# Predicting Protein Interactions with Computational Geometry and Machine Learning Methods

Lukas Willy Bruhn
born 13th March 1994 in Kiel, Germany

Master Thesis
Supervisor: Prof. Dr. Volkmar Liebscher
Second Supervisor: PD Dr. Christopher Lillig

*A thesis submitted in fulfillment of the requirements for the degree of Master of Science*

Institute of Mathematics and Computer Science
University of Greifswald

November 20, 2019

# Contents

# List of Figures

# List of Tables

# Abstract

It is widely believed that proteins with similar geometrical properties also have similar functional properties [20]. A more specific hypothesis, is that proteins with similar *electrostatic fields* have similar functional properties [4]. In this thesis an implementation is presented that is capable of predicting protein-protein-interactions based on comparison of the geometry of the electric field, derived from pure *protein-data-base*-files with *Visualize-molecular-Dynamics* (VMD) [17]. The proposed method is built upon the ideas presented in [26] and involves approximating a distribution of feature-distributions of the geometry of 3D-objects. The method is based on the *Gromov-Wasserstein distance* for which a computational fast lower-bound was proposed in [26]. In this thesis an approximation of said lower-bound is used in combination with a sampling procedure to generate different features of the 3D-objects. These features are then used to train a neural net for classification. The method proposed in [26] was re-implemented in this thesis. The two approaches were compared on the publicly available data-set containing 3D-models of animals [32]. Furthermore the implementation was tested on the publicly available data-set containing 3D-models of furniture (*ModelNet10*) [35] achieving reasonable prediction accuracy without further model-specific engineering.

The accuracy of predicting protein-interactions was measured on several pdb-data-sets including a set of Thioredoxins, glutathiones and a set of randomly selected pdbs of which target-interaction-information was taken from the *drug bank* [34].

The implementation is primarily written in *R* [28] using *keras* [6] with *tensorflow* [25]. The repository containing the implementation can be found at **https://github.com/WillyBruhn/PredictingProteinInteractions** and is freely available to the public running under the *GNU*-public-license.

# Chapter 1

# Predicting Protein Interactions

## 1.1 Introduction

The idea of inferring functional similarity through geometrical similarity is not new. In [20] the authors propose that proteins with geometrical similar properties can have similar functional properties. In this case *geometrical similarity* refers to the 3-dimensional structure of the proteins, that is simply speaking the relative positions of the amino acids. Another geometrical property of a protein can be derived from the *electric field*. Each protein consists of amino acids which are built from atoms containing protons and electrons. Protons contain a positive charge and electrons contain a negative charge. Since the protons are placed in the core of the atom and the electrons float around the core in some distance, different charges are occurring in different proximity to the core. That way, given an atom in Euclidean space it is possible to assign each point in the Euclidean space its corresponding charge. Then fixing one specific value, and selecting all points whose charge is equal to the selected value, one obtains an *iso-surface*. The iso-surface is the analogue to the *iso-line* in 2-dimensional space.

One major research interest of the group of Prof. Lillig is the question in how far similarity in the iso-surfaces of proteins can be used to infer functional similarity.

In [4] Lillig et. al. proposed that the relevance of the geometrical complementarity of the iso-surfaces of two interacting proteins is not only locally around the active site important but also that long range interactions around the iso-surface play an important role. The analogy next to the key-lock-model and induced-fit-theory [21] is the following: In order for the key to work with a lock it also needs to be guided toward the lock. Protein iso-surfaces need not only to match well enough at the active site, but surface

interactions far away from the active site are necessary to lead the active sites to each other.

In order to verify this hypothesis Lillig et. al. did the following: Given a set of proteins of which the capability to functionally replace each other in vitro, that means in a living cell is known, the iso-surfaces are computed with *Visualize-molecular-Dynamics* (VMD) [17]. Then Lillig claims that just with the naked eye the geometrical properties can be used to make an accurate prediction of the above mentioned functionality. Many experiments were conducted and one of special interest for this thesis, to which we will refer as the *106-redoxin-test-set*, was conducted in E.coli. Lillig achieved a relatively high accuracy with his method. However the method involves tideous steps and require an expert to manually examine each protein. When I joined Lilligs group, my first contribution was to automatize the time-consuming steps, that are necessary to obtain an image of all the iso-surfaces of the proteins that one wants to examine (for more details see section 6.1.1). However this still needs an expert and hence the goal was to also automatize the prediction-step. A program capable of predicting functionality of proteins purely based on pdb-files could be used large-scale on databases, e.g. the protein-data-bank [3]. Another thinkable use-case could be to query a database for functional candidates of a given target-protein. Also the proposed hypothesis of long-range-interactions can be put in mathematical models and can be examined.

## 1.2 Test-Set and Articulation of the Problem

Thioredoxins (Thrx) form a family of proteins that contain about 100 amino acids and a di-sulfid-bond as the active-center. Thioredoxins function as electron-donors [27] and have a high relevance in practically any known organism.

In a previously conducted experiment by a colleague of Prof. Lillig in E.choli a gene-knock-down was performed. For E.choli the capability to produce Thrx is of vital importance, as it controls many regulatory reactions. Usually loosing the capability of Thrx-production leads to cell-death. In the conducted experiment the genes responsible for the production of Thrx were cut out of the genome and instead the genes for other proteins were inserted through gene-transfer. This was carried out for 106 different other proteins of the Gluta-redoxin-family. In the majority of cases the cells died, when they lost the capability to produce Thrx. In some cases however the cells continued to live. The conclusion was, that the replacing proteins were capable to replace Thrx in vital reactions. That means the proteins were *functionally similar*. Lars et. al. initially did not have an explanation as to why these

proteins seemed to functionally replace Thrx. They conducted Lillig to take a look at this experiment. Lillig used his above described approach on this data-set blindfolded and retained a relatively high prediction-accuracy, just with the naked eye.

This biochemical experiment can be simplified as follows: We are given one molecule $X$ and a set of molecules $Y_i$. Given one specific biochemical reaction that $X$ is involved in, we want to predict which of the $Y_i$ behave similarly to $X$ in that biochemical reaction. For each $Y_i$ a label $f_i$ exists indicating if $X$ and $Y_i$ are similar. Since the labels are known, this problem is a supervised learning problem.

## 1.3 Goals of This Research

Motivated by the promising results from his above described approach Lillig asked both me and Felix Berens of the mathematical institute of Greifswald to join his work-group for a master-thesis. The main focus in both our theses was the automatization of the *naked-eye-approach*. Lillig pointed out three major components that he saw as relevant features for the automatization when using the iso-surfaces:

- the proximity to the *active center*
- regions in which potentials meet (in this thesis referred to as *boarder regions*)
- the importance of *long-range-interactions*.

This is discussed in more detail in section 3.6.

## 1.4 Related Work and Similar Approaches

A common trend that can be observed in many scientific fields, is the rapid rate at which new data can be obtained. In order to organize and analyze these collections of data many new algorithms have been developed. The field of machine-learning makes great profit of the scale at which available data is growing.

3D-laser-scans are a popular tool for generating 3D-models automatically and are one such an example where huge amounts of data can be acquired very fast. A laser is used to measure the distance over a grid of the surface of an object. These distances are then used to generate a 3D-model of the surface. Measuring more distances lead to a higher resolution of the model. A typical task is to merge multiple scans into one model. This means translating and rotating the different point clouds in order to establish a one-to-one

correspondence between subsets of the point clouds. This problem is called point-set-registration. For such rigid transformations algorithms such as the iterative-closest-point [15], [30] exist.

Non-rigid registrations form a more challenging problem. As Memoli points out in [26], *finding a notion of similarity that has some insensitivity to different poses of the same object* is important in such a case. One example of this nature is the articulation of an object. In figure 1.1 different poses of the same model are displayed [32]. Robust point matching (RPM) [13] is one approach to this problem.

**Figure 1.1:** Different articulations of the same model. Displayed is a 3D-model of a horse from the *animal-test-set* [32].

# Chapter 2

# Metric Geometry for Object Comparison

In this chapter the main definitions and concepts that are presented in [26] and form the basis for the approach are recalled. The theory is recapitulated in a condensed from as my colleague Felix Berens already examined the theory in his Master-thesis [2].

In summary the concepts relevant for this thesis from [26] are the following: The main concept is to compare the distances inside objects rather than finding a one-to-one-correspondence of the points from one object to another. This notion of similarity is called *intrinsic distances* as opposed to distances of points from different objects which is called *extrinsic distances*.

In [26] a theoretically appealing formalism is introduced that connects the idea of the *extrinsic distances* and *intrinsic distances*. The term *gromovization* is coined which describes formally how the two approaches are connected. Informally the idea goes as follows: An object is modeled as a set of points in some metric space. Additionally each point is assigned a weight, making it a metric-measure-space. One now tries to find some rule of how to transfer the mass from the points from the first object to the second object that minimizes some notion of dissimilarity. With the *extrinsic* approach the objects have to be embedded in a common space and then one object has to be rotated and translated in such a way that the objects align as best as possible. With the *intrinsic* approach finding such an embedding and transformation of the second object becomes obsolete. This second approach is called *Gromov-Wasserstein-Distance*. Memoli derives a computationally fast lower bound $\mathbf{FLB}_p$ [26, Def. 6.1] for the Gromov-Wasserstein-Distance. Then another bound for the $\mathbf{FLB}_p$ is introduced which will be called $\mathbf{DE}$ in this thesis and is based on distributions of the eccentricities of the objects. In this thesis I proof that in the finite case it holds that $\mathbf{FLB}_1 = \mathbf{DE}_1$, which is a

very important result from a computational standpoint, because computational simulations are always dealing with finite amounts of points.

Memoli proceeds to examine how well the $\mathbf{FLB}_p$ can be used to measure the dissimilarity between 3D-objects. For this purpose he applies the method on the publicly available data-set of 3D-models of different animals [32] to which will be referred as the *animal-test-set*.

Memolis proposed method was re-implemented in this thesis. The corresponding results are reviewed in section 2.3.

## 2.1 A Lower Bound for the Gromov-Wasserstein Distance

**2.1 Definiton** *(Push Forward Measure)[26]*
*For a measurable map $f : X \to Y$ between two compact metric spaces $X$ and $Y$, and $\mu$ a measure on $X$, the* push-forward-measure *$f_{\#\mu}$ on $Y$ is given by*

$$f_{\#\mu}(A) := \mu(f^{-1}(A)) \quad for \quad A \in \mathcal{B}(Y) \tag{2.1}$$

*where $\mathcal{B}(Y)$ is the Borel $\sigma$-algebra of $Y$.*

**2.2 Definiton** *(Metric Measure Space)[26, Def. 5.1]*
*A* metric probability space *(mp-space for short, in [26] it is called metric measure-space) is a triple $(X, d_X, \mu_X)$ where*

- *$(X, d_X)$ is a compact metric space.*
- *$\mu_X$ is a Borel probability measure on $X$ i.e., $\mu_X(X) = 1$, and $\mu_X$ has full support:* $\text{supp}[\mu_X] = X$.

The triple $(X, d_X, \mu_X)$ will be denoted by $\mathbb{X}$. The reason for imposing $\mu_X(X) = 1$ is that one thinks of $\mu_X$ as a modeling of the acquisition process or sampling procedure of an object. Two mp-spaces $(X, d_X, \mu_X)$ and $(Y, d_Y, \mu_Y)$ are called *isomorphic* iff there exists an isometry $\Psi : X \to Y$ such that $\Psi_{\#}\mu_X = \mu_Y$. Furthermore, by $\mathcal{G}_w$ the collection of all mp-spaces will be denoted.

**2.3 Definiton** *(Measure coupling) [26, Def. 5.6]*
*Given two metric measure spaces $(X, d_X, \mu_X)$ and $(Y, d_Y, \mu_Y)$ one says that a measure $\mu$ on the product space $X \times Y$ is a* coupling *of $\mu_X$ and $\mu_Y$ iff*

$$\mu(A \times Y) = \mu_X(A), \quad and \quad \mu(X \times A') = \mu_Y(A') \tag{2.2}$$

*for all measurable sets $A \subset X, A' \subset Y$. Denote by $\mathcal{M}(\mu_X, \mu_Y)$ the set of all couplings of $\mu_X$ and $\mu_Y$.*

**2.4 Definiton** *[26]*
*For metric spaces* $(X, d_X)$ *and* $(Y, d_Y)$ *let* $\Gamma_{X,Y} : (X \times Y) \times (X \times Y) \to \mathbb{R}^+$ *be given by*

$$\Gamma_{X,Y}(x, y, x'y') := |d_X(x, x') - d_Y(y, y')|. \tag{2.3}$$

**2.5 Definiton** *(Gromov-Wasserstein Distance)[26, Def. 5.7]*
*For* $\infty \geq p \geq 1$ *one defines the* Gromov-Wassertein-distance $\mathfrak{D}_p$ *between two mp-spaces* $\mathbb{X}$ *and* $\mathbb{Y}$ *by*

$$\mathfrak{D}_p(\mathbb{X}, \mathbb{Y}) := \inf_{\mu \in \mathcal{M}(\mu_X, \mu_Y)} \mathbf{J}_p(\mu) \tag{2.4}$$

*where for* $p \in [1, \infty)$ *and* $\mu \in \mathcal{M}(\mu_X, \mu_Y)$

$$\mathbf{J}_p(\mu) := \frac{1}{2}\Big(\int_{X\times Y}\int_{X\times Y} (\Gamma_{X,Y}(x, y, x', y'))^p \mu(dx \times dy)\mu(dx' \times dy')\Big)^{\frac{1}{p}}$$
$$\Big(= \frac{1}{2}||\Gamma_{X,Y}||_{L^p(\mu\otimes\mu_Y)}\Big) \tag{2.5}$$

*and* $p = \infty$

$$\mathbf{J}_\infty(\mu) := \frac{1}{2} \sup_{\substack{x,x'\in X \\ y,y'\in Y \\ s.t.(x,y),(x',y')\in R(\mu)}} \Gamma_{X,Y}(x, y, x', y') \Big(= \frac{1}{2}||\Gamma_{X,Y}||^\infty_L(R(\mu) \times R(\mu)).\tag{2.6}$$

**2.6 Definiton** *(Eccentricity) [26, Def. 5.3]*
*Given* $p \in [1, \infty]$ *and an mp-space* $\mathbb{X} = (X, d_X, \mu_X)$ *define the* $p$-eccentricity function *of* $X$ *by*

$$s_{\mathbb{X},p} : X \to \mathbb{R}^+ \quad given\ by \quad x \mapsto \Big(\int_X d_X(x, x')^p \mu_X(dx')\Big)^{\frac{1}{p}} \Big(= ||d_X(x, \cdot)||_{L^p(\mu_X)}\Big) \tag{2.7}$$

*for* $1 \leq p \leq \infty$*, and by*

$$s_{\mathbb{X},\infty} : X \to \mathbb{R}^+ \quad given\ by \quad x \mapsto \sup_{x'\in supp[\mu_X]} d_X(x, x') \tag{2.8}$$

*for* $p = \infty$.
*Let* $S_{\mathbb{X},p} : \mathbb{R} \to [0, 1]$ *be given by* $t \mapsto \mu_X(\{x \in X | s_{\mathbb{X},p}(x) \leq t\})$*, i.e.,* $S_{\mathbb{X},p}$ *is the distribution function of* $s_{\mathbb{X},p}$ *under* $\mu_X$*. Additionally denote :*

$$s_{\mathbb{X}}(x) := s_{\mathbb{X},1}(x) \quad for \quad x \in X$$
$$S_{\mathbb{X}}(t) := S_{\mathbb{X},1}(t) \quad for \quad t \in \mathbb{R}.$$

**2.7 Definiton** *(Distance of Distributions of Eccentricities)*
*For $\mathbb{X}, \mathbb{Y} \in \mathcal{G}_\omega$ define:*

$$\boldsymbol{DE}(X,Y) := \frac{1}{2}\left(\int_{\mathbb{R}} |S_{X,1}(t) - S_{Y,1}(t)|dt\right). \tag{2.9}$$

**2.8 Definiton** *(First Lower Bound) [26, Def. 6.1]*
*For $\mathbb{X}, \mathbb{Y} \in \mathcal{G}_\omega$ define for $p \in [1,\infty)$:*

$$\boldsymbol{FLB}_p(X,Y) := \frac{1}{2} \inf_{\mu \in \mathcal{M}(\mu_X,\mu_Y)} \left(\int_{X\times Y} |s_{X,p}(x) - s_{Y,p}(y)|^p \mu(dx \times dy)\right)^{\frac{1}{p}}. \tag{2.10}$$

*Additionally denote:*

$$\boldsymbol{FLB}(X,Y) := \boldsymbol{FLB}_1(X,Y).$$

Memoli uses the following Lemma to prove **FLB** $\geq$ **DE**. The Proof of the Lemma is cited here, as it is used in section 2.2.1 to proof that under certain conditions it holds **FLB** = **DE**.

**2.9 Lemma** *[26, Lemma. 6.1]*
*Let $(X, d_X, \mu_X)$ and $(Y, d_Y, \mu_Y)$ be two mp-spaces in $\mathcal{G}_\omega$. Let $f : X \to \mathbb{R}$ and $g : Y \to \mathbb{R}$ be continuous and $\phi : \mathbb{R} \to (0,\infty)$ be convex. Then*

$$\inf_{\mu\in\mathcal{M}}(\mu_X,\mu_Y) \int_{X\times Y} \phi(f(x) - g(y))\mu(dx \times dy) \geq \int_0^1 \phi(F^{-1}(t) - G^{-1}(t))dt \tag{2.11}$$

*where $F(t) := \mu_X\{x \in X | f(x) \leq t\}$ and $G(t) := \mu_Y\{y \in Y | f(y) \leq t\}$ are the distributions of $f$ and $g$, respectively, and their generalized inverses under $\mu_X$ and $\mu_Y$, respectively, are defined as:*

$$F^{-1}(t) = \inf\{u \in \mathbb{R} | F(u) > t\}.$$

*Furthermore, when $\phi(u) = |u|, u \in \mathbb{R}$, one can dispense with the inverses:*

$$\inf_{\mu\in\mathcal{M}}(\mu_X,\mu_Y) \int_{X\times Y} |f(x) - g(y)|\mu(dx \times dy) \geq \int_{\mathbb{R}} |F(u) - G(u)|du \tag{2.12}$$

*Proof [26, Lemma. 6.1, Proof]:* Fix $\mu \in \mathcal{M}(\mu_X,\mu_Y)$. Let $h : X \times Y \to \mathbb{R}^2$ given by $h = (f, g)$ and consider the measure $\nu = h_\#\mu$ on $\mathbb{R}^2$. By Theorem 4.1.11 of [11](applied to $T : X \times Y \to \mathbb{R} \times \mathbb{R}, (x, y) \mapsto (f(x), g(y)))$ one has

$$\int_{X\times Y} \phi(f(x) - g(y))\mu(d_X \times d_Y) = \int_{\mathbb{R}\times\mathbb{R}} \phi(t - s)\nu(dt \times ds).$$

Now, $\nu(I \times \mathbb{R}) = \mu(f^{-1}(I) \times g^{-1}(\mathbb{R})) = \mu(f^{-1}(I) \times Y) = \mu_X(f^{-1}(I))$, for any $I \in \mathcal{B}(\mathbb{R})$. Similarly, $\nu(\mathbb{R} \times J) = \mu_Y(g^{-1}(J))$ for any $J \in \mathcal{B}(\mathbb{R})$. Hence from the equality above,

$$\int_{X\times Y} \phi(f(x) - g(y))\mu(dx \times dy) \geq \inf_{\nu \in \mathcal{M}(f_{\#}\mu_X, g_{\#}\mu_Y)} \int_{\mathbb{R}\times\mathbb{R}} \phi(t-s)\nu(dt \times ds).$$

The conclusion follows from results on the transportation problem on the real line, see Remark 2.19 in [33], and then from the fact that $\mu \in \mathcal{M}(\mu_X, \mu_Y)$ was arbitrary and the right-hand side does not depend on it.

□

**2.10 Proposition** *[26, Prop. 6.1]*
*Let $X, Y \in \mathcal{G}_\omega$ and $p \in [1, \infty]$. Then,*

$$\mathfrak{D}_p(X, Y) \geq \boldsymbol{FLB}_p(X, Y). \tag{2.13}$$

Note that for $1 \leq p < \infty$, solving for $\mathbf{FLB}_p$ leads to a *Mass Transportation Problem* for the cost $c(x, y) := |s_{X,p}(x) - s_{Y,p}(y)|^p$.

Then, invoking Lemmma 2.9 with $\phi(u) = |u|^p$ one obtains

**2.11 Corollary**
*(Lower bound based on distribution of eccentricities) [26, Cor. 6.1]*
*For $X, Y \in \mathcal{G}_\omega$,*

$$\boldsymbol{FLB}(X, Y) \geq \boldsymbol{DE}. \tag{2.14}$$

**2.12 Remark** *(Matching measures between finite spaces) [26, Rem. 2.2]*
*When $X, Y \in \mathcal{C}_\omega(Z)$ are finite, say $n_X = |X|$ and $n_Y = |Y|$, then $\mathcal{M}(\mu_X, \mu_Y)$ is composed of matrices with non-negative entries of size $n_X \times n_Y$ satisfying $n_X + n_Y$ linear constraints:*

$$\sum_{x\in X} \mu(x, y) = \mu_Y(y) \quad \forall y \in Y \quad \textit{and} \quad \sum_{y\in Y} \mu(x, y) = \mu_X(x) \quad \forall x \in X. \tag{2.15}$$

## 2.2 Building up on Memolis Approach

In this section mainly two things are described that were found out about the **DE** in this thesis. First of all a proof is constructed that shows that in the finite case it holds that $\mathbf{DE}_1 = \mathbf{FLB}_1$. This is remarkable from a computational standpoint since in any computation one can only deal with a finite

amount of points, and hence the infinite case plays no role in computation. And hence the computation of the **DE** is much cheaper than the computation of the **FLB** this can already be a huge improvement to any algorithm that one wants to build up on and is restricted to some calculation-time. Further-more a proof is given that shows that the **DE** is a *pseudo-metric*.

### 2.2.1 DE = FLB

Memoli uses Lemma 2.9 in order to prove **FLB** $\geq$ **DE** in Corollary 2.11. The crucial part of Lemma 2.9 is the following equation:

$$\inf_{\mu\in\mathcal{M}}(\mu_X,\mu_Y)\int_{X\times Y}|f(x)-g(y)|\mu(dx\times dy)\geq\int_{\mathbb{R}}|F(u)-G(u)|du. \quad (2.16)$$

In the finite case $|X|,|Y|<\infty$ however it holds

$$\inf_{\mu\in\mathcal{M}}(\mu_X,\mu_Y)\int_{X\times Y}|f(x)-g(y)|\mu(dx\times dy)=\int_{\mathbb{R}}|F(u)-G(u)|du. \quad (2.17)$$

Analogously to corollary 2.11 it then follows **FLB** = **DE**.

Here I will give two proofs of (2.17). Both proofs involve constructing an optimal transport-map. In the first proof using Lemma 2.9 or more specifically (2.16) the equality follows directly with the optimal transport-map as it minimizes the left-hand-side of the equation. The second proof uses results from [33] which directly lead to the proof, using the constructed optimal-transport-map.

#### Construction of an Optimal Transport-Map

In the situation of Lemma 2.9 and the case that $\phi := |\cdot|$ let $X$ and $Y$ with $|X| = n < \infty$ and $|Y| = m < \infty$ be permuted such that $f(x_1) \leq f(x_2) \leq \cdots$ and $g(y_1) \leq g(y_2) \leq \cdots$. Then denote with

$$F_i := \sum_{k=1}^{i}\mu_X(x_k) \quad \text{for} \quad i\in\{1,\cdots,n\}$$

$$G_j := \sum_{k=1}^{j}\mu_Y(y_k) \quad \text{for} \quad j\in\{1,\cdots,m\}.$$

That means $F_i = F(t)$ for $t \in [f(x_i), f(x_{i+1}))$ for $i \in \{1,\cdots,n\}$ and $G_j = G(t)$ for $t \in [g(y_j), g(y_{j+1}))$ for $j \in \{1,\cdots,m\}$.

Now define

$$H : \mathbb{R}^2 \to \mathbb{R}, \quad H(s,t) := \min\{F(s), G(t)\}.$$
$$H_{ij} := H(f(x_i), g(y_j)) \quad \text{for } i \in \{1, \cdots, n\}, j \in \{1, \cdots, m\}.$$
$$\mu_{ij} := H_{ij} - \max\{H_{i-1j}, H_{ij-1}\} \quad \text{for } i \in \{1, \cdots, n\}, j \in \{1, \cdots, m\}.$$
$$\mu : X \times Y \to [0,1], \quad \mu(x_i, y_j) := \mu_{i,j} \quad \text{for } i \in \{1, \cdots, n\}, j \in \{1, \cdots, m\}.$$

**2.13 Lemma**
*Let* $\mu_X(x) > 0 \quad \forall x$ *and* $\mu_Y(y) > 0 \quad \forall y$. *Then*

$$H_{i,j+1} < H_{i+1,j} \Rightarrow \mu_{i',j+1} = 0 \quad \forall \quad i' \leq i \tag{2.18}$$
$$H_{i,j+1} > H_{i+1,j} \Rightarrow \mu_{i+1,j'} = 0 \quad \forall j' \leq j \tag{2.19}$$
$$H_{i,j+1} = H_{i+1,j} \Rightarrow \mu_{i',j'} = 0 \quad \forall j' \leq j, \quad \forall i' \leq i$$

*with* $i, i' \in \{1, \cdots, n\}$ *and* $j, j' \in \{1, \cdots, m\}$.

**Proof:** It is sufficient to prove (2.18) since (2.19) follows by switching the roles of $F$ and $G$. Let $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, m\}$ be given. Since $\mu_X(x_i) > 0$ it follows that $F_i < F_{i+1}$ and since $\mu_Y(y_j) > 0$ it follows that $G_j < G_{j+1}$. Let $i' \in \{1, \cdots, n\}$ with $i' \leq i$ be given. Assume $F_{i'} \leq G_j$. Then

$$\begin{aligned} \mu_{i',j+1} &= H_{i',j+1} - \max\{H_{i',j}, H_{i'-1,j+1}\} \\ &= \min\{F_{i'}, G_{j+1}\} - \max\{\min\{F_{i'}, G_j\}, \min\{F_{i'-1}, G_{j+1}\}\} \\ &= F_{i'} - \max\{F_{i'}, F_{i'-1}\} \\ &= F_{i'} - F_{i'} = 0. \end{aligned} \tag{2.20}$$

Assume $H_{i,j+1} < H_{i+1,j}$. From a case distinction one obtains that:

$$H_{i,j+1} < H_{i+1,j} \Rightarrow \min\{F_i, F_{i+1}, G_j, G_{j+1}\} = F_i$$

since

$$\begin{aligned} &1.) F_i < G_j \\ &\quad \Rightarrow H_{i+1,j} = \min\{F_{i+1}, G_j\} > F_i = \min\{F_i, G_{j+1}\} = H_{i,j+1} \quad \surd \\ &2.) F_i > G_j \\ &\quad \Rightarrow H_{i+1,j} = \min\{F_{i+1}, G_j\} = G_j < \min\{F_i, G_{j+1}\} = H_{i,j+1} \quad \lightning \\ &3.) F_i = G_j \\ &\quad \Rightarrow H_{i+1,j} = \min\{F_{i+1}, G_j\} = \min\{F_{i+1}, F_i\} = F_i \\ &= G_j = \min\{G_j, G_{j+1}\} = \min\{F_i, G_{j+1}\} = H_{i,j+1} \quad \lightning \end{aligned}$$

Therefore $F_{i'} < G_j \quad \forall i' \leq i$ in the case that $H_{i,j+1} < H_{i+1,j}$. Analogously for $H_{i,j+1} > H_{i+1,j}$. For the case that $H_{i,j+1}, = H_{i+1,j}$ it follows $F_i = G_j$ (see case distinction above). Therefore in all three cases $H_{i,j+1} < H_{i+1,j}$, $H_{i,j+1} > H_{i+1,j}$ and $H_{i,j+1} = H_{i+1,j}$ it holds $F_{i'} \leq G_j \quad \forall i' \leq i$. With (2.20) it follows $\mu_{i',j+1} = 0 \quad \forall i' \leq i$ respectively $\mu_{i+1,j'} = 0 \quad \forall j' \leq j$.

□

**2.14 Lemma**
*Let $n' \leq n$ and $m' \leq m$:*

$$\sum_{i=1}^{n'} \sum_{j=1}^{m'} \mu_{ij} = H_{n'm'}. \tag{2.21}$$

**Proof:** The statement is shown in two steps: First the statement is shown for the first row via proof by induction and the first column via proof by induction. Then the statement is shown via proof by induction for all columns and rows. Therefore: $H_{1,j} = \min\{F_1, G_j\}$. Proof by Induction: Anchor:

$$H_{1,1} = \min\{F_1, G_1\}, \quad \mu_{1,1} = H_{1,1} - \max\{0, 0\} = H_{1,1}.$$

Step:

$$\sum_{k=1}^{j} \mu_{1,k} = H_{1,j} \Rightarrow \sum_{k=1}^{j+1} = H_{1,j+1}.$$

$$\begin{aligned} \mu_{1,j+1} &= H_{1,j+1} - \max\{H_{1,j}, 0\} \\ &= H_{1,j+1} - H_{1,j} \\ &= H_{1,j+1} - \sum_{k=1}^{j} \mu_{1,k} \\ &\Leftrightarrow \\ \mu_{1,j+1} + \sum_{k=1}^{j} \mu_{1,k} &= H_{1,j+1} \\ \sum_{k=1}^{j+1} \mu_{1,k} &= H_{1,j+1} \end{aligned}$$

Step end.

It follows:

$$\sum_{k=1}^{j} \mu_{1,k} = H_{1,j} \quad \forall j$$
$$\sum_{r=1}^{i} \mu_{r,1} = H_{i,1} \quad \forall i.$$

Induction step 2:

$$\sum_{r=1}^{i+1}\sum_{k=1}^{j} \mu_{r,k} = H_{i+1,j} \Rightarrow \sum_{r=1}^{i+1}\sum_{k=1}^{j+1} \mu_{r,k} = H_{i+1,j+1}.$$

Proof of the induction 2:

$$\mu_{i+1,j+1} = H_{i+1,j+1} - \max\{H_{i,j+1}, H_{i+1,j}\}$$

Assume $H_{i,j+1} \leq H_{i+1,j}$, then:

$$\mu_{i+1,j+1} = H_{i+1,j+1} - \sum_{r=1}^{i+1}\sum_{k=1}^{j} \mu_{r,k}$$
$$\mu_{i+1,j+1} + \sum_{r=1}^{i+1}\sum_{k=1}^{j} \mu_{r,k} = H_{i+1,j+1}$$
$$\sum_{r=1}^{i+1}\sum_{k=1}^{j+1} \mu_{r,k} - \sum_{r=1}^{i} \mu_{r,j+1} = H_{i+1,j+1}.$$

With Lemma 2.13 it follows $\sum_{r=1}^{i} \mu_{r,j+1} = 0$.
Therefore $\sum_{r=1}^{i+1}\sum_{k=1}^{j+1} \mu_{r,k} = H_{i+1,j+1}$. Analogously for $Hi, j+1 > H_{i+1,j}$ □

**2.15 Lemma**
$\mu \in \mathcal{M}(\mu_X, \mu_Y)$.

**Proof:** Let $A = \{x_i\}$. It has to be shown that:

$$\mu(A \times Y) = \mu_X(A)$$
$$\Leftrightarrow \sum_{j=1}^{m} \mu_{ij} = \mu_X(A) = \mu_X(\{x_i\}) = F_i - F_{i-1}.$$

With Lemma 2.14 it follows:

$$\begin{aligned} Q := \sum_{k=1}^{i}\sum_{j=1}^{m} \mu_{k,j} = H_{i,m} &= \min\{F_i, G_m\} \\ &= \min\{F_i, \max_j\{G_j\}\} \\ &= \min\{F_i, \max_r\{F_r\}\} = F_i. \end{aligned}$$

$$P := \sum_{k=1}^{i-1}\sum_{j=1}^{m} \mu_{k,j} = H_{i-1,m} = \min\{F_i i - 1, G_m\} = F_i i - 1.$$

Then:

$$F_i - F_{i-1} = Q - P = \sum_{k=1}^{i}\sum_{j=1}^{m} \mu_{k,j} - \sum_{k=1}^{i-1}\sum_{j=1}^{m} \mu_{k,j} = \sum_{k=i}^{i}\sum_{j=1}^{m} \mu_{k,j} = \sum_{j=1}^{m} \mu_{i,j}.$$

Any set $A' \subseteq X$ is a finite union of sets like $A$ ($A' = \bigcup_{i\in I} A_i$). For the sets in $Y$ analogously.
Therefore $\mu \in \mathcal{M}(\mu_X, \mu_Y)$.

□

**2.16 Lemma**
*In the situation of Lemma 2.9 let $|X| = n$ and $|Y| = m$. Then it holds:*

$$\inf_{\mu\in\mathcal{M}}(\mu_X, \mu_Y) \int_{X\times Y} |f(x) - g(y)| \mu(dx \times dy) = \int_{\mathbb{R}} |F(u) - G(u)| du. \quad (2.22)$$

**Proof:** Denote with $d_{i,j}(t) := \mathbb{1}_{[f(x_i),\infty)}(t) - \mathbb{1}_{[g(y_j),\infty)}(t)$. Let $t \in \mathbb{R}$. Then $\forall i, j, k, l$ with $\mu_{i,j} > 0$ and $\mu_{k,l} > 0$ it holds that $\operatorname{sign}(d_{i,j}(t)) = \operatorname{sign}(d_{k,l}(t))$.
Proof by contradiction: Assume $\operatorname{sign}(d_{i,j}(t)) \neq \operatorname{sign}(d_{k,l}(t))$ or more specifically assume $d_{i,j}(t) > 0$ and $d_{k,l}(t) < 0$. Then it follows $f(x_i) \leq t < g(y_j)$ and $g(y_l) \leq t < f(x_k)$. It follows that $i < k, l < j$. By definition of $\mu$ it holds

$$\mu_{k,j} = H_{k,j} - \max\{H_{k-1,j}, H_{k,j-1}\}.$$

Assume $H_{k-1,j} \geq H_{k,j-1}$. Then with Lemma 2.13 it follows $\mu_{k,l} = 0$, which is a contradiction.
Assume $H_{k-1,j} \leq H_{k,j-1}$. Then with Lemma 2.13 it follows $\mu_{i,j} = 0$, which

is also a contradiction. Therefore it holds that $\forall i, j, k, l$ with $\mu_{i,j} > 0$ and $\mu_{k,l} > 0$, $\text{sign}(d_{i,j}(t)) = \text{sign}(d_{k,l}(t))$. With this it follows:

$$\begin{aligned}
&\int_{\mathbb{R}} |F(t) - G(t)|dt \\
&= \int_{\mathbb{R}} |\sum_{i=1}^{n} \mu_X(x_i) \cdot \mathbb{1}_{[f(x_i),\infty)}(t) - \sum_{j=1}^{m} \mu_Y(y_j) \cdot \mathbb{1}_{[g(y_j),\infty)}(t)|dt \\
&= \int_{\mathbb{R}} |\sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{i,j} \cdot \mathbb{1}_{[f(x_i),\infty)}(t) - \sum_{j=1}^{m}\sum_{i=1}^{n} \mu_{i,j} \cdot \mathbb{1}_{[g(y_j),\infty)}(t)|dt \\
&= \int_{\mathbb{R}} |\sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{i,j} \cdot (\mathbb{1}_{[f(x_i),\infty)}(t) - \mathbb{1}_{[g(y_j),\infty)}(t))|dt \\
&= \int_{\mathbb{R}} \sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{i,j} \cdot |\mathbb{1}_{[f(x_i),\infty)}(t) - \mathbb{1}_{[g(y_j),\infty)}(t)|dt \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{i,j} \cdot \int_{\mathbb{R}} |\mathbb{1}_{[\min\{f(x_i),g(y_j)\}, \max\{f(x_i),g(y_j)\})}(t)|f(x_i) - g(y_j)|dt \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{i,j} \cdot |f(x_i) - g(y_j)| \\
&= \int_{X\times Y} |f(x) - g(y)|\mu(d_X \times d_Y).
\end{aligned}$$

Lemma 2.9. showed that:

$$\inf_{\mu' \in \mathcal{M}}(\mu_X, \mu_Y) \int_{X\times Y} |f(x) - g(y)|\mu'(dx \times dy) \geq \int_{\mathbb{R}} |F(t) - G(t)|dt.$$

Since $\mu \in \mathcal{M}$ ( Lemma 2.15), and

$$\int_{\mathbb{R}} |F(t) - G(t)|dt = \int_{\mathbb{X}\times\mathbb{Y}} |f(x) - g(y)|\mu(d_X \times d_Y)$$

it follows that

$$\inf_{\mu' \in \mathcal{M}}(\mu_X, \mu_Y) \int_{X\times Y} |f(x) - g(y)|\mu'(dx \times dy) = \int_{X\times Y} |f(x) - g(y)|\mu(d_X \times d_Y).$$

Therefore

$$\inf_{\mu' \in \mathcal{M}}(\mu_X, \mu_Y) \int_{X\times Y} |f(x) - g(y)|\mu'(dx \times dy) = \int_{\mathbb{R}} |F(t) - G(t)|dt.$$

□

With the following theorems from Villani [33] an alternative proof of Lemma 2.16 can be formulated.

**2.17 Theorem** *(Optimal transportation theorem for a quadratic cost on $\mathbb{R}$) [33, Section 2.2 Theorem 2.18]*
*Let $\mu_F, \mu_G$ be two probability measures on $\mathbb{R}$, with respective cumulative distribution functions $F$ and $G$. Let $\Pi$ be the probability measure on $\mathbb{R}^2$ with joint two-dimensional cumulative distribution function*

$$H(s,t) = \min\{F(s), G(t)\}.$$

*Then, $\Pi$ belongs to $\mathcal{M}(\mu_F, \mu_G)$, and is optimal in the Kantorovich transportation problem between $\mu_F$ and $\mu_G$ for the quadratic cost function $c(r,s) = |r-s|^2$. Moreover, the value of the optimal transportation cost is*

$$\mathcal{T}_2(\mu_F, \mu_G) = \int_0^1 |F^{-1}(t) - G^{-1}(t)|^2 dt.$$

**2.18 Remark** *[33, Section 2.2 Remark 2.19]*

*(i) The* ***Hoeffding-Frechet theorem*** *states that a non-negative function $H$ on $\mathbb{R}^2$, non-decreasing and right-continuous in each argument, defines a probability measure $\pi$ on $\mathbb{R}^2$ with given marginals $\mu, \nu$ if and only if*

$$\forall (x,y) \in \mathbb{R}^2, \quad F(x) + G(y) - 1 \leq H(x,y) \leq \min(F(x), G(y)),$$

*where $F$ and $G$ are the cumulative distribution functions associated with $\mu$ and $\nu$ respectively.*

*(ii) The $\pi$ constructed in Theorem 2.17 is optimal whatever the convex cost. More precisely, $\pi$ is optimal as soon as the cost function $c(x,y)$ takes the form $c(x-y)$, where $c$ is a convex non-negative symmetric function on $\mathbb{R}$. In this case the optimal transportation cost is*

$$\mathcal{T}_1(\mu_F, \mu_G) = \int_0^1 |F^{-1}(t) - G^{-1}(t)| dt.$$

*(iii) The total transportation cost associated with the cost function $c(x,y) = |x-y|$ is*

$$\mathcal{T}_1(\mu_u, \mu_v) = \int_0^1 |U^{-1}(t) - V^{-1}(t)| dt = \int_{\mathbb{R}} |U(t) - V(t)| dt.$$

*Proof of Lemma 2.16 (second proof):*
Define $\mu_F := f_{\#\mu_X}$ and $\mu_G := g_{\#\mu_Y}$. Then

$$F(s) = \mu_F((-\infty, s]) = f_{\#\mu_X}((-\infty, s]) = \mu_X(f^{-1}((-\infty, s]))$$

and

$$G(t) = \mu_G((-\infty, t]) = g_{\#\mu_Y}((-\infty, t]) = \mu_Y(g^{-1}((-\infty, t])).$$

Then it follows:

$$\begin{aligned}\int_{R(s_0,t_0)} dh_{\#\mu} =& h_{\#\mu}(R(s_0, t_0)) \\ =& \mu(h^{-1}(R(s_0, t_0))) \\ =& \mu(h^{-1}(\{(s, t) : s \leq s_0, t \leq t_0\})) \\ =& \mu(\{(x_i, y_j) : f(x_i) \leq s_0, g(y_j) \leq t_0\}).\end{aligned}$$

Then for $n' := \max\{i : f(x_i) \leq s_0\}$ and $m' := \max\{j : g(y_j) \leq t_0\}$ it follows:

$$\mu(\{(x_i, y_j) : f(x_i) \leq s_0, g(y_j) \leq t_0\}) = \sum_{n'}^{i=1} \sum_{m'}^{j=1} \mu_{i,j}.$$

With Lemma 2.14 it follows that $\sum_{n'}^{i=1} \sum_{m'}^{j=1} \mu_{i,j} = H_{n',m'}$. Therefore

$$\sum_{n'}^{i=1} \sum_{m'}^{j=1} \mu_{i,j} = H_{n',m'} = H(s_0, t_0).$$

Therefore $H$ is the cumulative distribution function of $h_{\#\mu}$. Therefore with Theorem 2.17 and $\pi := h_{\#\mu}$ it follows that $\mu \in \mathcal{M}(\mu_X, \mu_Y)$ and is optimal in the Kantorovich transportation problem, that means

$$\inf_{\mu' \in \mathcal{M}}(\mu_X, \mu_Y) \int_{X \times Y} |f(x) - g(y)|^2 \mu'(dx \times dy) = \int_{X \times Y} |f(x) - g(y)|^2 \mu(dx \times dy)$$

and

$$\int_{X \times Y} |f(x) - g(y)|^2 \mu(dx \times dy) = \int_0^1 |F^{-1}(t) - G^{-1}(t)|^2 dt.$$

.

With Remark 2.18(*ii*) it follows

$$\inf_{\mu' \in \mathcal{M}} (\mu_X, \mu_Y) \int_{X \times Y} |f(x) - g(y)| \mu'(dx \times dy) = \int_{X \times Y} |f(x) - g(y)| \mu(dx \times dy).$$

and

$$\int_{X \times Y} |f(x) - g(y)| \mu(dx \times dy) = \int_0^1 |F^{-1}(t) - G^{-1}(t)| dt.$$

.

Furthermore with Remark 2.18(*iii*) it follows

$$\int_{X \times Y} |f(x) - g(y)| \mu(dx \times dy) = \int_{\mathbb{R}} |F(t) - G(t)| dt.$$

Therefore it follows

$$\inf_{\mu' \in \mathcal{M}(\mu_X, \mu_Y)} \int_{X \times Y} |f(x) - g(y)| \mu'(dx \times dy) = \int_{\mathbb{R}} |F(t) - G(t)| dt.$$

□

**2.19 Corollary** *(DE = FLB)*
*Let $|X| = n$ and $|Y| = m$. Then it holds:*

$$\boldsymbol{FLB}(\mathbb{X}, \mathbb{Y}) = \boldsymbol{DE}(\mathbb{X}, \mathbb{Y}). \tag{2.23}$$

*Proof:* Analogously to [26, Cor. 6.1], but instead of using Lemma 2.9 which implies "$\geq$" using Lemma 2.16 which implies "=". □

### 2.2.2 Examples of Optimal Transport Maps

**Example 1**

$$\mu_X = \frac{1}{6} \cdot \begin{pmatrix} 1 & 4 & 1 \end{pmatrix}$$

$$f = \begin{pmatrix} 10 & 10 & 100 \end{pmatrix}$$

$$F = \frac{1}{6} \cdot \begin{pmatrix} 1 & 5 & 6 \end{pmatrix}$$

$$\mu_Y = \frac{1}{6} \cdot \begin{pmatrix} 1 & 3 & 2 \end{pmatrix}$$

$$g = \begin{pmatrix} 1 & 5 & 7 \end{pmatrix}$$

$$G = \frac{1}{6} \cdot \begin{pmatrix} 1 & 4 & 6 \end{pmatrix}$$

$$H = \frac{1}{6} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 4 & 5 \\ 1 & 4 & 6 \end{pmatrix}$$

$$\mu = \frac{1}{6} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

The left-hand-side of the equation looks like this:

$$\begin{aligned}
&\int_{X \times Y} |f(x) - g(y)| \mu(d_x \times d_y) \\
&= \sum_{i,j} \mu_{i,j} \cdot |f_i - g_j| \\
&= \mu_{1,1} \cdot |f_1 - g_1| + \mu_{2,2} \cdot |f_2 - g_2| + \mu_{2,3} \cdot |f_2 - g_3| + \mu_{3,3} \cdot |f_3 - g_3| \\
&= \frac{1}{6} \cdot |10 - 1| + \frac{3}{6} \cdot |10 - 5| + \frac{1}{6} \cdot |10 - 7| + \frac{1}{6} \cdot |100 - 7| \\
&= \frac{1}{6} \cdot |9| + \frac{3}{6} \cdot |5| + \frac{1}{6} \cdot |3| + \frac{1}{6} \cdot |93| \\
&= \frac{1}{6} \cdot (9 + 15 + 3 + 93) \\
&= \frac{1}{6} \cdot 120 \\
&= 20
\end{aligned}$$

.

The right-hand-side of the equation looks like this:

$$
\begin{aligned}
&\int_t |F(t) - G(t)| \\
&= \sum_{i=2}^{n_X+n_Y} |t_i - t_{i-1}| \cdot |F(t_{i-1}) - G(t_{i-1})| \\
&= |t_2 - t_1| \cdot |F(t_1) - G(t_1)| + |t_3 - t_2| \cdot |F(t_2) - G(t_2)| \\
&+ |t_4 - t_3| \cdot |F(t_3) - G(t_3)| + |t_5 - t_4| \cdot |F(t_4) - G(t_4)| \\
&= |5 - 1| \cdot |F(1) - G(1)| + |7 - 5| \cdot |F(5) - G(5)| \\
&+ |10 - 7| \cdot |F(7) - G(7)| + |100 - 10| \cdot |F(10) - G(10)| \\
&= |5 - 1| \cdot \frac{1}{6} \cdot |0 - 1| + |7 - 5| \cdot \frac{1}{6} \cdot |0 - 4| \\
&+ |10 - 7| \cdot \frac{1}{6} \cdot |0 - 6| + |100 - 10| \cdot \frac{1}{6} \cdot |5 - 6| \\
&= |4| \cdot \frac{1}{6} \cdot |-1| + |2| \cdot \frac{1}{6} \cdot |-4| \\
&+ |3| \cdot \frac{1}{6} \cdot |-6| + |90| \cdot \frac{1}{6} \cdot |-1| \\
&= \frac{2}{3} + \frac{4}{3} + 3 + 15 \\
&= 20
\end{aligned}
$$

.

**Example 2**

$$\mu_X = \frac{1}{20} \cdot \begin{pmatrix} 2 & 1 & 1 & 4 & 3 & 8 & 1 \end{pmatrix}$$

$$f = \begin{pmatrix} 5 & 10 & 11 & 14 & 19 & 25 & 100 \end{pmatrix}$$

$$F = \frac{1}{20} \cdot \begin{pmatrix} 2 & 3 & 4 & 8 & 11 & 19 & 20 \end{pmatrix}$$

$$\mu_Y = \frac{1}{20} \cdot \begin{pmatrix} 4 & 1 & 1 & 6 & 8 \end{pmatrix}$$

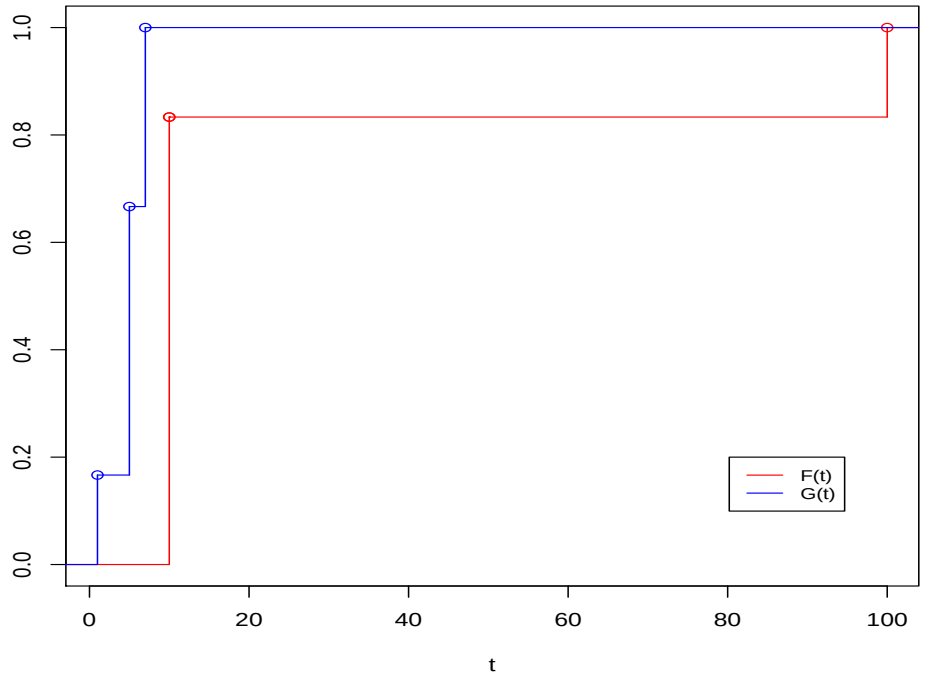


**Figure 2.1:** Graphs of the two cumulative distributions constructed in **Example 1** (section 2.2.2).

$$g = \begin{pmatrix} 1 & 5 & 7 & 14 & 35 \end{pmatrix}$$

$$G = \frac{1}{20} \cdot \begin{pmatrix} 4 & 5 & 6 & 12 & 20 \end{pmatrix}$$

$$H = \frac{1}{20} \cdot \begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 4 & 5 & 6 & 8 & 8 \\ 4 & 5 & 6 & 11 & 11 \\ 4 & 5 & 6 & 12 & 19 \\ 4 & 5 & 6 & 12 & 20 \end{pmatrix}$$

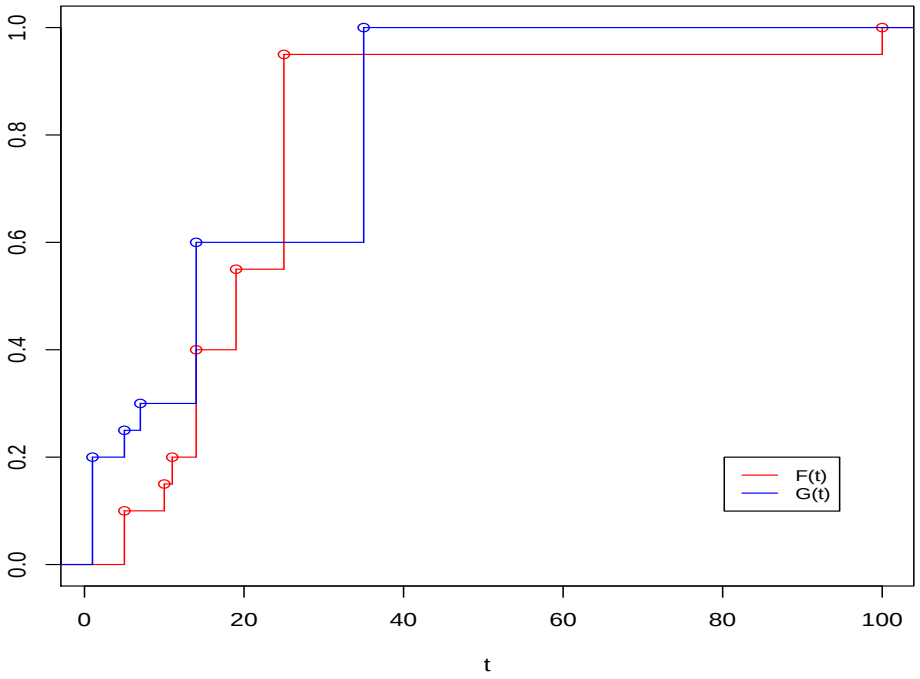


**Figure 2.2:** Graphs of the two cumulative distributions constructed in **Example 2** (section 2.2.2).

$$\mu = \frac{1}{20} \cdot \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

### 2.2.3 $\mathbf{DE}(X, Y)$ is a Pseudo-metric

**Claim:** $\mathbf{DE}(X, Y)$ is a pseudo-metric.
**Proof:** Let $(X, d_X, \mu_X)$,$(Y, d_Y, \mu_Y)$,$(Z, d_Z, \mu_Z)$ be given.

1. $\mathbf{DE}(X, X) = \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{X,1}(t) - S_{X,1}(t)|dt \Big) = 0$
2. $$\begin{aligned}\mathbf{DE}(X, Y) &= \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{X,1}(t) - S_{Y,1}(t)|dt \Big) \\ &= \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{Y,1}(t) - S_{X,1}(t)|dt \Big) \\ &= \mathbf{DE}(Y, X)\end{aligned}$$
3. $$\begin{aligned}\mathbf{DE}(X, Y) &= \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{Y,1}(t) - S_{X,1}(t)|dt \Big) \\ &= \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{X,1}(t) - S_{Z,1}(t) + S_{Z,1}(t) - S_{Y,1}(t)|dt \Big) \\ &\leq \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{X,1}(t) - S_{Z,1}(t)| + |S_{Z,1}(t) - S_{Y,1}(t)|dt \Big) \\ &= \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{X,1}(t) - S_{Z,1}(t)|dt \Big) + \frac{1}{2}\Big( \int_{\mathbb{R}} |S_{Z,1}(t) - S_{Y,1}(t)|dt \Big) \\ &= \mathbf{DE}(X, Z) + \mathbf{DE}(Z, Y)\end{aligned}$$

**2.20 Example** *(DE is only a pseudo-metric)*
*This example illustrates that* ***DE*** *is not a metric but only a pseudo-metric since it does not hold that* $\boldsymbol{DE}(\mathbb{X}, \mathbb{Y}) = 0 \Rightarrow \mathbb{X} = \mathbb{Y}$. *Let* $\mathbb{X} = (X, d_X, \mu_X)$ *and* $\mathbb{Y} = (Y, d_Y, \mu_Y)$ *with* $n_X = 2$, $n_Y = 3$ *and* $d \in \mathbb{R}$*:*

$$\mu_X = \left(\tfrac{1}{2}, \tfrac{1}{2}\right)$$
$$\mu_Y = \left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right)$$
$$d_X = \begin{pmatrix} 0 & d \\ d & 0 \end{pmatrix} \qquad d_Y = \begin{pmatrix} 0 & \frac{3}{4}d & \frac{3}{4}d \\ \frac{3}{4}d & 0 & \frac{3}{4}d \\ \frac{3}{4}d & \frac{3}{4}d & 0 \end{pmatrix}$$

*Then*

$$s_X(x_1) = d_X(x_1, x_2) \cdot \mu_X(x_2) = d \cdot \frac{1}{2}$$
$$s_X(x_2) = d_X(x_1, x_2) \cdot \mu_X(x_1) = d \cdot \frac{1}{2}$$
$$S_X(t) = \begin{cases} 0 & , t < d \cdot \frac{1}{2} \\ 1 & , t \geq d \cdot \frac{1}{2} \end{cases}$$

$$
\begin{aligned}
s_Y(y_1) &= d_Y(y_1, y_2) \cdot \mu_Y(y_2) + d_Y(y_1, y_3) \cdot \mu_Y(y_3) = \frac{1}{3} \cdot d \cdot \frac{3}{4} + \frac{1}{3} \cdot d \cdot \frac{3}{4} \\
&= d \cdot \frac{1}{2} \\
s_Y(y_2) &= d_Y(y_1, y_2) \cdot \mu_Y(y_1) + d_Y(y_2, y_3) \cdot \mu_Y(y_3) = \frac{1}{3} \cdot d \cdot \frac{3}{4} + \frac{1}{3} \cdot d \cdot \frac{3}{4} \\
&= d \cdot \frac{1}{2} \\
s_Y(y_3) &= d_Y(y_3, y_2) \cdot \mu_Y(y_2) + d_Y(y_1, y_3) \cdot \mu_Y(y_1) = \frac{1}{3} \cdot d \cdot \frac{3}{4} + \frac{1}{3} \cdot d \cdot \frac{3}{4} \\
&= d \cdot \frac{1}{2} \\
S_Y(t) &= \begin{cases} 0 & , t < d \cdot \frac{1}{2} \\ 1 & , t \geq d \cdot \frac{1}{2} \end{cases}
\end{aligned}
$$

*Then*

$$\boldsymbol{DE}(\mathbb{X}, \mathbb{Y}) = \int_{\mathbb{R}} |S_X(t) - S_Y(t)| dt = 0.$$

*Since by construction* $\mathbb{X} \neq \mathbb{Y}$ *it does not hold that*

$$\boldsymbol{DE}(\mathbb{X}, \mathbb{Y}) = 0 \Rightarrow \mathbb{X} = \mathbb{Y}.$$

## 2.3 Application of the Lower Bound

In this section the application of the **FLB** to the *animal-test-set* as done in [26] is described. Furthermore it is described how the experiments from [26] were repeated using *R*. The obtained results are discussed and compared to the results from [26].

### 2.3.1 Downsampling

The models in the *animal-test-set* [32] contain from 7000 to 30000 points each. Additionally the points are forming a triangle-mesh as an approximation of the surface. In [26] the presented idea is to select few characteristic points that represent each model well.

In [26] the **FLB** (2.8) was used as a measure of similarity between two given models. For this purpose only few points of the models are selected in a two-step-procedure, with the aim to retain a smaller model that represents

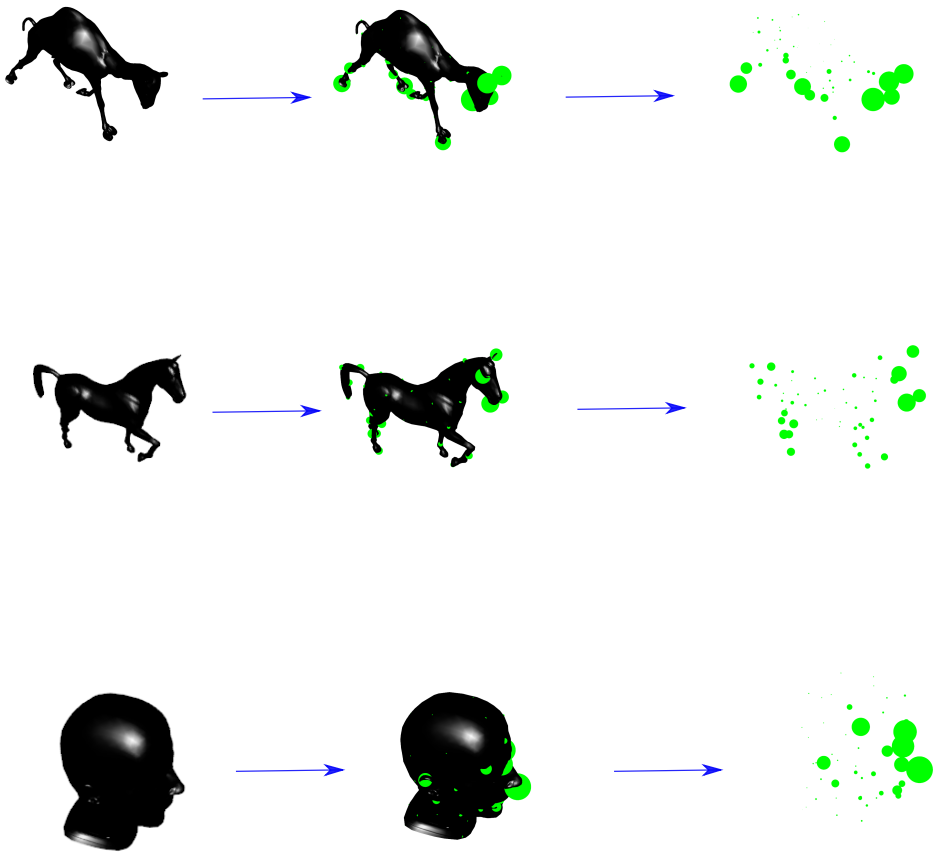

**Figure 2.3:** From each model a few characteristic points are selected. Displayed are the models of *camel*, *horse* and *head* from the *animal-test-set* [32].

the key characteristics of the model well. In the first step the *farthest point sampling procedure* is applied in the Euclidean space and $n_{\text{Euclidean}} = 4000$ points are selected.

Let $\hat{X}_k$ denote this reduced model. Then an intrinsic distance (or graph distance) was defined using Dijkstra's algorithm [9] on the graph $G(X_k)$ with vertex set $X_k$, where each vertex is connected by an edge to those vertices with which it shares a triangle (Dijkstra's algorithm is an algorithm to obtain shortest paths between pairs of nodes in a weighted graph. For the original paper refer to [9]). Since $\hat{X}_k \subset X_k$, by restriction, one endows $\hat{X}_k$ with this intrinsic distance as well. Memoli further sub-sampled $\hat{X}_k$, again using

the farthest point procedure ( with the distance computed using $G(X_k)$), and he retained only $n_{\text{dijkstra}} = 50.$ points. Denote the resulting set by $\mathbb{X}_k$. He then endowed $\mathbb{X}_k$ with the *normalized* distance metric inherited from the Dijkstra procedure described above and a probability measure based on Voronoi partitions: the mass (measure) at point $x \in \mathbb{X}_k$ equals the proportion of points in $\hat{X}_k$ which are closer to $x$ than to any other point in $\mathbb{X}_k$. From each model $X_k$ one thus obtains a discrete mm-space $(\mathbb{X}_k, d^{(k)}, \nu^{(k)})$. A matrix $((d_{ij}))$ is then computed with the **FLB**, such that $d_{ij} = \mathcal{D}(\mathbb{X}_i, \mathbb{X}_j)$.

The authors do not elaborate much on how the parameters $n_{\text{Euclidean}}$ and $n_{\text{dijkstra}}$ were chosen. However I assume that choosing more points, even if it might be computational feasible, does not always lead to a more precise similarity-measure. I think that a good balance between using too little points, that lead to over-generalizing the model and using too many points, that lead to over-specifying of the model, has to be found. This problem reminds of the under-and-over-fitting problem, that one often comes across in machine-learning [8]. It is also conceivable that the optimal number of points is problem-specific. That means that a good parameter for the *animal-test-set* does not necessarily transfer to the protein-test-set.

### 2.3.2 Nearest Neighbor Classification and Cross-Validation

Given the pairwise similarities of all models $((d_{ij}))$ the proposed classification-method in [26] is a $1-$NN-method (1-Nearest-Neighbor).

With any prediction-problem one is usually interested in how well the proposed method will work on new yet unseen data [5]. Hence one often splits the given data into a training-and test-set. An algorithm then computes a model with the training-set and the accuracy is then calculated based on the prediction-accuracy, that the model achieves on the test-set. The drawback of this approach is however, that the data available for the training is smaller. Also for small data-sets the choice of the test-set might not be representative and choosing a different test-set might lead to different results. By doing the split into a training- and test-set multiple times and averaging over the obtained performance, one gets an estimate for the actual prediction-accuracy of the model. This procedure is called cross-validation [19]. Many different variations of this approach exist. In [26] the following was done: from each class of *lions*, *cats*, *heads*, *horses*, *elephants*, *flamingos*, *camel* 1 model is chosen at random. The prediction for each chosen model is obtained from the nearest-neighbor computed with $((d_{ij}))$ of the remaining not selected models. This process is repeated 10000 times to get an estimate of how well

**Figure 2.4:** A shortest path between two points on the surface marked in blue.

the method works.

### 2.3.3 Results on the *animal-test-set*

I implemented this downsampling-procedure and classification-method in R and reused it on the *animal-test-set* for further investigation of the parameters $n_{\text{Euclidean}}$ and $n_{\text{dijkstra}}$. Memoli uses the **FLB**, whereas I use the **DE** for the calculation of the dissimilarity as I have shown that it holds that **FLB** = **DE** in the finite case in Theorem 2.19. Additionally attempting to reproduce the results from [26] assures that my implementation does not contain any larger deviations from the original implementation, let alone errors. Initially I used $n_{\text{Euclidean}} = 4000$ and $n_{\text{dijkstra}} = 50$. In figure 2.1 the confusion matrix obtained with the procedure described in 2.3.2 is shown. On the given models the method seems to work extremely well, only the classes *cats* and *lions* are separated poorly. However as described in [32] these models are computationally derived from similar 3D-models, on top of the obvious fact that lions and cats share some natural similarity, and hence are expected to be difficult to distinguish.

## 2.4 Approximating an Empirical Cumulative Distribution

The **DE** calculates the integral of the difference of two cumulative distributions. In this section it is explained how one can obtain an efficient representation of a cumulative distribution. Then in turn an approximation for the **DE** is derived that is computational more efficient than the **DE**. The key-

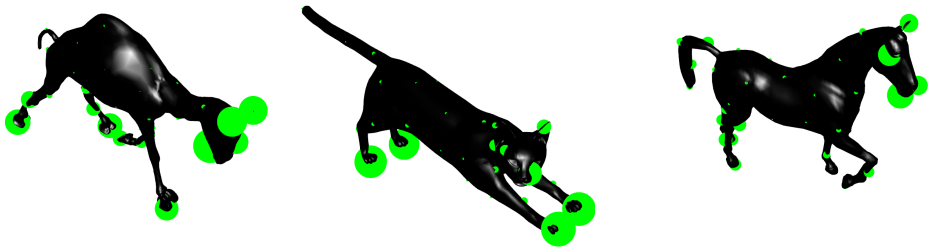

**Figure 2.5:** 50 points selected from the models. Bigger points indicate a higher weight.

| | **camel** | **cat** | **elephant** | **flamingo** | **head** | **horse** | **lion** |
|---|---|---|---|---|---|---|---|
| **camel** | 9153 | 0 | 0 | 0 | 0 | 0 | 847 |
| **cat** | 0 | 5130 | 0 | 102 | 0 | 905 | 3863 |
| **elephant** | 0 | 0 | 7236 | 914 | 0 | 1850 | 0 |
| **flamingo** | 0 | 0 | 0 | 10000 | 0 | 0 | 0 |
| **head** | 0 | 0 | 0 | 0 | 10000 | 0 | 0 |
| **horse** | 0 | 0 | 0 | 0 | 0 | 10000 | 0 |
| **lion** | 0 | 2888 | 0 | 0 | 0 | 0 | 7112 |

**Table 2.1:** Confusion matrix of 1−NN-classification on the *animal-test-set*. Row-wise the true labels are shown and column-wise the predicted labels are shown. For each model-class 1 model was chosen at random and using the remaining models predictions were made. This was repeated for 10000 times. Hence the sum in each row is 10000. For more details refer to section 2.3.2.

idea is to approximate the cumulative distribution by its quantiles. Then the approximation for the **DE** can be calculated by calculating the *manhattan-distance* between two such vectors of quantiles.

Let a set $X$ and a probability measure $\mu_X$ on $X$ and a measurable function $f : X \to [0, \infty)$ be given. Then the cumulative distribution function $F : \mathbb{R}_+ \to [0, 1]$ is defined as $F(t) = \mu_X(f^{-1}((-\infty, t]))$. Furthermore the generalized inverse under $\mu_X$ is defined as:

$$F^{-1}(t) = \inf\{u \in \mathbb{R} | F(u) > t\}.$$

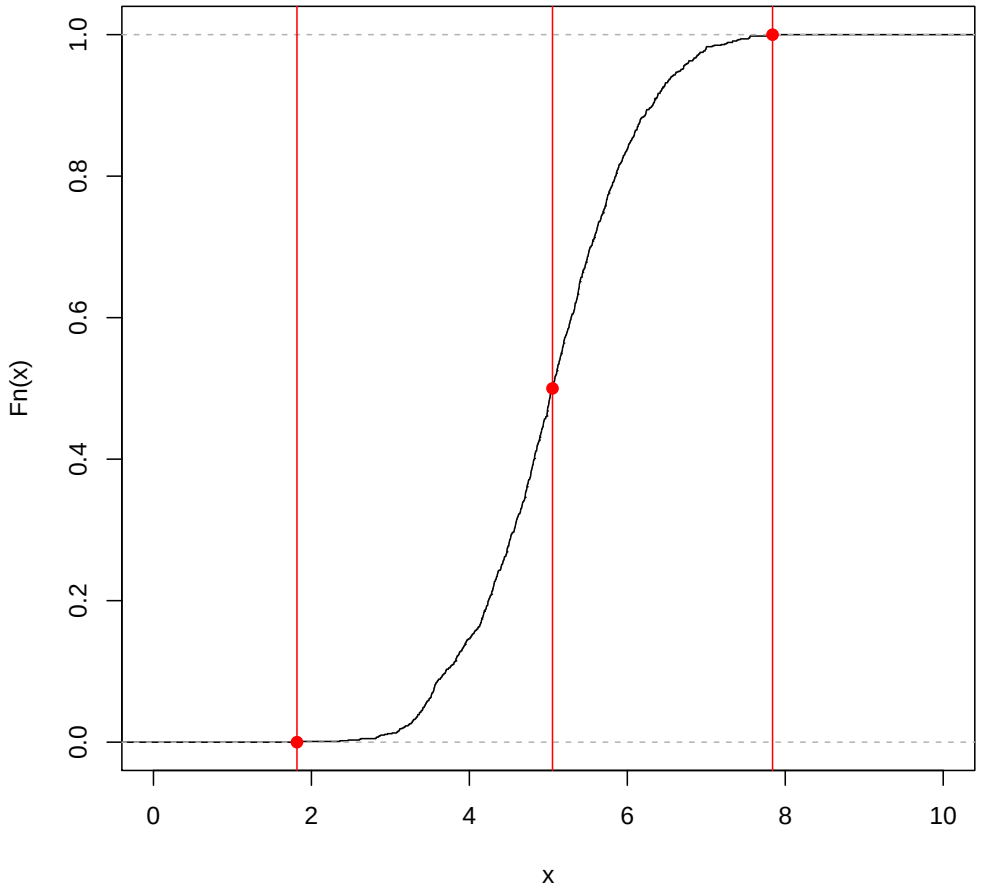


**Figure 2.6:** Approximation of a cumulative distribution by its quantiles marked in red.

Then define $\mathcal{A}(F, q) = (\mathcal{A}_0, \cdots, \mathcal{A}_{q+1})$ with

$$\mathcal{A}_i := F^{-1}(\frac{i}{q+1}) \quad \text{for} \quad i \in \{0, \cdots, q+1\}. \tag{2.24}$$

That means $\mathcal{A}(F, q)$ are the quantiles of $F$. An example is depicted in figure 2.6.

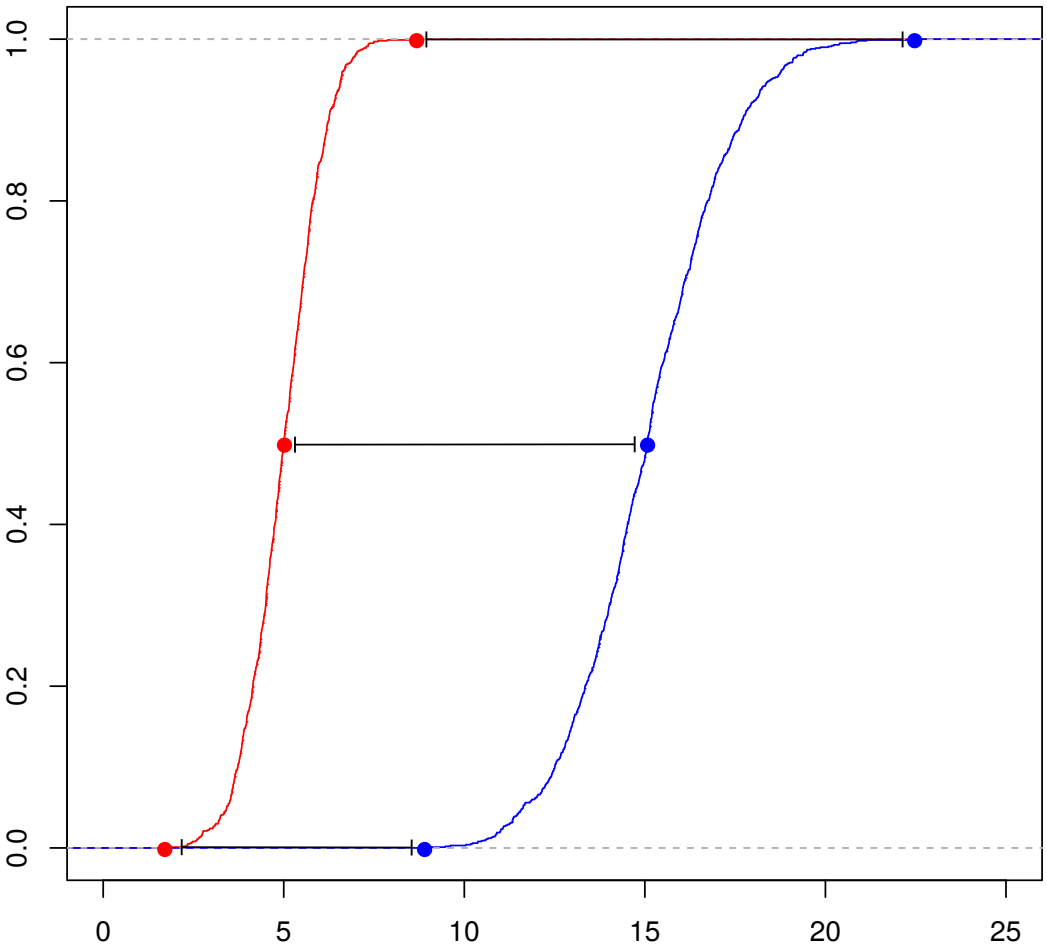


**Figure 2.7:** Illustration of the approximation of the **DE** as defined in equation (2.25). The difference between the quantiles of the two cumulative distribution functions is calculated.

Let two mp-spaces $(X, d_X, \mu_X)$ and $(Y, d_Y, \mu_Y)$ be given. Then define:

$$\mathbf{DE}_{\mathcal{A},q}(X,Y) := \frac{1}{q+2} \cdot \sum_{i=0}^{q+1} |\mathcal{A}(S_X, q)_i - \mathcal{A}(S_Y, q)_i|. \tag{2.25}$$

### 2.4.1 Upper Bound for the Error of Approximation

In the previous section a computational efficient approximation $\mathbf{DE}_{\mathcal{A},q}$ was introduced for the **DE**. In this section an upper bound for the error of this approximation is derived. Formally that means given two mp-spaces $(X, d_X, \mu_X)$ and $(Y, d_Y, \mu_Y)$ define as the error of the approximation:

$$Err_{\mathbf{DE}}(X,Y,q) := |\mathbf{DE}_{\mathcal{A},q}(X,Y) - \mathbf{DE}(X,Y)|$$
$$= |\frac{1}{q+2} \cdot \sum_{i=0}^{q+1} |\mathcal{A}(S_X,q)_i - \mathcal{A}(S_Y,q)_i| - \int_{\mathbb{R}} |S_X - S_Y||.$$

First an upper bound for the error between $S_X$ and $\mathcal{A}(S_X,q)$ is derived. Let a $q \in \mathbb{N}$ and a vector $\mathcal{A} \in \mathbb{R}^{q+2}$ be given. Define $\mathcal{P}_{\mathcal{A}} : \mathbb{R} \to [0,1]$:

$$\mathcal{P}_{\mathcal{A}}(t) := \max \frac{\{i \in \{0,\cdots,q+1\} \mid \mathcal{A}_i < t\} + 1}{q+2}. \tag{2.26}$$

That means $\mathcal{P}_{\mathcal{A}}(t)$ is a step-function whose graph jumps at each $\mathcal{A}_i$ by $\frac{1}{q+2}$. See figure 2.8 for an example.

It then holds:

$$\frac{1}{q+2} \cdot \sum_{i=0}^{q} (\mathcal{A}_{i+1} - \mathcal{A}_i) + x - \mathcal{A}_{q+1} = \int_0^x \mathcal{P}_{\mathcal{A}}(t)dt \quad \forall x \geq \mathcal{A}_{i+1}. \tag{2.27}$$

Given a $\mathcal{A} \in \mathbb{R}^{q+2}$ with $\mathcal{A}_i \leq \mathcal{A}_{i+1} \quad \forall i \in \{0,\cdots,q\}$, define:

$$\mathcal{B}_{Err}(\mathcal{A}) := \max_{F \in \mathcal{H}} \int_{\mathbb{R}} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt$$

where $\mathcal{H}$ is the set of all cumulative distribution functions $F$ with $F^{-1}(\frac{i}{q+1}) = \mathcal{A}_i \quad i \in \{0,\cdots,q+1\}$.

**Claim:**

$$B_{Err}(\mathcal{A}) \leq \frac{\mathcal{A}_{q+1} - \mathcal{A}_0}{q+2}$$

**Proof:** Let $i \in \{0,\cdots,q\}$ be given. Then for any $F \in \mathcal{H}$ it holds:

$$\frac{i}{q+1} \leq F(t) \leq \frac{i+1}{q+1} \quad \forall t \in [\mathcal{A}_i, \mathcal{A}_{i+1}).$$

Since $F(t_1) \leq F(t_2) \quad \forall t_1 \leq t_2 \in \mathbb{R} \quad \exists a \in [\mathcal{A}_i, \mathcal{A}_{i+1})$ with $F(a) = \frac{i+1}{q+2}$.

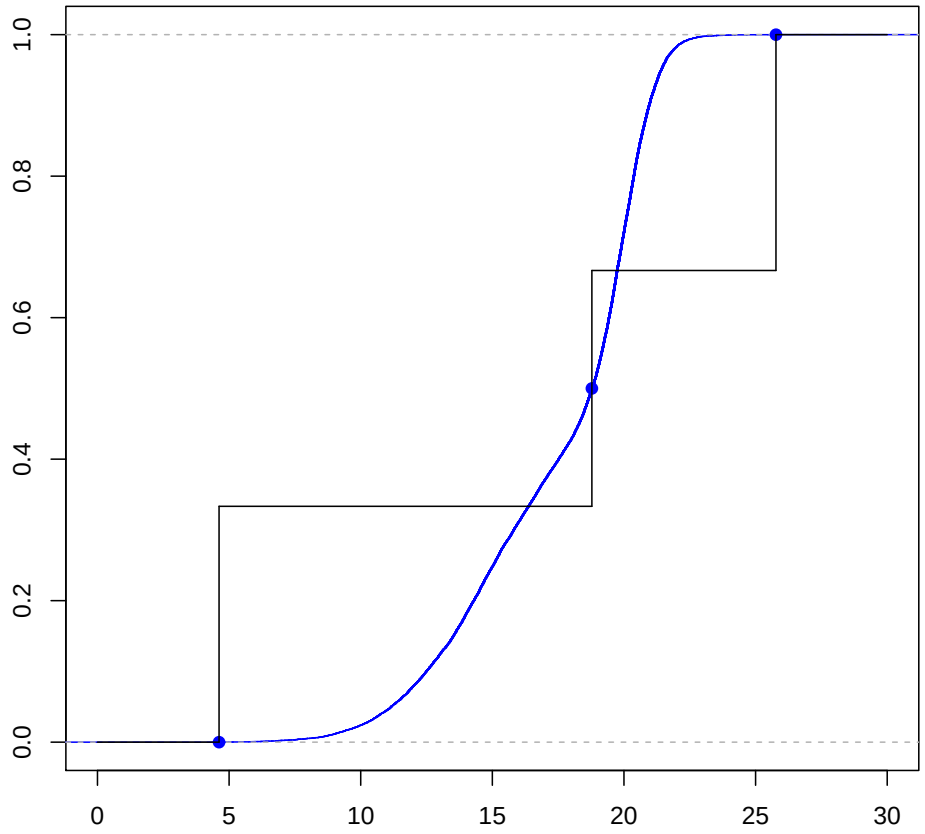


**Figure 2.8:** A cumulative distribution function with the step-function obtained from its quantiles $q = 1$.

Additionally it holds $\mathcal{P}_{\mathcal{A}}(t) = \frac{i+1}{q+2} \quad \forall t \in [\mathcal{A}_i, \mathcal{A}_{i+1})$. Therefore it follows

$$\begin{aligned}
&\int_{\mathcal{A}_i}^{\mathcal{A}_{i+1}} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt \\
&= \int_{\mathcal{A}_i}^{a} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt + \int_{a}^{\mathcal{A}_{i+1}} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt \\
&= \int_{\mathcal{A}_i}^{a} (\mathcal{P}_{\mathcal{A}}(t) - F(t))dt + \int_{a}^{\mathcal{A}_{i+1}} (F(t) - \mathcal{P}_{\mathcal{A}}(t))dt \\
&= \int_{\mathcal{A}_i}^{a} \mathcal{P}_{\mathcal{A}}(t)dt - \int_{\mathcal{A}_i}^{a} F(t)dt + \int_{a}^{\mathcal{A}_{i+1}} F(t)dt - \int_{a}^{\mathcal{A}_{i+1}} \mathcal{P}_{\mathcal{A}}(t)dt
\end{aligned}$$

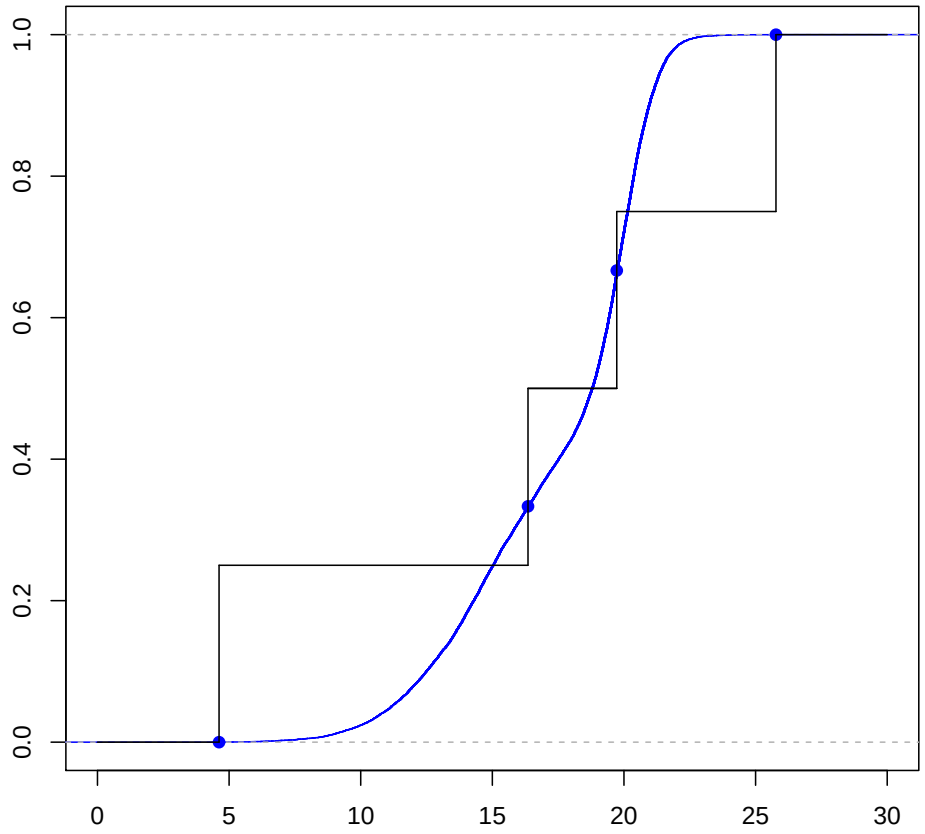


**Figure 2.9:** A cumulative distribution function with the step-function obtained from its quantiles with $q = 2$.

$$
\begin{aligned}
&= \int_{\mathcal{A}_i}^{a} \mathcal{P}_{\mathcal{A}}(t)dt - \int_{a}^{\mathcal{A}_{i+1}} \mathcal{P}_{\mathcal{A}}(t)dt - \int_{\mathcal{A}_i}^{a} F(t)dt + \int_{a}^{\mathcal{A}_{i+1}} F(t)dt \\
&= \frac{i+1}{q+2}(a - \mathcal{A}_i) - \frac{i+1}{q+2}(\mathcal{A}_{i+1} - a) - \int_{\mathcal{A}_i}^{a} F(t)dt + \int_{a}^{\mathcal{A}_{i+1}} F(t)dt \\
&= \frac{i+1}{q+2}(2a - \mathcal{A}_i - \mathcal{A}_{i+1}) - \int_{\mathcal{A}_i}^{a} F(t)dt + \int_{a}^{\mathcal{A}_{i+1}} F(t)dt. \qquad (2.28)
\end{aligned}
$$

In order to find a upper bound for equation (2.28) an upper bound for $\int_{\mathcal{A}_i}^{a} F(t)dt$ has to be found and a lower bound for $\int_{a}^{\mathcal{A}_{i+1}} F(t)dt$ has to be

found. It holds:

$$\int_{\mathcal{A}_i}^{a} F(t)dt \geq \frac{i}{q+1}(a - \mathcal{A}_i)$$
$$\int_{a}^{\mathcal{A}_{i+1}} F(t)dt \leq \frac{i+1}{q+1}(\mathcal{A}_{i+1} - a).$$

Plugging these bounds into equation (2.28) yields:

$$\begin{aligned}&\frac{i+1}{q+2}(2a - \mathcal{A}_i - \mathcal{A}_{i+1}) - \int_{\mathcal{A}_i}^{a} F(t)dt + \int_{a}^{\mathcal{A}_{i+1}} F(t)dt \\ &\leq \frac{i+1}{q+2}(2a - \mathcal{A}_i - \mathcal{A}_{i+1}) - \frac{i}{q+1}(a - \mathcal{A}_i) + \frac{i+1}{q+1}(\mathcal{A}_{i+1} - a) \qquad (2.29)\end{aligned}$$

Since (2.29) is linear in $a$, the maximum is achieved at $a = \mathcal{A}_i$ or $a = \mathcal{A}_{i+1}$. If $a = \mathcal{A}_i$, it follows:

$$\begin{aligned}&\frac{i+1}{q+2}(2\mathcal{A}_i - \mathcal{A}_i - \mathcal{A}_{i+1}) + \frac{i+1}{q+1}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{i+1}{q+2}(\mathcal{A}_i - \mathcal{A}_{i+1}) + \frac{i+1}{q+1}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{i+1}{q+1}(\mathcal{A}_{i+1} - \mathcal{A}_{i+1}) - \frac{i+1}{q+2}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{(i+1)(q+2)}{(q+1)(q+2)}(\mathcal{A}_{i+1} - \mathcal{A}_i) + \frac{(i+1)(q+1)}{(q+1)(q+2)}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{i+1}{(q+1)(q+2))}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &\leq \frac{q+1}{(q+1)(q+2))}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{1}{(q+2)}(\mathcal{A}_{i+1} - \mathcal{A}_i).\end{aligned}$$

The second-last equation comes from the fact that $i \leq q$.
If $a = \mathcal{A}_{i+1}$, it follows:

$$\begin{aligned}&\frac{i+1}{q+2}(2\mathcal{A}_{i+1} - \mathcal{A}_i - \mathcal{A}_{i+1}) - \frac{i}{q+1}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{i+1}{q+2}(\mathcal{A}_{i+1} - \mathcal{A}_i) - \frac{i}{q+1}(\mathcal{A}_{i+1} - \mathcal{A}_i) \\ &= \frac{(i+1)(q+2)}{(q+1)(q+2)}(\mathcal{A}_{i+1} - \mathcal{A}_i) - \frac{(i)(q+2)}{(q+1)(q+2)}(\mathcal{A}_{i+1} - \mathcal{A}_i)\end{aligned}$$

$$
\begin{aligned}
&= \frac{q-i+1}{(q+1)(q+2))}(\mathcal{A}_{i+1}-\mathcal{A}_i) \\
&\leq \frac{q+1}{(q+1)(q+2))}(\mathcal{A}_{i+1}-\mathcal{A}_i) \\
&= \frac{1}{(q+2)}(\mathcal{A}_{i+1}-\mathcal{A}_i).
\end{aligned}
$$

The second-last equation comes from the fact that $0 \leq i$.

In both of the cases $a = \mathcal{A}_i$ and $a = \mathcal{A}_{i+1}$ it holds:

$$
\int_{\mathcal{A}_i}^{\mathcal{A}_{i+1}} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt \leq \frac{1}{(q+2)}(\mathcal{A}_{i+1}-\mathcal{A}_i).
$$

Therefore it follows:

$$
\begin{aligned}
B_{Err}(\mathcal{A}) &= \max_{F\in\mathcal{H}} \int_{\mathbb{R}} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt \\
&= \max_{F\in\mathcal{H}} \sum_{i=0}^{q} \int_{\mathcal{A}_i}^{\mathcal{A}_{i+1}} |F(t) - \mathcal{P}_{\mathcal{A}}(t)|dt \\
&\leq \sum_{i=0}^{q} \frac{\mathcal{A}_{i+1}-\mathcal{A}_i}{q+2} \\
&= \frac{1}{q+2} \cdot (\mathcal{A}_{q+1} - \mathcal{A}_q + \mathcal{A}_q - \cdots - \mathcal{A}_1 + \mathcal{A}_1 - \mathcal{A}_0) \\
&= \frac{\mathcal{A}_{q+1}-\mathcal{A}_0}{q+2}.
\end{aligned}
$$

□

Using the abbreviations $\mathcal{A}^X := \mathcal{A}(S_X, q)$ and $\mathcal{A}^Y := \mathcal{A}(S_Y, q)$ the error for the approximation of the **DE** can be bounded as follows:

$$
\begin{aligned}
Err_{\mathbf{DE}}(X,Y,q) &= |\mathbf{DE}_{\mathcal{A}}(X,Y) - \mathbf{DE}(X,Y)| \\
&= |\frac{1}{q+2} \cdot \sum_{i=0}^{q+1} |\mathcal{A}_i^X - \mathcal{A}_i^Y| - \int_{\mathbb{R}} |S_X(t) - S_Y(t)|dt| \\
&= |\int_{\mathbb{R}} |\mathcal{P}_{\mathcal{A}^X}(t) - \mathcal{P}_{\mathcal{A}^Y}(t)|dt - \int_{\mathbb{R}} |S_X(t) - S_Y(t)|dt| \\
&= |\int_{\mathbb{R}} |\mathcal{P}_{\mathcal{A}^X}(t) - \mathcal{P}_{\mathcal{A}^Y}(t)| - |S_X(t) - S_Y(t)|dt| \\
&\leq |\int_{\mathbb{R}} |\mathcal{P}_{\mathcal{A}^X}(t) - \mathcal{P}_{\mathcal{A}^Y}(t) - S_X(t) + S_Y(t)|dt|
\end{aligned}
$$

$$
\begin{aligned}
&= |\int_{\mathbb{R}} |\mathcal{P}_{\mathcal{A}^X}(t) - S_X(t) + S_Y(t) - \mathcal{P}_{\mathcal{A}^Y}(t)|dt| \\
&\leq |\int_{\mathbb{R}} |\mathcal{P}_{\mathcal{A}^X}(t) - S_X(t)| + |S_Y(t) - \mathcal{P}_{\mathcal{A}^Y}(t)|dt| \\
&\leq |\int_{\mathbb{R}} |\mathcal{P}_{\mathcal{A}^X}(t) - S_X(t)|dt + \int_{\mathbb{R}} |S_Y(t) - \mathcal{P}_{\mathcal{A}^Y}(t)|dt| \\
&\leq \mathcal{B}_{Err}(\mathcal{A}^X) + \mathcal{B}_{Err}(\mathcal{A}^Y) \\
&= \frac{1}{q+2}(\mathcal{A}^X_{q+1} - \mathcal{A}^X_0) + \frac{1}{q+2}(\mathcal{A}^Y_{q+1} - \mathcal{A}^Y_0)
\end{aligned}
$$

This upper bound for the error shows that using larger values of $q$ leads to a smaller error. Furthermore, when the distributions $S_X$ and $S_Y$ are "steeper" the error is also smaller. With this upper bound a detour from $\mathcal{P}_{\mathcal{A}^X}$ to $S_X$ respectively $\mathcal{P}_{\mathcal{A}^Y}$ to $S_Y$ is taken. Hence it should be expected that this bound is rather pessimistic and it should be possible to find a bound that is smaller.

# Chapter 3

# Applying Metric Geometry to Protein-iso-surfaces

The theory as presented in [26] and the promising results on a somewhat comparable problem of classification of 3-dimensional objects (the *animal-test-set*) were presented in the preceding chapter. In this chapter it is at first discussed how a protein can be modeled in a reasonable way such that the **DE** can be applied.

Secondly in section 3.4 the implementation of Felix Berens [2] is described, formalized and minor changes are described, that were added to the approach. Additionally in section 3.5 it is described how the computation-time of the implementation was decreased.

Thirdly in sections 3.6 and 3.7 the idea of the **DE** is generalized by viewing each cumulative distribution function as a feature of a given object. The follow-up idea is then that a machine-learner can take these features as inputs, and combine them in more complex ways, than just calculating a difference as done in the **DE**.

## 3.1 Preprocessing - Generating the Surfaces from pdb-Files

Visual molecular dynamics (VMD) [17] is a molecular graphics program designed for the display and analysis of molecular structures. A pdb-file [3] (short for Protein-Data-Bank) is a standard format for proteins. For the purpose of this thesis a pdb-file can be thought of a file that stores geometrical properties which includes the relative positions of the atoms forming the protein. The pdb-files are converted to so called pqr-files with the program *PDB2PQR* [10]. The pqr-file can be thought of a pdb-file where additionally

information about the electric charge of each atom is added (The acronym is formed from: P for pdb, Q for charge, R for radius). With the corresponding plugin of VMD with the amber-force-field [1] the electrostatic potential of the given protein is calculated and as output a set of points in Euclidean space, all belonging to the same potential-value is obtained. The native exporting-option of VMD to export this surface to a wavefront-obj-file is used. The wavefront-obj-file-format contains the surface as a triangle-mesh, that means the Euclidean coordinates of the points and additionally connections between the points are stored. Triangle-meshes are a common way to store and visualize 3D-objects. Furthermore and especially important for the method proposed in this thesis, a geodesic distance, that is a distance on the surface of the object can be calculated using as the distance between two points, the shortest path between the points within the triangle-mesh. These triangle-meshes are however in most cases not connected and do not form a single connected component. Hence a calculation of a geodesic distance between any pair of points would not be possible. In order to use geodesic distances one needs a single connected component. To achieve this, the program *Manifold*[16] is used to create an *obj*-file with a single connected component. The program introduces new edges between points that are close together. The geodesic distances can then be calculated by calculating the length of a path between two points. This way also a geodesic distance between positive and negative potential is realized. This is done by merging the coordinates of the two potentials into one file. Then using *Manifold* one connected component is obtained, since additionally any points that are within the connected hull are discarded from the object. This is similar to the *naked-eye-approach* as only the information of the out-most potential is used, since the other parts are hidden behind the parts that are more in the front. The package *readobj* [18] in R is then used to read in the file.

Additionally the coordinates of the centers of the atoms forming the protein are read in from the pqr-file. Then the active center is extracted given a file containing motives, that means sequences of amino acids forming an active center. This is done with a *R*-script written for this thesis, which will be referred to as *centerSelect*.

All in all with the above steps completed, for each protein one is given:

- The coordinates in 3-dimensional-Euclidean space of the points that form the outer hull of the two potentials.
- A list that specifies for each point to which potential it belongs.
- A list of edges forming a triangle-mesh between the points of the hull.
- The coordinates in 3-dimensional-Euclidean space of the points that form the atoms of the protein.

- The coordinates in 3-dimensional-Euclidean space of the points that form the active center (that means these are a subset of the atom-coordinates).

## 3.2 Extending the Term mp-Space

A protein-iso-surface can be seen as a mp-space where the points are divided in two sub-classes, which in this case are the points that belong to the positive respectively negative potential. Following the notations in chapter 2.1:

A protein will be modeled in the following way:

$$\mathcal{X} = (X, d_X, d_{X_{\text{geo}}}, \mu_X, \text{pot})$$

where $d_X$ resemble the Euclidean distances, $d_{X_{\text{geo}}}$ resemble the geodesic distances, that means distances on the surface, $\mu_X$ is a probability measure and pot is a mapping $\text{pot} : X \to \{-1, 1\}$ specifying for each point to which potential it belongs. Introduce the following abbreviations:

$$\begin{aligned}
X^+ &:= \{x \in X : \text{pot}(x) = +\} \\
X^- &:= \{x \in X : \text{pot}(x) = -\} \\
\mathbb{X}^+ &:= (X^+, d_X, \frac{\mu_X}{\mu_X(X^+)}) \\
\mathbb{X}^- &:= (X^-, d_X, \frac{\mu_X}{\mu_X(X^-)}) \\
\mathbb{X} &:= (X, d_X, \mu_X) \\
\mathbb{X}^+_{\text{geo}} &:= (X^+, d_{X_{\text{geo}}}, \frac{\mu_X}{\mu_X(X^+)}) \\
\mathbb{X}^-_{\text{geo}} &:= (X^-, d_{X_{\text{geo}}}, \frac{\mu_X}{\mu_X(X^-)}) \\
\mathbb{X}_{\text{geo}} &:= (X, d_{X_{\text{geo}}}, \mu_X).
\end{aligned}$$

Then by construction $\mathbb{X}, \mathbb{X}^+, \mathbb{X}^-$ and $\mathbb{X}_{\text{geo}}, {\mathbb{X}_{\text{geo}}}^+, {\mathbb{X}_{\text{geo}}}^-$ are mp-spaces.

## 3.3 Reducing the Number of Points

It is common practice when dealing with point-clouds to apply a method to reduce the number of points in a first step. This is done for two reasons. Firstly using less points often increases the algorithm-performance, because less computational effort is necessary. Secondly and more importantly by

selecting few characteristic points, as done in [26] with the farthest-point-sampling-procedure, also the accuracy of the algorithm can increase. An additional case where a down-sampling-procedure is useful occurs, when the points are not uniformly distributed and the density varies.

In this section two approaches are introduced and formalized that are both a form of reducing the number of points in the model.

**3.1 Definiton** *(subset-induced-metric)*
*Define for a given metric space* $(M, d)$ *and a set* $A \subseteq M$ *the* subset-induced-metric $d_{|A} : A \times A \mapsto \mathbb{R}^+$

$$d_{|A}(a, b) := d(a, b) \quad \forall a, b \in A.$$

**3.2 Definiton** *(subset-induced-measure)*
*Define for a given measurable space* $(\Omega, \mathcal{A}, \mu)$ *and a set* $X \subseteq \Omega$ *the* $\sigma$*-algebra*

$$\mathcal{A}_{|X} := \{A \cap X : A \in \mathcal{A}\} \tag{3.1}$$

*which is called* trace-$\sigma$-algebra. *It holds* $A \in \mathcal{A}_{|X} \Leftrightarrow A \in \mathcal{A}$ *and* $A \subset X$. *Further define the measure* $\mu_{|X}$ *on* $\mathcal{A}_{|X}$ *as:*

$$\mu_{|X}(A) := \mu(A) \quad \forall A \in \mathcal{A}_{|X}. \tag{3.2}$$

**3.3 Definiton** *(Multiset)*
*Let a set* $M$ *with* $|M| < \infty$ *and a* $f : M \to \mathbb{N}_0$ *be given. The pair* $(M, f)$ *is called* multi-set. *Denote with* $supp((M, f)) := \{x \in M : f(x) > 0\}$. *Denote with* $|(M, f)| := |supp((M, f))|$ *and* $|f| := \sum_{x \in M} f(x)$.

Let $(X, d_X, \mu_X)$ be an mp-space and $(X, f)$ be a multi-set. Define

$$\begin{aligned} S_{\mathcal{N}}^{\text{supp}}(X, d_X, \mu_X, f) :=& (\text{supp}((X, f)), \\ & d_{X|\text{supp}((X,f))}, \\ & \frac{\mu_{X|\text{supp}((X,f))}(x) \cdot f(x)}{\sum_{x \in \text{supp}((X,f))} \mu_{X|\text{supp}((X,f))}(x) \cdot f(x)}). \end{aligned}$$

That means by definition of $S_{\mathcal{N}}^{\text{supp}}$ if for an $x \in X$ it holds that $f(x) > 1$, or in other words, if a point is sampled more than once, the mass of this point is multiplied with the times it was sampled. The following example illustrates this.

Define $S_{\mathcal{V}}^{\text{supp}}(X, d_X, \mu_X, f) := (\text{supp}((X, f)), d_{X|\text{supp}((X,f))}, \mu_{\mathcal{V}})$ with

$$c(x) := |\{x' \in X : d_X(x, x') \leq d_X(s, x') \quad \forall s \in \text{supp}((X, f))\}|$$

$$\mu_{\mathcal{V}}(x) := \frac{c(x)}{\sum_{x' \in \text{supp}((X,f))} \cdot c(x)}.$$

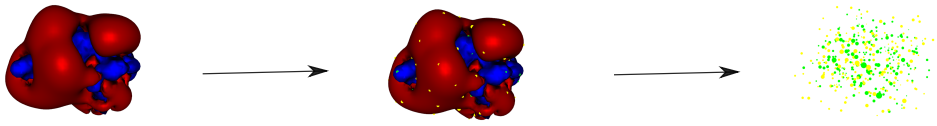

**Figure 3.1:** Only few points are selected from an *iso-surface*.

### 3.3.1 2-Step Downsampling

This is the approach that was presented in [26] and described earlier in section 2.3.1. A 3-dimensional object here is modeled as an mp-space $\mathbb{X} = (X, d_X, \mu_X)$ and an additional metric space $(X, d_{X_{geo}})$ specifying the geodesic distances of the points in the model. Now one wants to select few characteristic points that resemble the model well. Therefore in two steps points are selected from the model. In the first step with the Euclidean metric $(d_X)$ and in the second step from the now smaller model points are selected with the geodesic metric $(d_{X_{geo}})$.

Given a metric space $(M, d)$ and a $p \in M$ define the *farthest point procedure*-sequence $\mathrm{FPS}(M, d, p)$ as follows:

$$\mathrm{FPS}_1 := p$$
$$\mathrm{FPS}_i := \underset{a \in M \setminus \{\mathrm{FPS}_1, \cdots, \mathrm{FPS}_{i-1}\}}{\arg\max} \min_{b \in \{\mathrm{FPS}_1, \cdots, \mathrm{FPS}_{i-1}\}} d(a, b) \quad i \in \{1, \cdots, |M|\}.$$

## 3.4 Repeated Sub-Sampling

As mentioned earlier Felix Berens and I both worked on this topic in our master-theses. Berens used the **DE** as described in section 2 in his master-thesis [2] on the Euclidean distances. Since one can not compute the **DE** for all points in $X$ and $Y$ at once, since $(n_X, n_Y$ is too large), he performed a sampling procedure on the raw points in Euclidean space without any additional pre-processing. His idea was to sample uniformly $n$ points of both point-clouds leaving the measures $\mu_X, \mu_Y$ as the uniform distribution. Then calculate the **DE** for the two samples. This is repeated for $m$ times. The values for each sampling-step are stored in a vector. A histogram is then built from these values. This comparison is done once with $X, X$ and $X, Y$. Then the *earth mover's distance* or for short **emd** is calculated as the final

value between the two histograms. The **emd** was originally defined in [29]. For completeness the definition as in Berens Master-thesis [2] is repeated here:

**3.4 Definiton** *(earth mover's distance) [2, Def. 4.1]*
*Let $P = \{p_1, \cdots, p_k\}$ and $Q = \{q_1, \cdots, q_k\}$ be two histograms with $k$ bins of equal width. $p_i \in \mathbb{R}^{\geq 0}$ is the height(weight) of the $i$-th bin in the histogram $P$ respectively $q_j \in \mathbb{R}^{\geq 0}$ in $Q$. Also it should hold that:*

$$\sum_{i=1}^{k} p_i = \sum_{j=1}^{k} q_j =: R.$$

*The* earth mover's distance *between $P$ and $Q$ is the following linear program:*

$$\boldsymbol{emd}(P, Q) := \min \frac{\sum_{i,j=1}^{k} d_{i,j} f_{i,j}}{R},$$

*with the constraints:*

$$\begin{aligned} 0 \leq \frac{f_{i,j}}{R} \leq 1 \quad & \forall i, j \in \{1, \cdots, k\} \\ \sum_{j=1}^{k} f_{i,j} = p_i \quad & \forall i \in \{1, \cdots, k\} \\ \sum_{i=1}^{k} f_{i,j} = q_j \quad & \forall j \in \{1, \cdots, k\}. \end{aligned}$$

*$F = \{f_{i,j} | 1 \leq i, j \leq k\}$ denotes the transport from the $i$-th bin of $P$ to the $j$-th bin of $Q$. $d_{i,j} = |i - j|$ is the distance between the $i$-th and $j$-th bin.*

The approach described in the Thesis of Berens [2] can now be formalized in the following way: Given two mp-spaces

$$\begin{aligned} \mathbb{X} &= (X, d_X, \mu_X) \\ \mathbb{Y} &= (Y, d_Y, \mu_Y) \end{aligned}$$

and $n, m \in \mathbb{N}$ a distance should be calculated. Let $U_n : \mathbb{N} \to \mathbb{R}^+$

$$U_n(u) = \begin{cases} \frac{1}{n}, & \text{if } 1 \leq u \leq n \\ 0, & \text{otherwise} \end{cases}$$

Let $Z_X$ a random variable with density $U_{n_X}(u)$.
Let $s_{i,j} \sim Z_X$ i.i.d. for $j \in \{1, \cdots, n\}$ and for $i \in \{1, \cdots, m\}$.
Define $f_i(x) := \sum_{j=1}^{m} \mathbb{1}_{\{s_{i,j}\}}(x)$.
Denote with $R(\mathbb{X}, n, m) := (R_1, \cdots, R_m)$

$$R_i := (S_{\mathcal{N}}^{\text{supp}}(X, d_X, \mu_X, f_i)).$$

That means $R(\mathbb{X}, n, m)$ is a random-variable.
Let $R_X, R'_X \sim R(\mathbb{X}, n, m)$.
Denote with $Q^m(R_X, R'_X) = (Q_1, \cdots, Q_m)$

$$Q^m(R_X, R'_X)_i := \mathbf{DE}(R_{X_i}, R'_{X_i}).$$

Let $R_Y, R'_Y \sim R(\mathbb{Y}, n, m)$.
Denote

$$\begin{aligned}\mathbb{S}_{\mathbf{DE}}(\mathbb{X}, \mathbb{Y}) := \quad & \mathbf{emd}(Q^m(R_X, R'_X), Q^m(R_X, R_Y)) \\ & + \mathbf{emd}(Q^m(R_Y, R'_Y), Q^m(R_X, R_Y)).\end{aligned}$$

While in [2] the final value is originally calculated as

$$\mathbb{S}_{\mathbf{DE}}(\mathbb{X}, \mathbb{Y})' := \mathbf{emd}(Q^m(R_X, R'_X), Q^m(R_X, R_Y))$$

I argue that by adding the second term $\mathbb{S}_{\mathbf{DE}}(\mathbb{X}, \mathbb{Y})$ becomes symmetric.

With this notation the models examined in Berens' master-thesis are noted as follows:

$$\mathbb{S}_{\mathbf{DE}}(\mathbb{X}^+, \mathbb{Y}^+)'$$
$$\mathbb{S}_{\mathbf{DE}}(\mathbb{X}^-, \mathbb{Y}^-)'.$$

Additionally Berens experimented with the idea to combine the distances of the two different potentials by calculating the maximum or the average of both values. That is

$$\frac{\mathbb{S}_{\mathbf{DE}}(\mathbb{X}^+, \mathbb{Y}^+)' + \mathbb{S}_{\mathbf{DE}}(\mathbb{X}^-, \mathbb{Y}^-)'}{2}$$

and

$$\max\{\mathbb{S}_{\mathbf{DE}}(\mathbb{X}^+, \mathbb{Y}^+)', \mathbb{S}_{\mathbf{DE}}(\mathbb{X}^-, \mathbb{Y}^-)'\}.$$

Berens used these two functions to calculate all pairwise distances between all proteins. Then using these distance-matrices to build a hierarchical clustering led to very promising results.

The implementation which will be referred to as *CompareProteins* can be found at **https://github.com/ BerensF/ComparingProteins**.

## 3.5 Quick-Repeated-Sub-Sampling

In this section it will be described how one can speed up the calculation of the approach applied in *CompareProteins* [2]. There are three key-points that lead to a drastic decrease in computation-time:

- Use the approximation $\mathbf{DE}_{\mathcal{A}_q}$ instead of **DE**.
- For each protein generate each of the $m$ samples only once.
- Instead of calculating the **emd** of the distances between the distributions, calculate the geometrical center of the distributions. Then calculate the distances between the geometrical centers.

The implementation combining these three points will be referred to as *Quick-Repeated-Sampling*. The first point in the list does not require additional explanation: At the cost of introducing a small error calculate an approximation, that can be computed much faster.

The second and third point require more explanation.

### 3.5.1 Generate Only Once

In *CompareProteins* when calculating a distance between two potentials $X, Y$, for $m$ times a sample of $n$ points is drawn from both potentials and each time the **DE** is calculated. Then in an comparison with a third potential $Z$ the process is repeated. However I believe that this is not necessary. Instead what I suggest is to draw for each potential the $m$ samples only once. Then the distributions of eccentricities also have to be calculated only once and the quantiles can be stored. Then using these quantiles the **DE** can be approximated. This has two benefits: On the one hand one saves a huge amount of computation-time since the samples do not have to be reproduced whenever a new protein is added. On the other hand the comparisons are done, based on the same samples, which reduces the variance in the comparisons.

Furthermore in order to retain $m$ distances only $\sqrt{m}$ distributions have to be sampled. Calculating all pairwise distances between these distributions then again leads to $m$ distances. It is however questionable if this really leads to the same conclusions, since the $m$ distances are then no longer independently generated from each other. With no further investigation keeping the value $m$ higher seems to be more reasonable.

### 3.5.2 Geometrical Center vs. Earth-movers-Distance

In *CompareProteins* the approach is to generate $m$ distances from samples from $X$. Then additionally $m$ distances are generated between $X$ and $Y$.

These two lists of distances are used to generate two histograms. The histograms are then compared using the **emd**, leading to a single number as the final distance between $X$ and $Y$.

An alternative to the above approach is presented here. The idea is to calculate the *geometrical center* of the quantiles of $m$ samples from $X$. Additionally the *geometrical center* of the quantiles of $m$ samples from $Y$ has to be calculated. Then the final distance between $X$ and $Y$ is calculated as the Manhattan-distance between the geometrical centers.

Evidence that shows that the two approaches reveal similar information comes from the dendrograms that can be obtained. In figure 3.2 the dendrograms constructed with *CompareProteins* and *Quick-Repeated-Sampling* are displayed. The clusters in this particular example are very similar: The functional proteins in both clusters form one big cluster with the exception of the proteins $045, 046, 075$ and $097$.

### 3.5.3 Benchmarking

In table 3.1 the execution-times of *CompareProteins* and *Quick-Repeated-Sampling* is shown using different parameters. Using smaller values for $q$ leads to shorter computation-times. Larger values of $m$ lead to much longer computation-times. Using the geometrical centers instead of calculating the *earth-mover's-distance* leads to shorter computation-times. As pointed out in section 3.5.1 it might suffice to sample only $\sqrt{m}$ instead of $m$ samples in contrast to *CompareProteins*. In this case computation-time can be reduced by a factor of approximately 1000. This however would need further investigation. Since the main-focus in this thesis was to build a more complex classifier based on the approximation of the **DE**, this was not further investigated. Until then it might be safe to assume that *Quick-Repeated-Sampling* runs approximately 25 times faster than *CompareProteins*.

## 3.6 Modeling Protein-Similarity

It is expected that the geometry in close neighborhood to the active site plays the more important role, but still a similarity of the protein as a whole is necessary. That makes the modeling more difficult. On the one hand one wants a global similarity, which is needed to guide two proteins to each other in the correct conformation, but on the other hand once the correct alignment of the two proteins is achieved, the proteins have to be able to react with each other. The first part is expected to be influenced by long-range interactions which means that the general shape of the two proteins is similar. The second

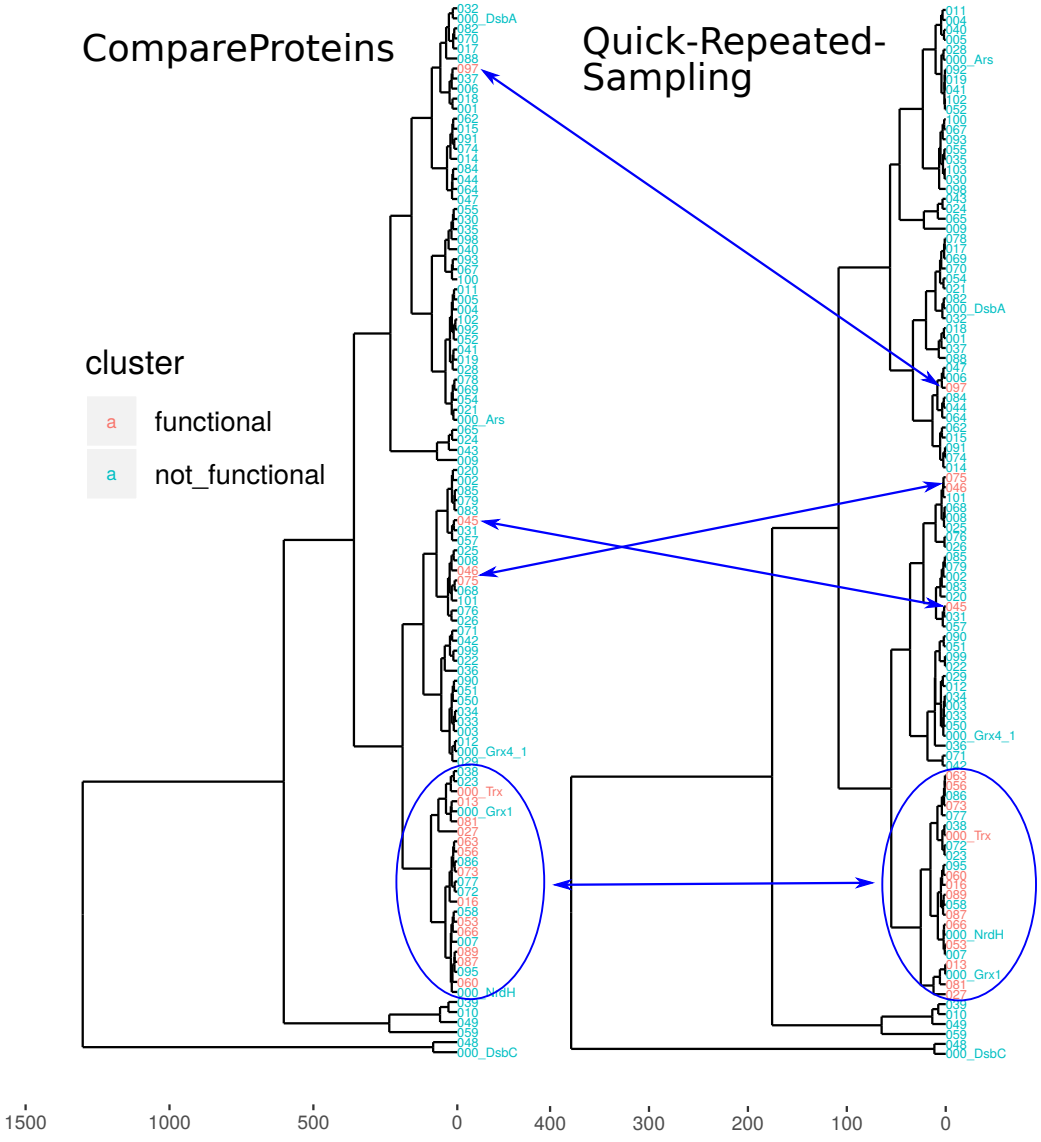


**Figure 3.2:** From left to right: Dendrograms constructed with *CompareProteins* and *Quick-Repeated-Sampling* with the positive potential. The names colored in turquoise are the proteins that can not functionally replace *Trx* in an investigated reaction. The names colored in red are the proteins that can. The blue arrows point at the similarities of the dendrograms. As parameters were set for *CompareProteins* $n = 100, m = 500$ and for *Quick-Repeated-Sampling* $n = 100, m = 500, q = 20$.

| | time in seconds | factor |
|---|---|---|
| quickRepSampling_n_100_m_22_q_1_emd | 34 | 0.06 |
| quickRepSampling_n_100_m_22_q_1_geo | 8 | 0.02 |
| quickRepSampling_n_100_m_22_q_20_emd | 36 | 0.07 |
| quickRepSampling_n_100_m_22_q_20_geo | 11 | 0.02 |
| quickRepSampling_n_100_m_500_q_1_geo | 132 | 0.25 |
| quickRepSampling_n_100_m_500_q_20_geo | 526 | 1.00 |
| CompareProteins_n_100_m_500 | 14400 | 27.38 |

**Table 3.1:** Average elapsed computation-times of the implementations *CompareProteins* and *Quick-Repeated-Sampling* in 10 repetitions using a single core. For *Quick-Repeated-Sampling* the computation-times using different parameters are displayed. For an explanation of $m, n$ refer to section 3.4. The parameter $q$ specifies the number of quantiles, *emd* or *geo* specify if the *earth mover's distance* or *geometric center* was calculated (see section 3.5.2).

part seems to be influenced by a more strict similarity in the geometry in close neighborhood to the active center.

In an attempt to model these geometrical properties of a protein three components seem reasonable to be used:

- distance to the active center
- distance to a boarder-area (that means an area where positive and negative potential meets)
- long range vs. short-range interactions.

The idea is now to detect the regions of interest by using the distance to the active center and the boarder-areas. Then a method should be applied locally to the region of interest. Additionally a method should be applied to model the similarity of the surface as a whole.

In figure 3.3 the iso-surface of a protein together with the coordinates of the atoms is shown. The atoms that form the active center are colored in yellow.

### 3.6.1 The Active Center

The active site is of particular interest in many protein-protein-reactions. In order to model this, the measure of the points can be changed in such a way that point that are in close proximity to the active center get a higher weight.

**Figure 3.3:** The reduced model with the atom-coordinates and the active center. The black points represent the atom-coordinates, the yellow points represent the active center. The blue and red points are the two potentials.

Let $\mathcal{A} \in \mathbb{R}^3$ be the atom-coordinates of the active center. Define:

$$\mu_{\mathcal{A}}(x) := \frac{\frac{1}{\min\limits_{a \in \mathcal{A}} d_X(x,a)}}{\mathcal{N}_a}$$

where $\mathcal{N}_a$ is the normalizing constant assuring that $\mu_{\mathcal{A}}$ is a probability measure.

### 3.6.2 The Boarder-Areas Between the Potentials

When taking two iso-surfaces of two potentials one would expect them to be disconnected and not to touch in any region. Otherwise this would imply a contradiction, since by definition each point in space can only have one specific potential-value. However when using the computed surface from VMD, one clearly sees regions where the different potentials come very close to each other. Prof. Lillig suggested that these regions may be of particular interest for predicting the protein-similarity. He believes that the occurrence of a biochemical reaction between two proteins is mainly influenced by the shape of these regions.

In order to model this knowledge the measure can be changed in such a way that points in close proximity to these important *boarder*-regions have a higher measure. Let a protein be given, represented by a set of points in $\mathbb{R}^3$ and a metric $d_{geo}$, noting the pairwise distances of all points on the surface of the protein. Additionally a map $pot : \mathbb{X} \to \{1, -1\}$ is given specifying for each point if it belongs to the positive or negative potential. Given a number $b$ one can now compute the measure for a point $p$ as follows:

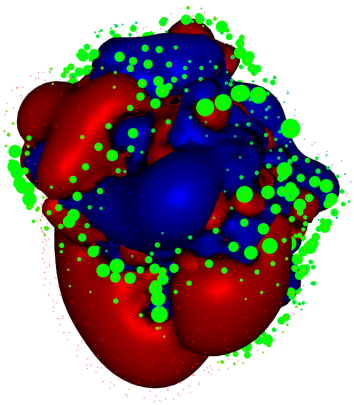

**Figure 3.4:** The boarder between positive and negative potential marked by the green points.

Let $kNN(p) \subset X$ be the $b$ points that are closest to $p$, that means $d_{geo}(p,x) \leq d_{geo}(p,x') \quad \forall x \in kNN(p) \quad \forall x' \in \mathbb{X}$. Then define:

$$\mu_{\mathcal{B},b}(p) := \frac{|\{x \in kNN(p) : pot(x) \neq pot(p)\}|}{\mathcal{N}_{\mathcal{B}}}$$

where $\mathcal{N}_{\mathcal{B}}$ is the normalizing constant that assures that $\mu$ is a probability measure. That means that points that have a lot of points in close neighborhood from the other potential get a higher weight. In figure 3.4 the iso-surfaces of a protein are displayed together with the measure of each point. The size of the green points indicate the value of the measure, where larger points indicate a high measure.

### 3.6.3 Combining Active Center and Boarder Area

In order to combine these two properties one can calculate the measure as follows:

$$\mu_{X\,\alpha,\beta,b}(x) := \frac{\mu_{\mathcal{A}}(x)^{\alpha} \cdot \mu_{\mathcal{B},b}(x)^{\beta}}{\mathcal{N}} \tag{3.3}$$

where $\mathcal{N}$ is the normalizing constant assuring that $\mu_X$ is a probability measure.

Then using as parameters $\alpha = 0$ and $\beta = 0$ one has a uniform distribution. Higher values for $\alpha$ model that the active center is very important, and higher values for $\beta$ model that the boarder areas are very important. In figure 3.5 a protein with the measure for different values of $\alpha$ and $\beta$ is shown.

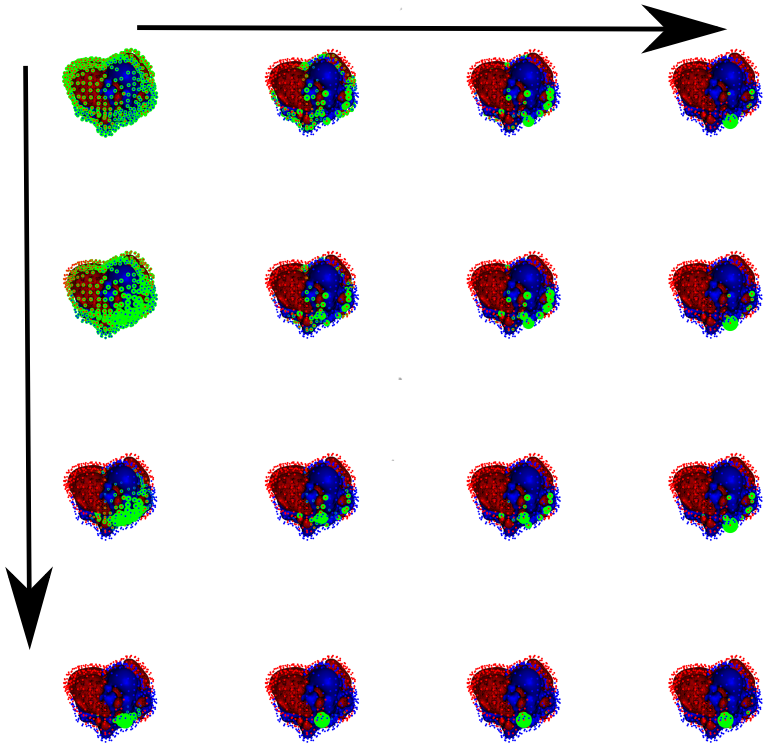

**Figure 3.5:** The effects on the measure with varying values for $\alpha$ and $\beta$. From left to right $\beta$ increases, from top to bottom $\alpha$ increases. The size of the green points indicate how much mass is put into the point. For an explanation of the parameters $\alpha$ and $\beta$ refer to equation (3.3).

### 3.6.4 Long-Range vs. Short-Range-Interactions

In order to model long-range vs. short-range-interactions the idea is the following: Given a number $k \in [0, 1]$ take for each point $x \in X$ the $k \cdot |X|$ closest neighbors with regard to $d_{X_{\text{geo}}}$. Doing this results in $|X|$ subsets for each of which different features can be calculated. In figure 3.6 different values of $k$ are displayed on four exemplary points. In each row $k$ increases from right to left.

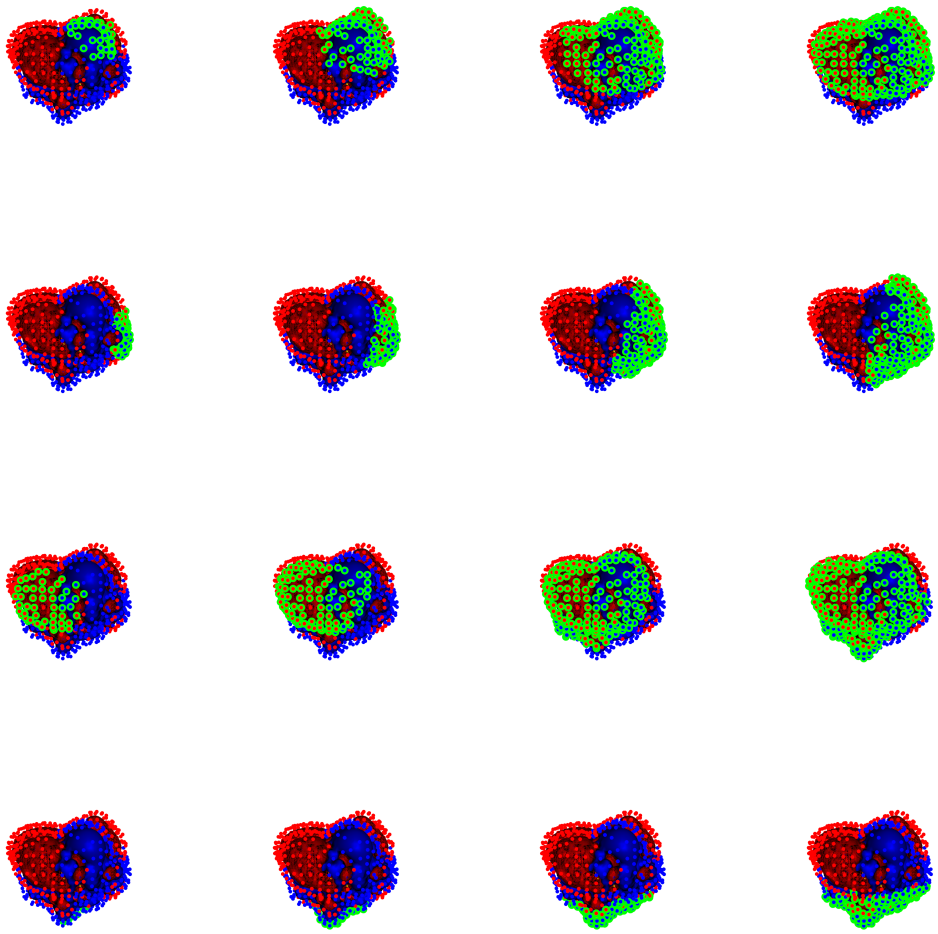

**Figure 3.6:** Example of four different points selected in each row. The nearest neighbors of the points are marked in green. From left to right the number of nearest neighbors increases.

## 3.7 Putting It All Together

In the previous sections approaches to calculate similarities between 3D-models have been introduced. However these approaches are always producing only a single number and only grasp the relation between two objects in a very specific way. In the case of the **DE** that is the difference of the integral of two cumulative distribution functions. The idea is here to instead of calculat-

| $q_1^+$ | $q_2^+$ | $q_3^+$ | $q_1^-$ | $q_2^-$ | $q_3^-$ | $q_1$ | $q_2$ | $q_3$ | $q^+_{\mathbf{geo}_1}$ | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.26 | 0.3 | 0.4 | 0.29 | 0.39 | 0.51 | 0.55 | 0.71 | 0.94 | 0.23 | 0.26 |
| 0.18 | 0.22 | 0.38 | 0.32 | 0.41 | 0.58 | 0.52 | 0.7 | 1.03 | 0.16 | 0.19 |
| 0.52 | 0.62 | 0.83 | 0.5 | 0.64 | 0.77 | 1.02 | 1.27 | 1.64 | 0.46 | 0.54 |
| 0.43 | 0.56 | 0.75 | 0.58 | 0.81 | 1.02 | 1.01 | 1.4 | 1.88 | 0.4 | 0.51 |
| 0.68 | 0.88 | 1.19 | 0.53 | 0.69 | 0.94 | 1.22 | 1.61 | 2.09 | 0.61 | 0.79 |

**Table 3.2:** Exemplary matrix of the quantiles of the cumulative distribution of the eccentricitie-function as features. Each row corresponds to one specific point of the iso-surface. The columns are the features of each point. For more details see section 3.7.

ing the **DE** directly, using the approximations of the cumulative distribution functions of the objects as features. Then using a machine-learning-method that takes as input these features, it seems natural that more complex relations can be learned. The complexity can then be increased even further by combining multiple of such features. In what follows it will be described how such a feature can be obtained from a protein $\mathcal{X} = (X, d_X, d_{X_{\text{geo}}}, \mu_X, \text{pot})$ to which the 2-step-downsampling-procedure with 2000 points in the first step and 1000 points in the second step was applied.

Given parameters $k \in [0,1], q, b \in \mathbb{N}, \alpha, \beta \in \mathbb{R}$ the features are computed as follows. For each point in $X$ the $k \cdot |X|$ nearest neighbors with regard to $d_{X\,\text{geo}}$ are drawn. Then the measure for each of these subsets is adjusted according to equation (3.3) using $\alpha, \beta$ and $b$. For each of these subsets then the mp-spaces $\mathbb{X}^+, \mathbb{X}^-, \mathbb{X}, \mathbb{X}^+_{\text{geo}}, \mathbb{X}^-_{\text{geo}}$ and $\mathbb{X}_{\text{geo}}$ are obtained. Then the approximations of the cumulative distributions of the eccentricities are calculated: $\mathcal{A}(S_{\mathbb{X}^+}, q), \mathcal{A}(S_{\mathbb{X}^-}, q)$, $\mathcal{A}(S_{\mathbb{X}}, q)$, $\mathcal{A}(S_{\mathbb{X}^+_{\text{geo}}}, q)$, $\mathcal{A}(S_{\mathbb{X}^-_{\text{geo}}}, q)$ and $\mathcal{A}(S_{\mathbb{X}_{\text{geo}}}, q)$. For each approximation of a cumulative distribution that makes $q+2$ numbers. Therefore for each point in $X$ that makes $6 \cdot (q+2)$ numbers. Since the model was reduced to 1000 points that makes $1000 \cdot 6 \cdot (q+2)$ numbers for each protein. Denote these numbers in a matrix $\mathcal{F}$, in such a way that in each row the features for one specific point are noted. See table 3.2 for an example. In what follows denote by $\mathcal{F}(\mathcal{X}, k, q, b, \alpha, \beta)$ the feature-matrix derived with the specified parameters.

Multiple of such feature-matrices using different parameters can now be combined by keeping the same order of the points in the rows and concatenating the columns of the matrices. The idea is now to use these concatenated feature-matrices as input to build a classifier using machine learning.

# Chapter 4

# Predicting Protein-Interactions

In this chapter it will be discussed how a neural-net was built for classification of the proteins when using the features as described in 3.7. An example of multiple simulated distributions is investigated to motivate the approach. First some basics on machine-learning and neural-nets are recalled.

## 4.1 A Binary-Classification Problem

A *binary-classification-problem* can be formulated as follows: Given is a number $n \in \mathbb{N}$ and a random-variable $Z : \Omega \mapsto \mathbb{R}^n$ and a function $l : \mathbb{R}^n \to \{0, 1\}$. Then a sample of size $m \in \mathbb{N}$ is drawn from $Z$ forming a matrix $X \in \mathbb{R}^{m \times n}$ with $X_i \sim Z$. Denote $y \in \mathbb{R}^m$ with $y_i := l(X_i)$. A function $f : \mathbb{R}^n \to \{0, 1\}$ is then called a *binary-classifier*. Calculating $f(x)$, for an $x \in X$ is called *predicting*. The task is now to find a function $f$, that given a new sample $X' \in \mathbb{R}^{m' \times n}$ with $X'_i \sim Z$ predicts the correct labels $l(X'_i)$. In order to try to find such a function $f$ one is able to use $X$ and $y$. There exist many different techniques and algorithms that try to find such a function $f$. The field of *machine-learning* investigates these techniques extensively. The usual case is however that not all labels for a new sample $X'$ can be predicted correctly. Therefore different measures for evaluating the performance of a classifier $f$ have been invented. Define:

$$\mathbf{TP} := |\{i \in X' : f(X'_i) = y_i \quad \text{and} \quad y_i = 1\}|$$

$$\mathbf{TN} := |\{i \in X' : f(X'_i) = y_i \quad \text{and} \quad y_i = 0\}|$$

$$\mathbf{FP} := |\{i \in X' : f(X'_i) \neq y_i \quad \text{and} \quad y_i = 0\}|$$

$$\mathbf{FN} := |\{i \in X' : f(X'_i) \neq y_i \quad \text{and} \quad y_i = 1\}|,$$

where **TP**, **TN**, **FP**, **FN** stands for *True-positive*, *True-negative*, *False-positive*, *False-negative* respectively.

$$\mathbf{ACC} := \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{TN} + \mathbf{FP} + \mathbf{FN}}.$$

$$\mathbf{PPV} := \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}}.$$

$$\mathbf{TPR} := \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}}.$$

$$F_1 := 2 \cdot \frac{\mathbf{PPV} \cdot \mathbf{TPR}}{\mathbf{PPV} + \mathbf{TPR}},$$

where **ACC**, **PPV**, **TPR**, $F_1$ stands for *accuracy*, *positive-predicted-value*, *true-positive-rate*, $F_1$*-score*, respectively. The $F_1$-score is a more reliable measure of accuracy, than **ACC** when the classes are imbalanced. In tables 4.1 and 4.2 an example is illustrated with an unbalanced class distribution. $\mathbf{ACC} = \frac{30}{300} = 0.9$ even though not a single instance of the labels with $y_i = 1$ is correctly predicted. A naive classifier with $\hat{y}_i = 0 \quad \text{for all} \quad i$ leads to this accuracy. In table 4.2 the accuracy calculates as

$$\mathbf{ACC} = \frac{30 + 30 + 0 + 0}{30 + 30 + 240} = 0.9.$$

The $F_1$-scores of both examples calculate as

$$\begin{aligned} F_1 =& 0 \\ & \text{and} \\ F_1 =& 2 \cdot \frac{\frac{30}{30+30} \cdot \frac{30}{30}}{\frac{30}{30+30} + \frac{30}{30}} = 2 \cdot \frac{\frac{1}{2} \cdot 1}{\frac{1}{2} + 1} = \frac{2}{3}. \end{aligned}$$

That means in this case maximizing the $F_1$-score would lead to preferring the predictions that produced the confusion matrix in table 4.2.

**Table 4.1:** Example of a confusion matrix with low $F_1$-score.

| | $y_i = 1$ | $y_i = 0$ |
|---|---|---|
| $\hat{y}_i = 1$ | 0 | 0 |
| $\hat{y}_i = 0$ | 30 | 270 |

**Table 4.2:** Example of a confusion matrix with high $F_1$-score.

| | $y_i = 1$ | $y_i = 0$ |
|---|---|---|
| $\hat{y}_i = 1$ | 30 | 30 |
| $\hat{y}_i = 0$ | 0 | 240 |

## 4.2 Cross-Validation

Any machine-learning-method can be seen as a sophisticated method to make predictions about samples from a distribution. One is given only one random-sample of the distribution and based on that sample the task is to make predictions for further samples from that distribution. Therefore finding a model that fits well to only one very specific sample is usually not wanted. Instead a model that fits well to any sample is wanted. In order to measure the performance of a model on yet unseen data, taking all the data in a sample to train the model does not work, since no data is then left to test the model. Instead what is done, is a split into a *training-* and a *test*-set. The parameters of the model are then tuned in such a way that the model fits well to the data in the trainings-set. Then the ability of the model to generalize to new independent data is tested on the test-set. This procedure works well if enough data is available. However if the available data is limited and enough computational power is available a more preferable procedure is *k-fold-cross-validation*. With k-fold-cross-validation the data is split into $k$ (typically 10) instead of only 2 partitions. Each partition is called a fold. Then for each fold a model is trained on the remaining $k - 1$ folds. The performance is then measured on the fold that the model was not trained on. Then the performance is averaged over all models. The extreme case where $k = n$ where $n$ describes the size of the sample, is called *Leave-one-out-cross-validation*.

## 4.3 Neural-Net

In case of a binary-classification-problem a function $f : \mathbb{R}^n \to \{0, 1\}$ has to be found. One way of realizing this is an *artificial-neural-net*(ANN). A brief and very compact description of an ANN will be given here. For a more detailed description refer to [22]. An ANN consists of multiple layers $(l_1, \cdots, l_L)$. Each layer consists of multiple nodes. The first layer is called *Input layer*. The Input layer has $\mathbb{R}^n$ nodes. The nodes of consecutive layers

$l_i, l_{i+1}$ are *connected* with weights $w \in \mathbb{R}$, that means given some values in layer $l_i$ the values in layer $l_{i+1}$ are calculated by matrix-multiplication. After the matrix-multiplication, in resemblance to the biological neuron an activation-function is applied. The key-point with these activation-functions is that they are non-linear. For example

$$\text{relu}(t) := \begin{cases} 0 & , t \leq 0 \\ t & , t > 0 \end{cases}.$$

The question now is how the weights in the matrices between the layers have to be chosen in order to lean towards more correct outputs. This is done by calculating a loss between the prediction of the ANN and the actual label. With this a gradient with respect to the weights can be calculated. Then using *gradient-descend* the weights can be optimized. One way to make the calculation of the gradient for the weights more efficient, is called *back-propagation*. Using the chain-rule going from layer to layer backwards, beginning with the last layer, for each weight a gradient is calculated. Adding the gradients to the weights then decreases the loss. This process of feeding in new data and updating the weights in the matrices is called *learning* or *training*. Passing the whole data-set through the ANN is called an *epoch*. After a fixed number of epochs the training is ended. The weights stored in the matrices at this point then form the final model and are not altered any further. The model can then be applied to new data to make predictions.

## 4.4 Augmentation

In practice ANNs having more data at hand increases the chances of a successful training. When data is limited a common technique is to artificially generate new data. For a set of images that is for example rotating and translating the images. In the case of the proteins, reusing the notation as described in 3.7 the idea is the following: Each protein was reduced to 1000 points. For each point multiple features were computed. To increase the amount of data the following is done: For each $x \in X$ draw $m$ rows from $\mathcal{F}$ at random. This can be repeated multiple times. This form of augmentation basically means that parts of the data are dropped for each sample. Then the $m$ rows $\mathcal{F}_1, \cdots, \mathcal{F}_m$ are concatenated into a single vector. Since the order in which the rows are drawn should not have a different outcome, the vector is ordered in the following way: For each $\mathcal{F}_i$ the sum $s_i := \sum_j \mathcal{F}_{i,j}$ of the row is calculated. Then the $\mathcal{F}_i$ are put in the vector in decreasing order with regard to $s_i$. This vector then forms the input for the neural-net.

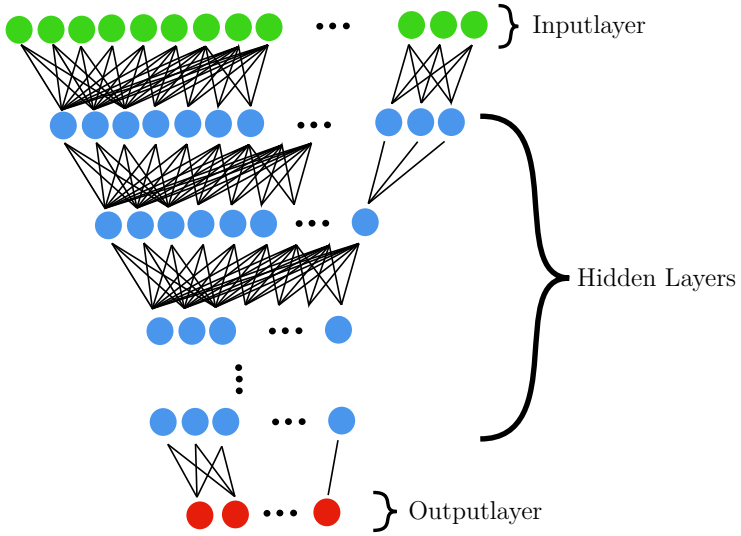


**Figure 4.1:** Schematic description of the architecture of a *feed-forward-net*. The green nodes form the inputs. The blue nodes are the *hidden nodes* and the red nodes are the outputs.

## 4.5 Auto-Encoder

An *auto-encoder* is a special type of neural-net that can be used to learn efficient representations of data in a unsupervised manner. The aim is to reduce the dimensions of the data. The architecture of an auto-encoder can have multiple layers ending with the so called *bottle-neck-layer*. This layer, as the name suggests, forms a bottle-neck in that sense that the number of nodes is usually much smaller than the number of nodes in the other layers. The goal is to find for each input a representation that can be extracted from the bottle-neck-layer. During training the net-architecture is changed in such a way that in regard to the bottleneck-layer the number of nodes for each layer is symmetrical. That means an Input layer with $n$ dimensions then also leads to an output with $n$ dimensions. The part of the net that leads from an input to the bottle-neck is called *encoder*. The part that comes after the bottleneck is called *decoder*. The idea is that from the input to the bottleneck the information is compressed that means *encoded*. Then in the second half of the net the information is decompressed or *decoded*. The training is now performed in such a way that the capability of the auto-encoder to encode and decode is optimized. This is done by comparing how similar the inputs and the encoded and decoded outputs are. The problem that sometimes occurs with deep neural nets is the *vanishing-gradient-problem*. That describes the

fact that, because of numerical errors during computation the gradient in the first few layers is not adjusted properly. To overcome this an alternative approach is to *pre-train* the first few layers with an auto-encoder. Then a deep neural net is built using the weights from the first few layers and adding more layers on top. In [31] an *auto-encoder* was first used in order to *pre-train* a *feed-forward-neural-network*.

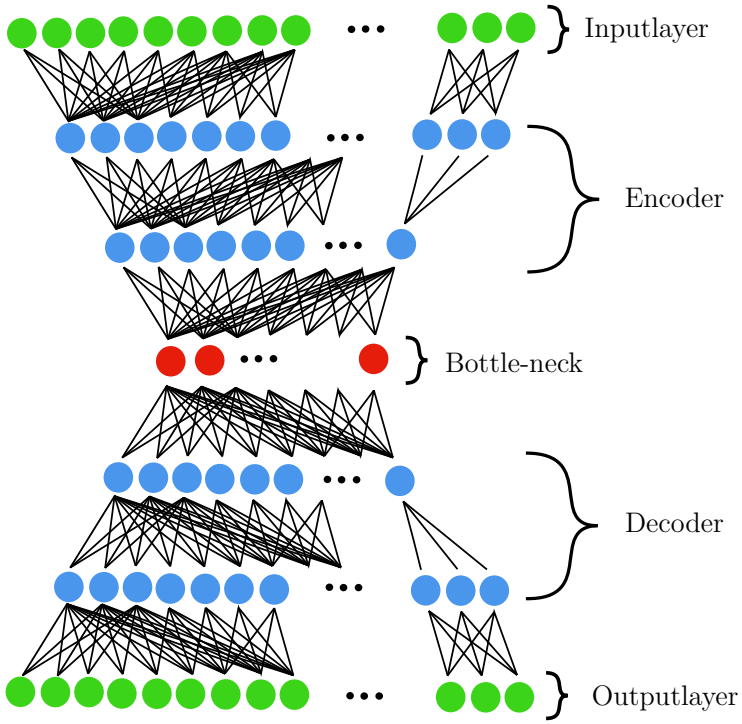


**Figure 4.2:** Architecture of a multi-layer auto-encoder. For more details refer to section 4.5.

## 4.6 A Neural Net for Predicting Protein Interactions

The idea is to combine multiple of such features as described in section 3.7 in one matrix $\mathcal{F}$. Then, as described in section 4.4 perform augmentation. That means sample $m$ times $u$ rows from $\mathcal{F}$. Each of these samples consisting of $u$ rows is then used as one train-/test-instance. As described in the section *Augmentation* the $u$ rows are concatenated into a single vector. Such a vector and the corresponding label together then form one training-pair. In a *pre-*

*training-step* an auto-encoder is used to learn a condensed representation of the data. After this step the layers from the Input layer to the bottle-neck-layer of the auto-encoder are extracted. On top of these layers the neural-net is then trained. All in all the process can be summarized as follows:

- Calculate the features $\mathcal{F}_1, \mathcal{F}_2, \cdots$.
- Perform data Augmentation. That means sample $u$ rows for $m$ times and concatenate the rows into a single vector. One can think of this step as leaving out $1000 - u$ points from the model.
- Create $p \in \mathbb{N}$ permutation dependencies. That means create new matrices $X_p, X'_p$ from $X$ by sampling for each protein $p$ rows for $X_p$ and $p$ rows for $X'_p$.
- Train an auto-encoder with $X_p, X'_p$. The idea behind this step is to learn the relation between the different augmented samples from each protein. Different samples from the same protein should be recognized as coming from the same object.
- Extract the layers from input to the bottleneck from the auto-encoder. Freeze the weights. Build an ANN on top of the frozen model.
- Train the neural-net with $X'$ and $y$.

This process is depicted in figure 4.3. In order to evaluate this approach on the protein-data-sets 10-fold cross-validation was performed. That means the whole process of generating the auto-encoder and building an ANN on top is repeated for each fold.

### 4.6.1 Clustering

*Clustering* describes the process of finding partitions of a set in such a way that the elements in each set have high similarity to elements within the same set and high dissimilarity to elements from other sets.

The auto-encoder produces a condensed representation of the inputs using the encoder-layers of the net. These encoded inputs can be used for clustering. A hierarchical clustering-algorithm has been used to cluster the proteins as follows. For each protein the representation $X'$ has been calculated. Then applying the auto-encoder the condensed representation is obtained. Then the geometric center of these condensed representations was calculated. Then the R-packages *cluster* [24] and *ggdendro* [7] were used to perform a hierarchical clustering.

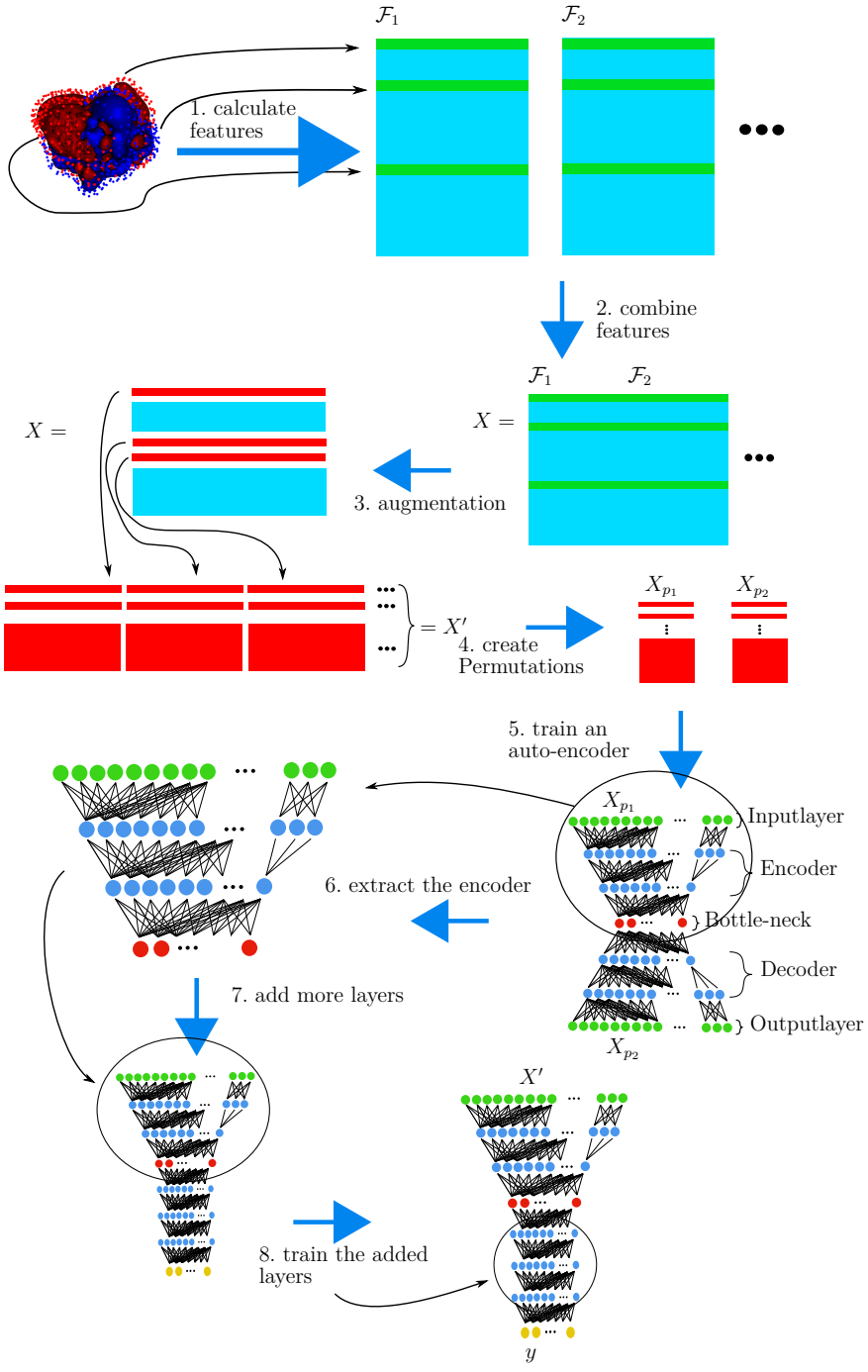


**Figure 4.3:** Overview of how the neural-net is generated. For more details refer to 4.6

# Chapter 5

# Results

The confusion matrices in this chapter are ordered in the following way: The columns contain the true labels and the rows contain the predictions. In section 5.1 the results of applying the implementation on the data-sets *animal-test-set* [32] and *modelNet10* [35] 3D-models are presented. In section 5.2 the results of applying the implementation to three protein-data-sets are presented.

## 5.1 Classifiying 3D-Models

In order to test the method that was proposed in section 4.6 experiments with two data-sets of 3D-models were conducted. The goal of these experiments was to show that the method can be used for 3D-object-recognition, independent of any biologically motivated modeling. For this purpose the method from section 4.6 was slightly altered and will be described here: With the Euclidean farthest-point-procedure $n_{\text{euclidean}} = 4000$ points were selected. Then using the geodesic farthest-point-procedure $n_{\text{dijkstra}} = 100$ points were selected. Then the measure was calculated using the Voronoi-partitions.

After these steps, which are the same as in the reference done by Memoli [26] a similar procedure as described in section 4.6 was applied. For each model 96 of the 100 points were drawn randomly for 100 times, resulting in 100 subsets with 96 points each. Then for each of these subsets the approximation of the distribution of eccentricities (see equation (2.24)) with $q = 1$ was calculated leading to 100 quantile-vectors, each with a size of 3. Then for 100 times 10 of such quantile-vectors were randomly chosen and concatenated leading to a total of 100 different representations for each model. Each of these representations then contained $3 \cdot 10$ values. These representations together with the corresponding class-labels were then used to train a

neural-net.

Unlike as in section 4.6 no pre-training with an auto-encoder was done for these data-sets.

### 5.1.1 The *animal-test-set*

The neural-net for this data-set was a fully-connected 3-layer neural-network with sizes $(300, 10, 10)$ and *relu* as activation-function between the layers. The model was trained with a batch-size of 10 and for 30 epochs. 10-fold cross-validation was performed. Averaging the accuracy over all ten folds lead to an accuracy of 0.99. In table 5.1 the confusion matrix is shown. In figure 5.1 the approximations of the different animals are shown (excluded are the heads, because they are much larger than the other models, and hence easy to classify anyways). Since $q = 1$ each quantile-vector is a 3-dimensional vector. The image was taken as a snapshot after manual adjustment of the view-angle. As one can see the classes *horse* and *camel* have a slight overlap in some points. This also reflects in the predictions. The classes *horse* and *camel* have few false predictions. The same is true for *camel* and *flam*. It is remarkable in that regard that the achieved prediction-accuracy using this method exceeds the prediction-accuracy using the **FLB** in combination with a 1-nearest-neighbor-method as mentioned in [26], which was 0.86.

| | camel | cat | elephant | flam | head | horse | lion |
|---|---|---|---|---|---|---|---|
| camel | **1062** | 0 | 0 | 0 | 0 | 66 | 0 |
| cat | 0 | **1000** | 0 | 0 | 0 | 0 | 1 |
| elephant | 0 | 0 | **1100** | 0 | 0 | 0 | 0 |
| flam | 38 | 0 | 0 | **1100** | 0 | 0 | 0 |
| head | 0 | 0 | 0 | 0 | **1000** | 0 | 0 |
| horse | 0 | 0 | 0 | 0 | 0 | **934** | 0 |
| lion | 0 | 0 | 0 | 0 | 0 | 0 | **999** |

**Table 5.1:** Confusion matrix from the *animal-test-set*. Each class contained between ten and eleven models and for each model 100 representations were created. Hence the large numbers in each class.

### 5.1.2 The ModelNet10-data-Set

The results on the animal-test-set were promising. There is however one bias in the *animal-test-set*. Each model of the same class is in fact derived from the same 3D-model, just in different poses.

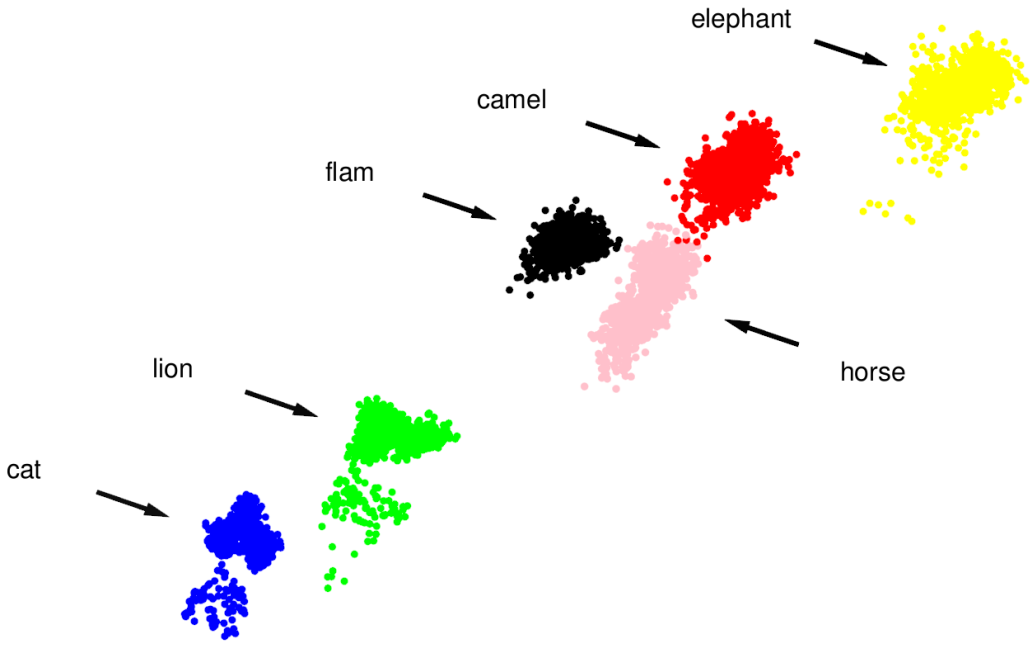


**Figure 5.1:** Displayed are the quantiles of the models from the *animal-test-set*. From each model $n = 96$ points were randomly drawn and then the quantiles with $q = 1$ of the distribution of eccentricities were calculated. Each point in the image corresponds to one such distribution. For each model this was done $m = 100$ times. For more details refer to section 5.1.1.

A probably more challenging test-set is the *modelNet10-dataset*. This data-set contains 40000 objects of 10 different classes of furniture as 3D-models and serves as an open source benchmark-test set for 3D-object-classification-algorithms. The models are completely unique and no two objects are derived from the same 3D-model. As a first step the data-set needed some pre-processing. The meshes were not connected and hence a geodesic distance was not given. The tool *Manifold* from [16] provides an option to make single connected meshes out of the meshes. Each model was preprocessed with the program *manifold*, yielding also a geodesic distance.

Then the method as described in 5.1 was applied to retain for each model 100 representations each with $3 \cdot 10$ values.

The *modelNet10-dataset* has a predefined split into a training- and a test-set. A neural-net was trained on the training-set and evaluated on the test-set. The achieved prediction accuracy is 0.67 on the test-set. The confusion matrix is shown in table 5.2 and 5.3.

| | bathtub | bed | chair | desk | dresser | monitor | night | sofa | table | toilet |
|---|---|---|---|---|---|---|---|---|---|---|
| bathtub | **30** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| bed | 3 | **75** | 0 | 17 | 3 | 12 | 0 | 9 | 3 | 4 |
| chair | 0 | 2 | **72** | 1 | 0 | 5 | 5 | 0 | 4 | 4 |
| desk | 0 | 3 | 0 | **3** | 0 | 0 | 0 | 0 | 1 | 0 |
| dresser | 1 | 0 | 3 | 5 | **60** | 1 | 5 | 2 | 0 | 2 |
| monitor | 3 | 7 | 7 | 5 | 4 | **61** | 1 | 5 | 2 | 3 |
| night | 0 | 0 | 4 | 0 | 13 | 2 | **59** | 0 | 0 | 4 |
| sofa | 11 | 7 | 2 | 11 | 3 | 3 | 1 | **82** | 0 | 6 |
| table | 0 | 4 | 5 | 42 | 3 | 2 | 4 | 1 | **90** | 0 |
| toilet | 1 | 1 | 7 | 1 | 0 | 12 | 11 | 0 | 0 | **77** |

**Table 5.2:** Confusion-matrix of the *ModelNet10-data-set*. This result was achieved by applying the approach described in 5.1.

| | bathtub | bed | chair | desk | dresser | monitor | night | sofa | table | toilet |
|---|---|---|---|---|---|---|---|---|---|---|
| bathtub | **0.61** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 |
| bed | 0.06 | **0.76** | 0.00 | 0.20 | 0.03 | 0.12 | 0.00 | 0.09 | 0.03 | 0.04 |
| chair | 0.00 | 0.02 | **0.72** | 0.01 | 0.00 | 0.05 | 0.06 | 0.00 | 0.04 | 0.04 |
| desk | 0.00 | 0.03 | 0.00 | **0.04** | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| dresser | 0.02 | 0.00 | 0.03 | 0.06 | **0.70** | 0.01 | 0.06 | 0.02 | 0.00 | 0.02 |
| monitor | 0.06 | 0.07 | 0.07 | 0.06 | 0.05 | **0.62** | 0.01 | 0.05 | 0.02 | 0.03 |
| night | 0.00 | 0.00 | 0.04 | 0.00 | 0.15 | 0.02 | **0.69** | 0.00 | 0.00 | 0.04 |
| sofa | 0.22 | 0.07 | 0.02 | 0.13 | 0.03 | 0.03 | 0.01 | **0.82** | 0.00 | 0.06 |
| table | 0.00 | 0.04 | 0.05 | 0.49 | 0.03 | 0.02 | 0.05 | 0.01 | **0.90** | 0.00 |
| toilet | 0.02 | 0.01 | 0.07 | 0.01 | 0.00 | 0.12 | 0.13 | 0.00 | 0.00 | **0.77** |

**Table 5.3:** Confusion-matrix of the *ModelNet10-data-set* (normalized). This result was achieved by applying the approach described in 5.1. The values in the matrix are the values from table 5.2 divided by the column-sums.

The achieved prediction accuracy can not compete with any of the top-listed prediction-models at *https://modelnet.cs.princeton.edu/*, ranging from 0.77 to 0.98 prediction-accuracy. It shows however that in principle this method is also applicable to other use-cases and more specifically the biological questions of this thesis.

## 5.2 Predicting Protein Interactions

The data-set *106-Redoxins* was used to tune reasonable parameters for the features and the neural-net. A mixture of manual tests, random-search and

grid-search were conducted testing different parameter-combinations with 10-fold stratified-cross-validation.

The following parameters achieved reasonable results, maximizing the F1-score:

$$\mathcal{F}(\mathcal{X}, k = 0.1, q = 1, b = 10, \alpha = 3, \beta = 3)$$
$$\mathcal{F}(\mathcal{X}, k = 0.2, q = 1, b = 10, \alpha = 3, \beta = 3)$$
$$\mathcal{F}(\mathcal{X}, k = 0.3, q = 1, b = 10, \alpha = 3, \beta = 3)$$
$$\mathcal{F}(\mathcal{X}, k = 0.5, q = 1, b = 10, \alpha = 3, \beta = 3)$$
$$\mathcal{F}(\mathcal{X}, k = 0.8, q = 1, b = 10, \alpha = 3, \beta = 3).$$

This result is achieved when the weights are adjusted in such a way that each class gets a weight equal to the inverse of its occurrence in the train-set. These weights can however also be used to specify the importance of a FN vs. a FP.

### 5.2.1 106 Redoxins

Using the parameters that maximize the F1-score one obtains an F1-score of 0.55 and an accuracy of 0.82. The confusion matrix is shown in 5.4. Tuning the weights for the two classes in order to maximize either the $TPR$ or $TNR$ one obtains the two confusion-matrices shown in table 5.5 respectively table 5.6.

| | functional | not_functional |
|---|---|---|
| functional | 0.62 | 0.14 |
| not_functional | 0.38 | 0.86 |

**Table 5.4:** Confusion-matrix from the 106-Redoxins-test-set maximizing the $F1$-Score.

| | functional | not_functional |
|---|---|---|
| functional | 0.88 | 0.51 |
| not_functional | 0.12 | 0.49 |

**Table 5.5:** Confusion-matrix from the 106-Redoxins-test-set with higher weights for the true-positives.

The clustering obtained with *Quick-Repeated-Sampling* is shown in figure 5.2.

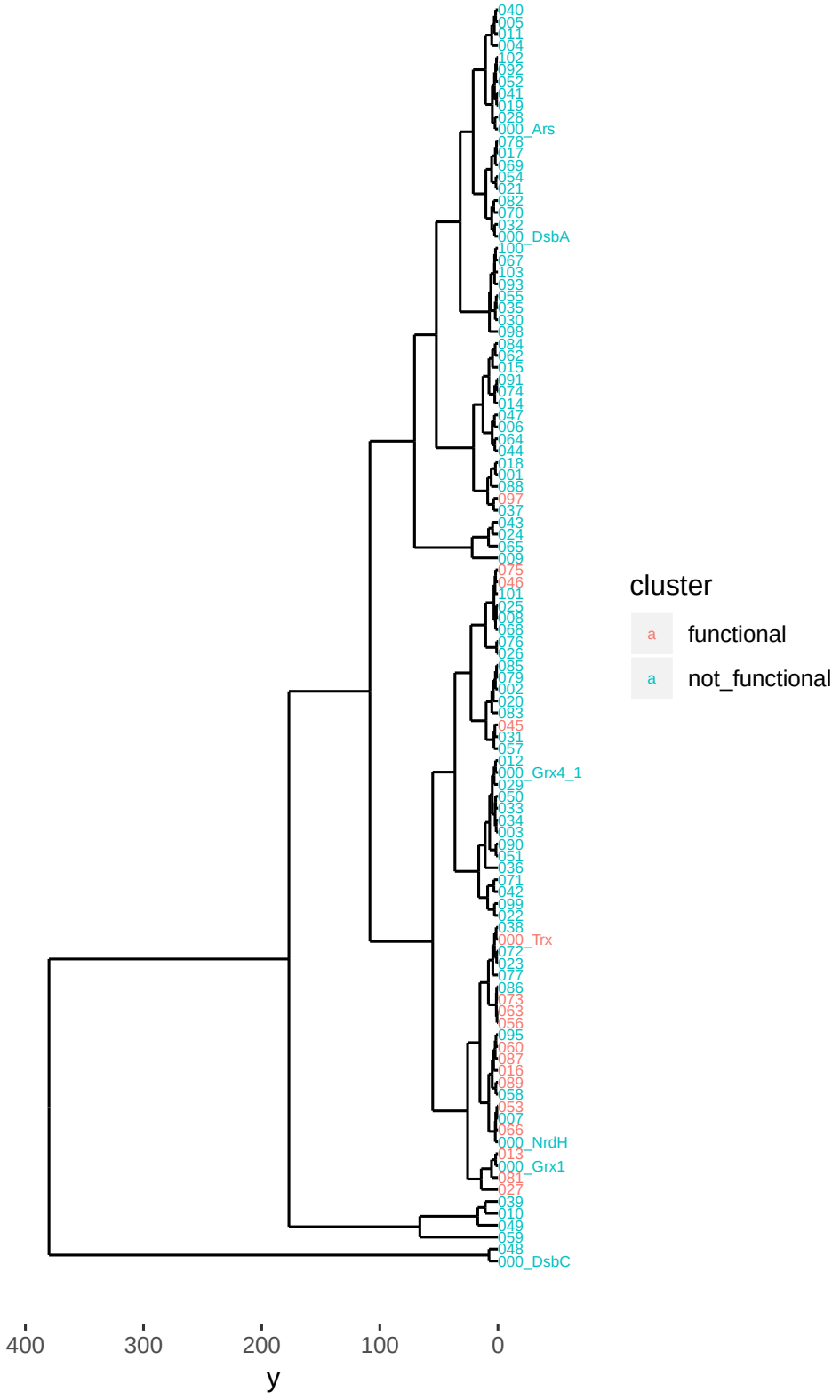


**Figure 5.2:** Clustering performed with *Quick-Repeated-Sampling*.

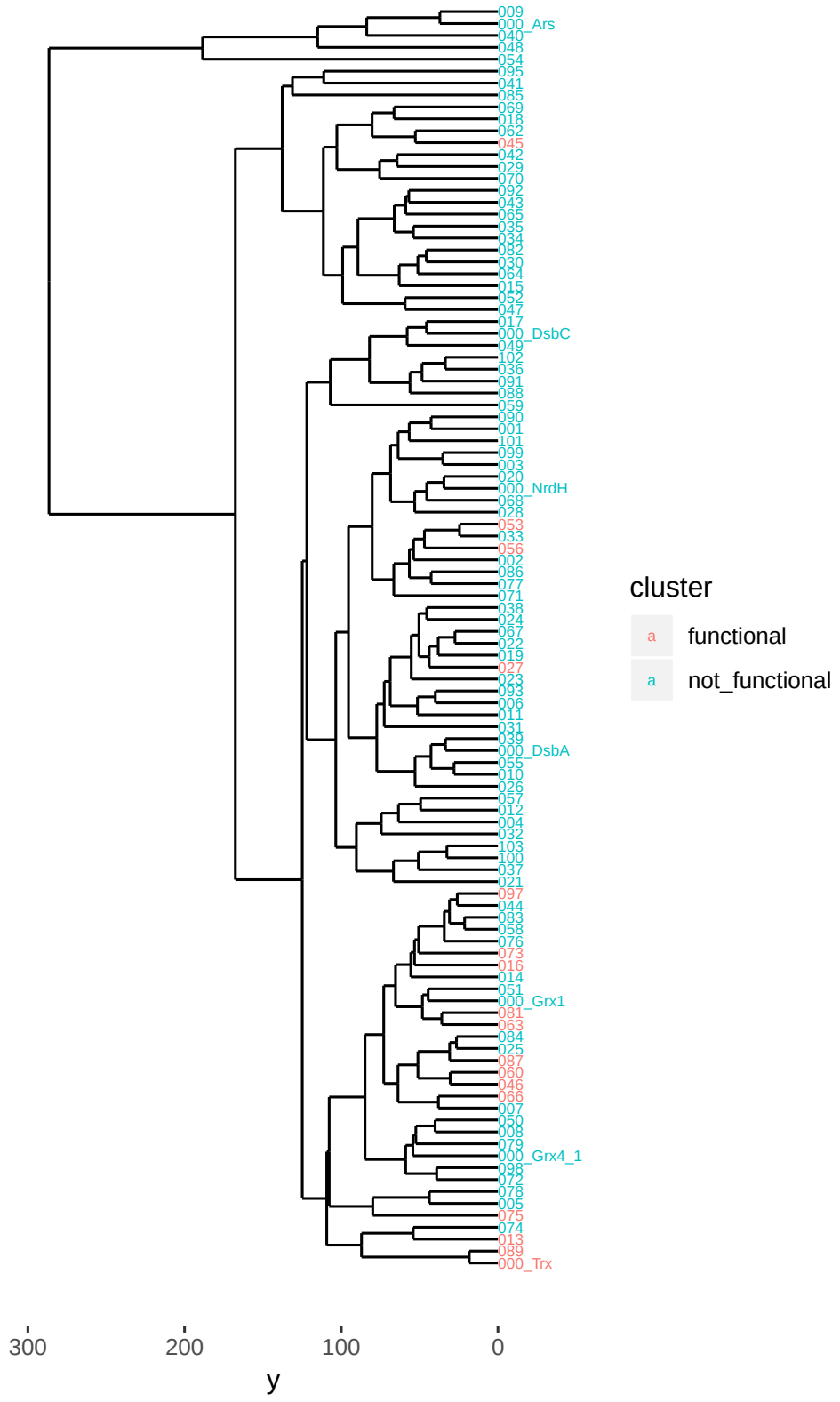


**Figure 5.3:** Clustering performed with the auto-encoder.

| | functional | not_functional |
|---|---|---|
| functional | 0.21 | 0.03 |
| not_functional | 0.79 | 0.97 |

**Table 5.6:** Confusion-matrix from the 106-Redoxins-test-set with higher weights for the true-negatives.

### 5.2.2 120 glutathiones

In this experiment a biochemical reaction was studied involving a glutathione. For each of the 120 proteins it was tested in vitro, if the protein reacts with the glutathione. Hence for each protein a label is given indicating if the proteins binded to each other. An experiment with 10-fold stratified cross-validation was conducted leading to an accuracy of 0.67 .

| | glutathionineBinding | NotGlutathionineBinding |
|---|---|---|
| glutathionineBinding | 0.59 | 0.30 |
| NotGlutathionineBinding | 0.41 | 0.70 |

**Table 5.7:** Confusion-matrix from the 120-Glutathionines-test-set.

### 5.2.3 De Novo Predictions on a few Proteins from the Drug Bank

For this experiment 86 pdbs were downloaded from the drug bank *https://www.drug bank.ca/* [34]. More specifically 4 targets were selected at random and then all available pdbs that have no overlap in these targets were downloaded. That means for each of the pdbs the class-label that needs to be predicted are the targets. The achieved accuracy on this data-set is 0.49. The confusion matrix is shown in table 5.8.

| | P03372_pdbs | P06401_pdbs | P10275_pdbs | P11511_pdbs |
|---|---|---|---|---|
| P03372_pdbs | **0.49** | 0.26 | 0.05 | 0.45 |
| P06401_pdbs | 0.14 | **0.38** | 0.15 | 0.24 |
| P10275_pdbs | 0.11 | 0.32 | **0.76** | 0.11 |
| P11511_pdbs | 0.25 | 0.04 | 0.04 | **0.20** |

**Table 5.8:** Confusion-matrix from pdbs obtained from the *drug-Bank*.

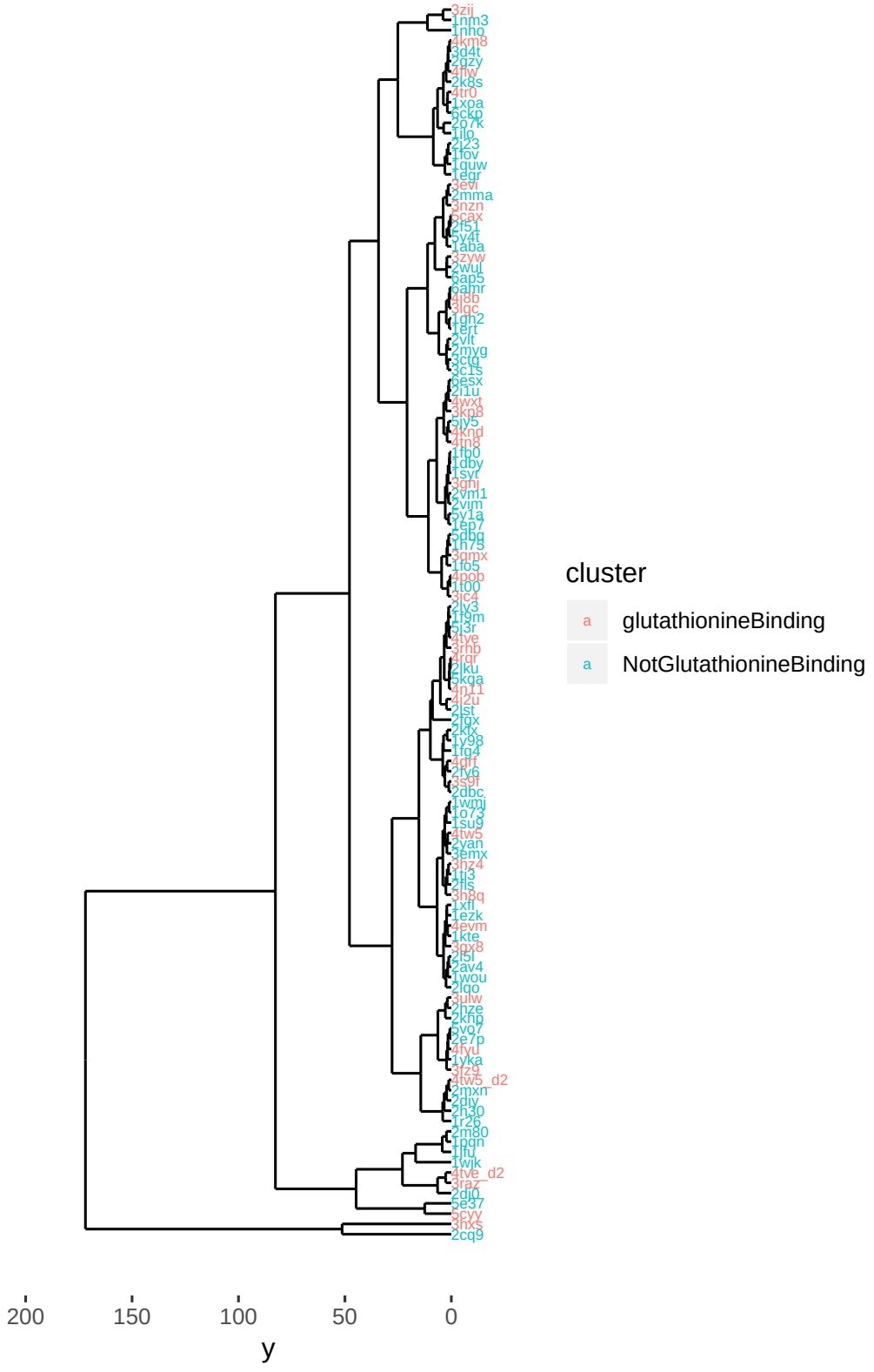


**Figure 5.4:** Clustering of *120 glutathiones* performed with *Quick-Repeated-Sampling*.

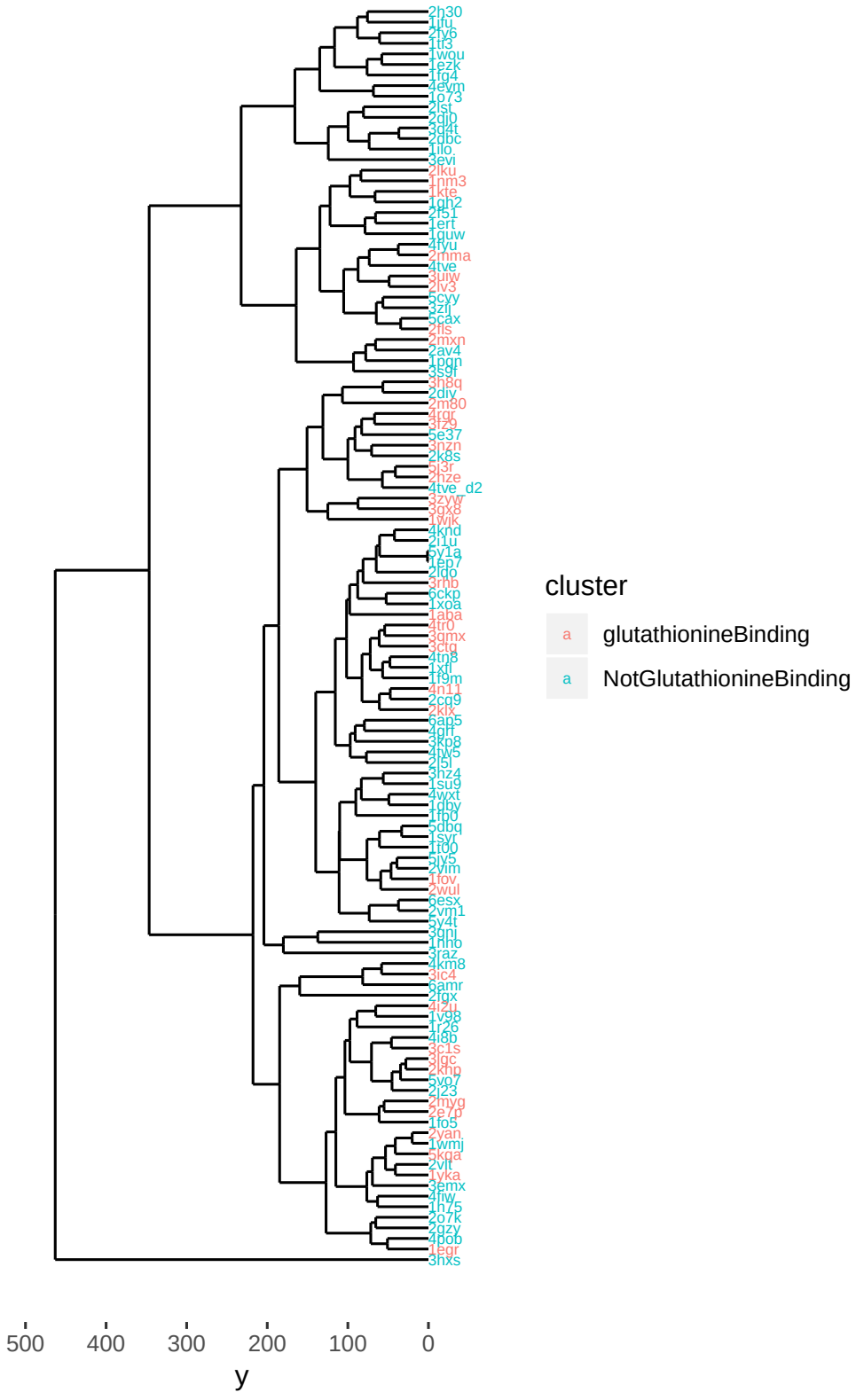


**Figure 5.5:** Clustering of *120 glutathiones* performed with the auto-encoder.

## 5.3 Conclusion

The results from this thesis can be divided into three topics:

- Theoretical findings regarding the Gromov-Wasserstein distance, that is a proof for the equality of **DE** and **FLB**.
- An implementation for predicting protein-protein-interactions. Evaluation of the implementation on diverse data-sets.
- The biological conclusions that can be drawn from the application of the implementation to the data-sets.

The results will be discussed in the above listed order.

### 5.3.1 A lower bound for the Gromov-Wasserstein Distance

Memoli proposed two lower bounds **FLB** and **DE** for the Gromov-Wasserstein distance $\mathfrak{D}$ [26]. Furthermore he proved that it holds $\mathfrak{D} \geq \mathbf{FLB} \geq \mathbf{DE}$. In this thesis it was proven that it holds $\mathbf{FLB} = \mathbf{DE}$ in the finite case. This is remarkable, since the computational cost for the **FLB** is much higher since it involves solving a linear program as opposed to calculating an integral, as done with the **DE**. Then using the quantiles of the distribution-functions an efficient approximation for the **DE** was derived. Using this technique the formerly proposed implementation by Felix Berens [2] could be sped up by a factor of approximately 50 on a data-set with 106 proteins. Here a more in-depth run-time-analysis would be necessary to quantify the performance increase on larger data-sets.

Another advantage of using the quantiles over calculating the **DE** directly, is that these values can be used as features to build more complex models.

### 5.3.2 A Software-Tool

The implementation automatizes tedious steps that generate the electrostatic potential from raw pdb-files. This is realized by utilizing the tools *pdb2pqr* [10] and *VMD* [17]. Then using *R* multiple features are calculated for each input protein based on the quantiles of the distribution of eccentricities. From these features, given a set of training-labels a neural-net can be trained using *keras* [6] respectively *tensorflow* [25] for classification. Alternatively if no labels are known a clustering can be computed. Applying the implementation to biochemically unrelated $3d$-models [32], [35] showed that it is in principle capable of distinguishing objects based on their geometry. The implementation offers different parameters to adjust the importance of the active site

and the boarder regions of the potentials (see sections 3.6.1, 3.6.2). This can serve as a starting-point in order to answer more biochemical questions regarding the mechanisms of reactions. The implementation offers parameters to set the importance of *specificity* in contrast to *sensitivity*.

A possible use-case of the implementation is to scan a library of candidate proteins quickly and exclude the non-relevant ones. Then applying techniques that are more computationally extensive, e.g. docking-tools such as *Swiss-Dock* [14], can be applied, making it possible to scan a much larger set of candidates in a given amount of time.

The implementation runs under the *GNU*-public license and can be downloaded at **https://github.com/WillyBruhn/PredictingProtein Interactions**.

### 5.3.3 Biochemical Conclusions

The hypothesis that similarity in the electrostatic potentials implies functional similarity, formerly stated by Lillig [4] was supported by the findings of this thesis. This conclusion is drawn from the fact that it was possible to build a classifier taking as input the geometry of the electrostatic potentials. The details of how the mechanisms of the reactions reflect in the iso-surfaces has yet to be investigated more extensively and is an ongoing research-topic in the work-group of Lillig. The implementations that were developed in the scope of this thesis and the thesis of Berens [2] can be seen as a ground-stone towards addressing the above mentioned research-topic.

The importance of the *active center*, the *boarder areas* (that is regions where positive and negative potential are very close to each other) and the influence of *long-range interactions* can be investigated with the implementation.

The implementation evaluates the similarity of two given proteins and hence their capability to functionally replace each other. The long-term goal however is to establish a method that predicts if two proteins can react with each other. Following the approaches in this thesis that would mean to evaluate in how far the geometry of the two proteins appears to be complementary.

# Chapter 6

# Appendix

## 6.1 Technical Details Regarding the Implementation

### 6.1.1 Automatization of the Visual Approach

This section roughly describes the workflow of the visual approach and how time-consuming steps in it were automatized.

The basic setting is the following: given the primary-structure of a protein $X$ and a set of proteins $Y_i$ (with their primary-structures), for each $Y_i$ the similarity to $X$ should be evaluated. In order to do so, VMD [17] a visualization-tool in order to get a 3D graphical representation of the iso-surfaces of the proteins as well as of the primary-structure and the secondary-structure was used. The graphical-representation is set on the active center of the protein in a beforehand manually set distance. The distance to be set, depends on the class of proteins that are examined and needs expert-knowledge. Larger molecules should be looked at from a further distance than smaller molecules. The key-point is that the distance is similar for all proteins that are investigated. Then a snapshot, meaning a 2D image of what is to be seen on the screen is taken. This is done for each protein. All the images of the iso-surfaces are compiled manually and put together in a big sheet. The biological expert then examines the images and decides based on the similarity of the images of $X$ and $Y_i$, if the proteins can be classified as similar or not similar. *VMD* (Visualize molecular Dynamics) is a molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting.

The pqr-files of the proteins are needed as Input. APBS solves the equations of continuum electrostatics for large biomolecular assemblages. The

output of the APBS-calculation is saved in *dx-files*. A dx-file specifies a voxel grid with a certain potential-value at each grid-point. E.g. a cubic voxel grid of size 128 contains $128^3 = 2097152$ different points. At each of these points the potential is stored. Given a certain potential *VMD* selects the points and interpolates between them with the marching-cubes algorithm [23] to create a visual representation of the iso-surface. This visual representation can be exported as wave-front-obj-file. This file then contains the surface as a *triangle-mesh*.

In order to extract the active center we used the pdb-files as input. Additionally we are given a list of motives that form active centers in the class of the redoxins. In our test set with 106 redoxins we are using a list that contains approximately 50 different motives, each with a length of 4 bases. The pdb-file is searched for each of the motives and outputs a warning if multiple motives are found. The corresponding position of the bases that form the active center is then used to calculate the geometric center of the bases. This geometric center is defined as location of the active center in all further methods. Note that since we only take the geometric center, we loose the information of the orientation of the active center. A script that is capable of doing this step can be found at **https://github.com/WillyBruhn/centerSelect.git**.

As a preliminary step for generating the data that was needed for the here presented thesis and the thesis of Berens [2] I implemented a shell-script that automatizes the above steps.

The proteins are given in pdb(protein-data-base-format). We use the command-line-tool *pdb2pqr* to convert the pdb-files to pqr-files. The pqr-file-format allows users to add charge and radius parameters to existing pdb data. This can be thought of as putting the pdbs in a water-simulation. Then an apbs-run is started that calculates the forces of the different atoms in the molecule. With the information added with the apbs-run, the different iso-surface values are added as graphical representation. The iso-surface values $-1$ and 1 were chosen. $-1$ in red and 1 as blue. Then the camera is positioned accordingly so that the focus is on the position of the active center in a fixed distance.

Additionally to the iso-surface-representation the primary-structure and secondary-structure is plotted. These three images per protein are then compiled into one larger overview as can be seen in figure 6.1

Additionally I implemented a GUI for easier usage as can be seen in figure 6.2. The different parameters can here be set easily and saved for different data-sets. This was done with C++ and the Qt-framework [12].

The implementation can be found at **https://github.com/WillyBruhn/MutComp.git**.

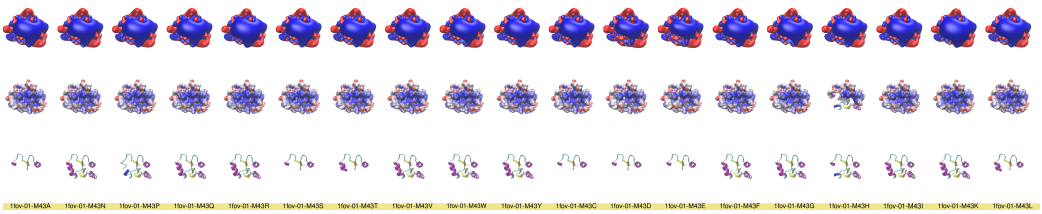

**Figure 6.1:** Output of MutComp - A compiled image of different proteins

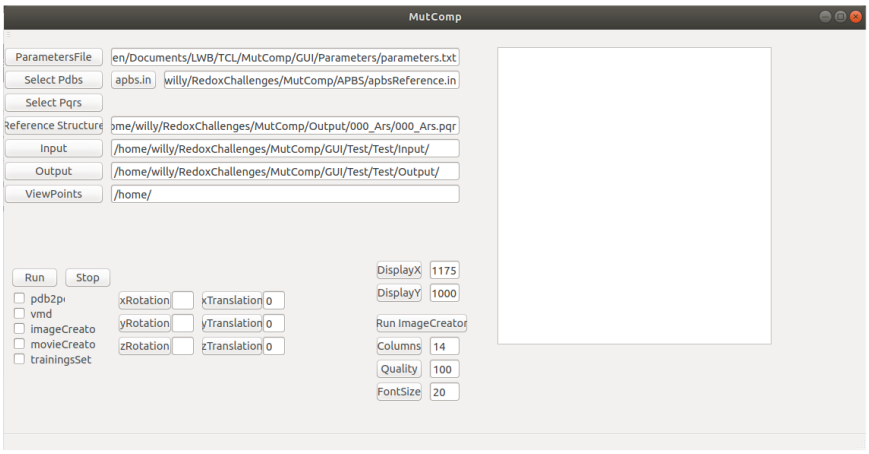


**Figure 6.2:** The graphical user-interface of *MutComp* - different parameters can be set here.

### 6.1.2 Downloading pdbs Automatically

In order to test the implementation on additional data I have written a *python*-script to download pdb-files automatically. As Input the name of a protein-target available at *https://www.drug bank.ca* has to be specified. A list of possible proteins that can interact with a specified drug from *https://www.drug bank.ca* are downloaded. Then the pdb-files are downloaded from *https://www.rcsb.org/*, if available. A file with the labels is then generated, where each pdb is assigned as functionality the target-protein according to *https://www.drug bank.ca*.

# Bibliography


[1] Nathan A Baker et al. "Electrostatics of nanosystems: application to microtubules and the ribosome". In: *Proceedings of the National Academy of Sciences* 98.18 (2001), pp. 10037–10041 (cit. on p. 38).

[2] Felix Berens. "Quantitative comparison of protein isosurfaces with approximated Gromov-Wasserstein-distance". MA thesis. Germany: University of Greifswald, 2019 (cit. on pp. 5, 37, 41–44, 71, 72, 74).

[3] Helen M Berman et al. "The protein data bank". In: *Nucleic acids research* 28.1 (2000), pp. 235–242 (cit. on pp. 2, 37).

[4] Carsten Berndt, Jens-Dirk Schwenn, and Christopher Horst Lillig. "The specificity of thioredoxins and glutaredoxins is determined by electrostatic and geometric complementarity". In: *Chem. Sci.* 6 (12 2015), pp. 7049–7058. URL: http://dx.doi.org/10.1039/C5SC01501D (cit. on pp. 1, 72).

[5] Gavin C Cawley and Nicola LC Talbot. "On over-fitting in model selection and subsequent selection bias in performance evaluation". In: *Journal of Machine Learning Research* 11.Jul (2010), pp. 2079–2107 (cit. on p. 26).

[6] François Chollet, JJ Allaire, et al. *R Interface to Keras*. https://github.com/rstudio/keras. 2017 (cit. on pp. 1, 71).

[7] Andrie de Vries and Brian D. Ripley. *ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'*. R package version 0.1-20. 2016. URL: https://CRAN.R-project.org/package=ggdendro (cit. on p. 59).

[8] Tom Dietterich. "Overfitting and undercomputing in machine learning". In: *ACM computing surveys* 27.3 (1995), pp. 326–327 (cit. on p. 26).

[9] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 25).

[10] Todd J Dolinsky et al. “PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations”. In: *Nucleic acids research* 35.suppl_2 (2007), W522–W525 (cit. on pp. 37, 71).

[11] RM Dudley. “Real analysis and probability, 2002 ed”. In: *Cambridge Studies in Advanced Mathematics* 74 () (cit. on p. 8).

[12] Qt Development Frameworks. *Qt*. 1995. URL: https://www.qt.io/ (visited on 04/09/2019) (cit. on p. 74).

[13] Steven Gold et al. “New algorithms for 2D and 3D point matching: Pose estimation and correspondence”. In: *Pattern recognition* 31.8 (1998), pp. 1019–1031 (cit. on p. 4).

[14] Aurelien Grosdidier, Vincent Zoete, and Olivier Michielin. “SwissDock, a protein-small molecule docking web service based on EADock DSS”. In: *Nucleic acids research* 39.suppl_2 (2011), W270–W277 (cit. on p. 72).

[15] Berthold K. P. Horn. “Closed-form solution of absolute orientation using unit quaternions”. In: *J. Opt. Soc. Am. A* 4.4 (Apr. 1987), pp. 629–642. URL: http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629 (cit. on p. 4).

[16] Jingwei Huang, Hao Su, and Leonidas Guibas. “Robust Watertight Manifold Surface Generation Method for ShapeNet Models”. In: *arXiv preprint arXiv:1802.01698* (2018) (cit. on pp. 38, 63).

[17] William Humphrey, Andrew Dalke, and Klaus Schulten. “VMD: visual molecular dynamics”. In: *Journal of molecular graphics* 14.1 (1996), pp. 33–38 (cit. on pp. 1, 2, 37, 71, 73).

[18] Gregory Jefferis and Syoyo Fujita. *readobj: Fast Reader for 'Wavefront' OBJ 3D Scene Files*. R package version 0.3.2. 2019. URL: https://CRAN.R-project.org/package=readobj (cit. on p. 38).

[19] Ron Kohavi et al. “A study of cross-validation and bootstrap for accuracy estimation and model selection”. In: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145 (cit. on p. 26).

[20] Walter A Koppensteiner et al. “Characterization of novel proteins based on known protein structures”. In: *Journal of molecular biology* 296.4 (2000), pp. 1139–1152 (cit. on p. 1).

[21] Daniel E Koshland Jr. “The key–lock theory and the induced fit theory”. In: *Angewandte Chemie International Edition in English* 33.23-24 (1995), pp. 2375–2378 (cit. on p. 1).

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436 (cit. on p. 55).

[23] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169 (cit. on p. 74).

[24] Martin Maechler et al. *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.6 — For new features, see the 'Changelog' file (in the package source). 2017 (cit. on p. 59).

[25] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on pp. 1, 71).

[26] Facundo Mémoli. "Gromov–Wasserstein Distances and the Metric Approach to Object Matching". In: *Foundations of Computational Mathematics* 11.4 (Aug. 2011), pp. 417–487. URL: https://doi.org/10.1007/s10208-011-9093-5 (cit. on pp. 1, 4–9, 18, 24, 26, 27, 37, 40, 41, 61, 62, 71).

[27] Garth Powis and William R Montfort. "Properties and biological activities of thioredoxins". In: *Annual review of biophysics and biomolecular structure* 30.1 (2001), pp. 421–455 (cit. on p. 2).

[28] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2008. URL: http://www.R-project.org (cit. on p. 1).

[29] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: *International Journal of Computer Vision* 40.2 (Nov. 2000), pp. 99–121. URL: https://doi.org/10.1023/A:1026543900054 (cit. on p. 42).

[30] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the icp algorithm." In: *3dim*. Vol. 1. 2001, pp. 145–152 (cit. on p. 4).

[31] Terence D Sanger. "Optimal unsupervised learning in a single-layer linear feedforward neural network". In: *Neural networks* 2.6 (1989), pp. 459–473 (cit. on p. 58).

[32] Robert W. Sumner and Jovan Popović. "Deformation Transfer for Triangle Meshes". In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 399–405. URL: http://doi.acm.org/10.1145/1015706.1015736 (cit. on pp. 1, 4, 6, 24, 25, 27, 61, 71).

[33] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. URL: https://books.google.de/books?id=GqRXYFxe0l0C (cit. on pp. 9, 10, 16).

[34] David S Wishart et al. "DrugBank: a comprehensive resource for in silico drug discovery and exploration". In: *Nucleic acids research* 34.suppl_1 (2006), pp. D668–D672 (cit. on pp. 1, 68).

[35] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920 (cit. on pp. 1, 61, 71).