PLS) regression was developed by Herman Wold in 1966 in order to be used in social science especially economy. However, it has been generally used in computational chemistry since 1980s [42].

The aim of PLS regression is to find dependent variable $\mathbf{Y}$ given independent variables $\mathbf{X}$ and extract their common statistical properties. Ordinary multiple regression can handle this if $\mathbf{Y}$ is a vector and $\mathbf{X}$ has the maximum number of linearly independent columns. The regression method does not work when the number of independent variables is larger than the number of samples because of multicollinearity[1]. There are several methods to avoid multicollinearity such as Principal Component Regression (PCR). In PCR, some variables are eliminated by carrying out Principal Component Analysis (PCA) of $\mathbf{X}$. The resulting principal components of $\mathbf{X}$ are used as predictors of $\mathbf{Y}$. Multicollinearity does not occur since the principal components are orthogonal. Nevertheless, the method has the problem of selecting the optimum subset of regressors. It is difficult to determine choosing components which are appropriate for $\mathbf{Y}$. On the other hand, PLS regression tries to find components of $\mathbf{X}$ which are relevant to $\mathbf{Y}$. Also, these components explain the covariance between $\mathbf{X}$ and $\mathbf{Y}$ as a desirable property. This is a generalization of PCA. After this step, decomposition of $\mathbf{X}$ allows us to predict $\mathbf{Y}$ [42].

4.2.1 PLS Regression Algorithm

Let $\mathbf{X}$ be a $n \times k$ matrix of independent variables and $\mathbf{Y}$ be a $n \times 1$ vector of dependent variables. Steps of PLS algorithm are explained in [43] as follows:

[1]When the correlation between the predictors is of high degree, multicollinearity occurs. In this case, effect of the predictors over prediction becomes hard to separate.

1. Initialize $i = 1$, $\mathbf{X}_1 = \mathbf{X}$ and $\mathbf{Y}_1 = \mathbf{Y}$.

2. For $i = 1, ..., g$, do the Steps

3. The unit vector $\mathbf{w}_i$ is obtained by standardizing the covariance matrix of $\mathbf{X}_i$ and $\mathbf{Y}_i$ as follows
$$\mathbf{w}_i = \frac{\mathbf{X}_i^T \mathbf{Y}_i}{\| \mathbf{X}_i^T \mathbf{Y}_i \|}$$

4. The score vector $\mathbf{t}_i$ is obtained by combining the columns of $\mathbf{X}_i$ with $\mathbf{w}_i$ linearly:
$$\mathbf{t}_i = \mathbf{X}_i \mathbf{w}_i$$

5. The regression coefficient c_i is computed by
$$c_i = \frac{\mathbf{t}_i^T \mathbf{Y}_i}{\| \mathbf{t}_i^T t_i \|}$$
which is the ordinary linear regression of $\mathbf{Y}_i$ on $\mathbf{t}_i$.

6. The vector $\mathbf{p}_i$ is obtained by
$$\mathbf{p}_i = \frac{\mathbf{X}_i \mathbf{t}_i^T}{\| \mathbf{t}_i^T \mathbf{t}_i \|}$$
which is the linear regression of $\mathbf{X}_i$ on $\mathbf{t}_i$

7. Calculate the residuals of $\mathbf{X}_i$ and $\mathbf{Y}_i$ after regressing on $\mathbf{t}_i$ by
$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i \mathbf{p}_i^T$$
and
$$\mathbf{Y}_{i+1} = \mathbf{Y}_i - \mathbf{t}_i c_i$$
respectively.

After the PLS part of the algorithm, the $k \times g$ matrices $\mathbf{W} = [\mathbf{w}_1 ... \mathbf{w}_g]$ and $\mathbf{P} = [\mathbf{p}_1 ... \mathbf{p}_g]$ and $n \times g$ matrix $\mathbf{T} = [\mathbf{t}_1 ... \mathbf{t}_g]$ are constructed. Moreover, the $g \times 1$ vector $\mathbf{C}$ is formed from the regression coefficients $c_1, ..., c_g$. Predicted values of $\mathbf{X}$ and $\mathbf{Y}$ are defined as follows:
$$\hat{\mathbf{X}} = \mathbf{T}\mathbf{P}^T = \sum_{i=1}^{g} \mathbf{t}_i \mathbf{p}_i$$
and
$$\hat{\mathbf{Y}} = \mathbf{T}\mathbf{C} = \mathbf{X}\mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1}\mathbf{C}$$

4.2.2 Advantages of PLS

The advantages of PLS are as follows[44]:

- It can handle the data when the number of attributes is greater than the number of samples.
- There is no need for additional feature selection methods.
- It works extremely fast.

CHAPTER 5

PREDICTING BINDING AFFINITY OF MOLECULES USING PARTIAL LEAST SQUARES AND SUPPORT VECTOR REGRESSION

The aim of this thesis is to predict binding affinities of the molecules which are introduced in Figure 3.1. We assume that binding affinity is related to the topological properties of molecules and their electrostatic potential values. So, we apply regression methods on the model obtained in Chapter 3 for predicting binding affinities. As regression methods, PLS regression and SVR with RBF-kernel and ε-loss function are chosen. The main reason of this choice is that the number of attributes are larger than the number of samples. Data preprocessing, software, performance measures, implementation details and results are explained with this order in the following sections.

5.1 Data Set

The models of 38 compounds which are obtained in Chapter 3 are used in the implementation. Each sample includes 42 attributes corresponding to the electrostatic potential values of 42 neurons of the spherical self-organizing map. It should be noted that the location of points of the tessellated sphere is fixed, and self-organizing maps provide "internally ordered" representation of molecules. These properties of self-organizing maps allow molecules to be represented with respect to their topology and electrostatic potentials on their surfaces. Consequently, each attribute carries topological and electrical information of the molecule which it belongs to.

Prior to the implementation, data is scaled in range $[-1, 1]$ in order to prevent attributes to be rounded up during calculation since attributes are close to 0.

5.2 Software

Support vector regression is implemented with LIBSVM 2.34 developed by Chang and Lin [39]. Partial least squares regression models are calculated by Unscrambler 9.7 30-day trial [45]. Results are evaluated using MATLAB. Support vector regression calculations are carried out on Linux platform. Other calculations are performed on Windows XP Professional.

5.3 Performance

Root mean squared error ($RMSE$) and coefficient of determination (R^2) are chosen as the performance measures of this study.

Let y_i and $\hat{y}_i$ be observed and predicted values of the i^{th} dependent variable respectively. Here, $i = 1, ...N$ where N is the number of samples. Further, $\overline{y}$ is the mean of the observed values. The formulas of $RMSE$ and R^2 are as follows:

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

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

$RMSE$ is a common error measure used in regression problems. It indicates the distance between the observed and the predicted value. On the other hand, R^2 determines the percentage of "variability" in dependent variables explained by the independent variables [46]. In Equation 5.2, the term "variability" is denoted by the sum of squares. Further, SSE is the explained sum of squares and SST is the total sum of squares. In regression problems, R^2 measures how well the approximated values are fitted to the observed values. R^2 ranges between 0 and 1. $R^2 = 1$ means that the variability of y is totally explained by the model. On the other hand, there is no relationship between the observed values and regressors when $R^2 = 0$. In other words, larger R^2 results in better predictions.

5.4 Implementation and Results

5.4.1 Parameter Selection for SVR

Optimal parameters for SVR should be determined in order to obtain reliable results in prediction. These parameters are C, ε and γ. The parameter C controls the relationship between the model complexity and the error tolerance. In other words, it is a trade-off between those two features. For instance, a large C tries to minimize the error regardless of the complexity of the model. ε determines the width of the ε-tube which controls the number of support vectors. It should be noted that the large number of support vectors leads to flat predictions. γ is the width of the RBF-kernel controls the range or distribution of the independent variables in the data [47]. There are two methods of searching for optimal parameters used in this study: practical parameter selection and grid search.

The first method, practical parameter selection, is proposed by Cherkassky and Ma [47]. According to their approach,

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

where $\overline{y}$ is the average of observed dependent variables and σ_y is the standard deviation of dependent variables. The value of ε has a relationship with the noise of the data and the number of the samples. The proposed value of ε is as follows:

$$\varepsilon = \tau\sigma_y\sqrt{\frac{\ln N}{N}} \tag{5.4}$$

where N is the number of samples. Also, $\tau = 3$ works well according to [47].

In this study, $C = 7.8477$ and $\varepsilon = 0.7176$ is found by using Equations 5.3 and 5.4 respectively. After obtaining C and ε, 5-fold cross-validation is performed to determine the value of γ. The data is divided into 5 groups randomly. At each iteration, SVR is trained with 4 groups and tested with the remaining group. $RMSE$ is computed for the test set. The implementation runs for 5 steps. Computed errors are summed up. The result can be seen in Figure 5.1. The best result is obtained for $\gamma = 0.232$. The SVR which uses this parameter set is denoted by SVR-1 in the rest of the implementation.

The second approach is the grid search which is explained in [38, 9]. In grid search, cross-validation is carried out for each set of parameters and error is computed. The parameter set which gives the minimum error is selected. Although, grid search is a popular method for parameter selection, it has some drawbacks. For instance, it is very time consuming. Assume n is the possible values for each parameter. In the case of SVR, there are 3 parameters and it

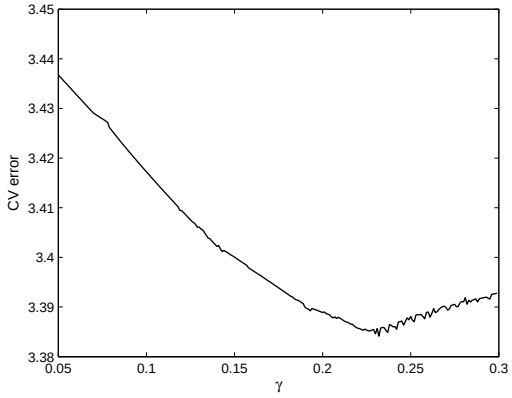


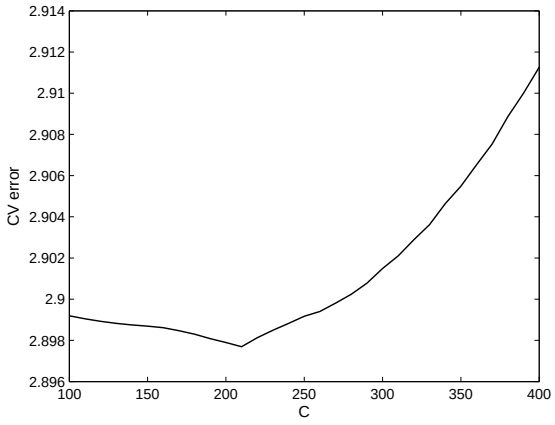
Figure 5.1: Cross-validation error vs. γ when $C = 7.8477$ and $\varepsilon = 0.7176$

has $O(n^3)$ complexity to search for the optimum parameter set. In our experiments, several ranges for parameters C, ε and γ are tried with 5-fold cross-validation. The best results are obtained with parameters $C = 210, \varepsilon = 0.43$ and $\gamma = 0.004$ as in Figure 5.2. The SVR which uses this parameter set is denoted by SVR-2 in the rest of the implementation.

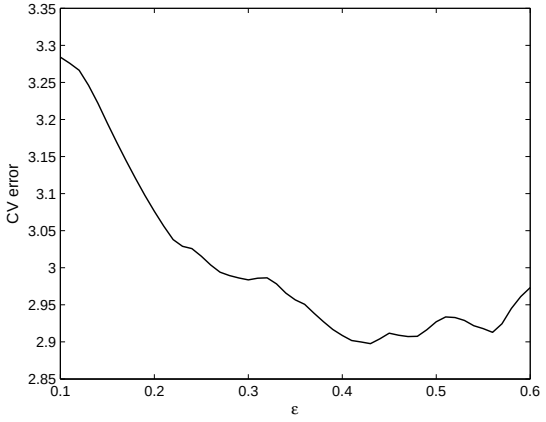
5.4.2 Training and Test Set Selection

After selecting the parameters, we test them on training and testing sets which are chosen with a predefined heuristic and randomly. According to the heuristic, the data is sorted with respect to the descending binding affinity values. The samples with the highest and the lowest output values are put into the training set. 8 samples among 36 remaining ones are chosen for the test set with equal intervals. In other words, the sorted samples with indices $4, 8, 12, ..., 32$ construct the test set. All the remaining samples join the training set. We aim to distribute the binding affinity values among the test and training sets equally. In Table 5.1, the $RMSE$ values of SVR-1 and SVR-2 for several training and testing sets are demonstrated. Random1, Random2 and Random3 are obtained by trying SVR-1 and SVR-2 over at least 1000 randomly generated training and testing sets containing 30 and 8 samples respectively.

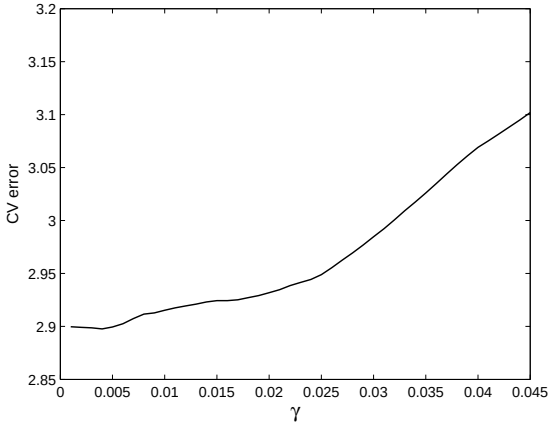
The selection Random3 results in the lowest testing $RMSE$ for both SVR-1 and SVR-2. For test set, SVR-1 outperforms SVR-2 with $RMSE$ values. However, training set performance of SVR-1 is worse than test set performance. Therefore, we will compare the performance of SVR-1, SVR-2 and PLS on Random3 sets since the best prediction results for test set are obtained with Random3.



(a)



(b)



(c)

Figure 5.2: Cross-validation error vs. (a) C when $\varepsilon = 0.43$ and $\gamma = 0.004$, (b) ε when $C = 210$ and $\gamma = 0.004$, (c) γ when $C = 210$ and $\varepsilon = 0.43$

Table 5.1: $RMSE$ values of SVR-1 and SVR-2

	$RMSE$			
	SVR-1		SVR-2	
Selection	Training	Testing	Training	Testing
Heuristic	0.5595	0.5672	0.3716	0.6605
Random1	0.5762	0.4652	0.3792	0.5521
Random2	0.5648	0.6885	0.3663	0.5956
Random3	**0.5889**	**0.4181**	**0.3845**	**0.4566**

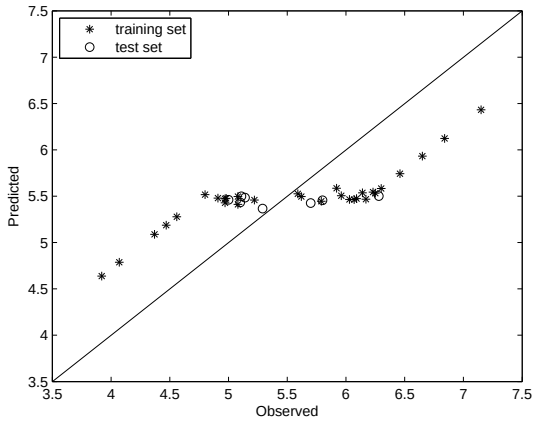
5.4.3 Comparison of SVR-1, SVR-2 and PLS

In this section, the performances of SVR-1, SVR-2 and PLS regression are compared on Random3 training and test sets. When PLS regression is performed on Random3, $RMSE$ values 0.6045 and 0.5022 are obtained for training and test sets respectively. The overall statistical results are revealed in Table 5.2. It is easily seen that SVR-2 outperforms both SVR-1 and PLS while they have similar results. SVR-2 results in both lowest $RMSE$ and highest R^2 among three methods. According to the definition of R^2, nearly 70% of the variability in observed values can be explained by our model using SVR-2. On the other hand, models using SVR-1 and PLS cannot approximate 50% explained variability in binding affinity.

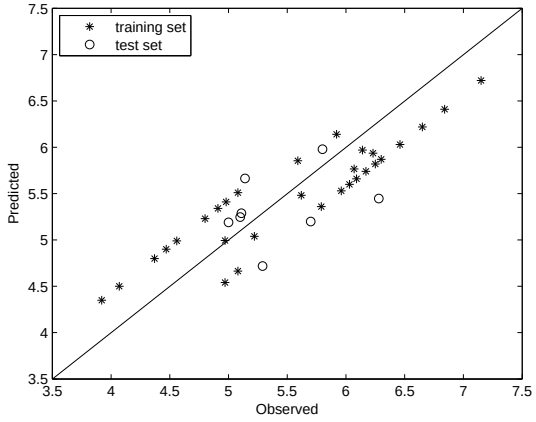
Table 5.2: $RMSE$ and R^2 values of SVR-1, SVR-2 and PLS

	SVR-1	SVR-2	PLS
$RMSE$	0.5573	0.4008	0.5845
R^2	0.4663	0.7240	0.4130

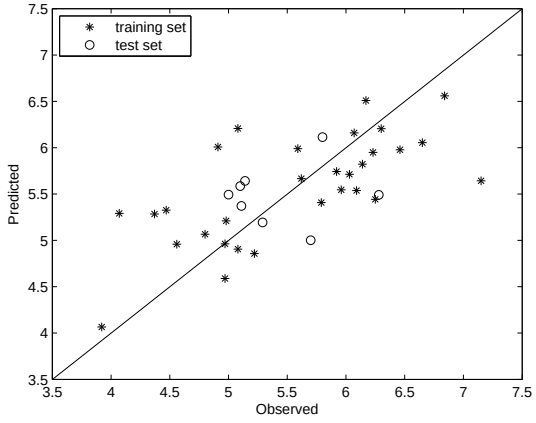
The correlation between observed and predicted binding affinity values obtained by SVR-1, SVR-2 and PLS is represented in Figure 5.3. In Figure 5.3.a, the results obtained by SVR-1 can be seen. The points would be located on the line, if the model perfectly fitted the observation. However, most of the points are not lying on the line. The horizontal distance between the points and the line shows the error between observed and predicted values. It



(a)



(b)



(c)

Figure 5.3: Predicted vs. Observed values obtained by (a)SVR-1, (b)SVR-2, (c)PLS

is observed that SVR-1 tends to approximate the binding affinity values, especially test set values, to the mean of the values since most of the points reside on a horizontal line. In other words, $RMSE$ and R^2 values are sometimes misleading. Results of SVR-2 are demonstrated by Figure 5.3.b. It seems that the error between the observed and the predicted values do not vary much especially in the training set. In test set, the error is distributed unlike SVR-1. Figure 5.3.c shows relationship between observation and prediction obtained by PSL. The deviations in binding affinity appears to be large. Observed and predicted values are shown numerically in Table 5.3 for test set and Table 5.4 for training set.

For all three methods, over-fitting is not significantly observed in the plots and by $RMSE$ values in the training set. However, the methods should be optimized in order to prove their generalization ability. The results also show the importance of parameter selection for SVR. The practical parameter selection is not suitable for this problem. Moreover, grid search does not always provide the best parameters. It is difficult to examine all ranges for parameters because the search is very time consuming. Also, different prediction methods can be used such as neural networks and fuzzy systems. At last, improvement of the model obtained in Chapter 3 can lead better results as a future work.

Table 5.3: Observed and predicted values for test set

	Observed	Predicted		
No.		SVR-1	SVR-2	PLS
2	6.28	5.50095	5.44639	5.491
4	5.7	5.42539	5.1989	5.001
5	5.8	5.45547	5.97842	6.114
10	5.11	5.49815	5.28853	5.372
12	5.29	5.36634	4.71811	5.194
25	5	5.46054	5.19025	5.493
32	5.1	5.43247	5.24645	5.586
34	5.14	5.48444	5.66326	5.641

Table 5.4: Observed and predicted values for training set

No.	Observed	Predicted		
		SVR-1	SVR-2	PLS
1	7.15	6.43208	6.71991	5.643
3	6.14	5.53698	5.97033	5.822
6	6.3	5.58247	5.86982	6.205
7	5.92	5.58593	6.14013	5.743
8	5.79	5.44497	5.36015	5.408
9	5.08	5.41149	4.66342	4.906
11	6.23	5.54367	5.93647	5.949
13	4.8	5.51721	5.22994	5.067
14	6.07	5.4631	5.76625	6.16
15	6.25	5.53245	5.81966	5.443
16	6.09	5.47392	5.65973	5.538
17	5.62	5.49673	5.48175	5.666
18	4.37	5.08792	4.79957	5.285
19	4.97	5.42836	4.54004	4.589
20	6.03	5.46546	5.59993	5.713
21	4.91	5.47923	5.34052	6.007
22	5.08	5.49562	5.51015	6.208
23	6.65	5.93231	6.21984	6.055
24	4.47	5.18748	4.89974	5.326
26	4.07	4.7877	4.50017	5.291
27	6.84	6.12284	6.41017	6.56
28	4.98	5.47662	5.41041	5.211
29	5.59	5.52552	5.85383	5.989
30	5.22	5.45843	5.03887	4.857
31	6.46	5.74246	6.03003	5.978
33	5.96	5.50523	5.52983	5.546
35	3.92	4.6377	4.35004	4.066
36	6.17	5.46755	5.7403	6.509
37	4.97	5.47189	4.99364	4.962
38	4.56	5.27743	4.99015	4.958

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this study, we attempt to obtain a model for predicting binding affinity of the molecules. The work consists of two phases:

1. Building a model of molecular electrostatic potential (MEP) surfaces by using spherical self-organizing map.

2. Predicting binding affinity from the obtained model by using support vector regression (SVR) and partial least squares (PLS).

In the first phase, the MEP surfaces of 38 PCP-like compounds are built in order to obtain the Cartesian coordinates of surface points and corresponding electrostatic potential values. However, we decide to build a simpler model since it is not feasible to use coordinates and electrostatic potential values directly. The molecular surfaces are mapped onto a tesselated sphere by using self-organizing map algorithm. The spherical self-organizing map is chosen because of its ability to protect the geometry of the surfaces. Also, it does not have the topological defects as two-dimensional and toric maps do. At the end of the implementation, we obtain a topology preserving model with electrical properties of the molecule. A desirable property of the model is that it reduce the number of attributes by clustering the number of points into a fewer number of neurons.

The model obtained in the first phase is used for predictive purposes in the second phase. SVR and PLS is chosen as prediction methods because they both can handle sparse data in high-dimensional space. In this case, our data includes 38 samples with 42 attributes. The attributes are internally ordered and represent the corresponding electrostatic potential values. The results show that SVR outperforms PLS in means of $RMSE$ and R^2 values. Also, it is observed that parameter selection is a crucial point in SVR.

As future work, there is work to do in both modeling and prediction parts. In this study, we work with electrostatic potential values of the molecules. Although, electrostatic potential is one of the main factors which affect the binding tendency of molecules, there are other important electrical properties such as polarizability, hydrophobicity, and lipophilicity [48]. The model can be improved with these electrical properties. A weighting mechanism such as an evolutionary algorithm can also be used in order to define how much a property influences the binding affinity.

In prediction part, the performance of SVR can be improved by using more effective parameter selection techniques. Further, different machine learning algorithms including fuzzy systems, neural networks etc. can be performed to obtain better prediction results.

REFERENCES

[1] M. Takatsuka, "An application of the self-organizing map and interactive 3-d visualization to geospatial data," in *the 6th International Conference on GeoComputation*, (Brisbane, Australia), 2001.

[2] T. Kohonen, *Self-Organizing Maps*. New York: Springer, 2001.

[3] A. Sangole and G. Knopf, "Geometric representations for high-dimensional data using a spherical sofm," *International Journal of Smart Engineering System Design*, vol. 5, pp. 11–20, January-March 2003.

[4] Y. Wu and M. Takatsuka, "The geodesic self-organizing map and its error analysis," in *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*, (Darlinghurst, Australia, Australia), pp. 343–351, Australian Computer Society, Inc., 2005.

[5] F. Boudjemaï, P. Enberg, and J.-G. Postaire, "Surface modeling by using self organizing maps of kohonen," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2418–2423, Oct. 2003.

[6] A. Leontitsis and A. Sangole, "Estimating an optimal neighborhood size in the spherical self-organizing feature map," *International Journal Of Computational Intelligence*, vol. 2(2), pp. 94–98, 2005.

[7] U. H. J. P. J. S. A. T. S. Anzali, J. Gasteiger and M. Wagener, "The use of self-organizing neural networks in drug design," in *3D QSAR im Drug Design - Volume 2* (G. F. H. Kubinyi and Y. C. Martin, eds.), pp. 273–299, Dordrecht, NL: Kluwer/ESCOM, 1998.

[8] E. Buyukbingol, A. Sisman, M. Akyildiz, F. Alparslan, and A. Adejare, "Adaptive neuro-fuzzy inference system (anfis): A new approach to predictive modeling in qsar

applications: A study of neuro-fuzzy modeling of pcp-based nmda receptor antagonists," *Bioorganic & Medicinal Chemistry*, vol. 15, pp. 4265–4282, 2007.

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

[10] G. Schneider, "Neural networks are useful tools for drug design," *Neural Networks*, vol. 13, pp. 15–16, 2000.

[11] B. B. R. Burbridge, M. Trotter and S. Holden, "Drug design by machine learning: support vector machines for pharmaceutical data analysis," *Computers and Chemistry*, vol. 26, pp. 5–14, 2001.

[12] D. W. F.R. Burden, M.G. Ford and D. Winkler, "Use of automatic relevance determination in qsar studies using bayesian neural networks," *Journal of Chemical Information and Computer Sciences*, vol. 40, pp. 1423–1430, 2000.

[13] S.-S. So and M. Karplus, "Evolutionary optimization in quantitative structure-activity relationship: An application of genetic neural networks," *Journal of Medicinal Chemistry*, vol. 39, pp. 1521–1530, 1996.

[14] B. B. Braunheim and S. D. Schwartz, "Neural network methods for identification and optimization of quantum mechanical features needed for bioactivity," *Journal of Theoretical Biology*, vol. 206, pp. 27–45, 2000.

[15] V. L. S. Benjamin B. Braunheim, Robert W. Miles and S. D. Schwartz, "Prediction of inhibitor binding free energies by quantum neural networks. nucleoside analogues binding to trypanosomal nucleoside hydrolase," *Biochemistry*, vol. 38, pp. 16076–16083, 1999.

[16] D. J. Wild and P. Willett, "Similarity searching in files of three-dimensional chemical structures. alignment of molecular electrostatic potential fields with a genetic algorithm," *Journal of Chemical Information and Computer Sciences*, vol. 36, pp. 159–167, 1996.

[17] J. Gasteiger and X. Li, "Mapping the electrostatic potential of muscarinic and nicotinic agonists with artificial neural networks," *Angewandte Chemie International Edition in English*, vol. 33, pp. 643–646, 1994.

[18] H. Bauknecht, A. Zell, H. Bayer, P. Levi, M. Wagener, J. Sadowski, and J. Gasteiger, "Representation of molecular electrostatic potentials by topological feature maps," *Journal of Chemical Information and Computer Sciences*, vol. 36, pp. 1205–1213, 1996.

[19] H. J. Wolters, "Geometric modeling applications in rational drug design: a survey," *Computer Aided Geometric Design*, vol. 23, no. 6, pp. 482–494, 2006.

[20] *HyperChem Version 5.1.* http://www.hyper.com.

[21] *VegaZZ Version 2.0.8.* http://www.vegazz.net.

[22] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, Sep 1990.

[23] J. Kangas, T. Kohonen, and J. Laaksonen, "Variants of self-organizing maps," *IEEE Transactions on Neural Networks*, vol. 1, pp. 93–99, Mar 1990.

[24] A. Sangole and G. Knopf, "Visualization of randomly ordered numeric data sets using spherical self-organizing feature maps," *Computers & Graphics*, vol. 27, pp. 963–976, 2003.

[25] U. Seiffert and B. Michaelis, "Three-dimensional self-organizing maps for classification of image properties," in *ANNES '95: 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems*, p. 310, 1995.

[26] Y. Wu and M. Takatsuka, "Spherical self-organizing map using efficient indexed geodesic data structure," *Neural Networks*, vol. 19, no. 6, pp. 900–910, 2006.

[27] H. B. A. Zell and H. Bauknecht, "Similarity analysis of molecules with self-organizing surfaces - an extension of the self-organizing map," in *ICNN'94: International Conference on Neural Networks*, pp. 719–724, 1994.

[28] G. Knopf and A. Sangole, "Trinocular data registiration using a three-dimensional self-organizing feature map," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 2863–2868, 2000.

[29] A. Krishnamurthy, S. Ahalt, D. Melton, and P. Chen, "Neural networks for vector quantization of speech and images," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 1449–1457, October 1990.

[30] D. DeSieno, "Adding a conscience to competitive learning," vol. 1, pp. 117–124, IEEE International Conference on Neural Networks, July 1988.

[31] R. Kamimura, *Competitive Learning by Information Maximization: Eliminating Dead Neurons in Competitive Learning*, vol. 2714 of *Lecture Notes in Computer Science*, pp. 99–106. Springer Berlin / Heidelberg, 2003.

[32] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European Conference on Machine Learning (ECML)*, Spring 1997.

[33] T. Joachims, "A statistical learning model of text classification with support vector machines," in *the Conference on Research and Development in Information Retrieval (SIGIR)*, ACM, 2001.

[34] C. Campbell, "Algorithmic approaches to training support vector machnies: A survey," in *ESANN2000*, p. 8, 2000.

[35] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10,number 5, pp. 1048–1054, 1999.

[36] D. Zhang and W. S. Lee, "Question classification using support vector machines," in *the* 26^{th} *Annual International ACM SIGIR conference on Research and Development in Informaion Retrieval*, ACM, 2003.

[37] W.-Q. L. H.-Y. Z. H.-L. W. G.-L. S. Yan-Ping Zhou, Jian-Hui Jiang and R.-Q. Yu, "Boosting support vector regression in qsar studies of bioactivities of chemical compounds," *European Journal of Pharmaceutical Sciences*, vol. 28, no. 4, pp. 344–353, 2006.

[38] A. Smola and B. Schölkopf, "A tutorial on support vector regression," tech. rep., NeuroCOLT2, 1998.

[39] 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.

[40] V. Kecman, *Support Vector Machines - An introduction*, vol. 177 of *Studies in Fuzziness and Soft Computing*, ch. 1, pp. 1–49. Springer-Verlag Berlin Heidelberg, 2005.

[41] S. Gunn, "Support vector machines for classification and regression," tech. rep., ISIC, 1998.

[42] H. Abdi, "Partial least squares (pls) regression," in *Encyclopedia for research methods for the social sciences* (A. B. M. Lewis-Beck and T. Futing, eds.), pp. 792–795, 2003.

[43] B. Jorgensen and Y. Goegebeur, *Partial least squares regression I.* http://statmaster.sdu.dk/courses/ST02, 2007.

[44] A.-L. Boulesteix and K. Strimmer, "Partial least squares: a versatile tool for the analysis of high-dimensional genomic data," *Briefings in Bioinformatics*, vol. 8, no. 1, pp. 32–44, 2006.

[45] *Unscrambler Version 9.7.* http://www.camo.com.

[46] S. Weisberg, *Applied Linear Regression.* Wiley Series in Probability and Statistics, Wiley–Interscience, 3^{rd} ed., 2005.

[47] V. Cherkassky and Y. Ma, "Practical selection of svm parameters and noise estimation for svm regression," *Neural Networks*, vol. 17, no. 1, pp. 113–126, 2004.

[48] S. Moro, M. Bacilieri, B. Cacciari, and G. Spalluto, "Autocorrelation of molecular electrostatic potential surface properties combined with partial least squares analysis as new strategy for the prediction of the activity of human a_3 adenosine receptor antagonists," *Journal of Medicinal Chemistry*, vol. 48, no. 18, pp. 5698–5704, 2005.

Appendix A

VISUALIZATION OF THE MOLECULES

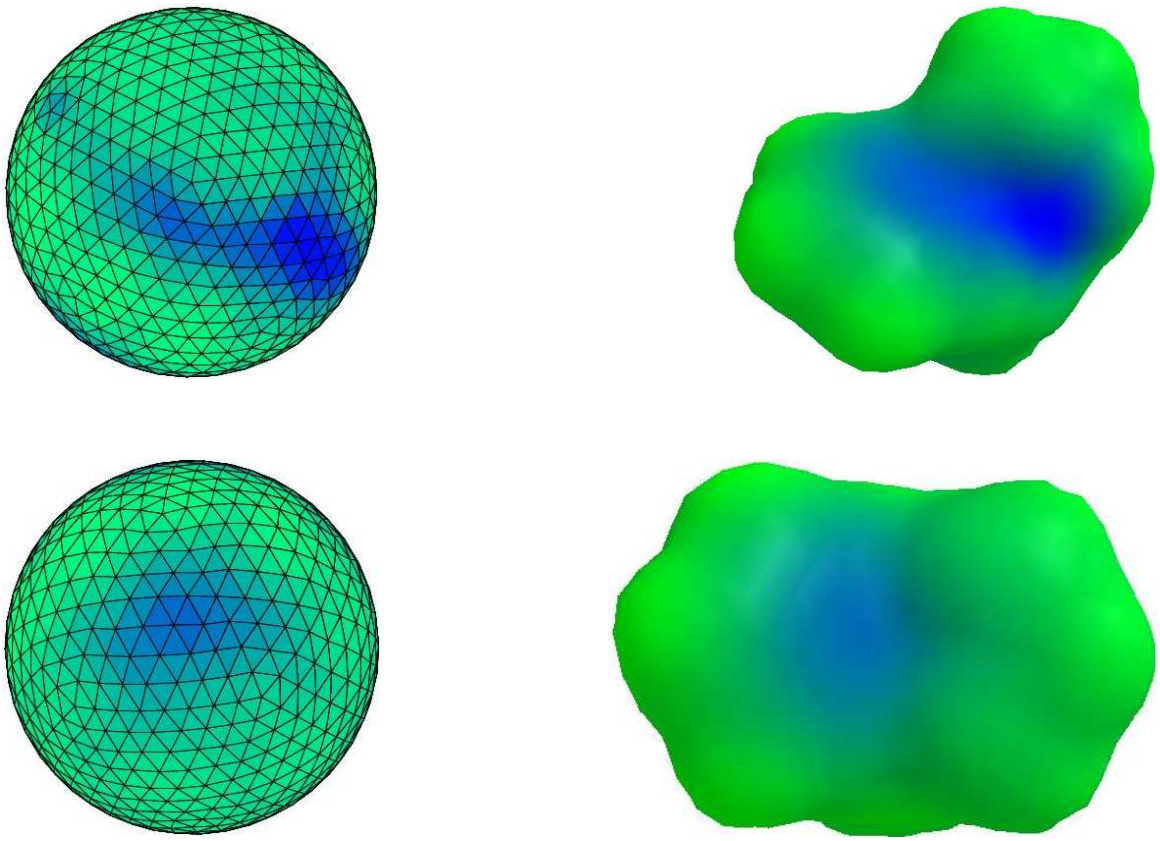


Figure A.1: Molecule 1

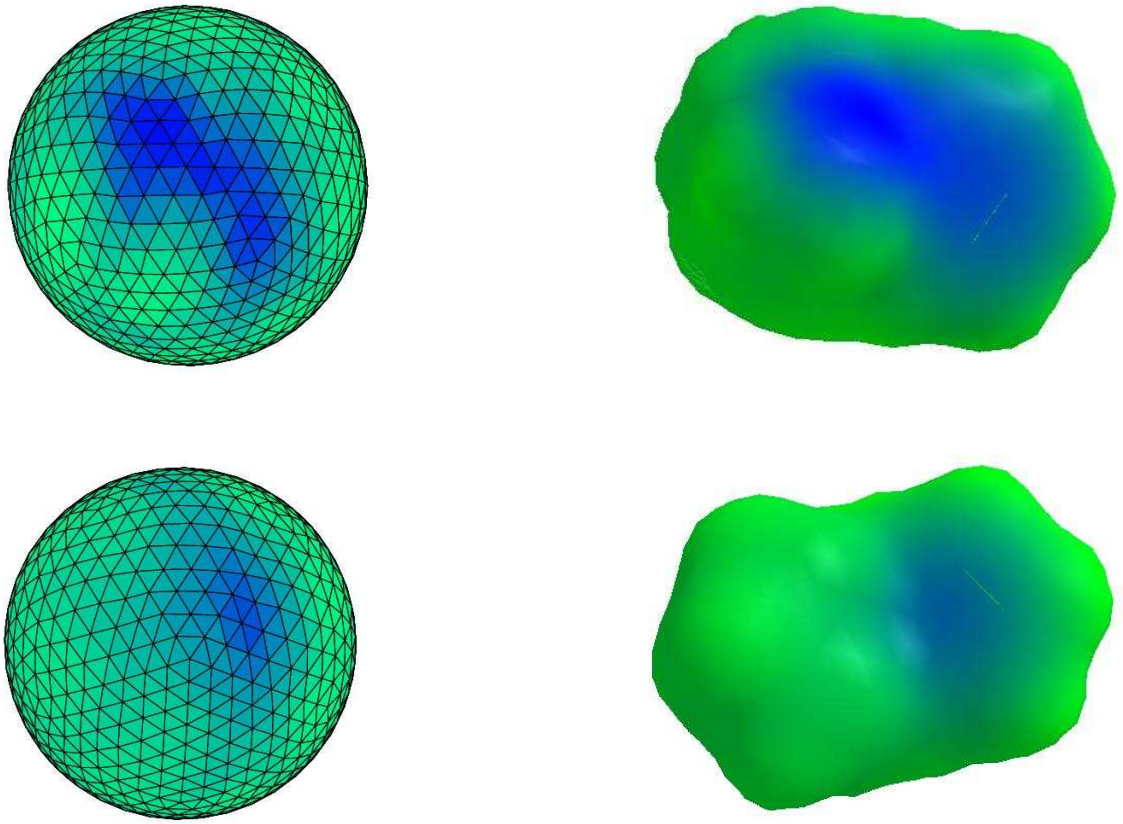


Figure A.2: Molecule 2

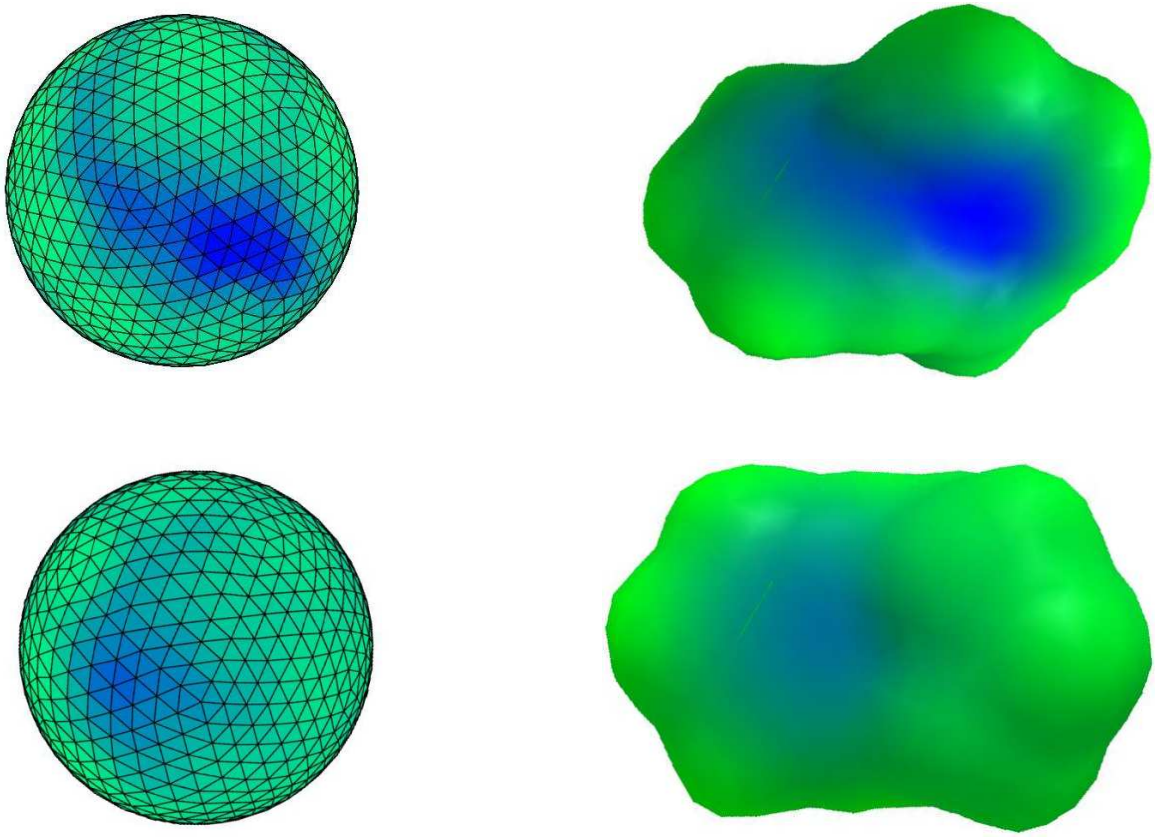


Figure A.3: Molecule 3

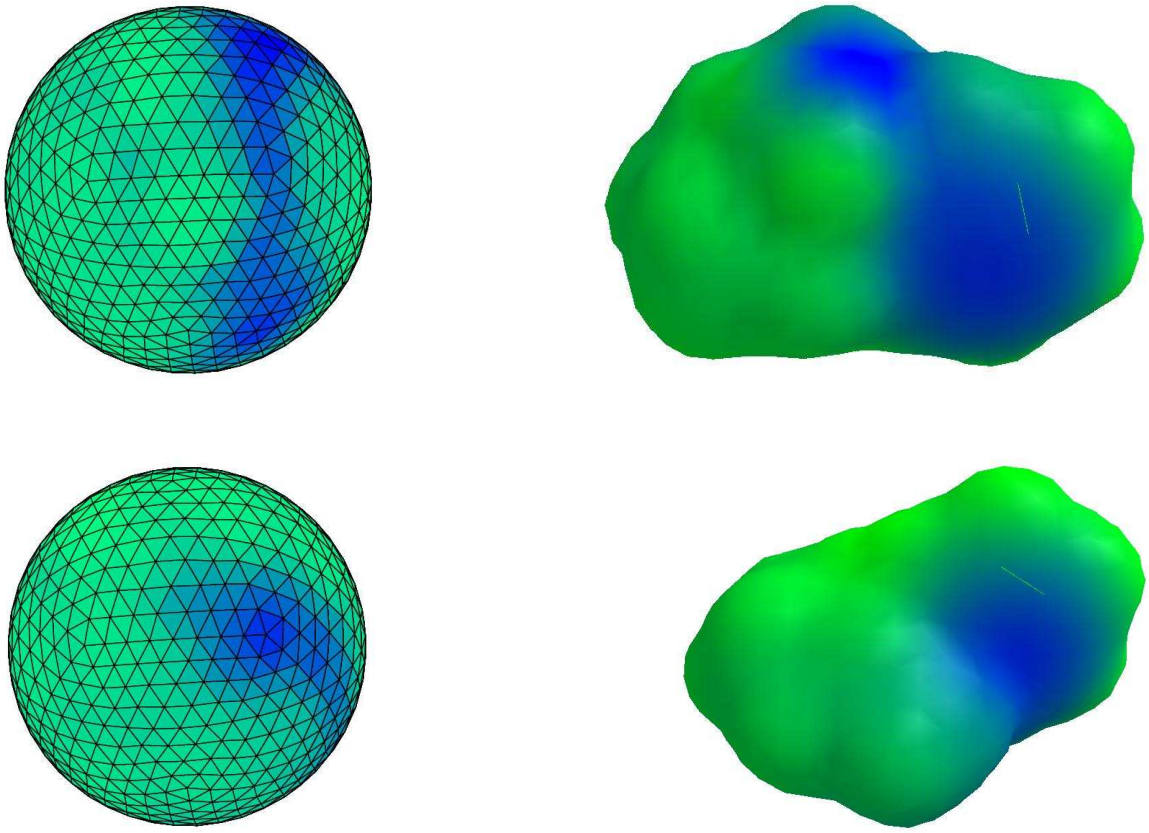


Figure A.4: Molecule 4

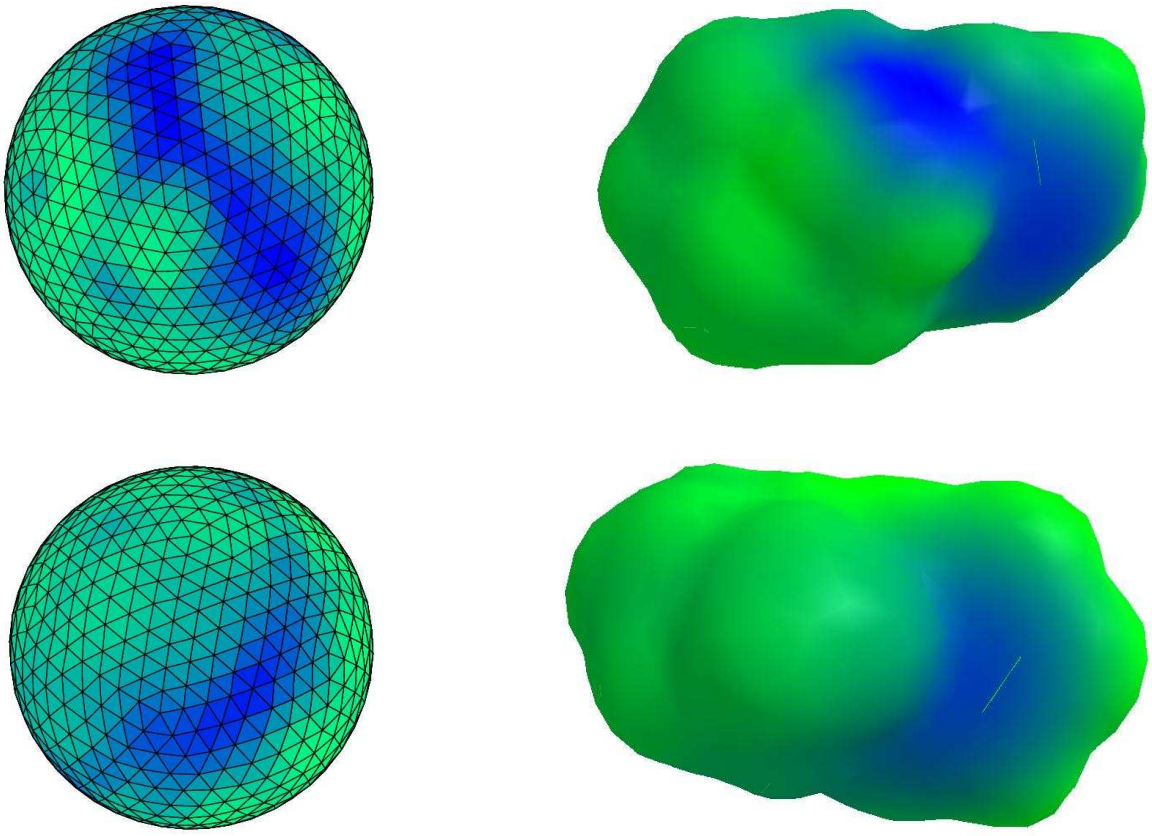


Figure A.5: Molecule 5

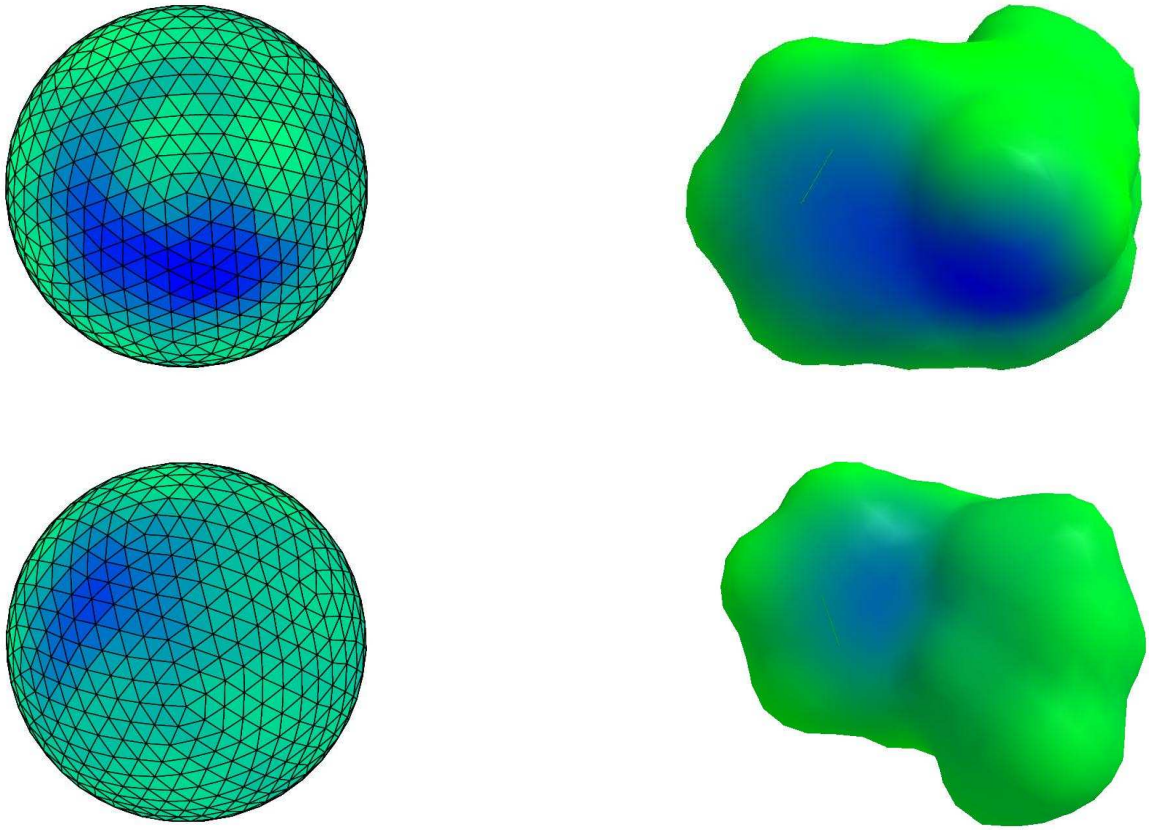


Figure A.6: Molecule 6

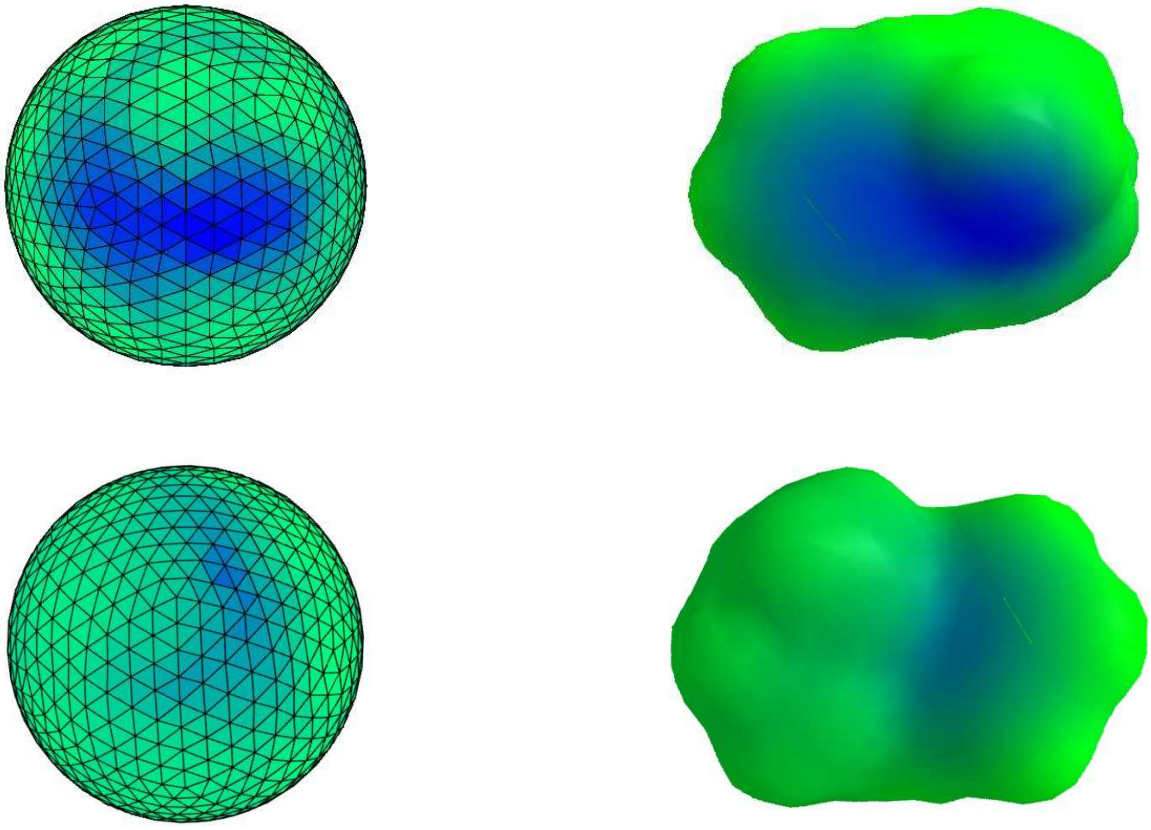


Figure A.7: Molecule 7

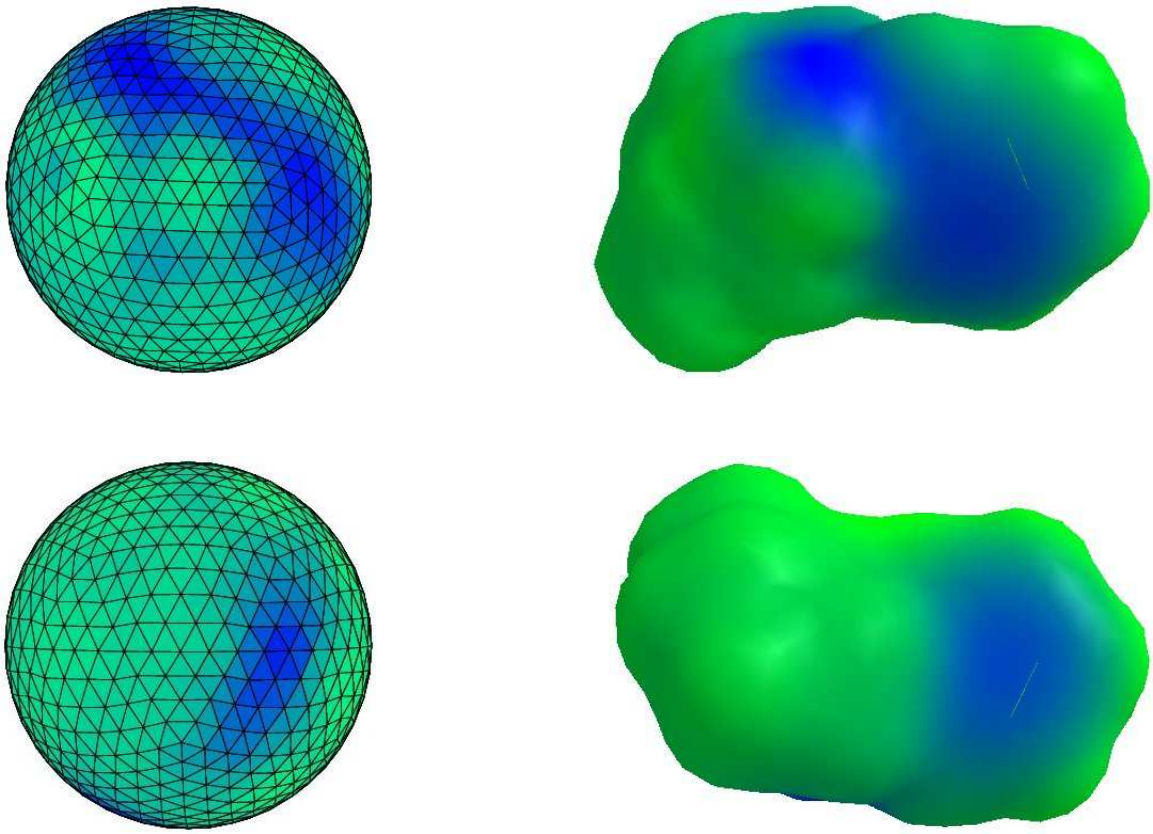


Figure A.8: Molecule 8

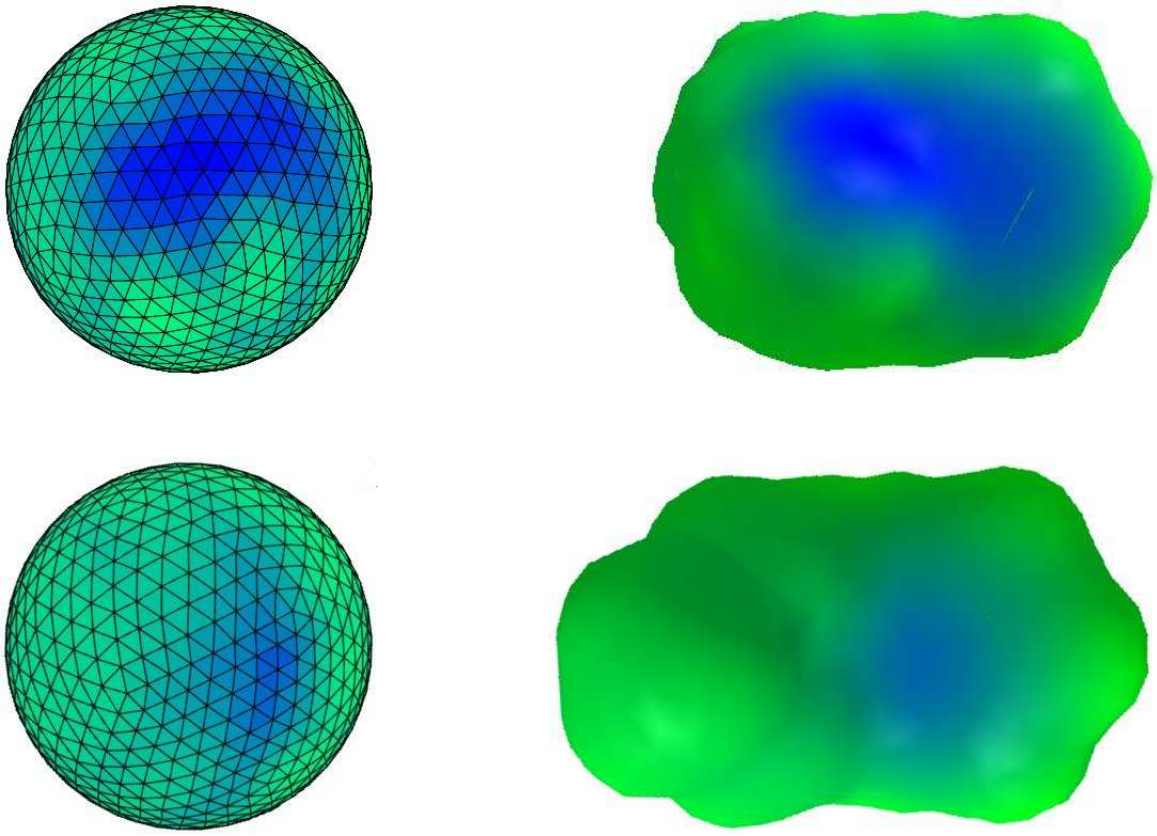


Figure A.9: Molecule 9

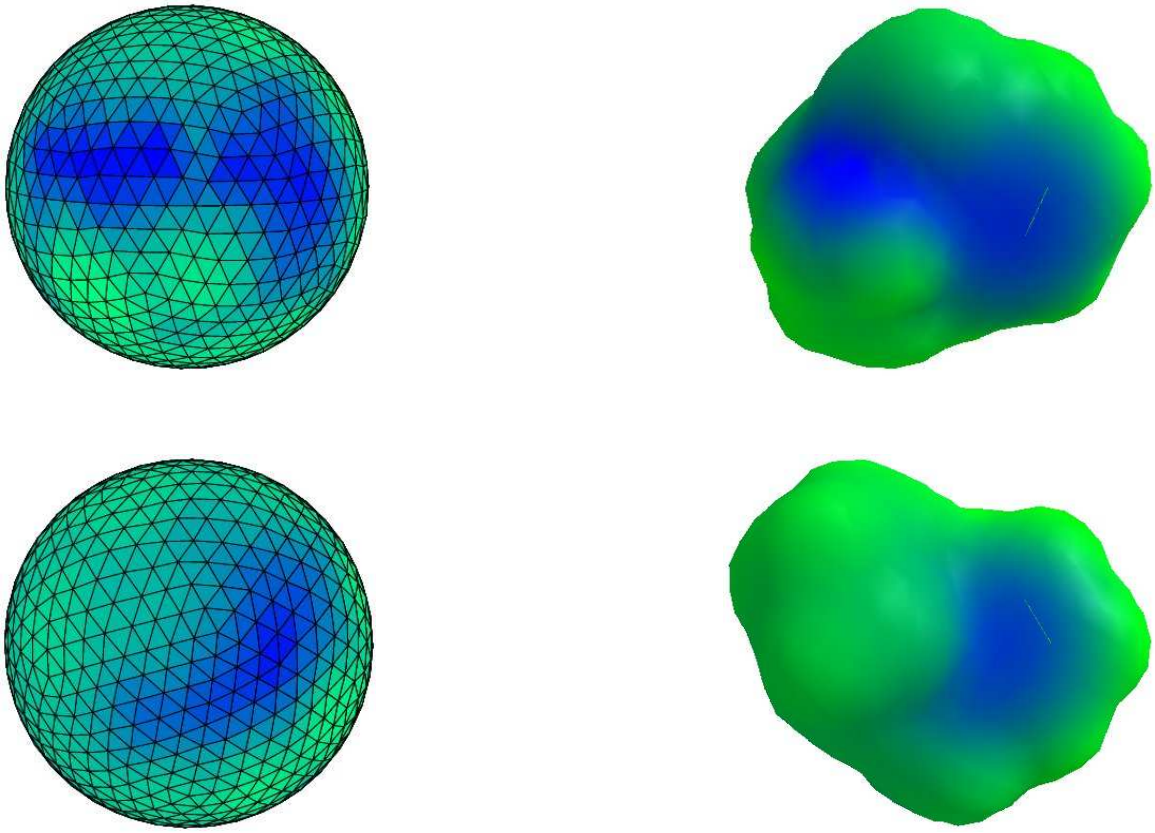


Figure A.10: Molecule 10

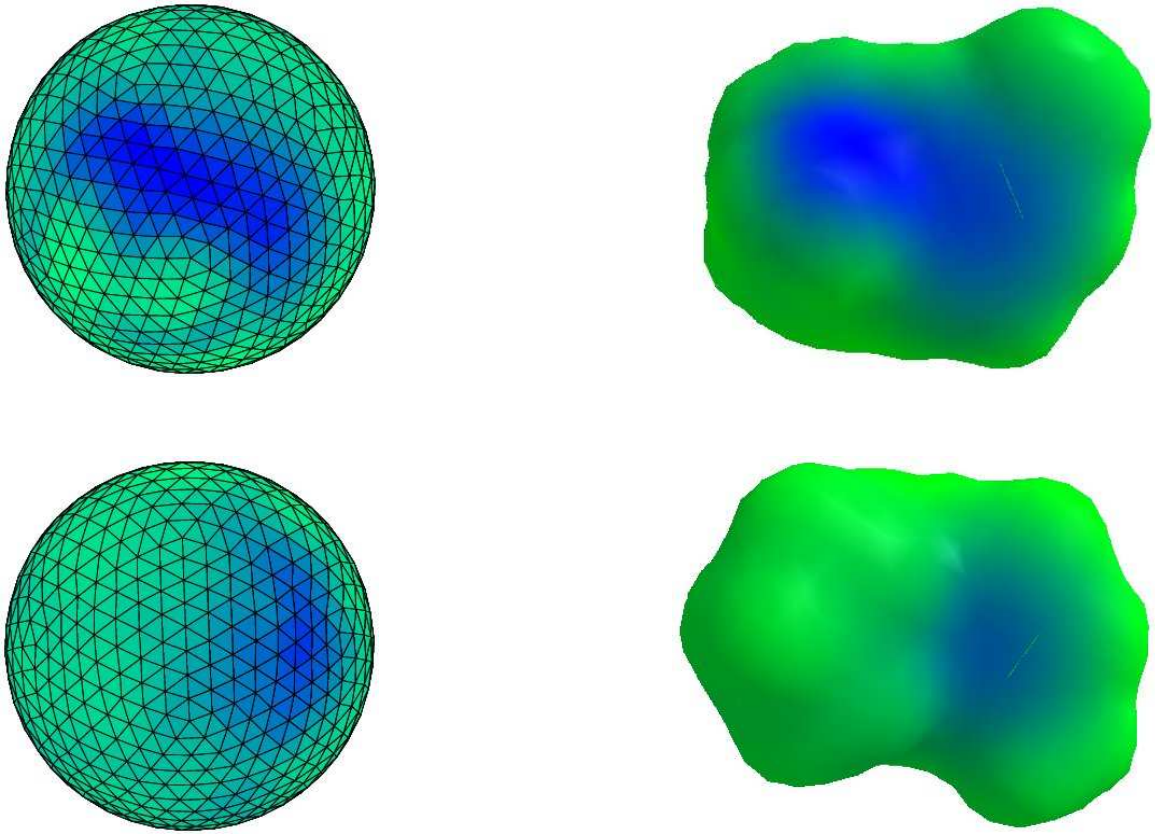


Figure A.11: Molecule 11

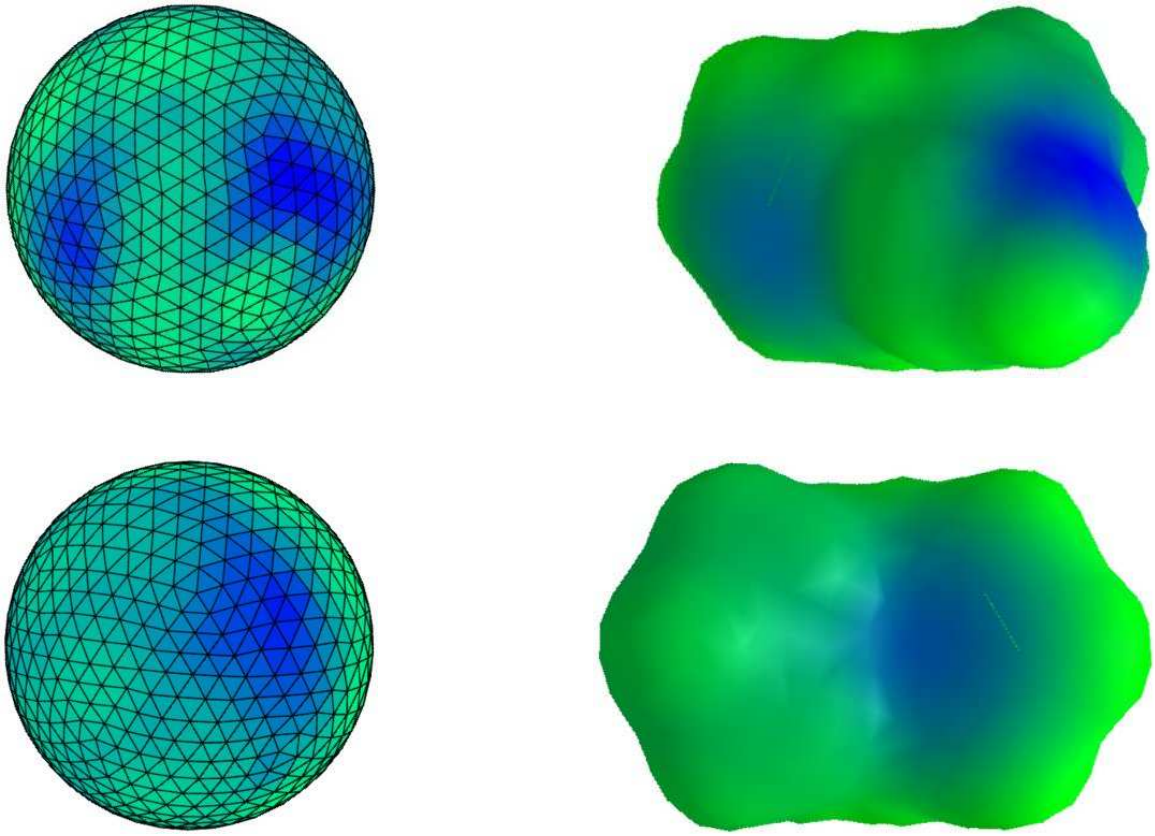


Figure A.12: Molecule 12

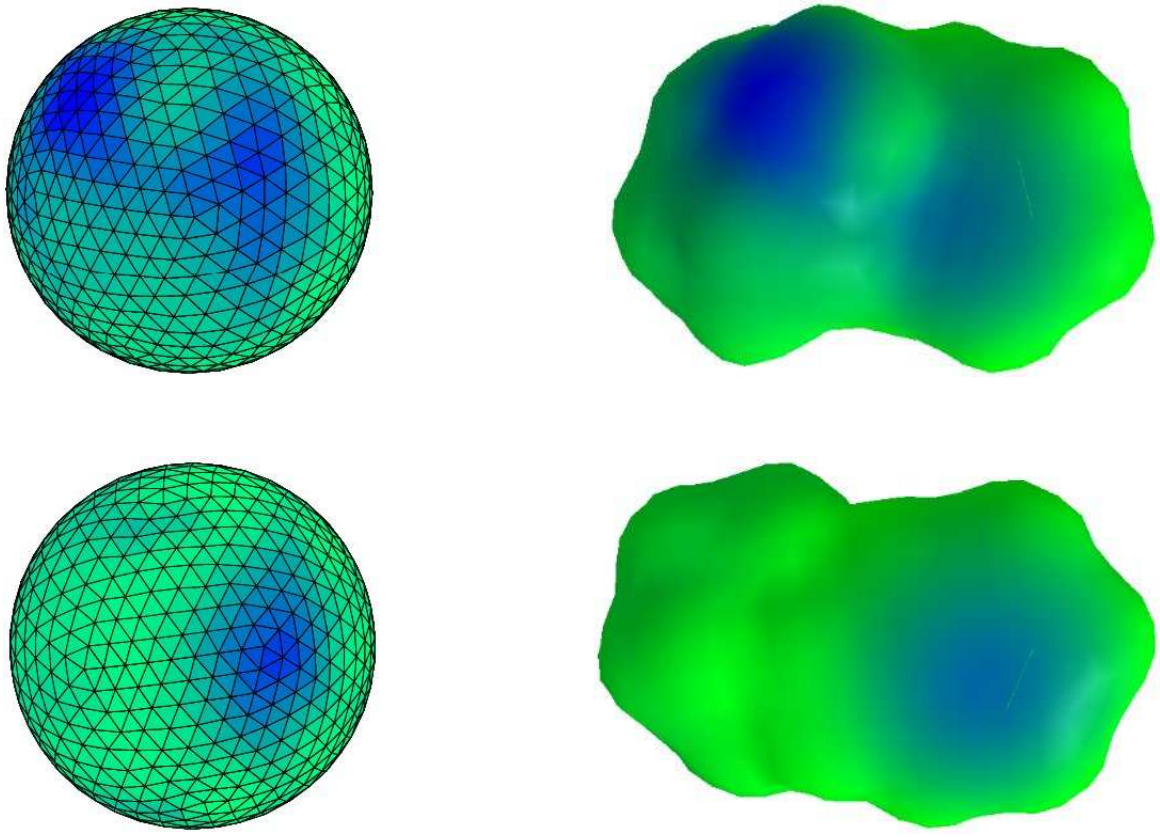


Figure A.13: Molecule 13

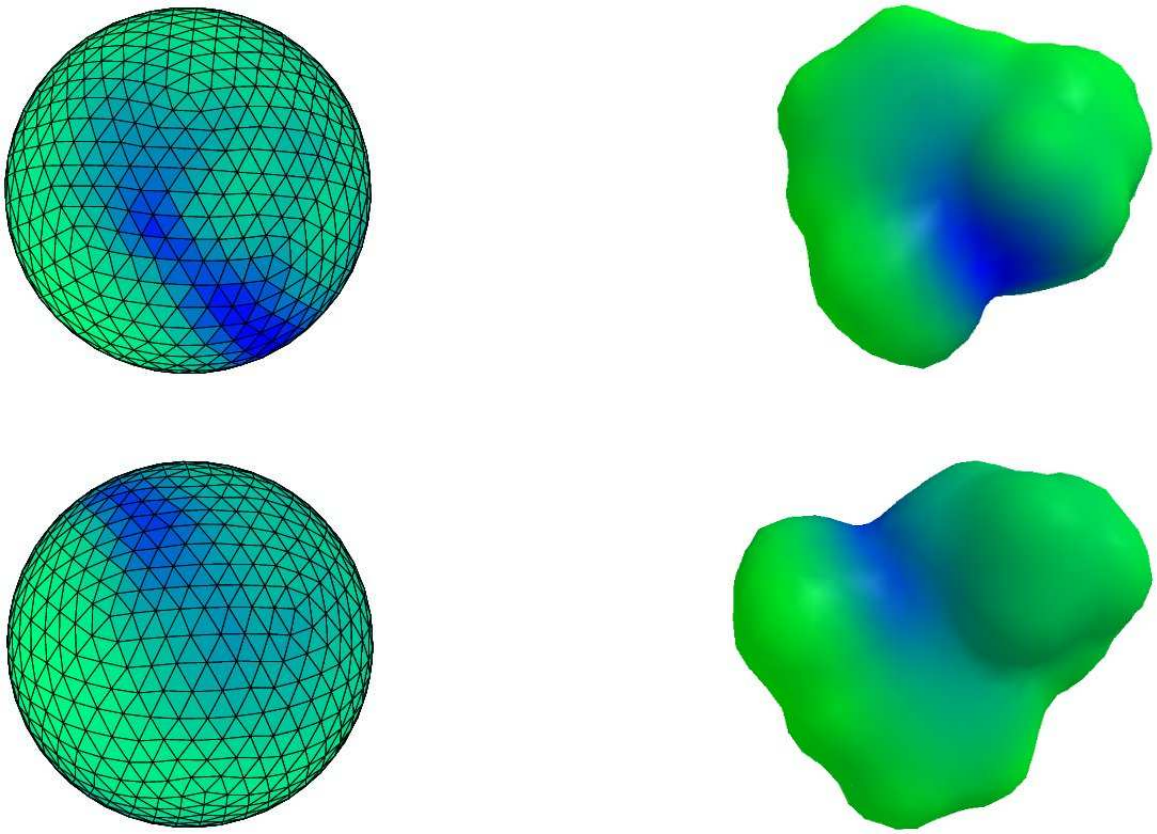


Figure A.14: Molecule 14

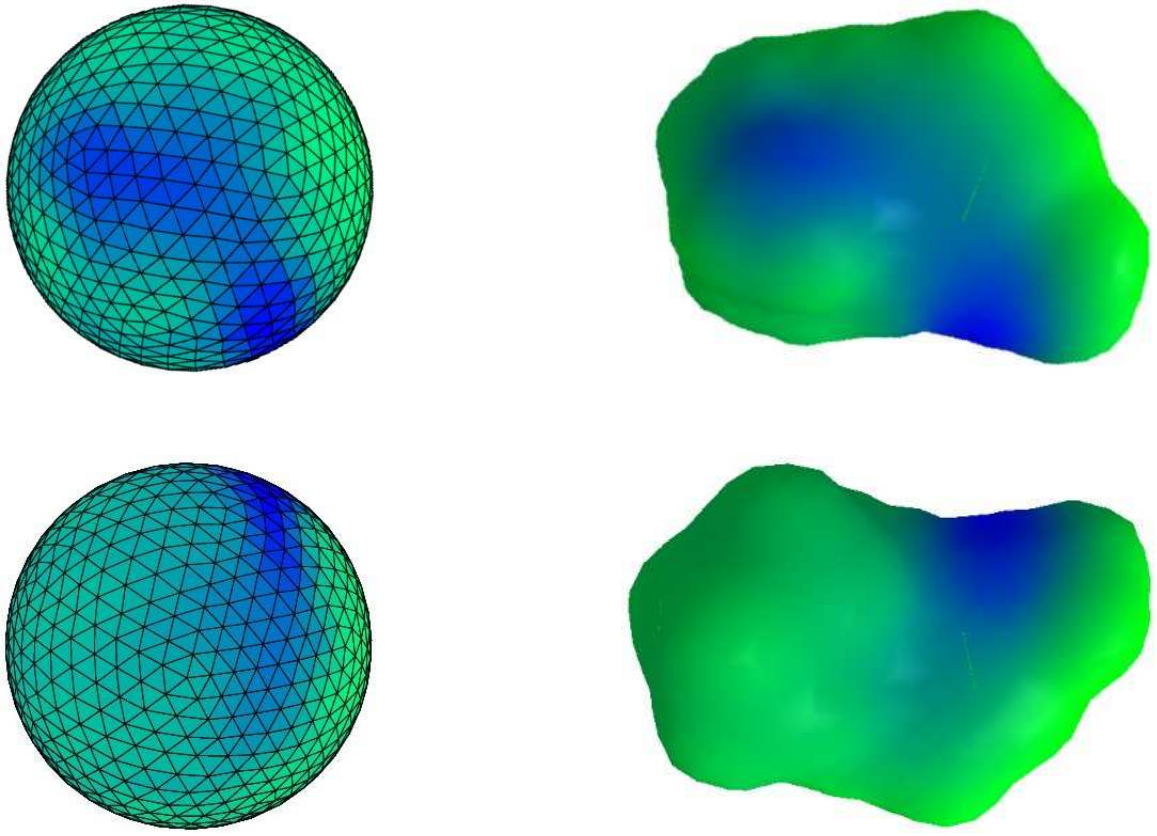


Figure A.15: Molecule 15

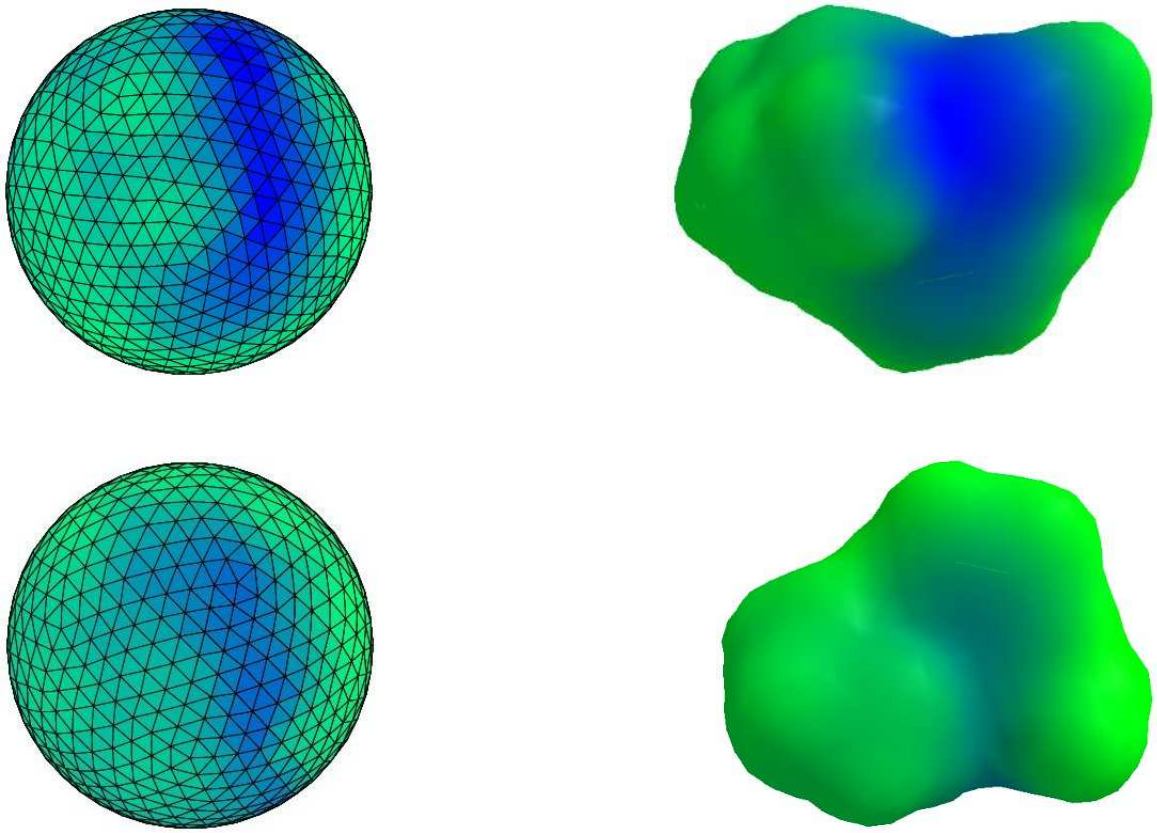


Figure A.16: Molecule 16

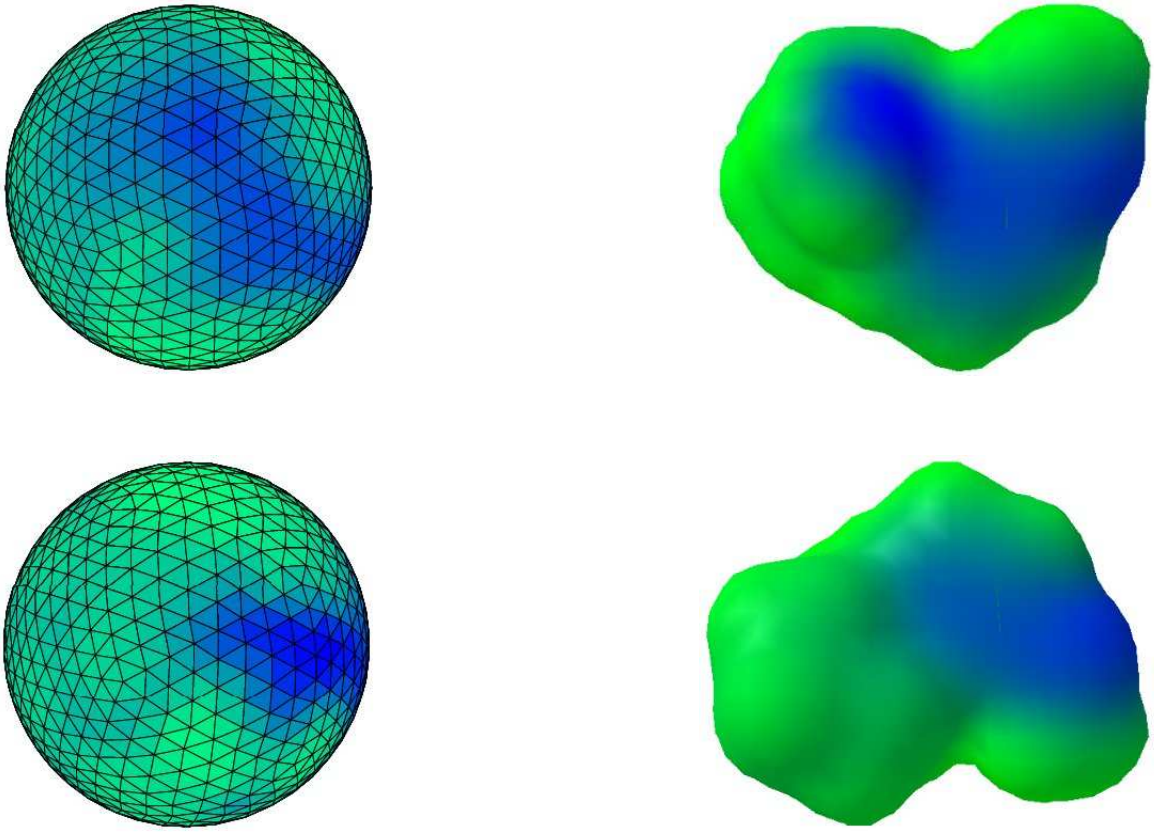


Figure A.17: Molecule 17

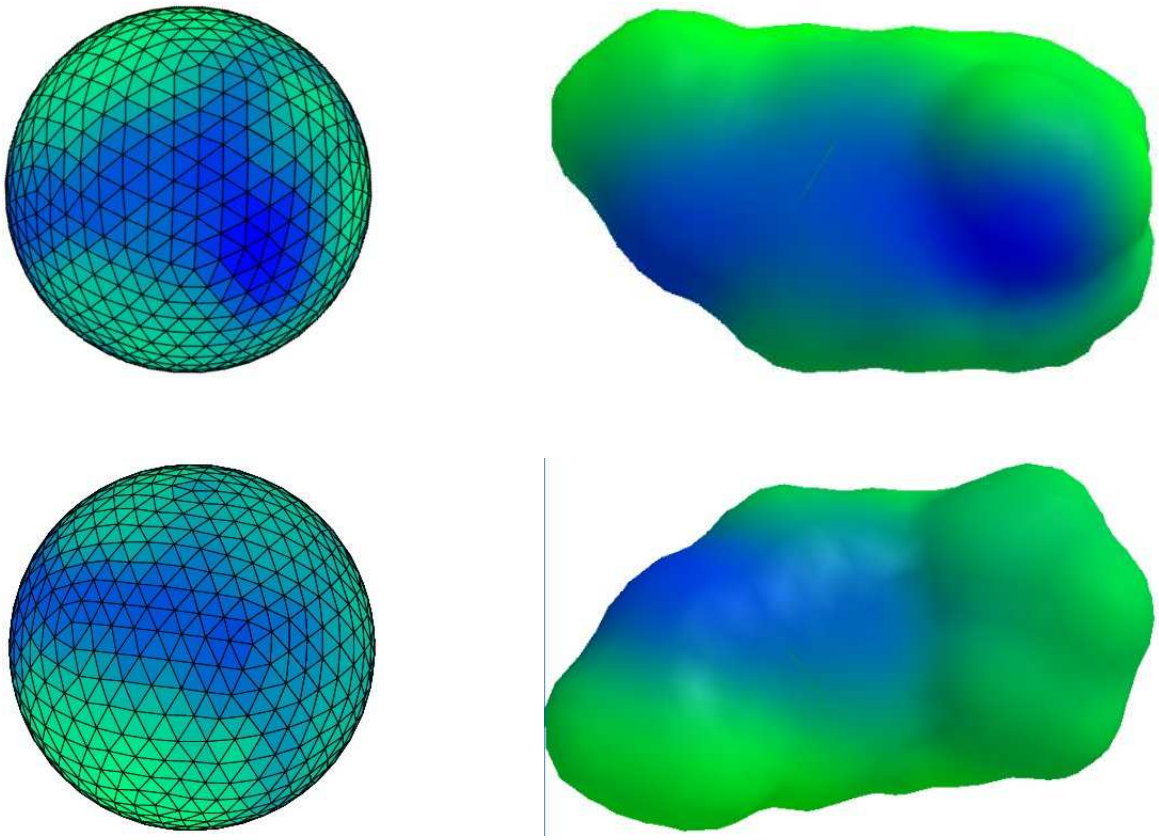


Figure A.18: Molecule 18

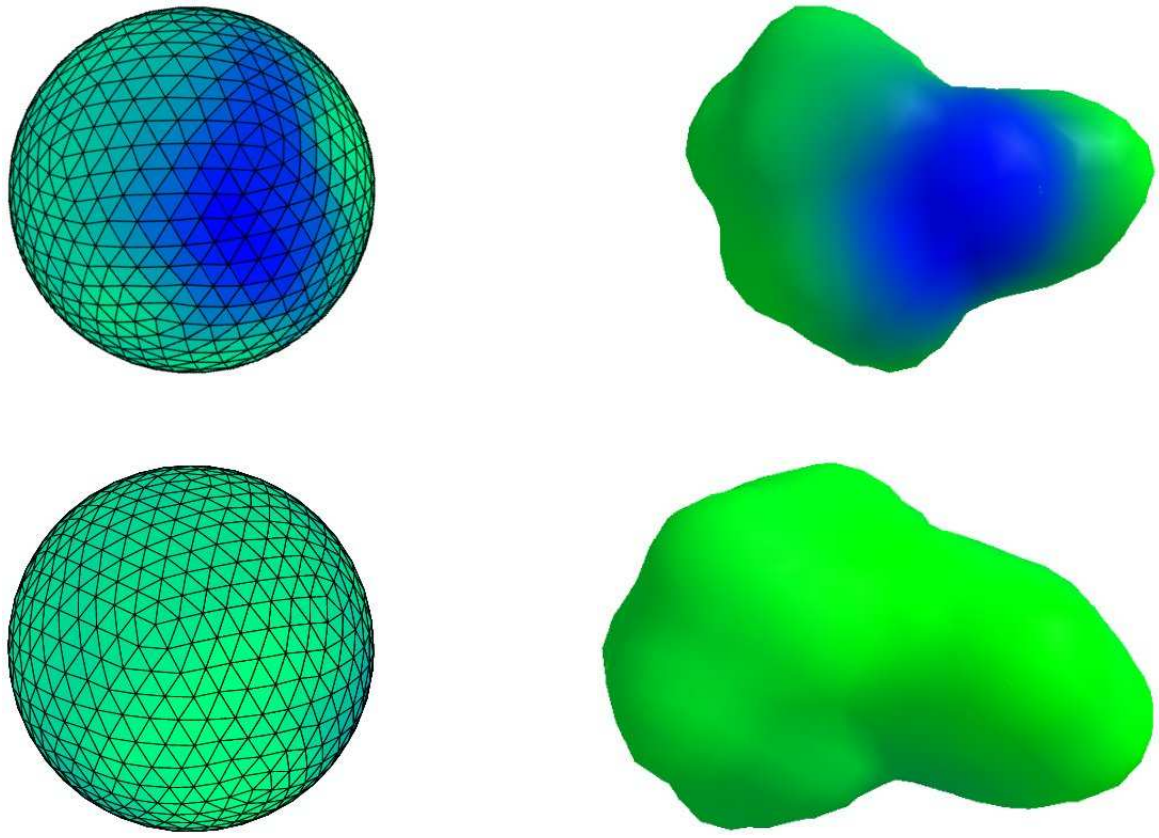


Figure A.19: Molecule 19

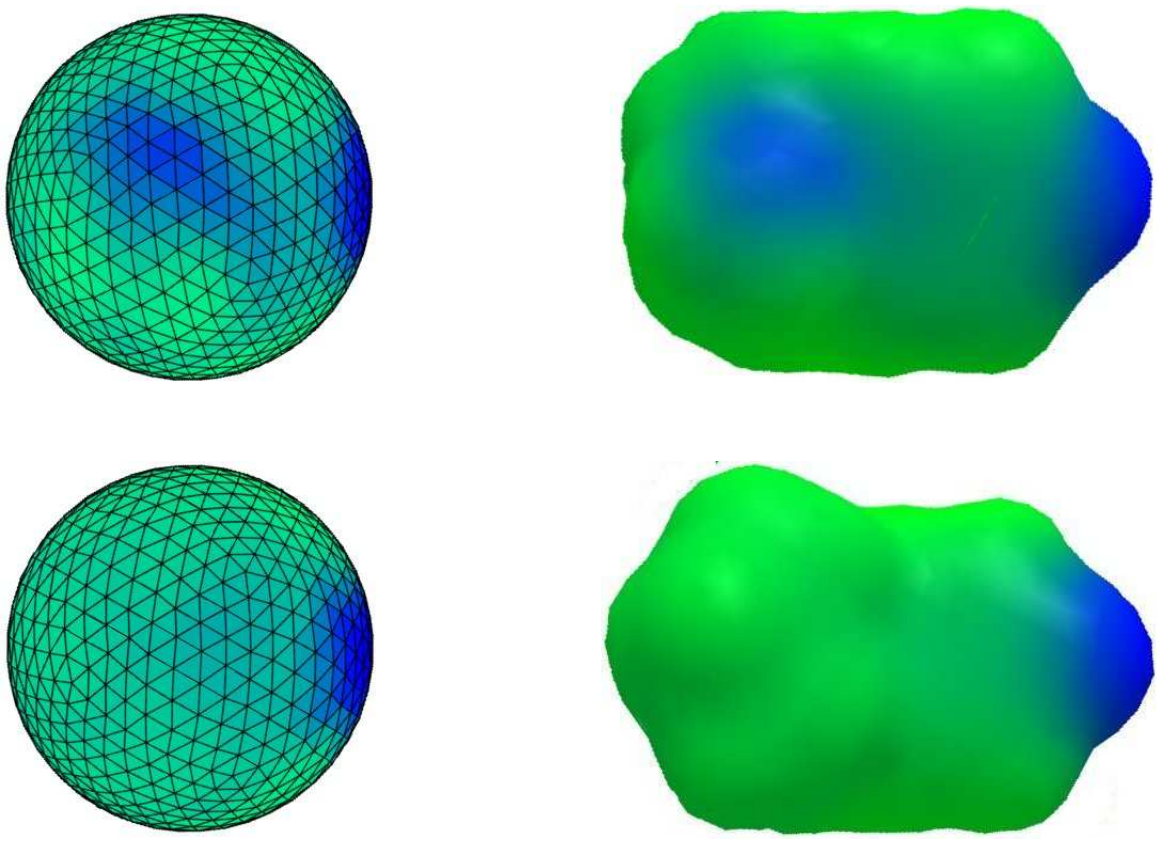


Figure A.20: Molecule 20

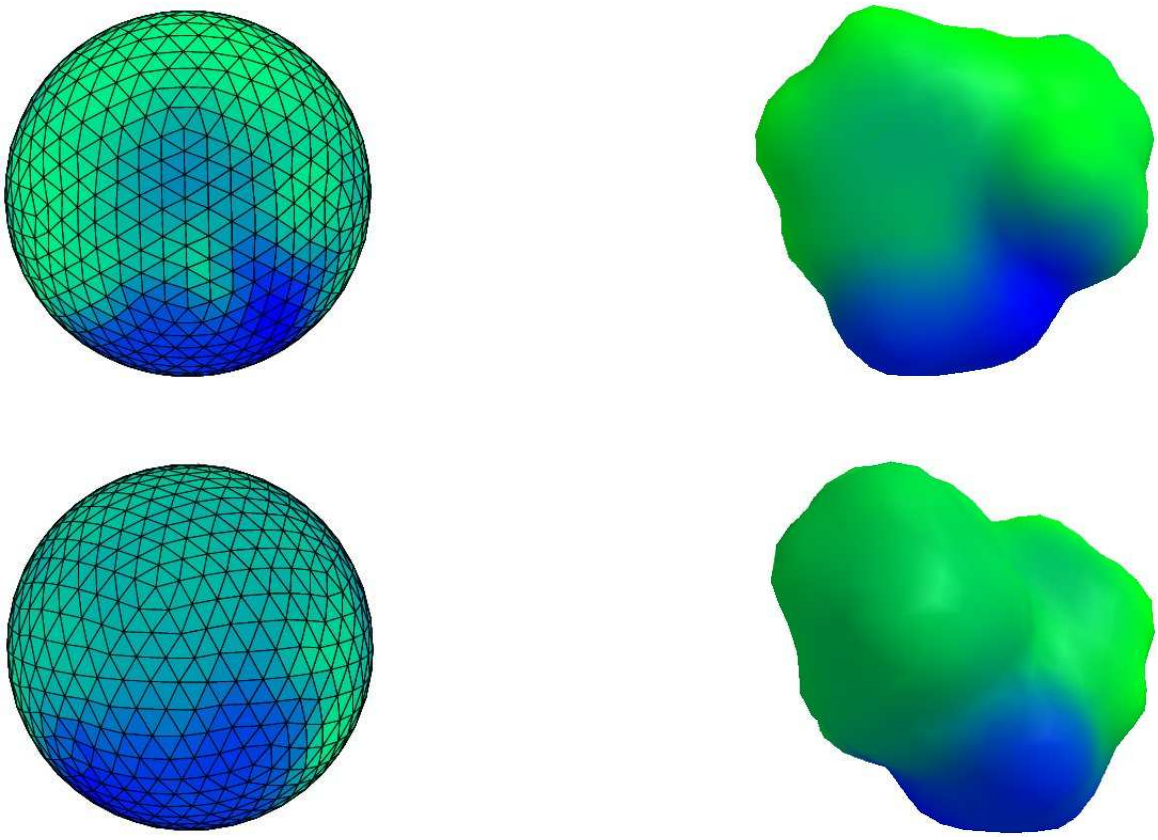


Figure A.21: Molecule 21

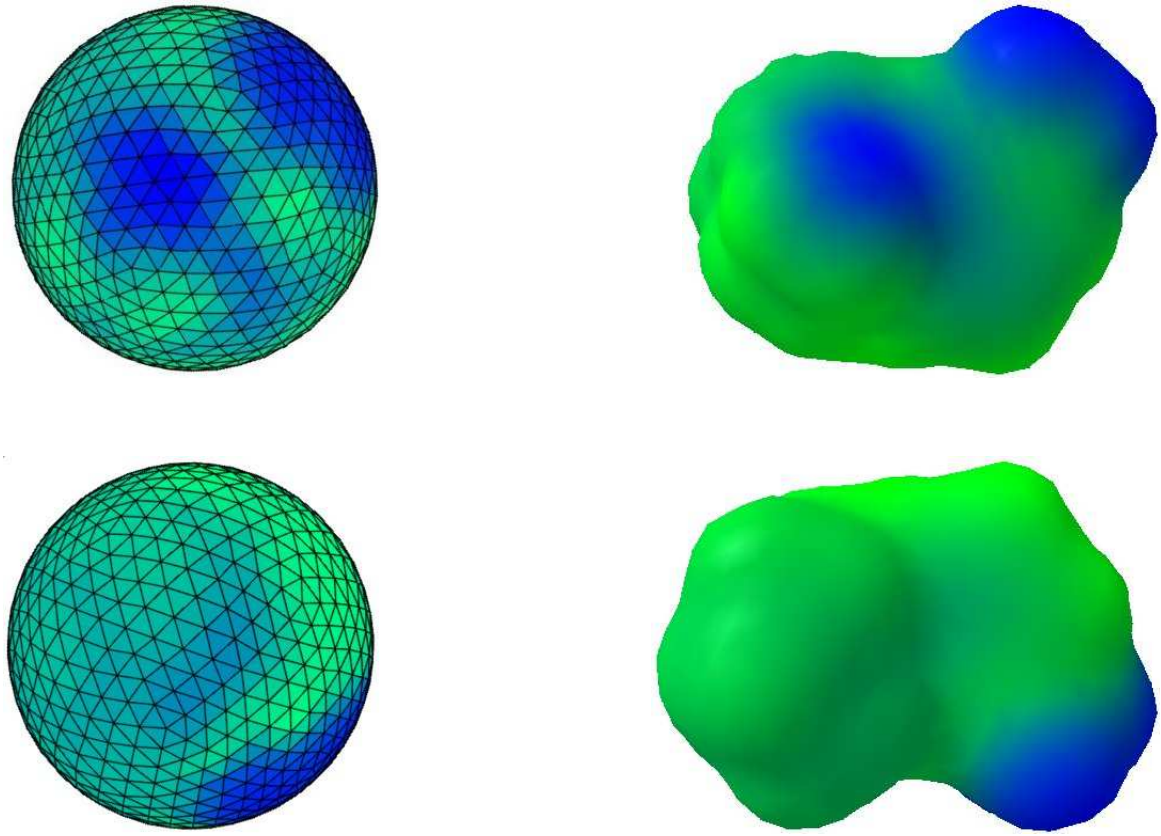


Figure A.22: Molecule 22

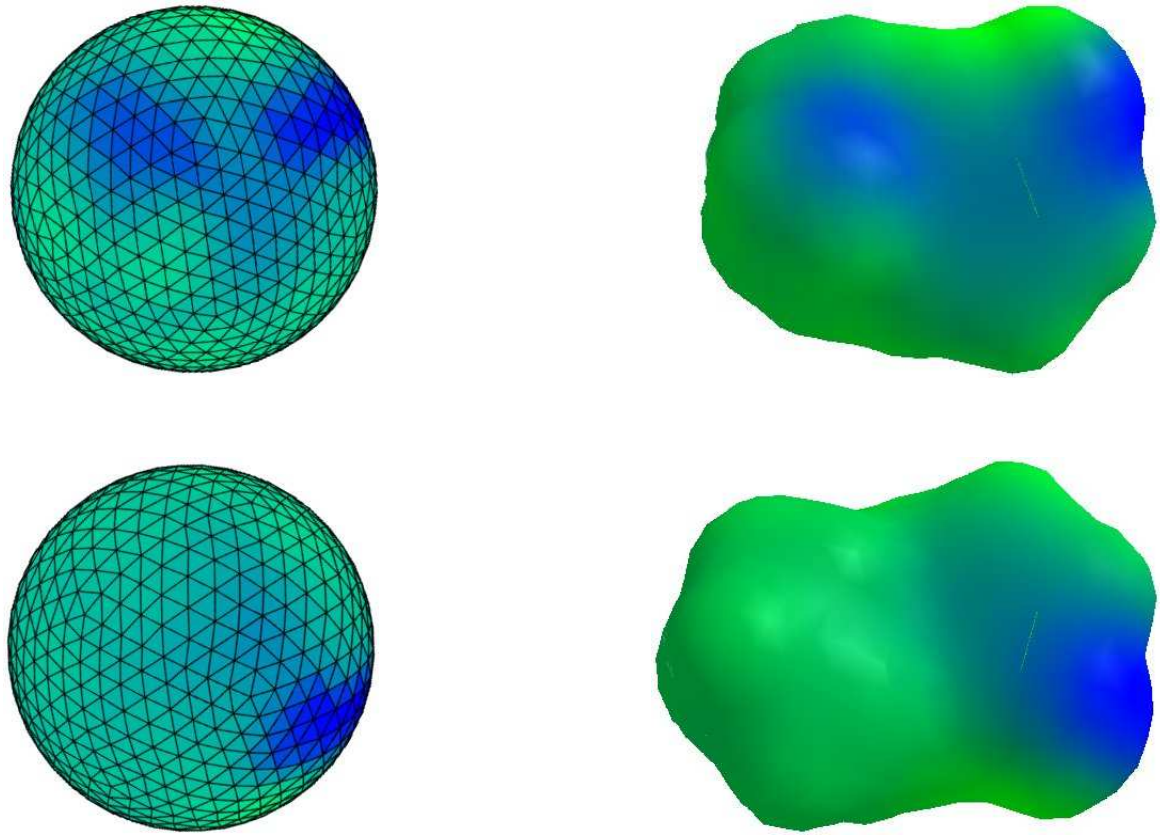


Figure A.23: Molecule 23

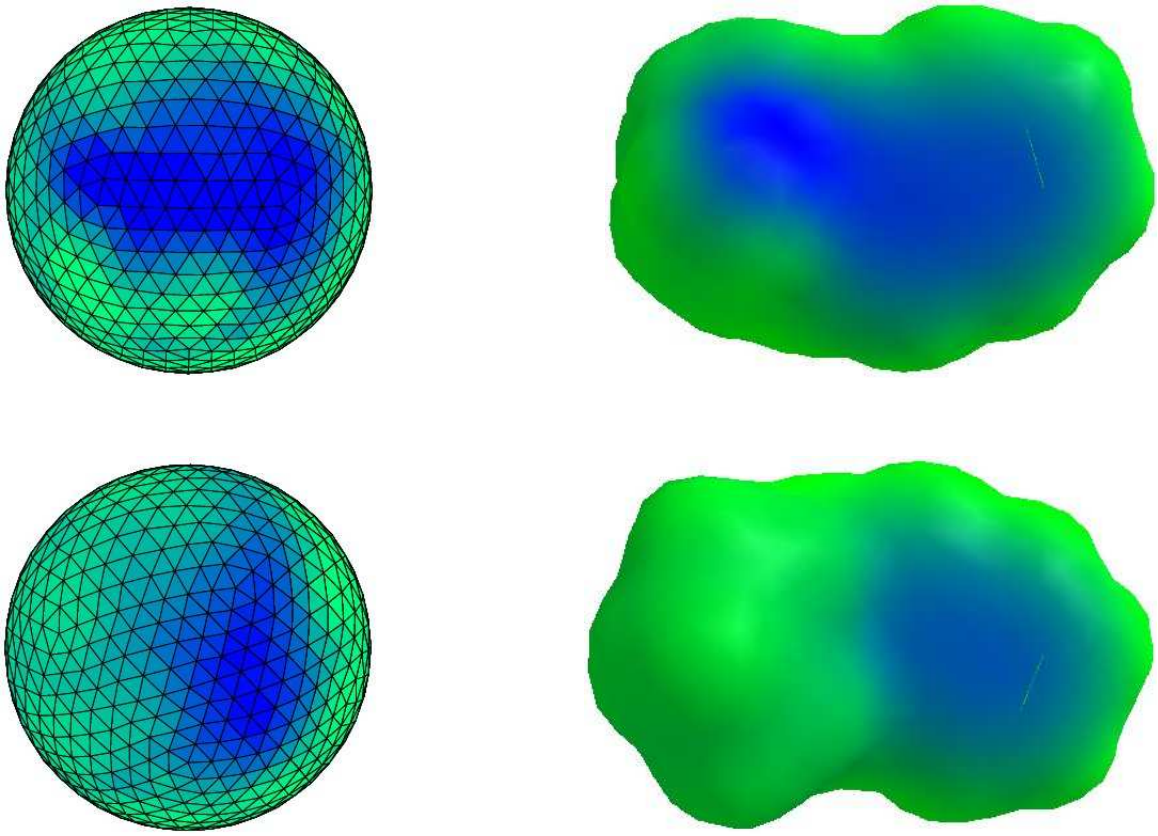


Figure A.24: Molecule 24

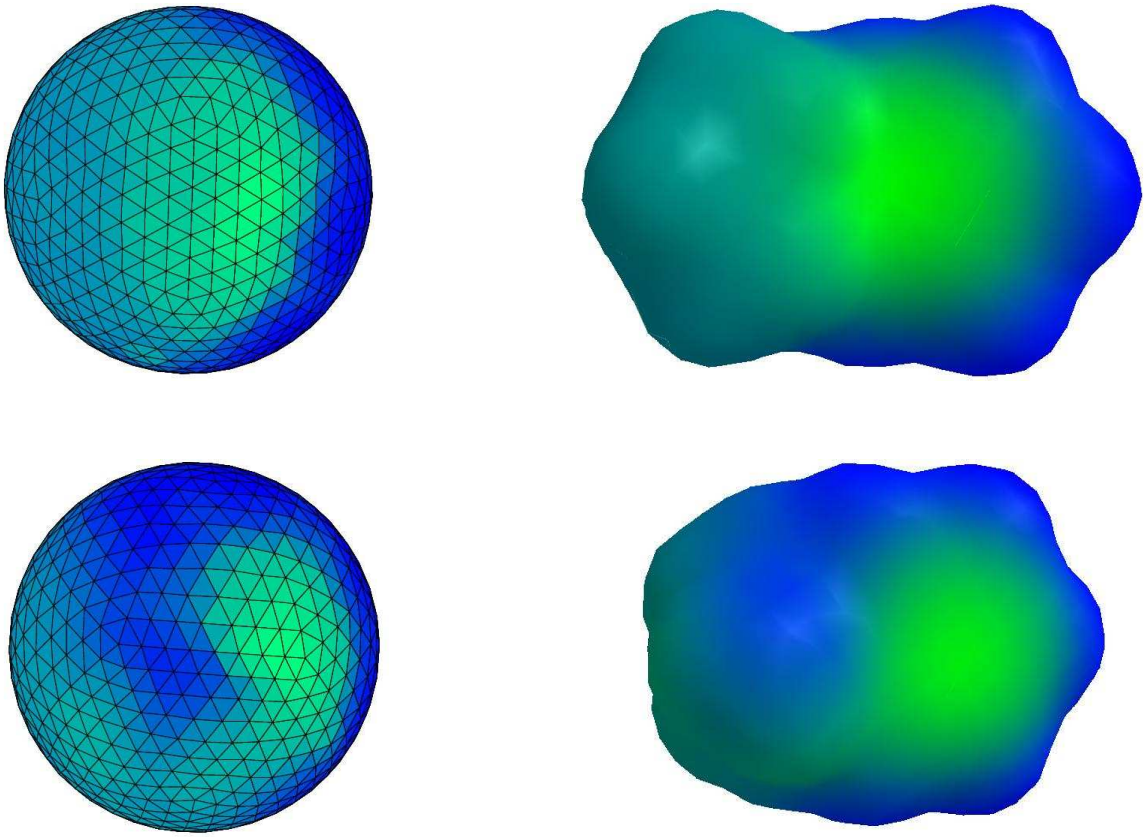


Figure A.25: Molecule 25

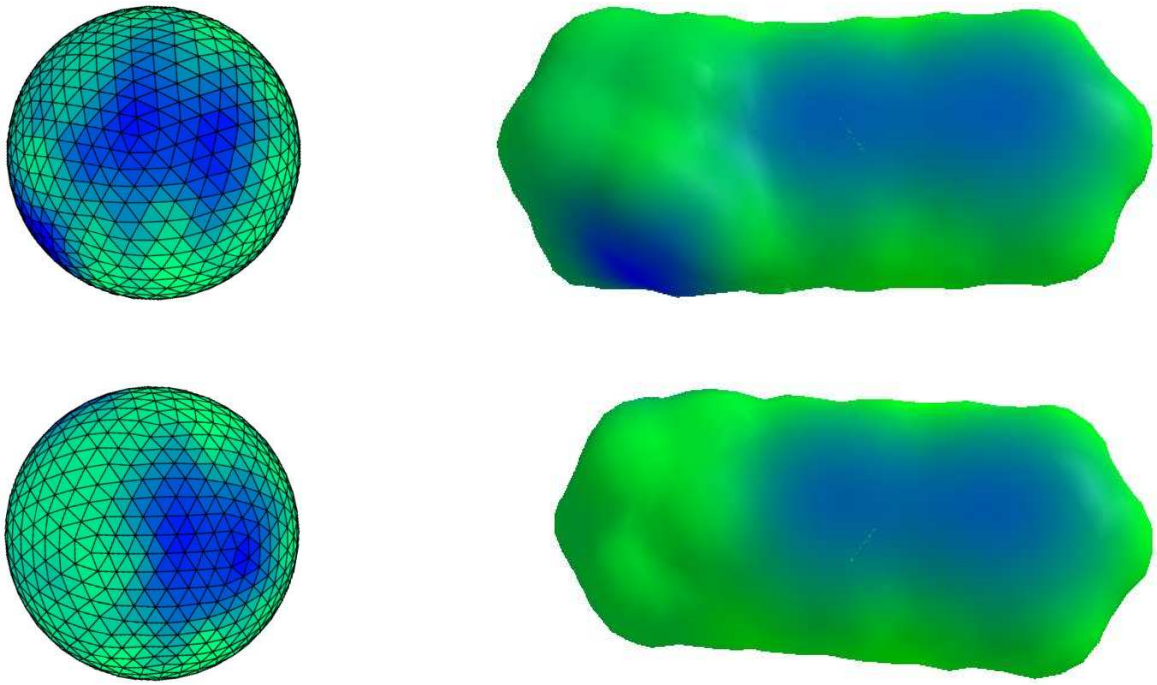


Figure A.26: Molecule 26

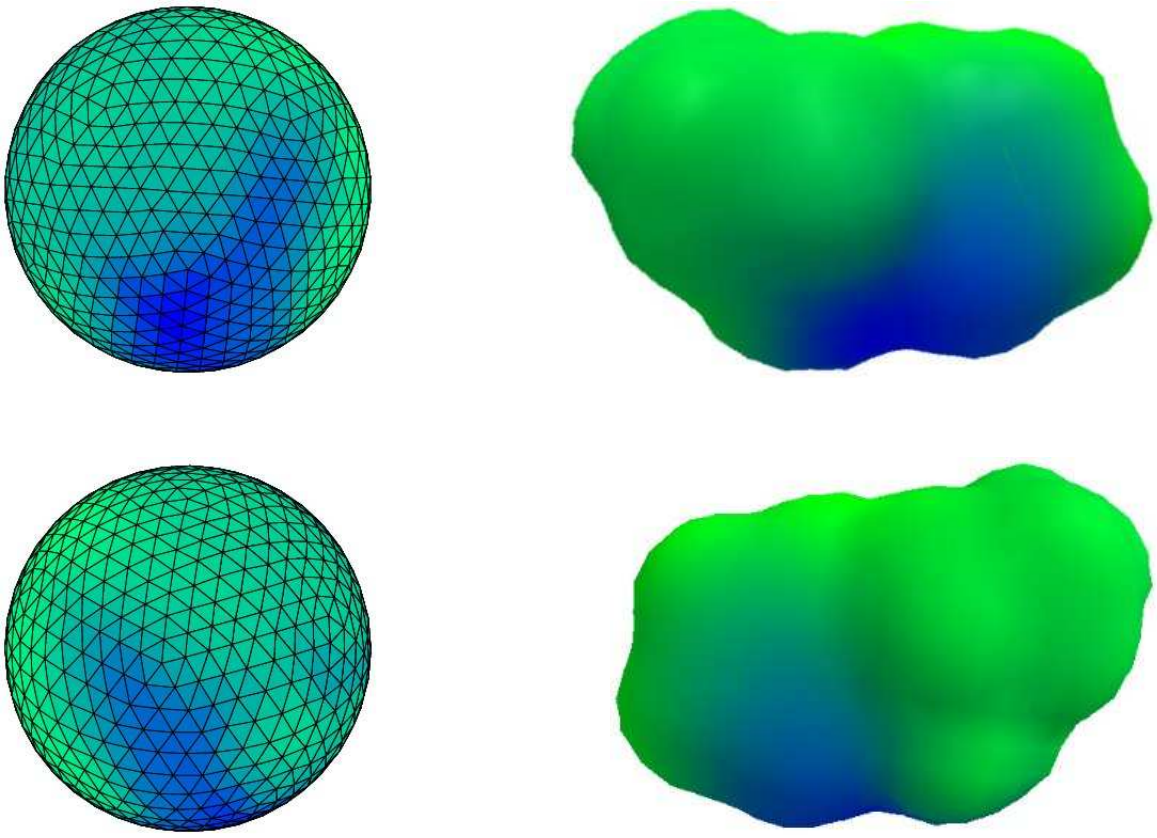


Figure A.27: Molecule 27

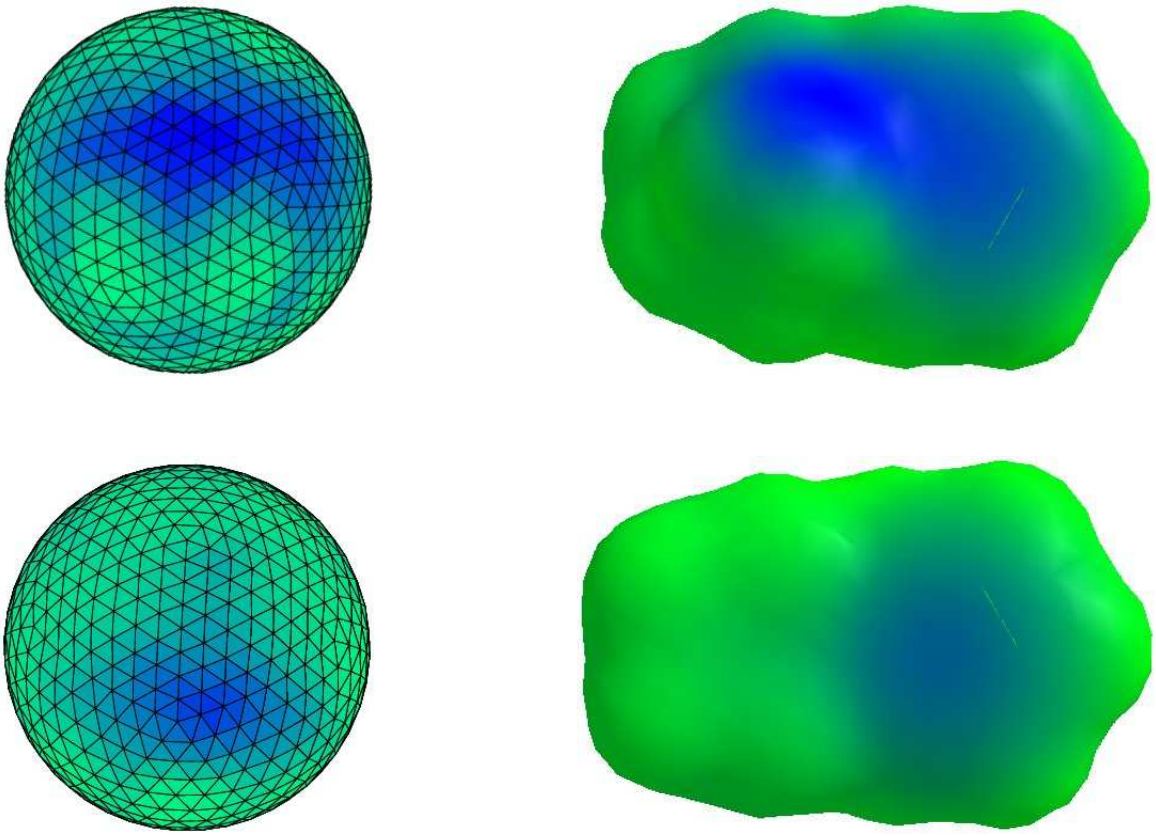


Figure A.28: Molecule 28

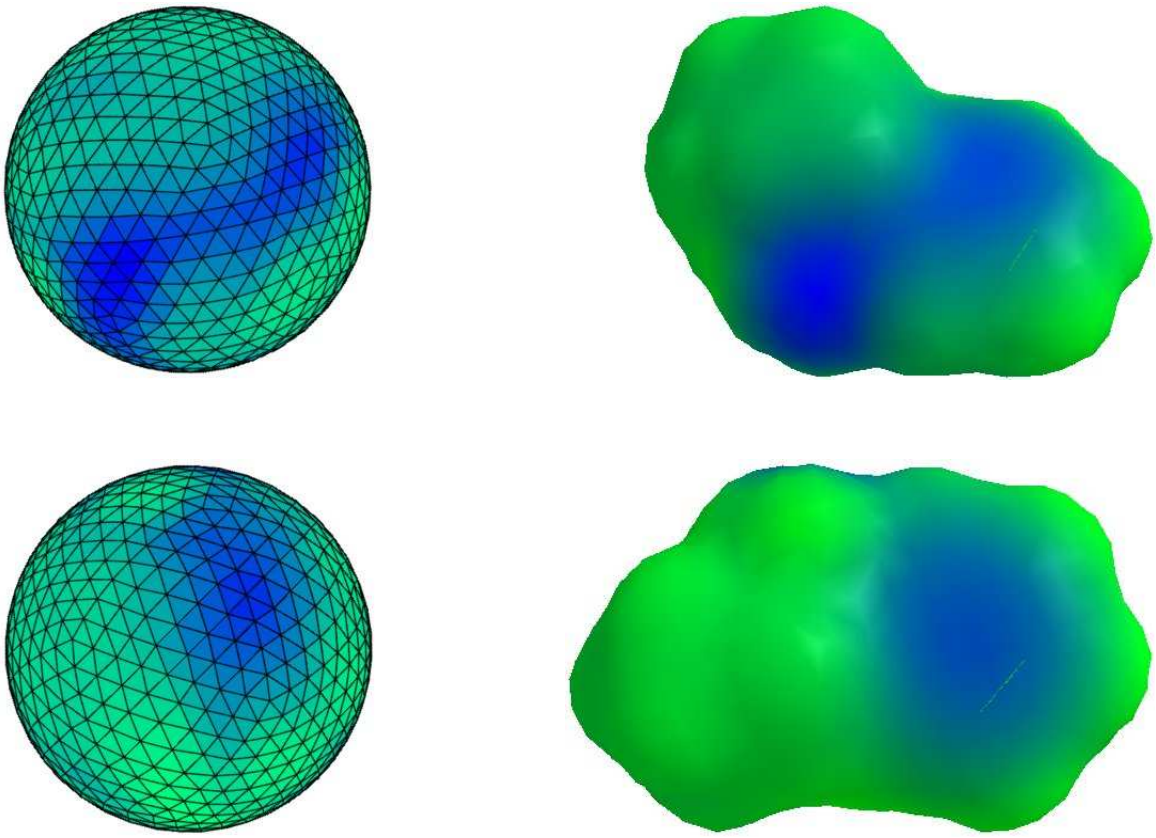


Figure A.29: Molecule 29

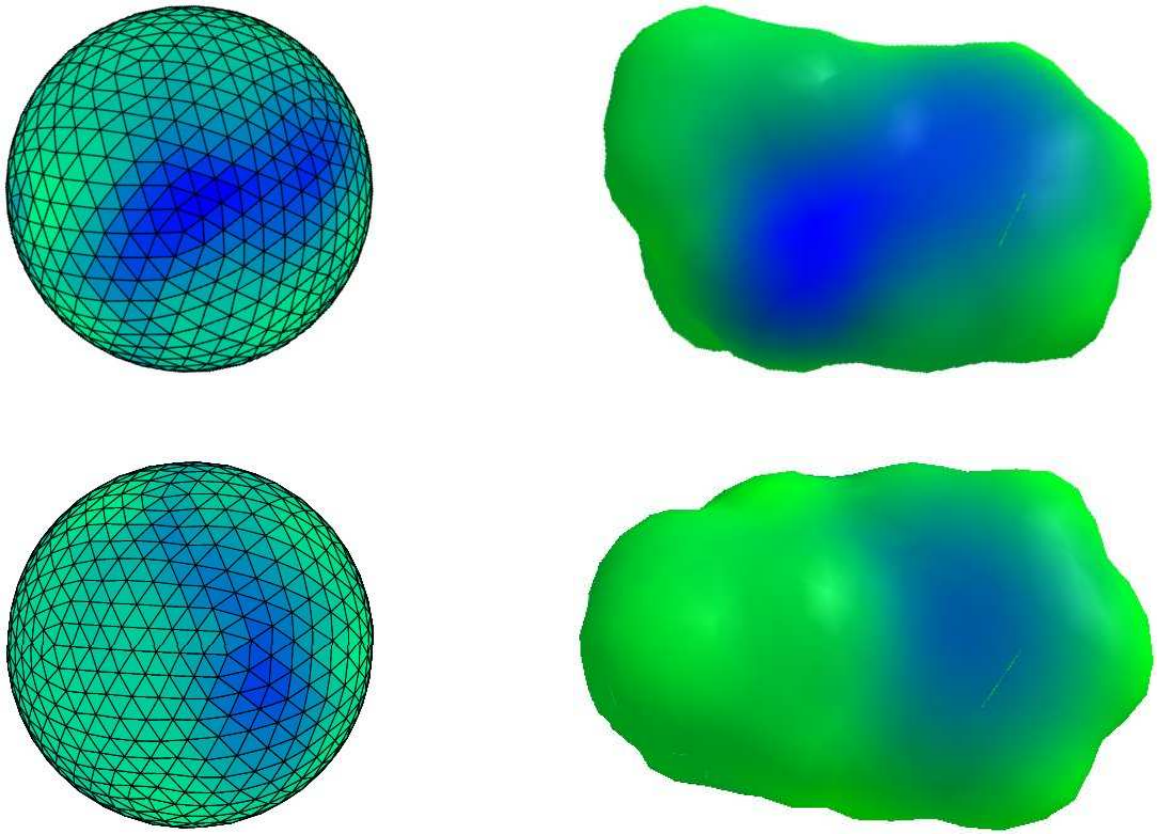


Figure A.30: Molecule 30

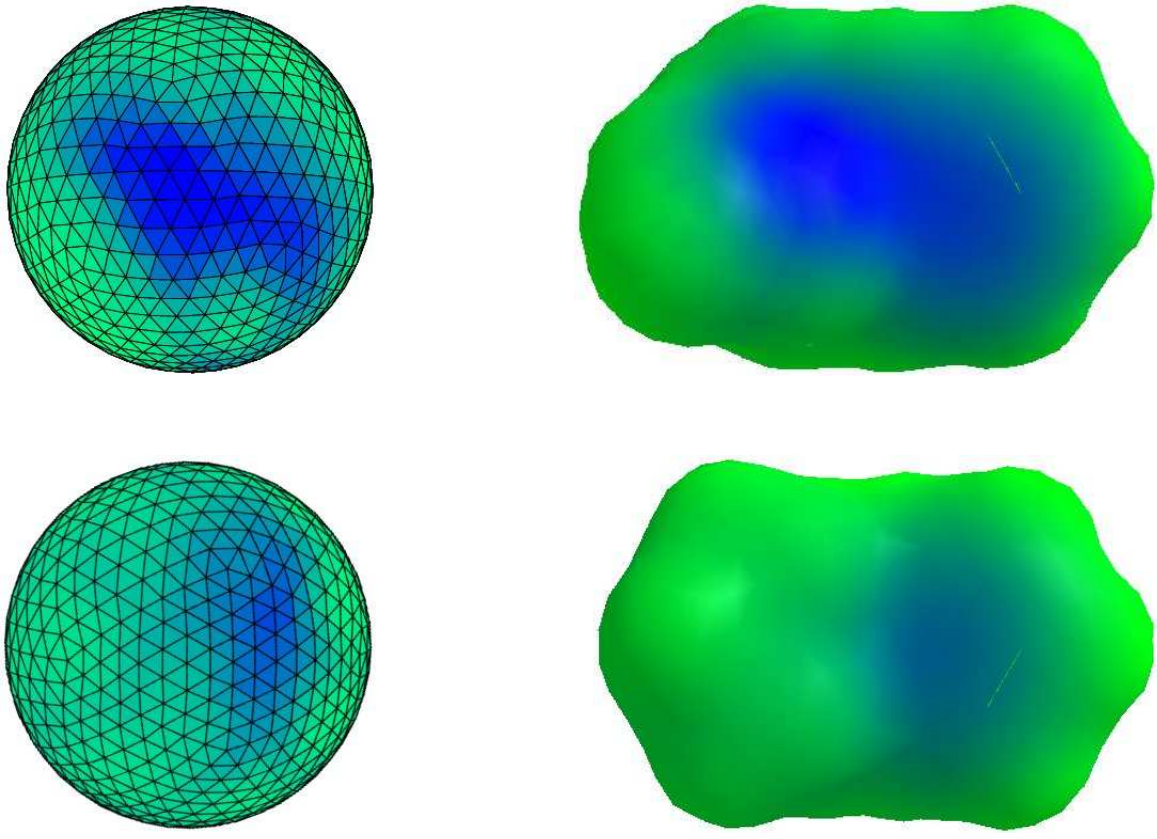


Figure A.31: Molecule 31

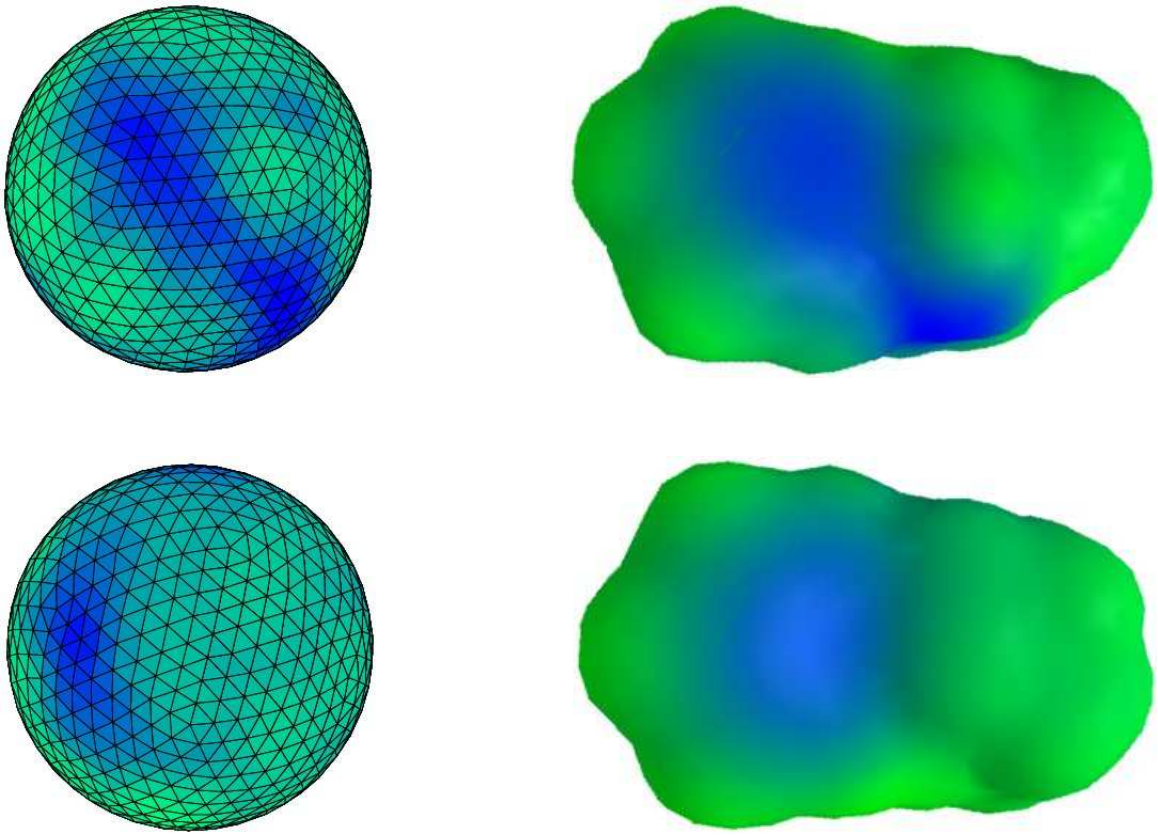


Figure A.32: Molecule 32

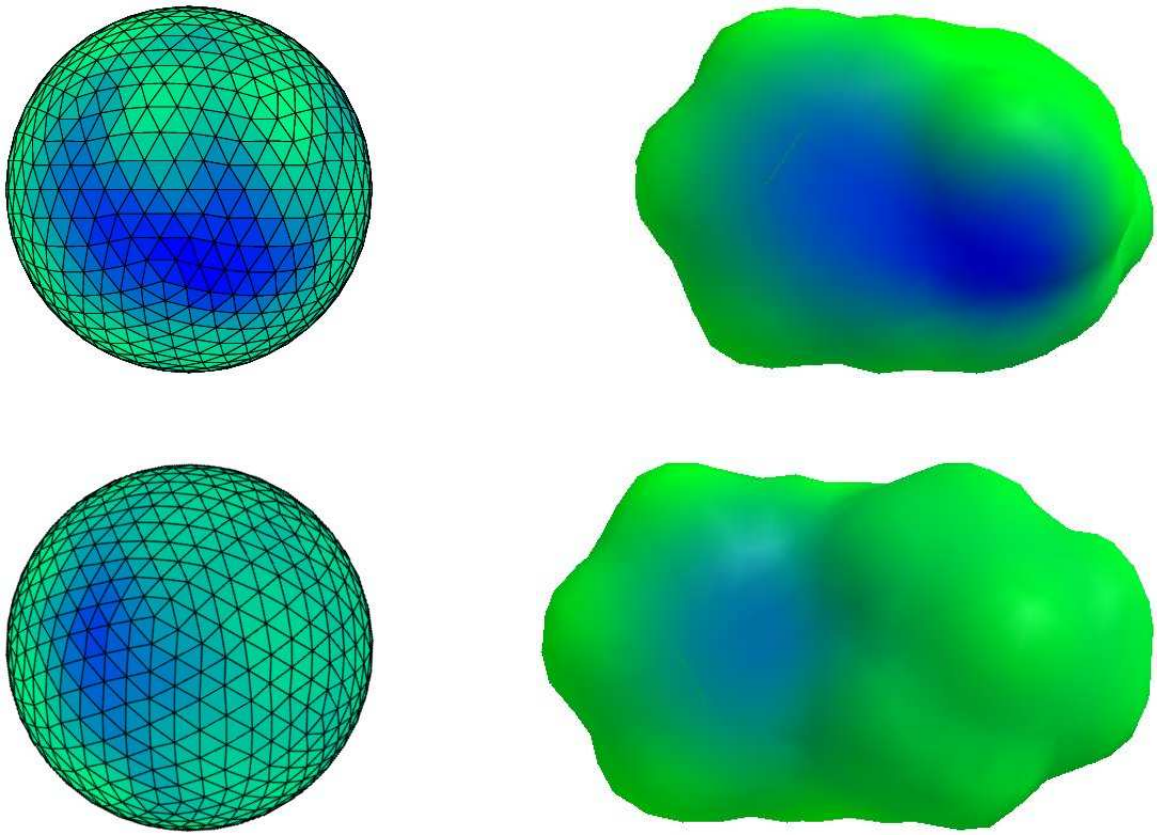


Figure A.33: Molecule 33

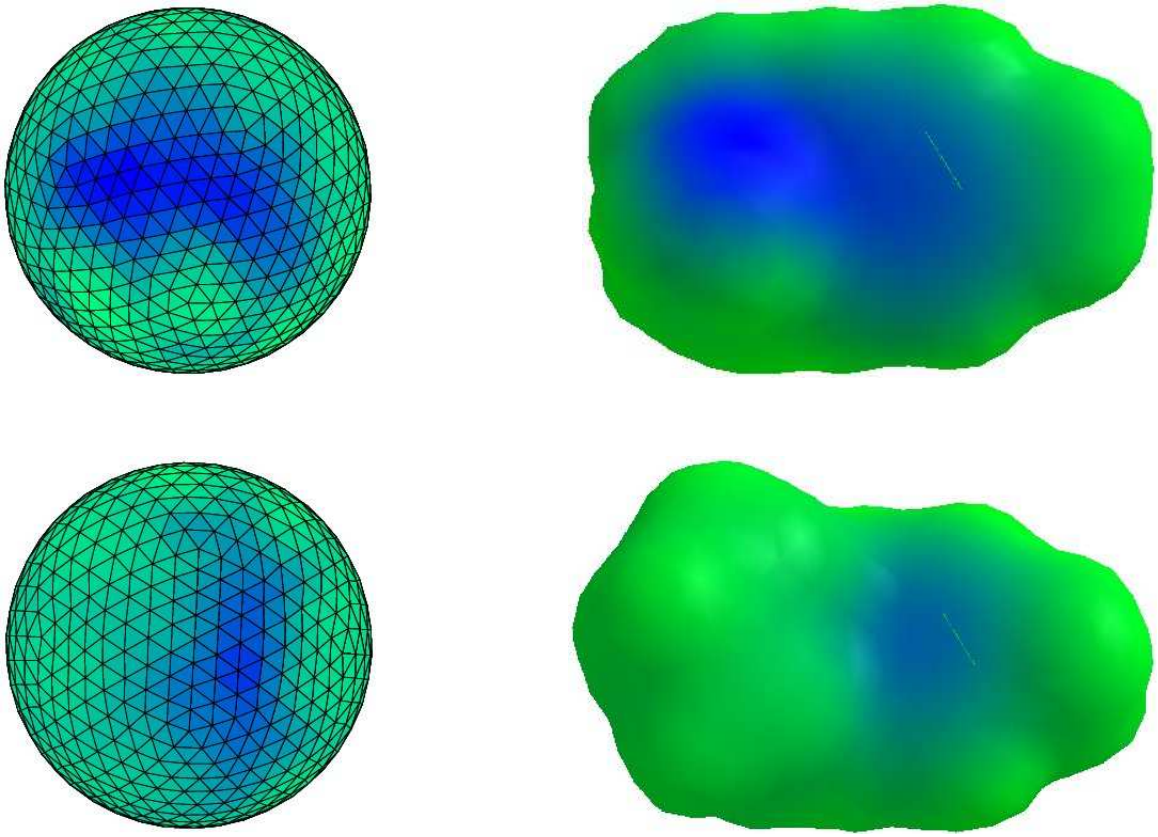


Figure A.34: Molecule 34

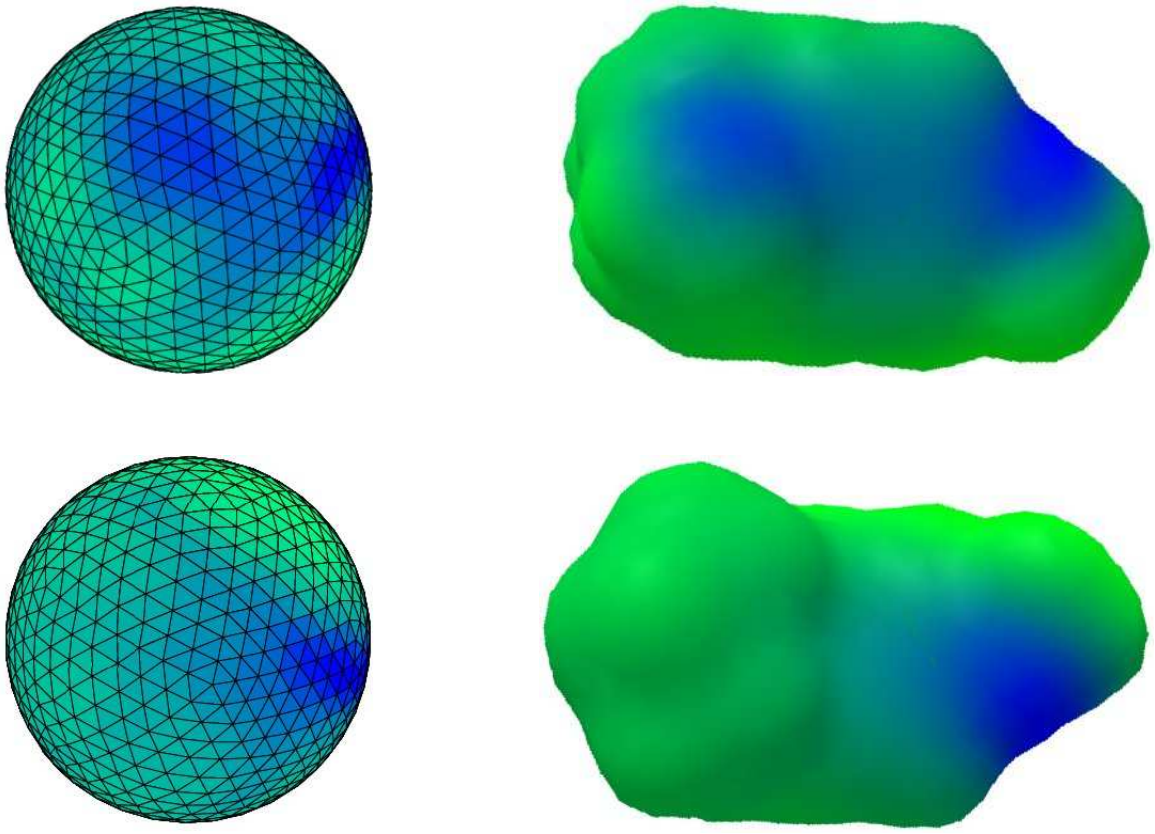


Figure A.35: Molecule 35

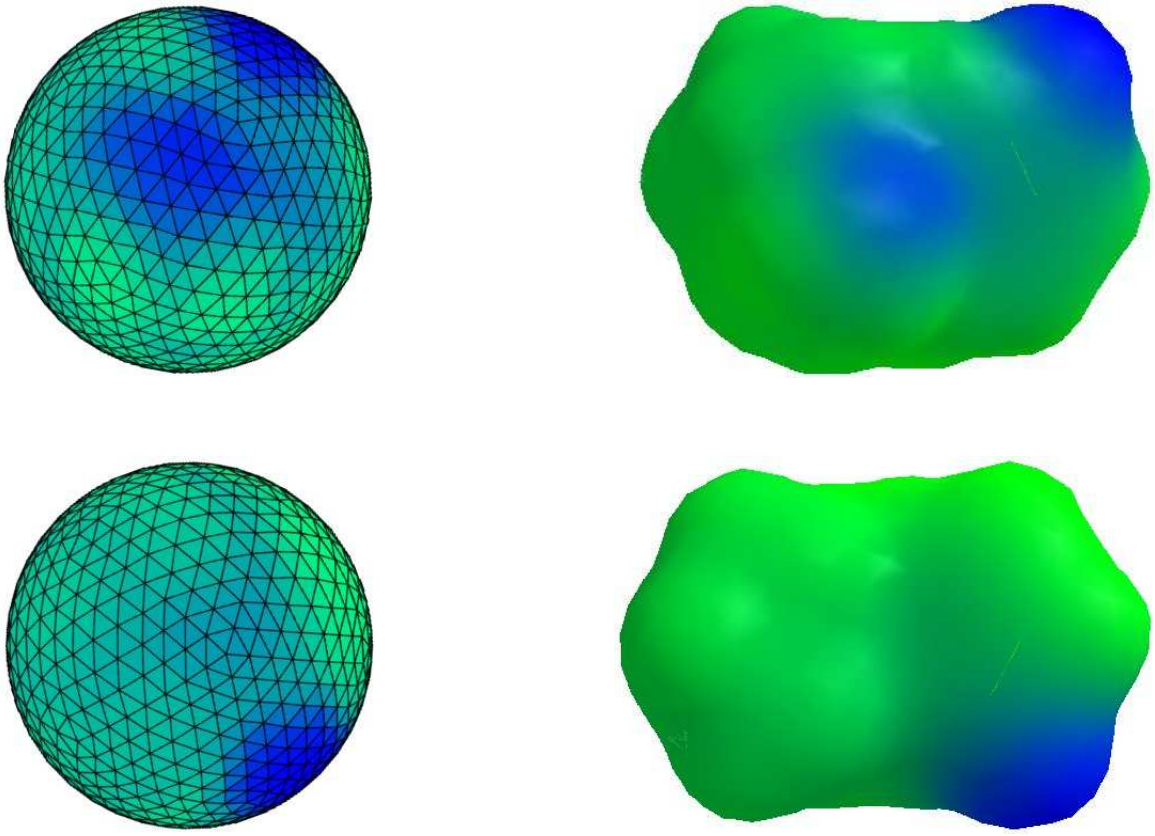


Figure A.36: Molecule 36

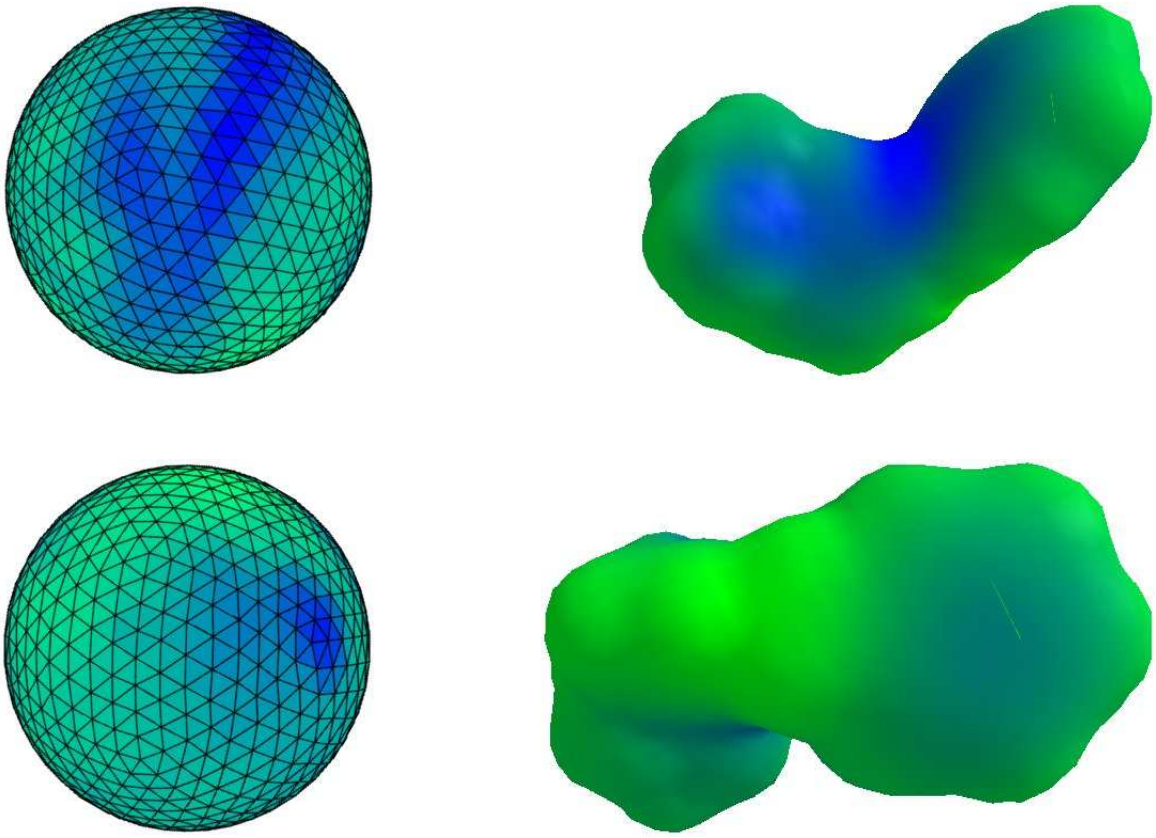


Figure A.37: Molecule 37

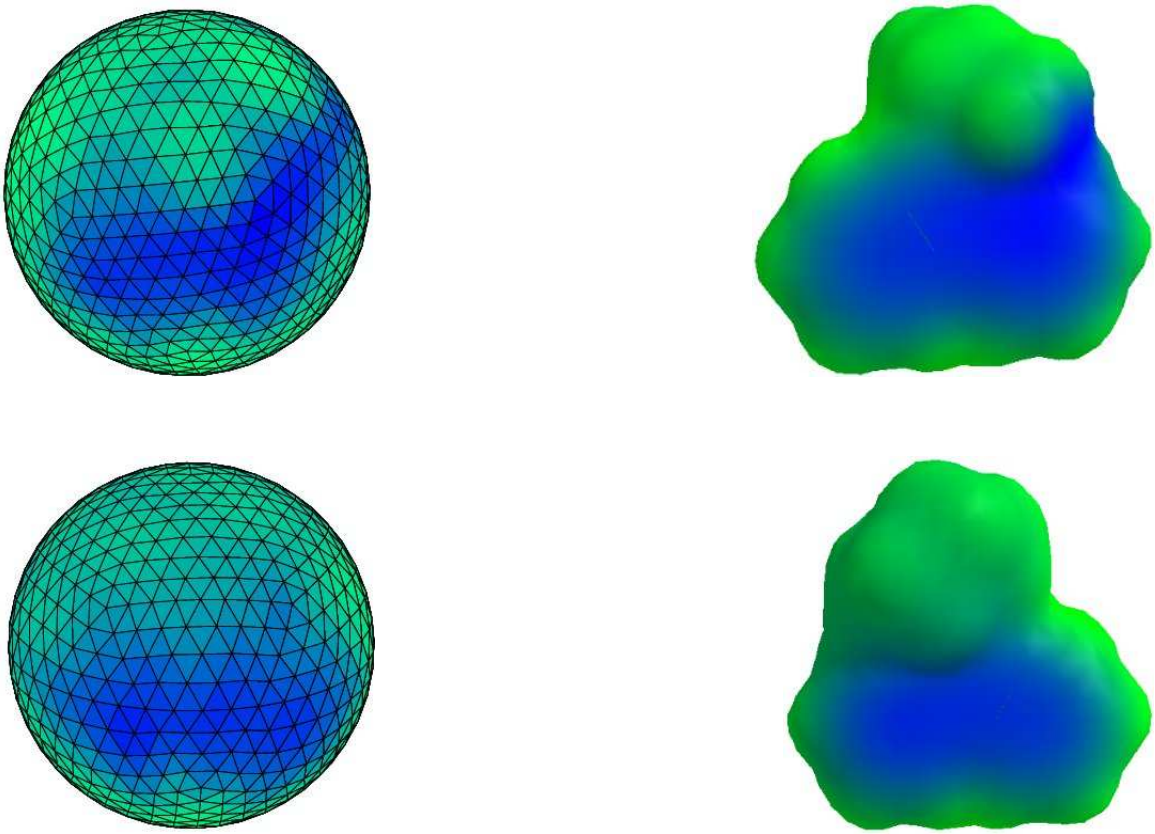


Figure A.38: Molecule 38