Modeling protein stability with Gaussian processes

Emmi Jokinen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 1.8.2016

Thesis supervisor:

Assist. Prof. Harri Lähdesmäki

Thesis advisor:

PhD Markus Heinonen

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Emmi Jokinen

Title: Modeling protein stability with Gaussian processes

Date: 1.8.2016 Language: English Number of pages: 6+63

Department of Computer Science

Professorship: Computational Systems Biology

Supervisor: Assist. Prof. Harri Lähdesmäki

Advisor: PhD Markus Heinonen

Proteins are used in various applications by different industries. In order to refine the processes they are used in or to create new applications, protein engineering is applied to alter the properties of proteins by introducing mutations to them. It is often desirable to improve the stability of proteins as they should be stable in the conditions of industrial processes. Protein stability predictors provide a way to estimate how mutations affect the stability. When a novel protein is being designed, the predictors can thus be used to reduce the amount of proteins to be tested experimentally.

This master's thesis introduces two machine learning approaches for predicting stability changes of proteins upon mutations. They both utilise Gaussian processes and a graph presentation of proteins, but by using different kernels and different notions of similarity, they adapt to different situations. The first approach uses experimental stability measurements only from the protein of interest. When enough data is available it can reach excellent results. For example, when we trained this model using a stability data set of 349 measurements for bacteriophage T4 lysozyme and leave-one-out cross validation, we achieved a correlation of 0.90 and root mean squared error of 0.76 kcal/mol and outperformed the current state-of-art prediction methods. This method can predict the effects of single and multiple simultaneous mutations and can also incorporate information from predictors relying on energy functions to further improve stability predictions.

The second approach exploits data from multiple proteins and can be applied even when only little or no experimental data is available from the protein of interest. We trained this model using a previously published data set of 2648 mutations from 131 proteins. When a set of 350 mutations of this data set was excluded for testing and the rest of the data was used for training, we achieved reasonable results, a correlation of 0.54 and a mean squared error of 1.32 kcal/mol.

Keywords: Protein stability, Gaussian processes, graph kernels, MKL

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Emmi Jokinen

Työn nimi: Proteiinien stabiilisuuden mallintaminen Gaussin prosesseilla

Päivämäärä: 1.8.2016 Kieli: Englanti Sivumäärä: 6+63

Tietotekniikan laitos

Professuuri: Laskennallinen systeemibiologia

Työn valvoja: apul.prof. Harri Lähdesmäki

Työn ohjaaja: FT Markus Heinonen

Proteiineja hyödynnetään useissa sovelluksissa eri teollisuuden aloilla. Kun halutaan tehostaa proteiineja käyttäviä prossesseja tai kehitetään uusia sovelluksia, niin proteiinien ominaisuuksia voidaan muokata tekemällä niihin mutaatioita. Proteiinien stabiilisuuden parantaminen on usein tarpeellista, sillä niiden tulisi olla stabiileja teollisuusprosessien olosuhteissa. Proteiinien stabiilisuusennustimien avulla voidaan arvoida miten mutaatiot vaikuttavat proteiniien stabiilisuuteen. Uusia proteiineja suunniteltaessa ennustimien käyttö voi siten vähentää kokeellisesti testattavien proteiinivarianttien määrää.

Tässä diplomityössä esitellään kaksi koneoppimismenetelmää, joilla voidaan ennustaa stabiilisuuden muutoksia kun proteiineihin tehdään mutaatioita. Molemmat metodit hyödyntävät Gaussin prosesseja ja esittävät proteiinit verkkoina, mutta ne käyttävät eri kerneleitä ja erilaisia kuvauksia samankaltaisuudelle ja mukautuvat siten eri tilanteisiin. Ensimmäinen malli hyödyntää vain tarkasteltavasta proteiinista saatua stabiilisuusdataa. Kun dataa on tarpeeksi, niin mallilla saadaan erinomaisia tuloksia. Esimerkiksi käyttämällä 349 mutaation datasettiä bakteriofagi T4 lysosyymille ja yksi-pois -ristiinvalidointia saimme korrelaation 0.90 ja virheen neliöllisen keskiarvon 0.76 kcal/mol ja suoriuduimme siten paremmin kuin muut stabiilisuusennustimet. Tätä mallia käyttäen voidaan ennustaa sekä yksittäisten että useiden samanaikaisten mutaatioiden vaikutuksia ja hyödyntää informaatiota energiafunktioita käyttäviltä ennustimilta stabiilisuusennustuksien parantamiseksi.

Toinen esitelty malli hyödyntää stabiilisuusdataa useista proteiineista ja sitä voidaan siten käyttää myös silloin, kun tarkasteltavasta proteiinista on saatavilla vain vähän tai ei lainkaan dataa. Tämän mallin kanssa käytimme aikaisemmin julkaistua datasettiä, jossa on mutaatioita 131 eri proteiinista. Datasetin 2648 mutaatiosta erotettiin 350 mutaatiota testausta varten ja loppuja käytettiin mallin kouluttamiseen. Näin saavutimme kohtuulliset tulokset, korrelaation 0.54 ja virheen neliöllisen keskiarvon 1.32 kcal/mol.

Avainsanat: Proteiinien stabiilisuus, Gaussin prosessit, verkkokernelit, MKL

Preface

First of all I want to thank my advisor Markus Heinonen and supervisor Harri Lähdesmäki for all their guidance and insight during this process. It is thanks to them that I will now continue my studies as a graduate student. The CSB research group has been a great group to work in and I am glad that I will be staying in it.

I would also like to thank all my friends who have made my studies here in Aalto such an excellent time, especially Bratislawa Youghurt, who provided me a new home right when I started my studies here and for many years after that.

Otaniemi, 1.8.2016

Emmi Jokinen

Contents

Symbols and abbreviations

Symbols

ΔG	Difference in the Gibbs energy between a folded and unfolded protein
$\Delta\Delta G$	Change in ΔG between the wild type and mutated protein, $\Delta G_{mut} - \Delta G_{wt}$
r	Pearson's correlation
rmse	Root mean squared error

Abbreviations

BFGS	Broyden-Fletcher-Goldfarb-Shanno
BLOSUM	Blocks substitution matrix
GP	Gaussian process
GPR	Gaussian process regression
L-BFGS	Limited memory BFGS
MKL	Multiple kernel learning
MLL	Marginal log-likelihood
MLM	Marginal likelihood maximisation
NMR	Nuclear magnetic resonance
PCHIP	Piecwise cubic hermite interpolating polynomial
PDB	The Protein data bank
PDB ID	The id number of a protein structure in the Protein data bank
REU	Rosetta energy unit
WDK	Weighted decomposition kernel

1 Introduction

Enzymes are a class of proteins that determine all the chemical transformations that make and break covalent bonds in cells. They bind to one or more substrates over and over again converting them into one or more chemically modified products. They significantly speed up reactions in the cells without themselves being changed, that is, they act as catalysts. Furthermore, enzymes are the most selective and powerful catalysts known. [2] They are used in various industrial processes in detergent, starch, fuel, textile, food and pharmaceutical industries and their usage is growing steadily [30, 51]. As they are derived from renewable resources, are biodegradable, work under relatively mild conditions of temperature and are often highly selective, they provide important advantages over chemical catalysts [14]. Protein engineering is used to further improve the properties of enzymes, for example to enhance their catalytic activity, modify their substrate specificity or to improve their thermostability [46]. Increasing the stability of proteins is an important aspect of protein engineering, as the enzymes used in industry should be stable in the industrial process conditions,

which often involve higher than ambient temperature and may involve non-aqueous solvents [7].

In order to do this, alterations are introduced to the amino acid sequence of a protein. Most common alterations are mutations, but also insertions and deletions are possible. [46] Mutations in general tend to be destabilising, so the stability needs to be taken into consideration when proteins are being modified in any way. If too many destabilising mutations are made to a protein, it does not remain functional and compensatory stabilising mutations may be needed. [60] Various methods using different approaches have been developed to predict the changes in protein stability upon mutations [4, 10, 13, 17, 21, 24, 34, 43, 44, 59]. These methods utilise physics or knowledge based potentials, their combination, or different machine learning methods. However, it has been assessed that although on average many of these methods provide good results, they tend to fail on details [45].

In this master's thesis two machine learning approaches will be proposed for the prediction of changes in protein stability upon mutations. They both utilise Gaussian processes and a graph presentation of proteins, but by using different kernels and different notions of similarity, they adapt to different situations. The first approach exploits stability data gathered only from the protein of interest. This method can achieve very accurate results and even predict changes when multiple mutations are introduced simultaneously to the protein. The disadvantage of this method is however, that such amounts of data that are required to train the model is not publicly available for many proteins. It will be shown that in such situation the prediction accuracy may be improved by generating additional data with a prediction method that uses physics or knowledge based potentials, such as Rosetta [34].

The second approach predicts the changes in stability upon single mutations. It does not necessarily require data from the protein of interest, as it can exploit the stability measurements from different proteins. It may not be quite as accurate as the first approach, but it can be useful when one is interested in a protein that has not been previously studied or has only little experimental stability data.

The development of these kinds of machine learning methods has been made possible by the increased amount of available data in many databases and increased computational resources. The methods proposed here use the three dimensional protein structure models available from the Protein Data Bank[1] (PDB) [5] to create graph representations of the mutated proteins that can be used with kernel methods. The similarities of the amino acids in the proteins are assessed with the features provided by AAindex[2] [27] and the experimentally measured stability data was obtained from

[1]The Protein Data Bank: www.rcsb.org
[2]AAindex: http://www.genome.jp/aaindex

a thermodynamic database Protherm[3] [32]. The required calculations were enabled by the computational resources provided by the Aalto Science-IT project.

In order to predict the stability changes of proteins upon mutations, the proteins must be presented in a form that can be used with machine learning methods. It is therefore important to understand the structure and properties of proteins at some level, so that the important aspects can be captured by the chosen representation.

As protein structures cannot be directly observed, one needs to also decide which protein structure models to begin with. There exists a few methods that are commonly used to derive the three dimensional structures of proteins [48]. Understanding the basic principles of these methods can provide insight into which protein structure models to use and the limitations that should be taken into consideration when working with them. Chapter 2 introduces important concepts concerning the protein structure and Chapters 3 and 4 present the idea of protein engineering and some current protein stability prediction approaches. In Chapter 5 the used methods are explained and Chapters 6 and 7 present the two proposed approaches for predicting changes in protein stability upon mutations.

[3]Protherm: http://www.abren.net/protherm/

2 Protein structure

Proteins consist of 20 different types of amino acids that form a long polypeptide chain by linking to their neighbours through covalent peptide bonds. The core of each of the amino acids is the same. In the center of this core there is an α-carbon atom, to which an amino group (NH_2) and a carboxyl group (C=O) are attached to. These parts of the amino acids form the core of the polypeptide chain, referred to as polypeptide backbone. The amino acids also have side-chains, which are attached to the α-carbon atom as well. They are not involved in making the peptide bonds, but are responsible for the different properties of amino acids and take part in weaker bonds. The 20 naturally occurring amino acids are listed in Table 1 with their codes, abbreviations and side-chain properties. The side-chains can be polar and have negative, positive or neutral charge, or they can be nonpolar. They are also of different sizes and volumes. [2] Figure 1 shows a fraction of the amino acid sequence of bacteriophage T4 lysozyme. The backbone consists of the black α-carbon atoms, grey carbons, blue nitrogens and red oxygens. Side chain atoms are coloured green.

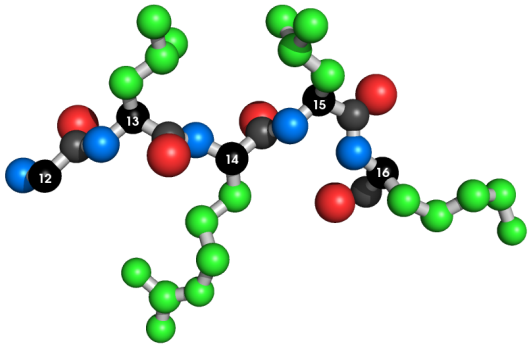


Figure 1: A fraction of the amino acid sequence of bacteriophage T4 lysozyme: Residues 12-16, GLRLK. The atoms that are part of the backbone are coloured by their type: α-carbons are black, other carbons grey, nitrogens blue and oxygens red. All side chain atoms are coloured green. The Hydrogen atoms are not shown.

Table 1: The 20 naturally occurring amino acid and their codes, abbreviations and side-chain properties.

name	**code**	**abbreviation**	**side-chain**
Alanine	A	Ala	nonpolar
Arginine	R	Arg	positive
Aspargine	N	Asn	uncharged polar
Aspartic acid	D	Asp	negative
Cysteine	C	Cys	nonpolar
Glutamine	Q	Gln	uncharged polar
Glutamic acid	E	Glu	negative
Glycine	G	Gly	nonpolar
Histidine	H	His	positive
Isoleucine	I	Ile	nonpolar
Leucine	L	Leu	nonpolar
Lysine	K	Lys	positive
Methionine	M	Met	nonpolar
Phenylanine	F	Phe	nonpolar
Proline	P	Pro	nonpolar
Serine	S	Ser	uncharged polar
Threonine	T	Thr	uncharged polar
Tryptophan	W	Trp	nonpolar
Tyrosine	Y	Tyr	uncharged polar
Valine	V	Val	nonpolar

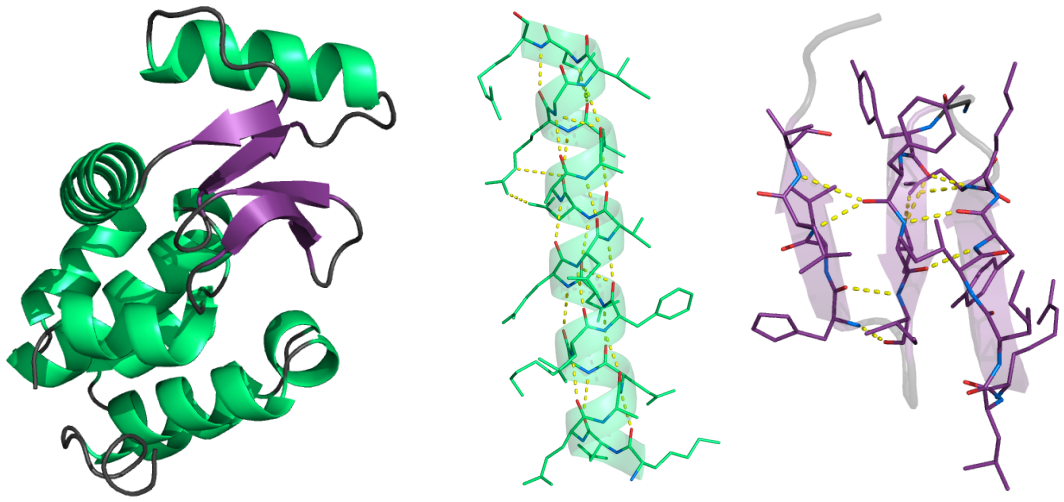
The order of the amino acids in the amino acid sequence is said to be the primary structure of the protein. It determines how the protein will fold and get its three dimensional structure. Each protein folds into a single stable conformation that in general minimises its free energy. This conformation can however change slightly

when the protein interacts with different molecules. In addition to the covalent bonds between the amino acids, there are also different weak bonds that can lower the energy of the conformation and stabilise the protein. [2]

There are three types of different weak non-covalent bonds: hydrogen bonds, electrostatic attractions and van der Waals attractions. Non-covalent bonds are 30-300 times weaker than the typical covalent bonds, but the combined strength of many non-covalent bonds determines the stability of the protein. Hydrophobic and hydrophilic amino acids also have a central role in determining the fold of a protein, as proteins are typically in aqueous environment. Therefore the hydrophobic amino acids tend to be forced together in order to minimise their disruptive effect on the hydrogen bonded water molecules. Hydrophilic amino acids on the other hand tend to be near the outside of the protein, where they can form hydrogen bonds with water and other polar molecules. The protein structure can also be stabilised by covalent cross-linkages between the amino acid side chains. Disulfide bonds can form between the -SH groups of two adjacent cysteines. They do not change the conformation of the protein, but reinforce it. These bonds do not generally form in the cell cytosol, but they may help the protein maintain its structure in extracellular conditions. [2]

Although the overall conformation of each protein is unique, there are two regular folding patterns that are often found in parts of them: α-helix and β-sheet. The secondary structure patterns of bacteriophage T4 lysozyme can be seen in Figure 2a, where α-helices are colored with green and β-sheets with violet. These patterns result from hydrogen-bonding between the N-H and C=O groups in the polypeptide chain. Since the bonding does not involve the side-chains, many different amino acids can form them. Figures 2b and 2c show examples of the hydrogen bonding within an α-helix and between β-sheets, respectively. Because of the weak bonds that form within these structures, they also stabilise the protein structure. These patterns are said to form the secondary structure of the protein. The three-dimensional structure of the polypeptide chain can be called as the tertiary structure and if the protein is formed of more than one polypeptide chain, the complex can be called as the quaternary structure of the protein. [2]

There are a few methods that can be used to determine the 3D-structure of a protein. A commonly used method for constructing the the 3D-structures is X-ray crystallography. The wavelengths of X-rays are small enough for the X-ray beams to be diffracted even by the smallest molecules. However, a single molecule produces a very weak scatter of X-rays and therefore the protein needs to be crystallised so that the resulting protein crystal contains multiple ordered molecules in identical orientations that diffract identically. The beams diffracted from the different molecules augment each other so that strong and detectable beams are produced. [48]



(a) (b) (c)

Figure 2: (a) shows the secondary structure of bacteriophage T4 lysozyme (PDB ID 2LZM), where α-helices are colored with green and β-sheets with violet. (b) displayes the hydrogen bonds within an α-helix and (c) three β-sheets and the hydrogen bonds between them. Oxygen atoms are colored with red and nitrogen atoms with blue.

Under certain conditions proteins solidify to form crystals in which the individual proteins adopt one or a few identical orientations, but under different conditions different orientations may occur. Also, structures defined with X-ray crystallography may differ from their natural structures, as proteins normally action in solution where the weak forces between the amino acids and water molecules can affect the protein structure. However, significant alterations are rare. Also, protein crystals are held together primarily by hydrogen bonds between hydrated protein surfaces and are thus very fragile and the crystals can be damaged by the free radicals generated by X-rays. Therefore the obtained structures should always be validated. [48]

Nuclear magnetic resonance (NMR) spectroscopy is another method for determining the protein structure. It derives the protein models in solution and assigns spectral peaks to all residues by chemical shift and decoupling experiments can be used to derive distance restraints that reveal local conformations and residues that are close in the folded protein but distant in the amino acid sequence. Based on these restrictions a set of chemically, stereochemically and energetically feasible models is calculated. NMR spectroscopy does not produce a single model, but an ensemble of possible models, that often agree well on some regions of the protein and not so well on others. A single model can be created from the ensemble for example by averaging the coordinates of the atoms. [48] NMR spectroscopy models benefit from the fact that the proteins are examined in their native form in solution. However, the models obtained with X-ray crystallography are often preferred in applications of protein design because of their high resolution. [53]

The Protein Data Bank (PDB) (www.rcsb.org) gathers these experimentally determined protein structure models in one place. The models are stored in coordinate files that contain the 3D coordinates for each of the atoms in the model. The number of protein structure models gathered in PDB has rapidly grown and currently (in May 2016) it contains 110290 protein structure models, out of which 99295 are created with X-ray crystallography and 9992 with NMR spectroscopy. There are also different metrics of the models available that can be used to asses the quality of the models. [5]

Proteins can also be presented in a more compact form as contact maps that still preserve much of the structural information. A contact map of a protein of n residues is a $n \times n$ binary matrix C, where $C_{ij} = 1$ if residues i and j are in contact and otherwise $C_{ij} = 0$. A contact occurs between two residues when they are within a given threshold of each other. The distance between the residues may be measured between C_α-atoms, C_β-atoms or the closest atoms of the two residues. [64] It has been suggested that the latter definition is not biased towards amino acid residues of any size, but rather introduces "white noise" as more contacts will be defined. [39] Contact maps can be derived from the coordinate files available from PDB. Figure 3 illustrates how the contact map is formed.

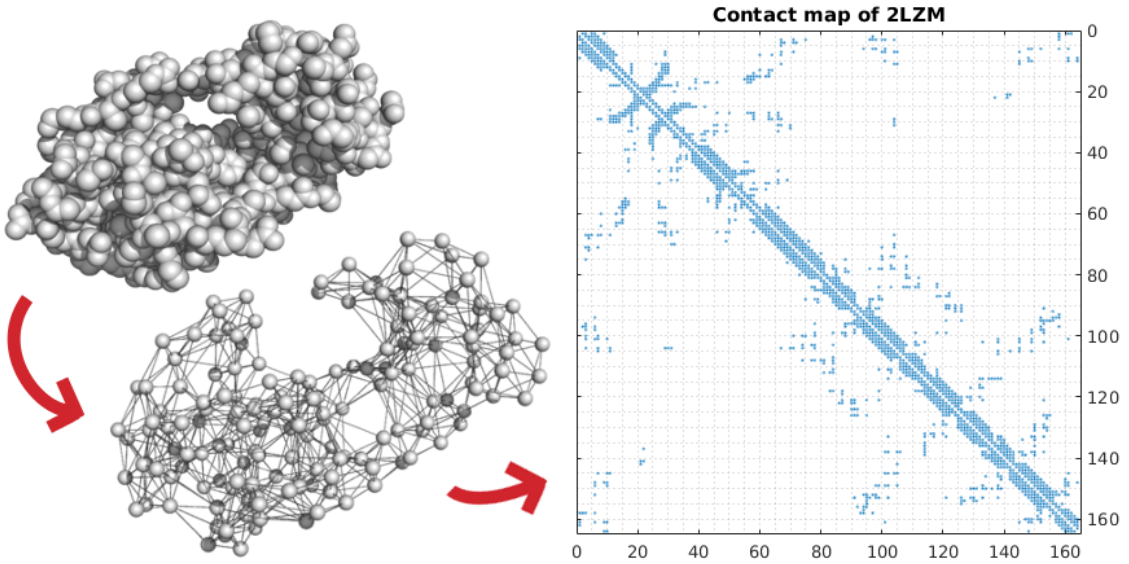


Figure 3: Contacts are defined between residues that are within the given threshold of each other. On the left they are shown as lines between the residues. Contact map presents these contacts in matrix form.

3 Protein engineering and design

Protein engineering is the process of constructing novel protein molecules. It can be used to provide new information of how proteins are assembled and what elements are essential for their structure, or to optimise the properties of a protein for a particular technology purpose. [46] Protein engineering often concentrates on the improvement of the properties of enzymes. Enzymes are a class of proteins that catalyse chemical reactions and they are among the most effective catalysts known [2]. Ideally, a biocatalyst should have high specific activity and high specificity on the reactant. It should also be stable at industrial process conditions, that often include higher than the ambient temperature, and partially or wholly nonaqueous solvents. [7] By substituting some of the amino acids of the protein, the protein can be altered so that for example its catalytic activity is enhanced, substrate specificity or pH profile is altered or stability improved [46].

The design of a new protein can be started from first principles (*de novo*) or by examining existing protein structures and then applying alterations to them. The

challenge of *de novo* protein design is that for any protein of n residues there are 20^n different possible sequences. Therefore it is often easier to decide which fold is appropriate and then identify a sequence that would be needed to generate that fold, since there are a range of sequences that can be accommodated into similar folds. [46] The design then starts by selecting a protein template and a protein engineering approach [7]. After the changes to the protein template have been decided, mutagenesis is used for the implementation of those changes and finally the created protein variants are evaluated for improved properties by screening or selection. There is a trade-off between the efforts needed for the protein engineering and the screening. For example, if the selected engineering approach is random mutagenesis, the engineering part is trivial, but the screening may be time consuming as it is likely that many random mutations need to be tested in order to find one that produces the desired effect. Good engineering approaches are valuable, as they can reduce the efforts needed for screening. [28]

One important aspect of protein engineering is protein stability. It can be defined as the difference in Gibbs energy ΔG between the native and denaturated state of a protein. The denaturated state can be thought as the state of the protein that exists after thermal denaturation [49]. More precisely the Gibbs energy difference determines the thermodynamic stability of the protein as it does not take into account the additional energy that may be needed for the transition between the two states. This additional energy that is needed determines the kinetic stability of the reaction. A protein is kinetically stable, if the activation barrier is substantial. [3] Figure 4 illustrates the difference between thermodynamic stability (ΔG_t) and kinetic stability (ΔG_k). Here only the thermodynamic stability will be considered and it will be referred to merely as stability ΔG. Globular proteins are only marginally stable, the free energy difference between the native and denaturated state of a protein is about 5-15 kcal/mol, which is not much more than the energy of a single hydrogen bond that is of the order 2-5 kcal/mol. [8] In globular proteins the polypeptide chain is folded up in a compact ball like shape with an irregular surface. Most of the enzymes are globular proteins. [2]

When mutations are introduced to proteins, their effect to the stability of the protein can be defined by the change they cause to the Gibbs energy ΔG, denoted as $\Delta\Delta$G. The stability ΔG of a protein can be estimated from the thermal, urea and guanidinium chloride (GdmCl) denaturation curves that are determined as the fraction of unfolded proteins at different temperatures or at different concentrations of urea or DdmCl. [40] According to Pace and Scholz [40], it seems that sometimes the $\Delta\Delta$G-values measured with thermal and urea denaturation are in good agreement and sometimes they are not, whereas according to Pace and Shaw [41] the ΔG-values acquired with these methods should be within the experimental error, which is approximately $\pm$0.3 kcal/mol. Therefore it might be safer to compare the stability

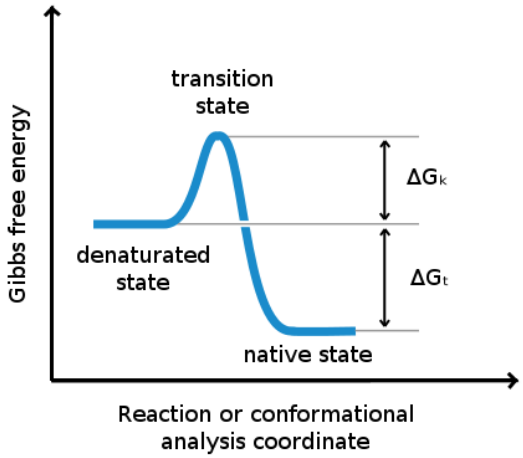


Figure 4: Gibbs free energy between different states. The free energy between the denaturated and native state is referred to as the thermodynamic stability.

values determined using only one of these methods. Many thermodynamic parameters such as the $\Delta\Delta$G-values from proteins and their mutants have been experimentally determined and collected in databases such as Protherm, which also specifies the method which is used to determine each of these stability values [32].

Improving the stability of a protein is often desirable as the enzymes used industry should be stable in the industrial process conditions [7]. In addition, it may be beneficial to consider the stability of a protein also when the function of the protein is being altered. That is because when new function mutations are introduced to a protein, they tend to destabilise the protein structure and compensatory stabilising mutations are often needed to maintain a functional protein. According to a stability threshold model, proteins stay functional when their stability remains within a certain margin. If the stability decreases below a certain threshold, the protein starts to unfold. Also, if the protein becomes too stable, the protein dynamics or regulation may be affected. [60] If the stability aspects are taken into consideration already in the phase of designing the protein, it can reduce the amount of protein variants to be screened.

When mutations are done, they are commonly defined by giving the one letter code of the wild type amino acid, the position number of the mutated residue in the amino acid sequence, and the one letter code of the mutated amino acid. For example A15I means that the 15th residue of the protein in question has been mutated from Alanine to Isoleucine. The codes and abbreviations of the amino acid are listed in Table 1.

Substitution matrices provide a way to classify amino acids by describing how well one amino acid can be substituted by another. These matrices are derived from

large sets of aligned sequences. A high value is indicated to the substitution, if the substitution is seen more often than expected by chance and conversely low values are assigned to rare substitutions. Substitution matrices provide rough guides of the quality of mutations, independent of the position in the sequence. [6].

BLOSUM matrices are substitution matrices that are derived from about 2000 ungapped blocks of aligned sequence segments collected from more than 500 groups of related proteins. Each block represents a conserved region of a protein family. Withing blocks sequence segments which have at least L percentage of identical amino acids are clustered and each cluster is weighted as a single sequence in counting pairs. The clustering reduces the contribution of closely related segments to the frequency table. A BLOSUM matrix created with clustering percentage L is referred to as BLOSUM L. Lower L value corresponds to longer evolutionary distances [18]. The scores $s(a, b)$ of the matrix between each amino acids a and b are estimated from the frequencies A_{ab} of observing residue a aligned against residue b in one cluster. This is defined by Equations 1-3, where q_a is the fraction of pairings that include a and p_{ab} is the fraction of pairings between a and b out of all observed pairings. [25]

$$s(a,b) = \log \frac{p_{ab}}{q_a q_b} \tag{1}$$

$$q_a = \frac{\sum_b A_{ab}}{\sum_{cd} A_{cd}} \tag{2}$$

$$p_{ab} = \frac{A_{ab}}{\sum_{cd} A_{cd}} \tag{3}$$

As stated earlier, amino acids have different side chains and thus different properties and these features can also be used to compare amino acids. AAindex [27] lists 544 indices that describe numerically the differences between the 20 naturally occurring amino acids. 531 of these indices contain values for each of the 20 amino acids. Here we scale all of the features between 0 and 1 and construct substitution matrices $\mathbf{S}_f$ using each of these features f as follows:

$$\mathbf{S}_f(a,b) = 1 - |f_a - f_b| \,. \tag{4}$$

Here f_a denotes the value of feature f given amino acid a.

4 Protein stability predictions

Several computational methods that rely on different approaches have been developed to predict the stability changes in proteins upon mutations. A group of these methods uses energy functions for the prediction task. They utilise potentials based on either physics [4, 44], knowledge [24] or both [34]. Also machine learning approaches that exploit the available experimental stability measurements have been developed. Some of these methods use only sequence level information of the proteins and can thus be used even if the structure of the protein is unknown [9, 19, 36] whereas others rely also on the information induced from three dimensional protein structures available from databases [13, 17, 21, 43, 59, 62]. In this section we will discuss the methods relying on potentials and machine learning methods and provide an example of both. The most commonly used metrics for the evaluation of these methods will also be presented.

4.1 Methods relying on potentials

Physics based potentials are obtained using principles of physics. The thermodynamic properties of the system are calculated using quantum mechanics or experimental measurements. The parameters of the atomic potentials are then obtained by combining the appropriate experimental and theoretical values. These atomic physics based potentials have been found promising, but it is computationally costly to apply them to large proteins that consist of large number of atoms. [31]

Knowledge based potentials are based on information extracted from sets of protein structures. They can be used to describe the interactions between residues or atoms in proteins approximately. Their advantage is therefore that they can describe interactions that would be difficult or impossible to formulate using laws of physics or to determine experimentally. An important source of the used structural information is the Protein Data Bank and as the size of the database increases, it becomes possible to derive more improved and more specific potentials. The knowledge based potentials can be atomic level or coarse-grained potentials. The coarse-grained potentials have lower computational costs but they are not always sufficient to reflect the entire landscape of a potential energy surface, although their performances are largely modulated by the choice of the coarse-graining scheme. [31]

Rosetta3 combines both physics and knowledge based potentials for the predictions of stability changes upon mutations. It is a software suite for the simulation and design of macromolecules that can be used for solving a wide variety of problems in structural biology, including prediction of changes in stability (the $\Delta\Delta$G) of a monomeric protein upon point mutations. [34] The predictions can be done with the ddg-monomer application. Given a preminimised crystal structure of the wild-type protein, the application generates a structural models of the point-mutant. The preminimisation reduces collisions that would otherwise introduce large amounts of noise. It is recommended that 50 models of the wild-type and each of the mutant structures would be generated. The change in stability is then calculated as the difference between the mean of the three mutant structures and the three wild-type structures with the best scores. [33]

There are different protocols for the ddg-monomer application that use different assumptions for the predictions. Kellogg et al. [29] compared these different protocols within Rosetta, ranging from an entirely fixed backbone approximation to full-protein flexibility. They concluded that the best correlations between experimental and predicted $\Delta\Delta$Gs as well as the classification accuracies to stabilising and destabilising mutations are achieved either using protocol 3 with fixed backbone and damped repulsive interactions or protocol 16 with flexible backbone and undamped repulsive interactions. [29] The protocol that used flexible backbone and undamped

repulsive interactions provided slightly better results and was therefore selected for the predictions executed here. Even though the backbone is considered flexible, some distance constraints are needed to prevent the backbone from moving too far from the starting conformation [29, 33].

The energy function that Rosetta uses for the stability predictions is a combination of physics-based and knowledge-based potentials. Therefore the energies are not measured in any actual physical energy units and are on an arbitrary scale, sometimes referred to as REU (Rosetta Energy Unit). These units cannot be directly compared to any experimental energies, but they can be converted to physical energy units by finding a line with best fit between the energies calculated by Rosetta and a set of benchmark cases with known experimental energies. [50] The equation of best-fit line obtained by Kellogg et al. [29] using a set of 1210 mutations was $y = 0.57x$.

4.2 Machine learning approaches

Machine learning methods use some set of experimental measurements of stability changes to train a model for the prediction of stability changes upon new mutations. They try to describe these values by different inputs, for example residue or atom features combined with sequence or structural information. Some methods also ensemble predictions from other methods and try to find an optimal combination of them [12, 42]. The machine learning methods try to either classify the stability changes to destabilising and stabilising mutations [9, 12, 19, 59, 63], destabilising, neutral and stabilising mutations [11, 36] or predict the actual change in stability [9, 12, 17, 42, 43, 59, 62, 63]. The most common approach among the current methods is to use support vector machines (SVM) [9, 11, 12, 13, 19, 36, 42], but also random forests [59, 62], Gaussian processes [43] and neural networks [17, 21] have been used to find the relation between the inputs and the training data.

mCSM [43] predicts the stability effects of mutations using Gaussian process regression with structure based signatures. The signatures are created by extracting information of the wild-type residue environment, changes in the pharmacophore counts between the wild-type and mutated residue and the experimental conditions. For a given mutation-site, the wild-type residue environment is defined by the pairwise distances and properties of atoms within a distance r from the geometric center of the mutation-site. A cumulative distribution of counts of atom pairs for all possible combinations of the atom properties is calculated by increasing the allowed distance between the atoms of a pair. These cumulative distributions are then concatenated into a single vector. The pharmacophores are used for capturing the physicochemical changes in atom types due to the mutation. Each amino acid is presented as a vector with counts of atoms with the different physicochemical properties. The difference between the

pharmacophore counts of the wild type and mutant residue is added to the signature. Also the pH and the temperature used in the experimental determination of the stability values are added to the signature, as well as the relative solvent accessibility of the residue to be mutated. The obtained signatures are used to train a Gaussian process regression model using Gaussian kernels.

4.3 Metrics for evaluating the predictions

When the stability changes upon mutations are predicted, the accuracy of the predictions is often assessed by correlation between the predicted and experimentally measured $\Delta\Delta$G-values [10, 17, 29, 43, 45]. The correlation coefficient, or more accurately Pearson product moment coefficient of correlation r, measures the strength of the linear relationship between two random variables $\mathbf{y}_*$ and $\bar{f}(\mathbf{x}_*)$, which in this case are the n experimentally measured and predicted $\Delta\Delta$G-values, respectively:

$$r = \frac{\sum_{i=1}^{n} \left(y_{*i} - \bar{\mathbf{y}}_*\right) \left(\bar{f}(x_{*i}) - \bar{f}(\mathbf{x}_*)\right)}{\sqrt{\sum_{i=1}^{n} \left(y_{*i} - \bar{\mathbf{y}}_*\right)^2 \sum_{i=1}^{n} \left(\bar{f}(x_{*i}) - \bar{f}(\mathbf{x}_*)\right)^2}}. \tag{5}$$

Here $\bar{\mathbf{y}}$ is the mean of the experimentally measured values and $\bar{f}(\mathbf{x}_*)$ is the mean of the predicted values. The value of r is always between -1 and 1. Correlation of $r = 0$ indicates no linear relationship between the experimental and predicted values. Values of r close to -1 or 1 indicate strong linear relationship. If $r = 1$, the predicted values increase as the experimental values increase and if $r = -1$, the predicted values decreases as the experimental values increase. [38]

Another commonly used metric in protein stability prediction is root mean squared error [17, 42, 43]:

$$\text{rmse} = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - \bar{f}(x_{*i})\right)^2}{n}}, \tag{6}$$

where n is the number of predictions, y is the experimentally determined $\Delta\Delta$G-value, and $\bar{f}(x_{*i})$ is the predicted value, i.e. the mean of the Gaussian distribution.

5 Kernel methods

In this section we will discuss ridge regression and how it can be generalised to kernel ridge regression and thus used with more complex data. After that Gaussian processes and their benefits will be introduced. We will also consider multiple kernel learning as it can be used to exploit different amino acid features and different representations of proteins.

5.1 Linear and kernel ridge regression

Suppose we have some data of the form

$$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l) \in \mathcal{X} \times \mathbb{R}, \tag{7}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is some set from which the inputs $\mathbf{x}_i$ are taken. In learning we would like to generalise to unseen data points y. That is, given some new input $\mathbf{x} \in \mathcal{X}$, we

would like to predict the corresponding $y \in \mathbb{R}$, which in our case are the changes in stability upon mutations. [54] Now if there is an approximately linear dependency between the inputs $\mathbf{x} \in \mathcal{X}$ and the corresponding y, we could just find a homogeneous real-valued linear function with a weight vector $\mathbf{w} \in \mathbb{R}^n$ such as

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}^\top \mathbf{x} = \sum_{i=1}^{n} w_i x_i \tag{8}$$

that best interpolates a given training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$. An one dimensional example of this is shown in Figure 5.

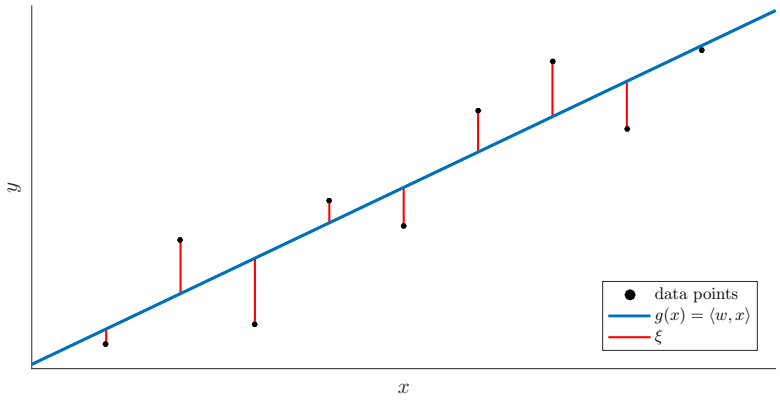


Figure 5: Example of linear regression.

Here ξ is the error of the linear function on a particular training example and it is calculated by the function

$$h((\mathbf{x}, y)) = |y - g(\mathbf{x})| = |y - \langle \mathbf{w}, \mathbf{x} \rangle| = |\xi|. \tag{9}$$

We would like to find a function $g(\mathbf{x})$ for which all of the training errors $|\xi|$ are small. For this we can define a loss function $\mathcal{L}$, whose value we try to minimise. A commonly used method is the least squares approximation, where the used loss function is the sum of the squares of the training errors, as defined by Equation 10. [56]

$$\mathcal{L}(g, S) = \mathcal{L}(\mathbf{w}, S) = \sum_{i=1}^{l} (y_i - g(\mathbf{x}_i))^2 = \sum_{i=1}^{l} \xi_i^2 \tag{10}$$

The learning problem has now become that of choosing $\mathbf{w} \in \mathbf{W}$ that minimises the collective loss. Using a notation where $\mathbf{X}$ denotes the matrix whose rows are the row vectors $\mathbf{x}_1^\top, \ldots, \mathbf{x}_l^\top$ and $\mathbf{y}$ denotes the vector $(y_1, \ldots, y_l)^\top$, we can write the errors as a vector $\xi = \mathbf{y} - \mathbf{X}\mathbf{w}$ and the loss function as:

$$\mathcal{L}(\mathbf{w}, S) = \|\xi\|_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}). \tag{11}$$

Now the optimal $\mathbf{w}$ can be sought by taking the derivatives of the loss function with respect to the parameters $\mathbf{w}$ and setting them equal to the zero vector

$$\frac{\partial \mathcal{L}(\mathbf{w}, S)}{\partial \mathbf{w}} = -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{0}, \tag{12}$$

which gives us the so called normal equations:

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}. \tag{13}$$

The second derivative of the loss function in Equation 11 is $\frac{\partial^2}{\partial \mathbf{w} \partial \mathbf{w}^\top} \mathcal{L}(\mathbf{w}, S) = 2\mathbf{X}^\top \mathbf{X} \geq 0$ as the square of $\mathbf{X} = \mathbf{X}^\top \mathbf{X}$ must be positive. Therefore according to the second derivative test, Equation 12 indeed leads to the minimum of the loss function [1]. If the inverse of $\mathbf{X}^\top \mathbf{X}$ exists, the solution of the least squares problem can be presented as

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{14}$$

and the predicted output on a new data point can be calculated with the prediction function $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$. [56]

When there is not enough information in the data to precisely specify the solution, as either there is not enough data to ensure that the matrix $\mathbf{X}^\top \mathbf{X}$ is invertible or there is too much noise in the data, one solution is to use regularisation and to restrict the choice of functions in some way. One simple regulariser is to favor functions that have smaller norms, that is, we can trade off the size of the norm against the loss. For the case of least squares regression, this gives the optimisation criterion of ridge regression, which corresponds to solving the optimisation

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{l} \left(y_i - g(\mathbf{x}_i)\right)^2. \tag{15}$$

Here the relative trade-off between norm and loss is defined by a positive number λ, which thus controls the degree of regularisation. With this formulation, the learning problem is reduced to solving an optimisation problem over $\mathbb{R}^n$. If we now take the derivative of this loss function with respect to the parameters, we obtain the equations

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n\right) \mathbf{w} = \mathbf{X}^\top \mathbf{y}, \tag{16}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix. In this case the matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n$ is always

invertible if $\lambda > 0$ and the solution can be given by

$$\mathbf{w} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n\right)^{-1} \mathbf{X}^\top \mathbf{y}, \tag{17}$$

which gives us the prediction function

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{y}^\top \mathbf{X} \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n\right)^{-1} \mathbf{x}. \tag{18}$$

This formulation of ridge regression can be used to identify linear relations between one selected variable y and the remaining features $\mathbf{x}$, where the relation is assumed to be functional. Often however, the sought relations are nonlinear and cannot accurately be estimated with a linear function, which is also the case with the data that we use with the protein stability predictions. To overcome this issue, we can formulate the primal solution in Equation 17 again in a different form known as the dual solution. If we write the Equation 16 in terms of $\mathbf{w}$, we obtain

$$\mathbf{w} = \lambda^{-1} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}^\top \alpha, \tag{19}$$

where $\alpha = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$ are the dual variables. Now with these dual variables α, $\mathbf{w}$ can be written as a linear combination of the training points, $\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i$ and we get the dual solution that is presented in Equation 23:

$$\begin{aligned}
& \alpha = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}) && (20)\\
\Rightarrow \quad & \lambda\alpha = \left(\mathbf{y} - \mathbf{X}\mathbf{X}^\top \alpha\right) && (21)\\
\Rightarrow \quad & \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_l\right) \alpha = \mathbf{y} && (22)\\
\Rightarrow \quad & \alpha = \left(\mathbf{K} + \lambda \mathbf{I}_l\right)^{-1} \mathbf{y}. && (23)
\end{aligned}$$

Here $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$, or component-wisely $\mathbf{K}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is referred to as the Gram matrix or kernel matrix. With $\mathbf{K}$, the prediction function can be given by

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{i=1}^{l} \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{l} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle = \mathbf{k}^\top (\mathbf{K} + \lambda \mathbf{I}_l)^{-1} \mathbf{y}, \tag{24}$$

where $k_i = \langle \mathbf{x}_i, \mathbf{x} \rangle$. The advantage of this formulation is that the predictive function in Equation 24 is now expressed in a form that requires only the inner products between the data points, which are given in the kernel matrix $\mathbf{K}$. [56]

As the predictive function presented in Equation 24 only requires the inner products between the data points, the features $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ can be mapped into a new feature space in such a way that the sought relations can be presented in a linear form,

allowing us to use the ridge regression algorithm to detect them. That is, we try to recode our data set S as $\hat{S} = \{(\phi(\mathbf{x}_1), y_1), \ldots, (\phi(\mathbf{x}_l), y_l)\}$, with an embedding map

$$\phi : \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \mapsto \phi(\mathbf{x}) \in \mathcal{F} \subseteq \mathbb{R}^N, \tag{25}$$

which allows us to look for a relation of the form

$$h((\mathbf{x}, y)) = |y - g(\mathbf{x})| = |y - \langle \mathbf{w}, \phi(\mathbf{x}) \rangle| = |\xi|. \tag{26}$$

This form of ridge regression, where the regularisation is done in the feature space rather than simply in the input space, can be called kernel ridge regression [47]. Now the predictive function in Equation 24 involves the kernel matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ with entries

$$\mathbf{K}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle , \tag{27}$$

where the matrix $\mathbf{X}$ consists of the feature vectors $\phi(\mathbf{x}_1)^\top, \ldots, \phi(\mathbf{x}_l)^\top$, and the vector $\mathbf{k}$ contains the values $k_i = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$. Sometimes the inner products can be computed as a direct function of the input features, without explicitly computing the mapping ϕ. This direct computation can be performed with a kernel function

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle , \tag{28}$$

where $\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) \in \mathcal{F}$ is the mapping from $\mathcal{X}$ to a feature space $\mathcal{F}$. [56] That is, given the two inputs $\mathbf{x}$ and $\mathbf{z} \in \mathcal{X}$, the kernel returns a real number characterising their similarity. This can also be represented as a $n \times n$ kernel matrix $\mathbf{K}$ of κ as

$$\mathbf{K}_{ij} := \kappa(\mathbf{x}_i, \mathbf{x}_j), \tag{29}$$

where $i, j = 1, 2, \ldots, n$ and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathcal{X}$. [54]

5.2 Gaussian processes

Gaussian processes provide a probabilistic approach to learning with kernels. Rather than restricting the class of functions to consider, a prior probability is assigned to every possible function. This prior represents the prior beliefs over the kinds of functions we expect to observe before seeing any data. If we are then given a set of training data, we may wish to only consider functions that go through or near these data points. The uncertainty of the underlying function is reduced close to these data points, which is illustrated by Figure 6: there is a continuous mean prediction

for the unseen data points, but there is more variance in the areas with no observed data points. This combination of the prior information and the data leads to the posterior distribution over functions. [47]

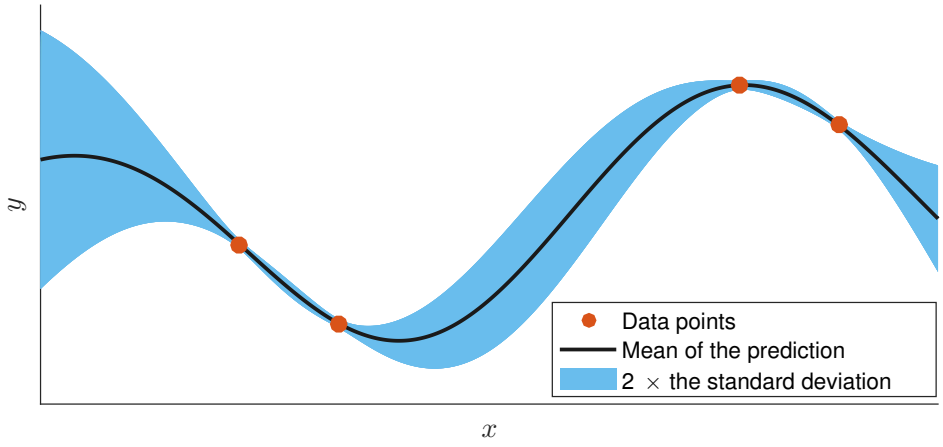


Figure 6: A Gaussian process over four data points. The uncertainty of the underlying function is reduced close to the data points.

A Gaussian process can be defined as a collection of random variables, any finite number of which have a joint Gaussian distribution, completely specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{30}$$

This implies a consistency requirement, also known as marginalisation property. Therefore if the Gaussian process specifies $(y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)$, it must also specify $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$. The mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of the real process $\mathbf{f}(\mathbf{x})$ are determined as follows:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{31}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{32}$$

The mean function is often considered to be zero, $m(\mathbf{x}) = \mathbf{0}$ for simplicity, but different functions can be used as well. [47]

For the prediction of unseen data points, we need to formulate the Gaussian processes in terms of the inputs corresponding to the training and test points. If the observations do not contain noise and we thus have $\{(\mathbf{x}_i, f_i)|i = 1, \ldots, n\}$, the joint distribution of the training outputs $\mathbf{f}$ and the test outputs $\mathbf{f}_*$ according to the prior is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) & \mathbf{K}(\mathbf{X},\mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}) & \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right). \tag{33}$$

When there are n training points and n_* test points, $\mathbf{K}(\mathbf{X},\mathbf{X}_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated at all pairs of training and test points. To get the posterior distribution over functions, the joint prior distribution is conditioned on the observations as [47]

$$\mathbf{f}_*|\mathbf{X}_*,\mathbf{X},\mathbf{f} \sim \mathcal{N}\Big(\mathbf{K}(\mathbf{X}_*,\mathbf{X})\mathbf{K}(\mathbf{X},\mathbf{X})^{-1}\mathbf{f}, \\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) - \mathbf{K}(\mathbf{X}_*,\mathbf{X})\mathbf{K}(\mathbf{X},\mathbf{X})^{-1}\mathbf{K}(\mathbf{X},\mathbf{X}_*)\Big). \tag{34}$$

When the observations do contain noise, we have $y = f(\mathbf{x}) + \epsilon$. If the noise is assumed to be independent and identically distributed Gaussian noise ϵ, with variance σ_n^2 the prior on the noisy observations becomes

$$\mathrm{cov}(\mathbf{y}) = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}, \tag{35}$$

and the joint distribution of the observed target values and the function values at the test locations under the prior become

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{K}(\mathbf{X},\mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}) & \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right). \tag{36}$$

Now when the conditional distribution is derived, we get the following predictive distribution and key predictive equations for Gaussian process regression:

$$\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{X}_* \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \mathrm{cov}(\mathbf{f}_*)\right), \text{ where} \tag{37}$$

$$\bar{\mathbf{f}}_* = \mathbb{E}[\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{X}_*] = \mathbf{K}(\mathbf{X}_*,\mathbf{X})\left(\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}, \tag{38}$$

$$\mathrm{cov}(\mathbf{f}_*) = \mathbf{K}(\mathbf{X}_*,\mathbf{X}) - \mathbf{K}(\mathbf{X}_*,\mathbf{X})\left(\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{K}(\mathbf{X},\mathbf{X}_*). \tag{39}$$

For a single test point $\mathbf{x}_*$ the mean and variance of the prediction can be more compactly denoted as

$$\bar{f}_* = \mathbf{k}_*^\top\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} \tag{40}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*, \tag{41}$$

where $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$. [47]

The advantage of using Gaussian processes instead of the regularised approach is that it can characterise the uncertainty in the predictions and handle multimodality in the posterior. Also, computation of the marginal likelihood can be very useful for setting parameters of the covariance function and for model comparison as well. Often the computations required for calculating the marginal likelihood are analytically intractable, but Gaussian process regression models with Gaussian noise are an exception as their integrals over the parameters are analytically tractable. The marginal likelihood, or evidence, $p(\mathbf{y}, \mathbf{X})$ is the integral of the likelihood times the prior

$$p(\mathbf{y}|\mathbf{X}, \theta) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}, \theta) p(\mathbf{f}|\mathbf{X}, \theta) \, \mathrm{d}\mathbf{f} \tag{42}$$

and the marginalisation is done over the function values $\mathbf{f}$. Under the Gaussian process model the prior is Gaussian and $\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ or

$$\log p(\mathbf{f}|\mathbf{X}) = -\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log 2\pi. \tag{43}$$

The likelihood is a factorised Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2\mathbf{I})$. The product of two Gaussians is another Gaussian:

$$\mathcal{N}(\mathbf{x}|\mathbf{a}, A)\mathcal{N}(\mathbf{x}|\mathbf{b}, B) = Z^{-1}\mathcal{N}(\mathbf{x}|\mathbf{c}, C), \tag{44}$$

where $\mathbf{c} = C(A^{-1}\mathbf{a} + B^{-1}\mathbf{b})$, $C = (A^{-1} + B^{-1})^{-1}$, the normalising constant is $Z^{-1} = (2\pi)^{-n/2} \, |A + B|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{a} - \mathbf{b})^\top (A + B)^\top (\mathbf{a} - \mathbf{b})\right)$ and n is the dimension of $\mathbf{x}$ and the covariance matrices are of size $n \times n$. Therefore after performing the integration of Equation 42, we get the log marginal likelihood

$$\log p(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^\top \left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} - \frac{1}{2}\log\left|\mathbf{K} + \sigma_n^2\mathbf{I}\right| - \frac{n}{2}\log 2\pi. \tag{45}$$

The first part of the equation, $\mathbf{y}^\top \left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}/2$, estimates the fit of the data regarding the observed targets $\mathbf{y}$, the second term $\log|\mathbf{K} + \sigma_n^2\mathbf{I}|/2$ is the complexity penalty that depends only on the covariance function and the inputs and the third term $n\log(2\pi)/2$ is a normalisation constant. [47]

The selection of the covariance function and its parameters is considered as training of a Gaussian process. The training can be done by maximising the marginal likelihood using gradients. For their calculation, we need the partial derivatives of the marginal likelihood with respect to the hyperparameters θ. They are obtained from Equation 45 and the derivative rules of the elements of an inverse matrix $\left(\frac{\partial}{\partial\theta}K^{-1} = -K^{-1}\frac{\partial K}{\partial\theta}K^{-1}\right)$ and of the log determinant of a positive definite symmetric matrix $\left(\frac{\partial}{\partial\theta}\log|K| = \mathrm{tr}\left(K^{-1}\frac{\partial K}{\partial\theta}\right)\right)$ as follows:

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|\mathbf{X}, \theta) = \frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_j} \mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \quad (46)$$

$$= \frac{1}{2}\mathrm{tr}\left(\left(\alpha\alpha^\top - \mathbf{K}^{-1}\right)\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \quad \text{where } \alpha = \mathbf{K}^{-1}\mathbf{y}. \quad (47)$$

The marginal likelihood with respect to the hyperparameters θ may have multiple local optima. Usually this should not be an unreasonable problem as every local maximum corresponds to a particular interpretation of the data. For example a complicated model with low noise may produce a local optima as well as a simpler model with more noise. When enough data is available it should become distinctive which optima is the most probable. [47] In order reach the global optima it is important to choose the initial values carefully or to experiment with different values when optimising the marginal likelihood with gradients.

5.3 Kernels

As already mentioned in Section 5.1, a kernel function $\kappa(\mathbf{x}, \mathbf{z})$ returns a real number characterising the similarity of the two inputs $\mathbf{x}$ and $\mathbf{z} \in \mathcal{X}$. They can thus be used to calculate the inner products of the feature vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{z}) \in \mathcal{F}$ as a direct function of the input features $\mathbf{x}$ and $\mathbf{z}$ without explicitly computing the mapping $\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) \in \mathcal{F}$ to the feature space $\mathcal{F}$. [56]

A property of kernel matrices is that they are positive semi-definite. This means that for kernel matrix $\mathbf{K}$ it holds that

$$\mathbf{v}^\top \mathbf{K}\mathbf{v} \geq 0 \quad (48)$$

for all vectors $\mathbf{v}$, which also implies that the kernel matrix must be symmetric, i.e. $\mathbf{K} = \mathbf{K}^\top$ and its eigenvalues are non-negative. Similarly, kernel matrix $\mathbf{K}$ is positive definite if $\mathbf{v}^\top \mathbf{K}\mathbf{v} > 0$ for all $\mathbf{v} \neq \mathbf{0}$, or equivalently if it is symmetric and its eigenvalues are positive. [56]

The most widely used kernels are squared exponential kernels, which are also known as Gaussian kernels. For $l > 0$, a Gaussian kernel is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = e^{-(\mathbf{x}-\mathbf{z})^2/2l^2}. \quad (49)$$

The parameter l controls the flexibility of the kernel: small values make the kernel more flexible whereas large values of l gradually reduce the kernel to a constant function. [56] Sometimes the parameter l is called the length scale of the kernel. The

Gaussian kernel is already in a normalised form as $\kappa(\mathbf{x}, \mathbf{z}) = 1$ if $\mathbf{x} = \mathbf{z}$, but for some other kernels this is not the case. Then normalisation, as defined below in Equation 50, can be used to ensure that $\kappa_{\mathrm{n}}(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x}$. [47]

$$\kappa_{\mathrm{n}}(\mathbf{x}, \mathbf{z}) = \frac{\kappa(\mathbf{x}, \mathbf{z})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x})\kappa(\mathbf{z}, \mathbf{z})}} \tag{50}$$

Kernels can also be defined between graphs [61]. A graph $G = (V_G, E_G)$ consists of a set of vertices V_G and a set of edges E_G. Each edge has a set of one or two vertices associated to it, which are called its endpoints that are joined by this edge and may be called neighbours. A multi-edge is a collection of two or more edges that have identical endpoints and a self-loop is an edge that joins a single endpoint to itself. Simple graphs are graphs that have no self-loops or multi-edges. A subgraph of G is a graph g such that $V_g \subset V_G$ and $E_g \subset E_G$. [23] When we consider proteins as graphs were residues are the vertices and the contacts between residues are the edges, it is enough to consider simple graphs.

When a kernel is defined between two graphs, the challenge is to capture the semantics inherent in the graph structure in a way that is reasonably efficient to evaluate [61]. Graph kernels have been categorised into three classes: graph kernels based on walks and paths, graph kernels based on limited-size subgraphs called graphlets and graph kernels based on subtree patterns. A walk is a sequence of consecutive nodes in a graph and a path is a walk with distinct nodes. A subtree is a subgraph of a graph, which has no cycles, but a designated root node. Subtree patterns extent the notion of subtrees by allowing repetition of nodes in a pattern, which makes it possible to model also subgraphs with cycles as subtrees. The construction of subtree patterns is illustrated in Figure 7. [58] However, these graph kernels are only defined between graphs with discrete labels. As we would like to exploit different amino acid features that are represented with continuous labels, we need a more flexible kernel.

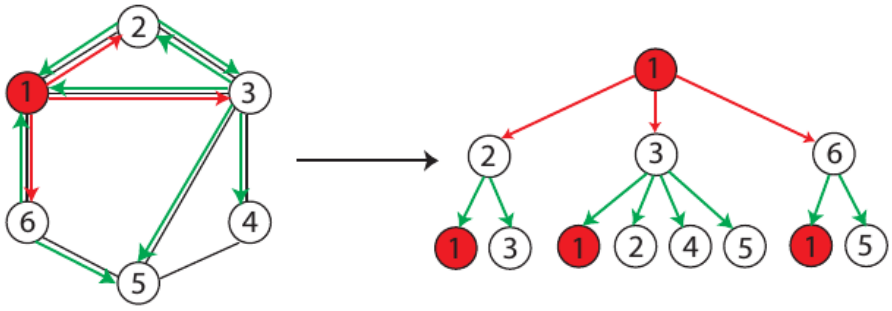


Figure 7: A subtree pattern of height two rooted at the node 1. The figure obtained from [58].

Weighted decomposition kernel (WDK) is a graph kernel that compares small substructures of graphs called selectors and matches them according to an equality predicate. The matches between the selectors are weighted by the similarity of their contexts, which are also subgraphs of the graphs. For example, the selectors could be single vertices and their contexts the neighbours of these vertices. WDK can be characterised as

$$\mathcal{R} = \langle \vec{G}, R, (\delta, \kappa_1, \ldots, \kappa_D) \rangle, \tag{51}$$

where $\vec{G} = (S, Z_1, \ldots, Z_D)$ is a $(1+D)$-tuple of non-empty subsets of subgraphs of G. $R(s, z_1, \ldots, z_D, G)$ is a finite parthood relation on $S \times Z_1 \times \cdots \times Z_D \times G$, which is true if and only if the selector $s \in S$ is a subgraph of graph G and $\vec{z} = (z_1, \ldots, z_D) \in Z_1 \times \cdots \times Z_D$ is a tuple of subgraphs of G called contexts of occurrences of s in G. δ is an exact matching kernel on $S \times S$ and $\kappa_d, d = 1, \ldots, D$ is a graph probability distribution kernel on $Z_d \times Z_d$. The corresponding kernel in a general form can be determined as

$$K(G, G') = \sum_{\substack{s,\, \vec{z} \in R^{-1}(G) \\ s',\, \vec{z}' \in R^{-1}(G')}} \delta(s, s') \sum_{d=1}^{D} \kappa_d(z_d, z'_d). \tag{52}$$

Figure 8 illustrates how substructures are compared in a weighted decomposition kernel, when the selectors s are single nodes that are matched if they are of the same colour and the context z of a selector s consists of the neighbours of that selector. [37]

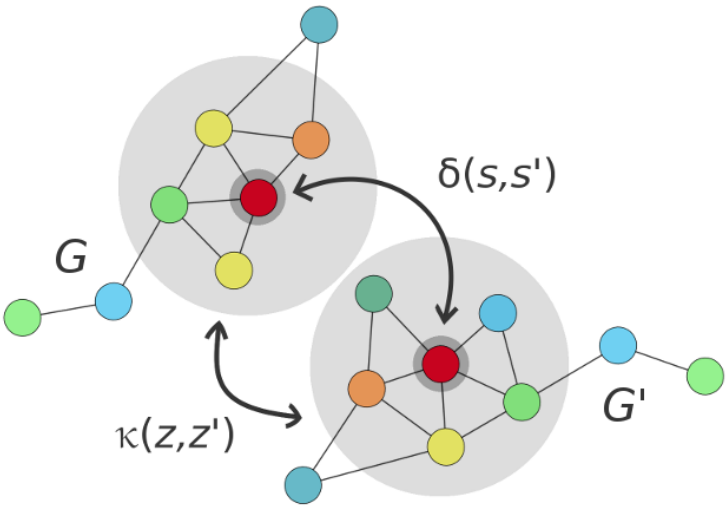


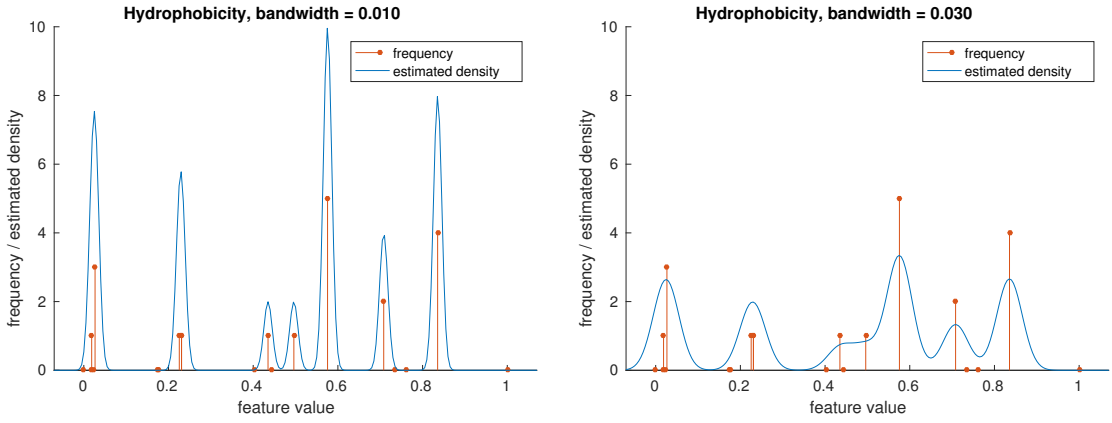
Figure 8: Comparison of substructures with a weighted decomposition kernel. The selectors s and s' are single nodes and contexts z and z' their neighbourhoods in graphs G and G', respectively.

In our case the selectors are residues that are matched either based on their position in the amino acid sequence or by the locations of mutations, and the context of a residue is its neighbourhood. If two neighbourhoods have the same structure, that is, the amount of residues in the neighbourhoods is the same and the edges between them are identical, but their amino acid labels may differ, they can be compared with a substitution matrix. This can be done by defining the kernel between these contexts as a sum of substitution matrix values from each of the residues in the neighbourhoods who are in the same position. However, if the neighbourhoods have different structures, a more general approach is needed to compare them.

A context of a residue can be described as a probability distribution, that is fitted to the residue neighbourhood based on the frequencies of the different amino acid feature values of the residues in that neighbourhood. This way we can compare the similarities of these probability distributions rather than the frequencies of discrete feature values and exploit the way that probability distributions capture similarity: if a distribution fit to one data point gives high likelihood to another data point, this should indicate that the two data points are similar. The graph probability distribution kernel, or the probability product kernel κ is designed to capture these similarities. It is defined as an integral of the product between a pair of probability distributions. In its special form, called Bhattacharyya kernel, a square root is taken from the product to insure the normalisation property $\kappa(z, z) = 1$, resulting to

$$\kappa(z, z') = \int \sqrt{p_{fz}(x) p_{fz'}(x)}\, \mathrm{d}x. \tag{53}$$

Here p_{fz} is the the estimated density of value x at $\rho_f(x)$, given an attribute ρ_f in subgraph z. [37, 26] With this formulation we can exploit the continuously labeled features f. Figure 9 shows how the density of hydrophobicity values (scaled between 0 and 1) of the amino acids in the neighbourhood of residue 27 in bacteriophage T4 lysozyme is estimated. A Gaussian distribution is located at each feature value, so that the mean of the Gaussian is centered at the feature value x. Since we have 20 different amino acids, we get up to 20 Gaussian distributions $\mathcal{N}(x_i, \sigma^2)$ that we sum up to form the combined estimated density. The variance of these Gaussian distributions affects how close the feature values have to be for them to be considered similar. Figure 9 illustrates this effect with two different bandwidths. In the case of Gaussian distributions, the bandwidth corresponds to the standard deviation σ [57].



(a) (b)

Figure 9: The estimated probability distribution of hydrophobicity in the neighbourhood of residue 27 in bacteriophage T4 lysozyme. Figures (a) and (b) use different bandwidths.

5.4 Multiple kernel learning

Simple kernels and kernel matrices can be manipulated and combined using certain operations in order to obtain more complex and useful kernels. This class of kernel functions preserves the finitely positive semidefiniteness 'kernel' property and is thus closed under such operations. If we have kernels $\kappa_1(\mathbf{x}, \mathbf{z})$ and $\kappa_2(\mathbf{x}, \mathbf{z})$ over $\mathcal{X} \times \mathcal{X}$, $\mathcal{X} \subset \mathbb{R}^n$, $a \in \mathbb{R}^+$, $s \in \mathbb{N}$, the operations include the following functions: [56]

1. $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$
2. $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$
3. $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$
4. $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})^s$

Multiple kernel learning (MKL) methods exploit these operations to combine multiple kernels that may correspond to using different notions of similarity or use information from multiple sources. Most MKL algorithms try to find an optimal combination of predefined kernels, but they can also be used to simultaneously search for optimal parameters for the kernels. There are different ways for choosing the optimal combination of the base kernels and the MKL methods can be categorised based on their learning method, functional form, target function and training method. [22]

The learning method determines the combination function that is used to combine the base kernels. If *fixed rules* are used, the combining function does not have any parameters and thus does not require any training. For example summation and multiplication of the kernels are considered as fixed rules. *Heuristic approaches* use a

parametrised combination function and find the parameters of this function, generally with some measure obtained from each kernel function separately. *Optimisation approaches* use parametrised combination functions as well, but they learn the parameters by solving an optimisation problem. *Bayesian approaches* assign priors to the parameters of the combination function and perform inference for learning them and the base kernel parameters. *Boosting approaches* add new base kernels to the combined learner until its performance stops improving. [22]

The functional form defines how the combination is done. The most popular methods are the *linear combination* methods. They can be unweighted or weighted sums of the base kernels. When a weighted sum is used, the combination function can be linearly parametrised as $\mathbf{K} = \sum_{i=1}^{m} \mu_i \mathbf{K}_i$ and the weights μ_i can be given different restrictions. *Nonlinear combination* methods use nonlinear functions for combining the base kernels, normally multiplication, power and exponentiation. *Data-dependent combination* methods assign specific weights for each data instance. This allows them to identify local distributions of the data and learn proper kernel combination rules for each region. [22]

The target function specifies the metric that is used for optimising the parameters of the combination function. *Similarity-based functions* calculate a similarity metric between the combined kernel matrix and an optimal kernel matrix calculated from the training data and search for the parameters of the combination function that maximise the similarity. *Structural risk functions* try to minimise the loss function, that in the case of ridge regression is presented by Equation 15. The restrictions on kernel weights can be integrated into the regularisation term. *Bayesian functions* estimate the quality of the combined kernel function using Bayesian formulation. Generally the likelihood or the posterior is used as the target function and the model parameters are selected using the maximum likelihood estimate or the maximum a posteriori estimate. [22]

Finally, the MKL methods can be classified to *one-step methods* and *two-step methods*. One-step methods find the combination function parameters and the parameters of the base kernels in a single pass, either sequentially or simultaneously. In the sequential approach the parameters of the combination function are determined first and after that the parameters of the base kernels. In the simultaneous approach both sets of parameters are learned together. Two-step methods first update the combination function parameters while fixing the base learner parameters and then update the base learner parameters while fixing the combination function parameters. These steps are repeated iteratively until convergence. [22]

With the optimisation approach to MKL we can use another Bayesian function as the target function, namely marginal likelihood and maximise it with respect to the combination function parameters and the base kernel parameters. With the

Gaussian processes the marginal likelihood can be calculated as shown in Equation 45 maximised with the help of gradients. For that we need the partial derivatives of the marginal likelihood with respect to the parameters θ, which can be calculated by Equation 47. For example, if we use a weight linear combination function

$$\mathbf{K} = \sum_{i=1}^{m} \mu_i \mathbf{K}_i^{\gamma_i}, \tag{54}$$

that is constructed of m base kernel matrices $\mathbf{K}_i$, we can differentiate the kernel matrix with respect to the weights μ and exponents γ as follows:

$$\frac{\partial \mathbf{K}}{\partial \mu_i} = \mathbf{K}_i^{\gamma_i} \tag{55}$$

$$\frac{\partial \mathbf{K}}{\partial \gamma_i} = \mu_i \ln \mathbf{K}_i^{\gamma_i} \odot \mathbf{K}_i^{\gamma_i}. \tag{56}$$

In Equation 56 $\odot$ denotes element-wise product of the matrices $\ln \mathbf{K}_i^{\gamma_i}$ and $\mathbf{K}_i^{\gamma_i}$.

Here we use a quasi-Newton strategy, where limited-memory BFGS updates are used in computing the step directions. The L-BFGS algorithm is suitable for the minimisation of a smooth nonlinear function $f : \mathbb{R}^n \to \mathbb{R}$ in the case where the number of variables n is large, and where analytic expressions for the function f and the gradient are available [35]. In Section 6 we will use a function that implements a two-metric projection method provided by [52] and in Section 7, where we have fewer kernel matrices, we will use Matlab's implementation for interior point method.

Alignf is a multiple kernel alignment algorithm proposed by [15], which finds an optimal combination of kernels by kernel alignment maximisation. Kernel-alignment is a measure of similarity between two kernel functions or between a kernel and a target kernel [16]. Alignf is thus a MKL method, that uses an optimisation approach to find an optimal linear combination of the kernels by using a similarity based target function.

The alignment between kernels K and K' exploits centered kernels. If the training and test points are drawn from distribution D, we can center the feature mapping $\phi : \mathcal{X} \mapsto \mathcal{F}$ by replacing it by $\phi - \mathrm{E}_{\mathbf{x}}[\phi]$, where $\mathrm{E}_{\mathbf{x}}$ denotes the expected value of ϕ when $\mathbf{x}$ is drawn from the distribution D. When a positive definite symmetric (PDS) kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is centered, any feature mapping ϕ associated with K needs to be centered. The centered kernel K_c associated to K is defined for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ by

$$K_c(\mathbf{x}, \mathbf{x}') = (\phi(\mathbf{x}) - \mathrm{E}_{\mathbf{x}}[\phi])^\top (\phi(\mathbf{x}') - \mathrm{E}_{\mathbf{x}'}[\phi]) \quad (57)$$

$$= K(\mathbf{x}, \mathbf{x}') - \mathrm{E}_{\mathbf{x}}[K(\mathbf{x}, \mathbf{x}')] - \mathrm{E}_{\mathbf{x}'}[K(\mathbf{x}, \mathbf{x}')] + \mathrm{E}_{\mathbf{x},\mathbf{x}'}[K(\mathbf{x}, \mathbf{x}')]\,. \quad (58)$$

The alignment between kernels K and K' with respect to the sample $S = \{x_1, \ldots, x_m\}$ is then defined as

$$\hat{\rho}(\mathbf{K}_c, \mathbf{K}'_c) = \frac{\langle \mathbf{K}_c, \mathbf{K}'_c \rangle_F}{\|\mathbf{K}_c\|_F \|\mathbf{K}'_c\|_F}, \quad (59)$$

where $\langle \mathbf{K}_c, \mathbf{K}'_c \rangle_F = \sum_{i,j=1}^m K_c(x_i, x_j) K'_c(x_i, x_j)$ is the Forbenius product between the centered kernel matrices and $\|\mathbf{K}_c\|_F$ is the Frobenius norm of the centered kernel matrix [15]. The alignment maximisation method takes correlations between the base kernel matrices into account and maximises the alignment between the convex combination kernel $\mathbf{K}_\mu = \sum_{k=1}^p \mu_k \mathbf{K}_k$ and the target kernel $\mathbf{K}_y = \mathbf{y}\mathbf{y}^\top$ by determining the mixture weights μ_k. With the set $\mathcal{M}' = \{\|\mu\|_2 = 1 \wedge \mu \geq \mathbf{0}\}$ the alignment maximisation problem is defined as

$$\mu^* = \arg\max_{\mu \in \mathcal{M}'} \frac{\mu^\top \mathbf{a}\mathbf{a}^\top \mu}{\mu^\top \mathbf{M} \mu}. \quad (60)$$

The matrix $\mathbf{M}$ is defined by $\mathbf{M}_{kl} = \langle \mathbf{K}_{kc}, \mathbf{K}_{lc} \rangle_F$, for $k, l \in \{1, \ldots, p\}$ and vector $\mathbf{a} = (\langle \mathbf{K}_{1c}, \mathbf{y}\mathbf{y}^\top \rangle_F, \ldots, \langle \mathbf{K}_{pc}, \mathbf{y}\mathbf{y}^\top \rangle_F)^\top$. The same can also be presented as a quadratic programming (QP) minimisation problem:

$$\min_{\mathbf{v} \geq \mathbf{0}} \mathbf{v}^\top \mathbf{M} \mathbf{v} - 2\mathbf{v}^\top \mathbf{a} \quad (61)$$

The solution μ^* of the alignment maximisation problem 60 is then $\mu^* = \frac{\mathbf{v}^*}{\|\mathbf{v}^*\|}$. [15]

Uniformly weight sum of the kernels can be thought as a special case of MKL, where fixed rules are used for the linear combination of the base kernels and thus no learning is actually required. This can be used as a base line comparison when evaluating other MKL methods. Here the mean of the base kernels is used for this purpose.

6 Predicting stability changes upon mutations within a single protein

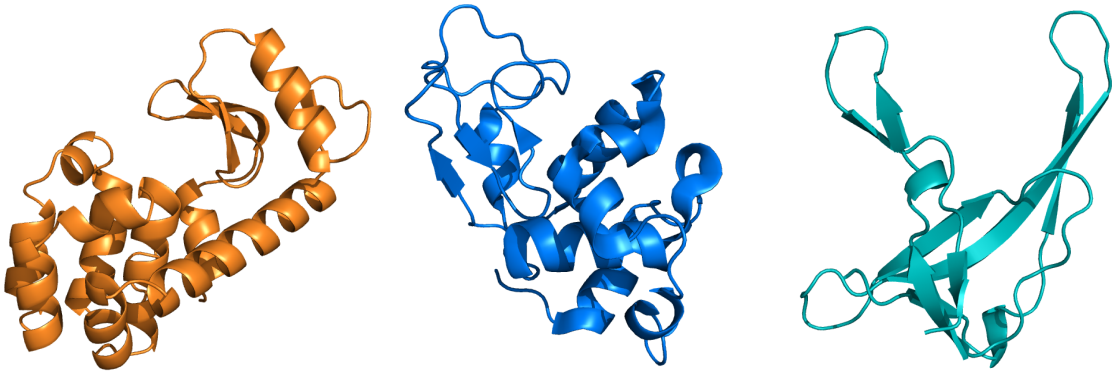
In this section we will describe the data and the model formulation used for the prediction of stability changes within a single protein when single or multiple mutations are introduced. The obtained results will also be presented.

6.1 Stability data sets

Three different proteins were selected for the prediction of changes in stability upon mutations within a single protein: bacteriophage T4 lysozyme, human lysozyme and enterobacteria phage m13 gene V protein. Lysozymes are enzymes that catalyse the cutting of polysaccharide chains in the cell walls of bacteria [2]. Enterobacteria phage m13 gene V protein is a DNA binding protein.

These proteins were selected because there exist several experimentally determined stability measurements from them in the Protherm database, both for single and multiple simultaneous mutations. They also have high resolution structures determined with X-ray crystallography, which are available from PDB. Only stability measurements obtained using thermal denaturation were used. There exist measurements for 349, 130 and 124 different mutations for bacteriophage T4 lysozyme, human lysozyme and Enterobacteria phage m13 gene V protein, respectively. An averaged value was assigned to each mutation with multiple measurements. These three data sets are presented in appendix A.

The secondary structures of these three proteins are presented in Figure 10. Bacteriophage T4 lysozyme (PDB ID 2LZM) has a polypeptide chain that consists of 164 residues, the human lysozyme (PDB ID 2NWD) consists of 130 residues and enterobacteria phage m13 gene V protein (PDB ID 1VQB) consists of 87 residues.

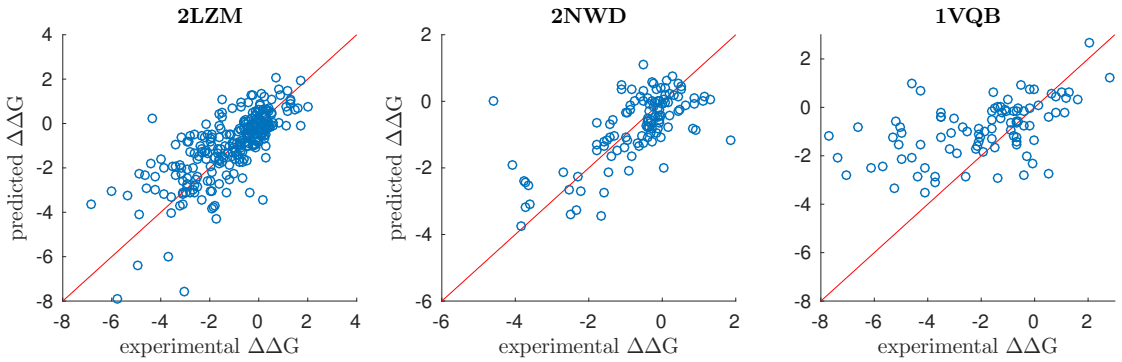


(a) (b) (c)

Figure 10: Secondary structures of (a) 2LZM, (b) 2NWD and (c) 1VQB created with PyMol [55]

For each of these proteins additional stability data y_{RREU} was generated using Rosetta [34]. This was done using protocol 16 described by [29]. This protocol uses flexible backbone and undamped repulsive interactions. Although the backbone is flexible, constraints were defined to ensure that the mutations do not cause too extreme changes to the conformation of proteins, as this would introduce unnecessary noise to the predictions. The same constraints were also applied here, so that C_α-atoms that are within 9 Ångstöms of each other in the wild type structure are not allowed to move more than 0.5 Ångstöms respect to each other. The wild-type structures of the proteins were also preminimised to reduce collisions. The additional data contained all possible single amino acid mutations for each of the proteins, that is $19 \cdot 164 = 3116$ mutations for 2LZM, $19 \cdot 130 = 2470$ mutations for 2NWD and $19 \cdot 87 = 1653$ mutations for 1VQB. The obtained data was scaled to physical units using the equation $y_{\mathrm{RP}} = 0.57 y_{\mathrm{RREU}}$ suggested by [29]. Figure 11 shows the

correlation of the predicted and experimental ΔΔG-values between the mutations that were in both the Rosetta generated and experimental data.



(a) r=0.73, rmse=1.12$\frac{\text{kcal}}{\text{mol}}$ (b) r=0.68, rmse=2.51$\frac{\text{kcal}}{\text{mol}}$ (c) r=0.52, rmse=2.19$\frac{\text{kcal}}{\text{mol}}$

Figure 11: Correlation between the Rosetta generated and experimental stability data with (a) Bacteriophage T4 lysozyme (PDB ID 2LZM) (b) Human lysozyme (PDB ID 2NWD) and (c) Enterobacteria phage m13 gene V protein (PDB ID 1VQB)

The data generated with Rosetta contained some extremely destabilising values. This might be due to the constraints described above, if they do not permit enough backbone flexibility for these mutations. However, a large fraction of mutations causes very little backbone movement, so overall using constraints for the backbone movement should provide better results [29]. To handle the extreme values in the Rosetta generated data, a data transformation $y_{\text{R}} = t(y_{\text{RP}})$ was done for the data used for the training of the Gaussian process regression model. The transformation $t(y_{\text{RP}})$ was defined as

$$t(y_{\text{RP}}) = \begin{cases} y_{\text{RP}} & \text{if } \; y_{\text{RP}} > 7 \text{ kcal/mol} \\ 8\sqrt[5]{y_{\text{RP}}/8} & \text{if } \; y_{\text{RP}} < -20 \text{ kcal/mol} \\ P(y_{\text{RP}}) & \text{if } \; -20 \text{ kcal/mol} \leq y_{\text{RP}} \leq 7 \text{ kcal/mol}, \end{cases} \tag{62}$$

where $P(y_{\text{RP}})$ is a piecewise cubic hermite interpolating polynomial (PCHIP) that was fitted between the other two fractions, so that the transformation is continuous and the transformed data points y_{R} preserve the original order. PCHIP is defined so that the resulting polynomial is monotonic between the any given data points and first derivative of the polynomial is continuous [20]. The data transformation that was done for the Rosetta generated stability data of bacteriophage T4 lysozyme (PDB ID 2LZM) is shown in Figure 12

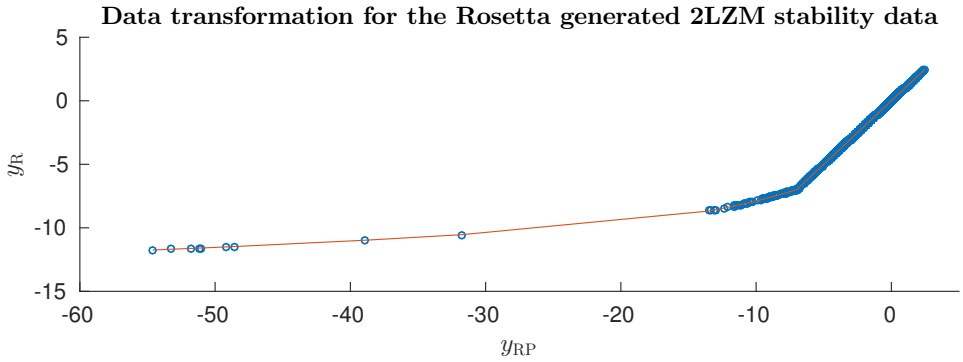


Figure 12: The data transformation for the Rosetta generated data that has been scaled to physical units (y_{RP}) that results to the transformed data y_{R} that is used for the training of the Gaussian process. The blue data points are connected with a red line.

6.2 Model formulation

When the stability effects upon mutations within a single protein are evaluated, the data consists only of protein variants of a single protein, who all have amino acid sequences of the same length. To compare these protein variants, a kernel matrix was defined between two protein variants a and b that are presented as graphs $G = (V, E)$, where the vertices V correspond to the the residues of the sequence and are labeled by their amino acid types. Edges E represent the contacts between the residues as defined by the contact map of the preminimised structure of the wild type protein. This same contact map is used for all of the protein variants, for the wild-type protein and its mutants. Therefore only the labels of the nodes differ between the graphs.

To compare the mutated proteins, a weighted decomposition kernel was constructed following the definition in Equation 52. Here the selectors are single residues and they are matched only when they have the same residue number, in other words when they have the same position in the protein sequence. The context z of a residue consists of its neighbours, which are the residues in contact with it according to the contact map. As each residue has their own position, each selector has exactly one neighbourhood and context and the number of contexts for one selector is $D = 1$. Unlike with the standard WDK, the matches are not weighted only by the similarity of their contexts, but also by the similarity of the match. The similarity of two amino acids is determined by a substitution matrix $\mathbf{S}_f$. The described kernel between protein variants a and b can be characterised as follows:

$$\mathbf{K}_f(a,b) = \sum_{i=1}^{n} \left(\mathbf{S}_f\left(a_i, b_i\right) \times \sum_{j \in \mathrm{nbs}(i)} \mathbf{S}_f\left(a_j, b_j\right) \right) \tag{63}$$

The above formulation allows the evaluation of the effects of multiple simultaneous mutations, as the complete protein structures are compared. However, as the wild type protein structure is used for all of the protein variants, changes that the mutations may cause to the protein structure are not taken into consideration. This may cause problems if mutations that alter the protein structure significantly are introduced – especially if many of them are introduced simultaneously. On the other hand, for example the BLOSUM62 substitution matrix should take this into account to some extend as these kinds of mutations are usually highly destabilising and do not occur often in nature.

This kernel matrix formulation was applied to five different settings. In the first and most simple case, only the experimental data from Protherm was used and only one kernel matrix was created using BLOSUM62 as the substitution matrix. Three hyperparameters, weight μ, exponent γ and noise of the data σ_n, were also introduced providing us the kernel matrix $\mathbf{K} = \mu\mathbf{K}_{\text{B62}}^{\gamma}$. The values of the hyperparameters were optimised using marginal likelihood maximisation as explained in Section 5.4.

For the second approach 531 additional substitution matrices were created using the formula in Equation 4 and the 531 features obtained from AAindex. These substitution matrices where then used to calculate 531 new kernel matrices using Equation 63. Marginal likelihood maximisation was used to assign each of the 532 kernel matrices their own weight μ_f and exponent γ_f, resulting to the following combination kernel matrix:

$$\mathbf{K} = \sum_f \mu_f \mathbf{K}_f^{\gamma_f}. \tag{64}$$

Also the noise σ_n was simultaneously optimised with marginal likelihood maximisation as in the first approach.

In the third method the mean of the previously created 532 kernel matrices was taken to form the combination kernel. As in the first approach, hyperparameters μ, γ and σ_n were introduced for this single kernel and optimised using marginal likelihood maximisation. This method is meant as a baseline for the other MKL methods.

Forth, a linear combination of the same 532 kernel matrices was also calculated using kernel alignment maximisation Alignf as discussed in Section 5.4. Each of the kernels was assigned its own weight μ_f, but a fixed exponent $\gamma = 1$ was used with all of the kernels and the noise was set to $\sigma_n = 0.1$ as these hyperparameters cannot be optimised with Alignf.

In the fifth setting also the data generated with Rosetta was used. In this case only one kernel created with BLOSUM62 was used and the hyperparameters μ and γ were optimised using marginal likelihood maximisation. This setting otherwise corresponds to the first model, but also the Rosetta generated data is used for the training of the Gaussian process and the noise of the data is modelled differently. Here it was

assumed that the experimental data is more accurate than the data generated with Rosetta and these data were thus assumed to have different noises. Furthermore, we assumed that the noise of the Rosetta generated data can be estimated from the generated stability values. One justification for this is that mutations that are highly destabilising often alter the folding of the protein, which complicates the stability prediction and as was said earlier, the constraints used with Rosetta may not permit enough backbone flexibility for these mutations. The noise was estimated with an inverse logit-function that has been scaled and shifted as follows

$$\sigma_{n,i} = 1.25 + 2.5 \cdot \frac{e^{-1.25y_{\mathrm{R},i}-1.25}}{e^{-1.25y_{\mathrm{R},i}-1.25}+1}, \tag{65}$$

which assigns low variances for stabilising and high variances for destabilising mutations. The estimated standard deviation of the transformed Rosetta generated data y_{R} in the case of 2LZM is shown in Figure 13. For the experimental data the noise was set to $\sigma_n = 0.1$. The noise for the experimental data could be optimised in the same manner as it was done when the model was trained using only the experimental data, but it was not implemented here.

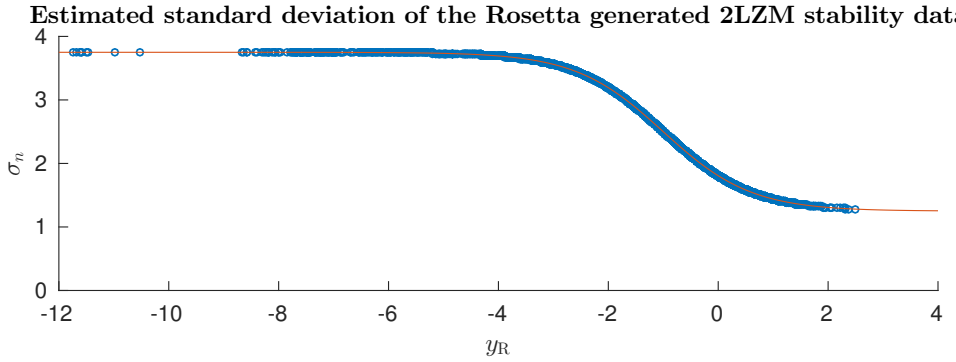
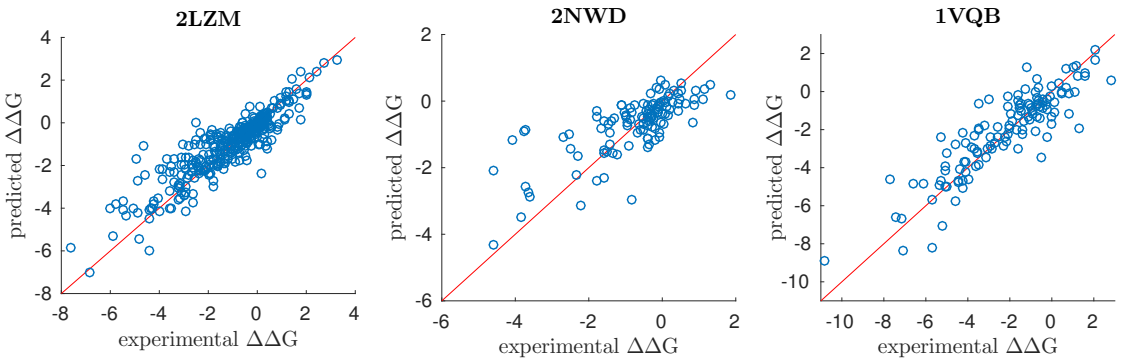


Figure 13: The estimated standard deviation of the transformed Rosetta generated data y_{R} in the case of 2LZM. The blue data points are connected with a red line, that corresponds to Equation 65.

6.3 Results

Gaussian process regression models were trained with the data for each of the three proteins, bacteriophage T4 lysozyme (2LZM), human lysozyme (2NWD) and enterobacteria phage m13 gene V protein (1VQB). When the 532 different kernels whose hyperparameters are optimised with marginal likelihood maximisation (GPMKL-model) using leave-one-out cross validation with the complete data sets, clear improvements can be seen with respect to both the correlation r and the root mean squared error rmse, when compared to the predictions obtained with Rosetta.

These results are shown in Figure 14. The Rosetta generated data contains all possible single mutations, so the results shown in Figure 11 are obtained using only the single mutations that are in the experimental data and the corresponding mutations in the Rosetta generated data.



(a) r=0.90, rmse=0.76 $\frac{\text{kcal}}{\text{mol}}$ (b) r=0.75, rmse=0.84 $\frac{\text{kcal}}{\text{mol}}$ (c) r=0.88, rmse=1.13 $\frac{\text{kcal}}{\text{mol}}$

Figure 14: Correlations of the predicted and experimental data with (a) Bacteriophage T4 lysozyme (2LZM) (b) Human lysozyme (2NWD) and (c) Enterobacteria phage m13 gene V protein (1VQB). The predictions are done with the GPMKL-model and leave-one-out cross validation using the whole data.

Figure 15 is based on Figure 14a, but the predictions are coloured by the number of simultaneous mutations. This figure shows that also the stability changes caused by multiple simultaneous mutations are predicted well.

In order to compare this method to existing methods, two web servers were used to predict the changes in stability upon the mutations, mCSM[4] and PoPMuSiC[5]. As these methods are applicable only for single mutations, they were used to predict all the single mutations in the experimental data sets. Table 3 shows the correlations and errors of these predictions as well as the results obtained with the GPMKL-model using all of the mutations and only the single mutations in the experimental data sets. Also the results obtained with Rosetta are included.

The GPMKL-model provides better results than the other methods with all of the three proteins. It should be noted that it has not been declared what data have been used to train the prediction models of the web servers and it is likely that their training data contains at least some of the mutations in the data sets used here.

Although the multiple kernel learning was done using kernels created with 532 different features, not all of them were used. The number of used features varied between 2–7 between the different folds. The models trained for 2LZM and 1VQB used kernels created with 20 different features, and the model trained for 2NWD

[4] http://bleoberis.bioc.cam.ac.uk/mcsm/stability
[5] http://dezyme.com

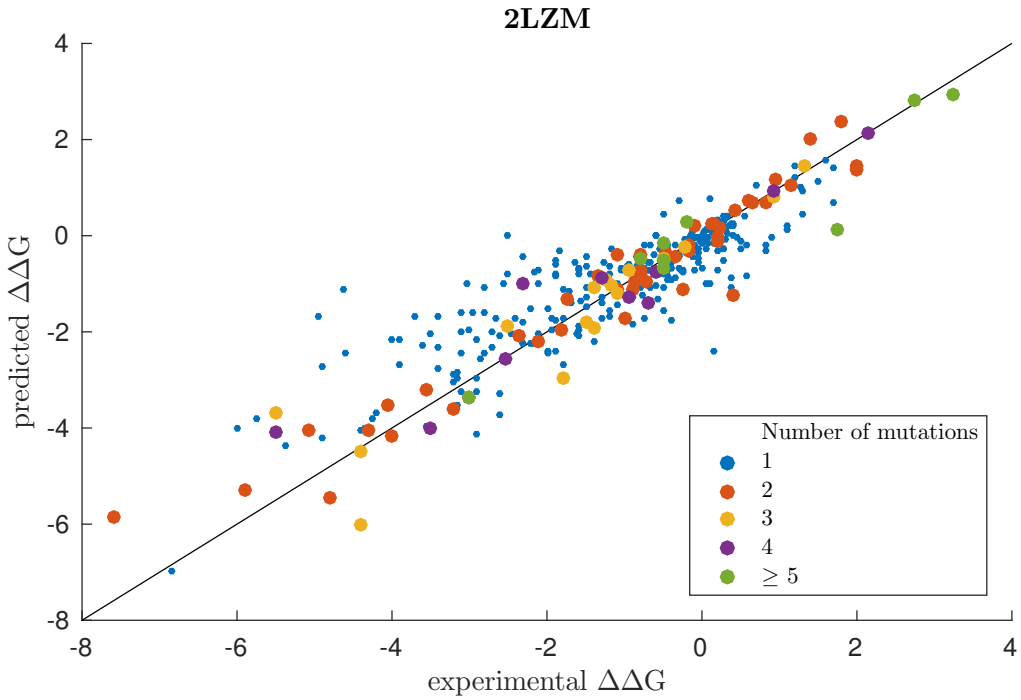


Figure 15: Correlations of the predicted and experimental data with 2LZM. This figure corresponds to Figure 14a, but the predictions are coloured by the number of simultaneous mutations.

Table 3: Correlations and root mean squared errors of stability change predictions with different methods for the experimental data for 2LZM, 2NWD and 1VQB. GPMKL (all) contains predictions for the multiple mutations in the data sets and the others have predictions only for the single mutations in the data sets.

	2LZM		2NWD		1VQB	
method	r	rmse	r	rmse	r	rmse
GPMKL (all)	0.90	0.76	0.75	0.84	0.88	1.13
GPMKL (single)	0.86	0.80	0.72	0.84	0.83	1.24
Rosetta	0.73	1.12	0.68	2.51	0.52	2.19
mCSM	0.57	1.27	0.63	1.02	0.53	2.24
PoPMuSiC	0.71	1.11	0.59	1.00	0.51	2.29

used 21 different features in all of the folds. However, the used features vary notable between the different proteins. This can be seen from the data provided in Appendix B. Therefore it is beneficial to use many different features for the training of these models, as it may not be evident which features should be used with a certain protein.

To evaluate the need for multiple kernel learning and to determine if the GPMKL model is a good choice for it, prediction models were constructed according to the five settings described in section 6.2. They were tested with different numbers of training samples to estimate how the amount of available training data affects the performance

of these models and to determine the learning curves. Figures 16-18 show how the predictions improve as more samples are used for the training. For each of the three proteins 100 randomly selected training sets were selected for each training set size. The same training sets were then used to train all of the models to ensure comparable results. We can see that with all the three proteins even the most basic model (B62) that uses only one kernel created with the BLOSUM62 substitution matrix reaches higher correlations between the predicted and experimental $\Delta\Delta$G-values than Rosetta, when a sufficient number of training samples is used.

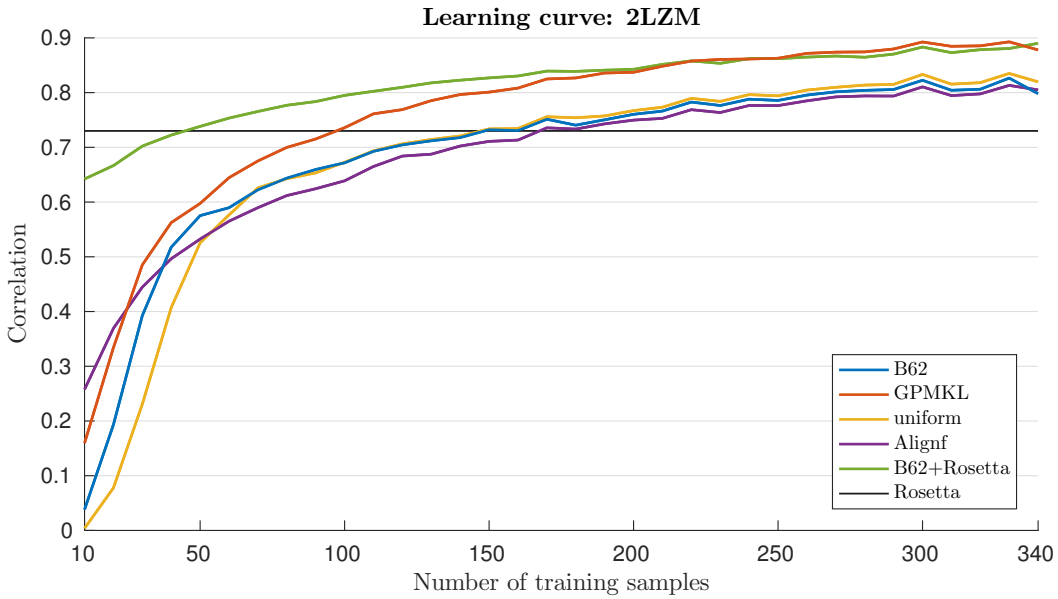


Figure 16: Correlation of predicted and experimental stability values with different amounts of training data with Bacteriophage T4 lysozyme (2LZM), averaged over 100 random folds.

The improvement that can be gained by using multiple kernel learning and training the model with more kernels that exploit the different features of the amino acids, seems to depend on the protein in question. With 2LZM and 1VQB the use of the 532 kernels whose hyperparameters are optimised with marginal likelihood maximisation (GPMKL-model), the correlation is always improved. With 2NWD the correlation is improved when only few training samples are used and is approximately the same when over 50 training samples are used. With 2LZM and 1VQB it seems clear that GPMKL is the a good method for the multiple kernel learning as it provides better results than the uniformly weight sum of the kernels and Alignf after more than 20 - 40 training samples are used. With 2NWD this is not as clear as even the predictions done with the model using only BLOSUM62 substitution matrix sometimes achieve higher correlations than the GPMKL-model.

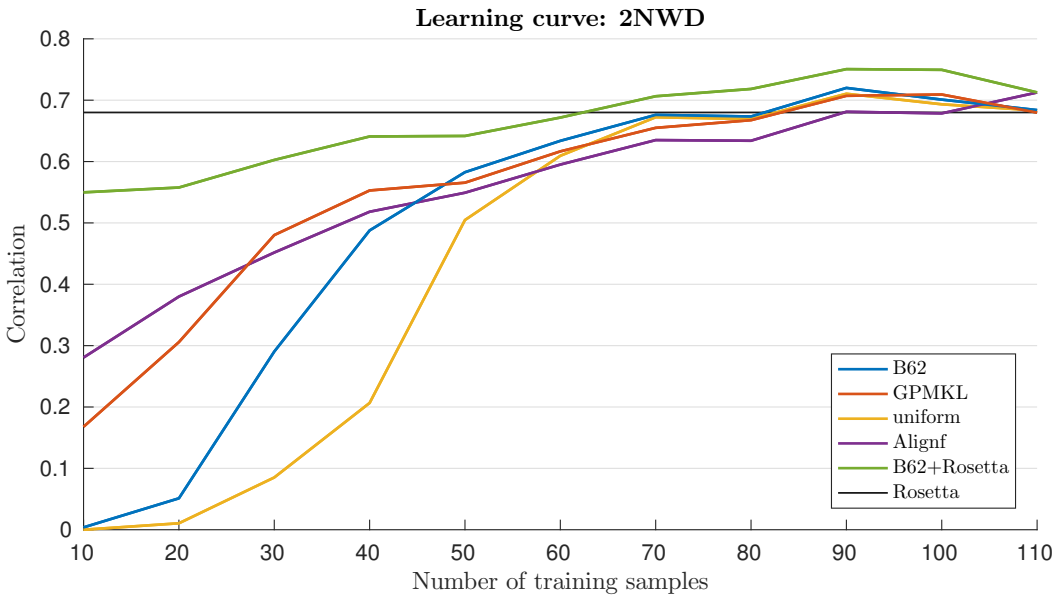


Figure 17: Correlation of predicted and experimental stability values with different amounts of training data with Human lysozyme (2NWD), averaged over 100 random folds.

When the Gaussian process regression model is trained with the additional Rosetta generated data (B62+Rosetta), the correlation is improved with all of the three proteins when a low number of training samples is used. With 2LZM and 2NWD the Rosetta generated data had high correlation with the experimental data, 0.73 and 0.68 respectively, and using this data provides always better or comparable results to the GPMKL-model. With 1VQB the correlation between the Rosetta generated and experimental data was relatively low, only 0.52. When more than 60 training samples are used the model exploiting the Rosetta generated data does not provide improvements to the correlations of the GPMKL-model and when more than 80 training samples are used its performance is even worse than that of the B62-model in terms of correlation. As this model exploiting the Rosetta generated data was trained using just one kernel created with the BLOSUM62 substitution matrix, it is likely that these results could be further improved by exploiting the different amino acid features, as this was the case when using only the experimental data. Also, it might be beneficial to optimise the noises of the experimental and Rosetta generated data instead of using a fixed value and a fixed function for them.

From the above results we can see that these models provide good results when enough data is available. However, it is also important to know how much these results depend on the selected training sets. Figure 19 shows the standard deviations of the correlations between the predictions and experimental measurements when the B62, GPMKL and B62+Rosetta models are used with the data for 2LZM. The

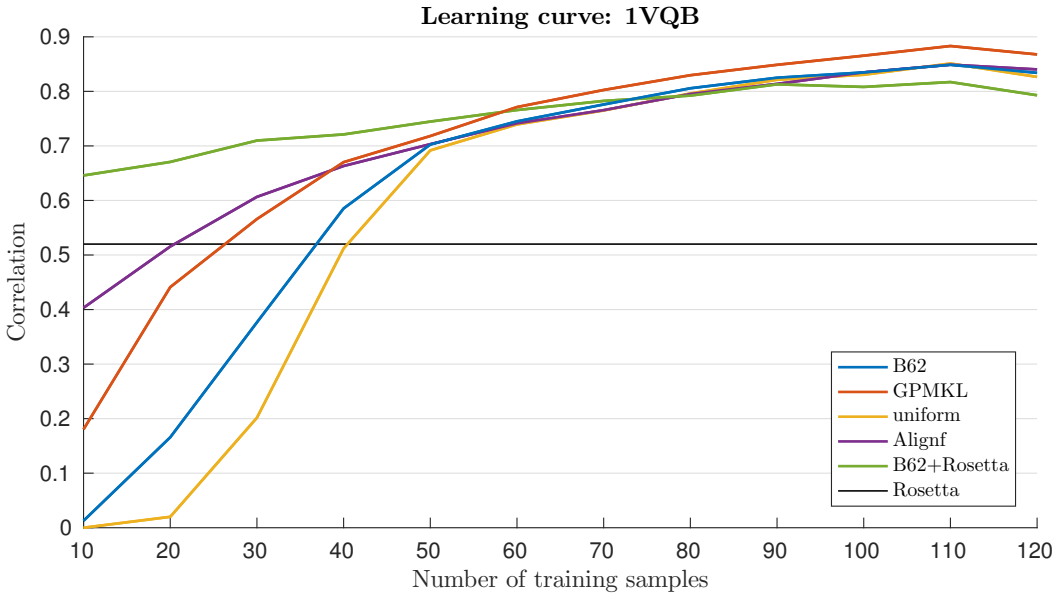


Figure 18: Correlation of predicted and experimental stability values with different amounts of training data with Enterobacteria phage m13 gene V protein (1VQB), averaged over 100 random folds.

same 100 random samples that were used with Figure 16 are used here as well and the mean of the predictions is thus the same. From these figures we can see that in addition to the improvements in correlation, also the variance of the predictions decreases when multiple kernels or the Rosetta generated data is used. When the Rosetta generated data is used, the correlations are least dependent on the selected training sets as the variance is smallest with this model.

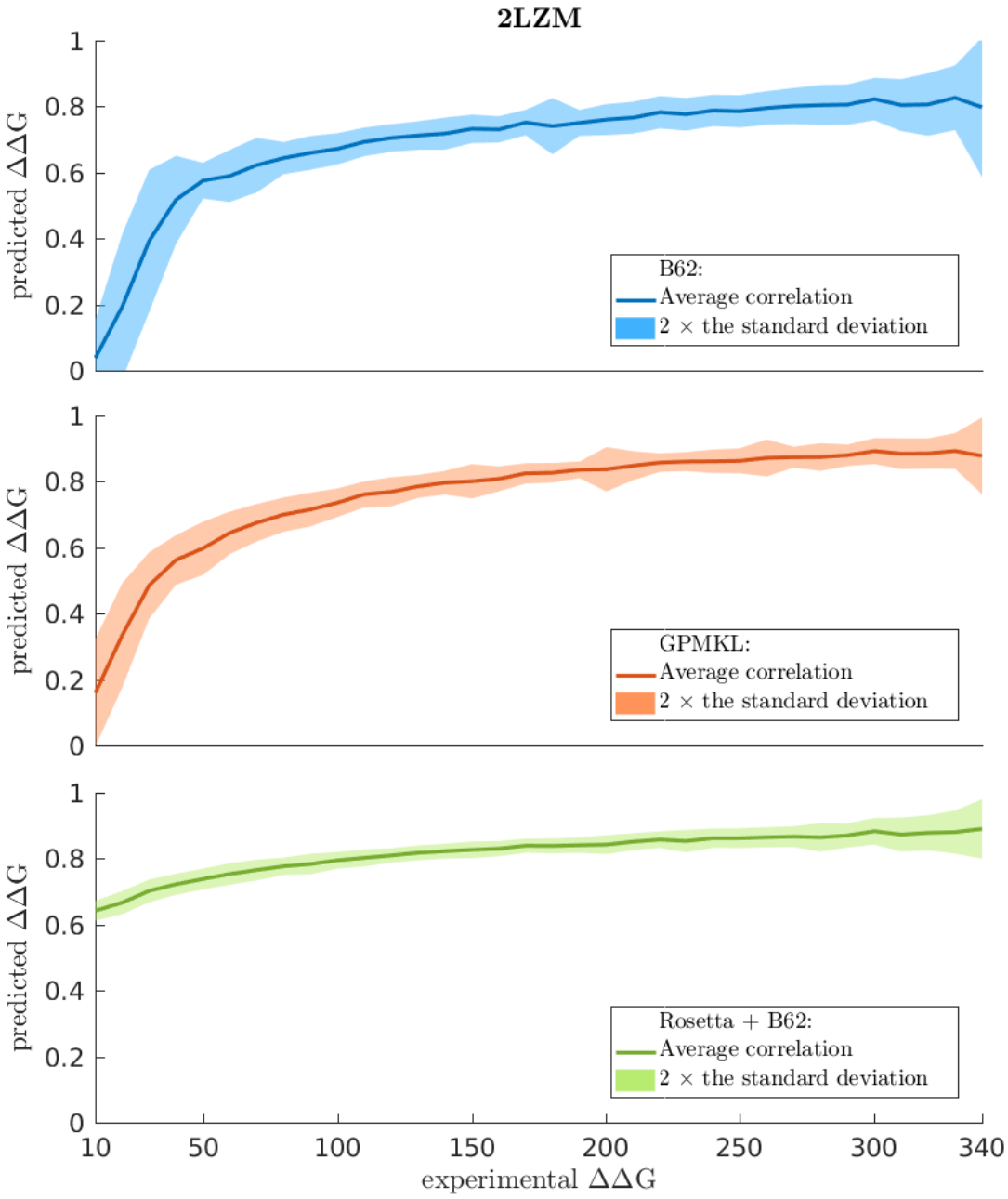


Figure 19: The correlation of predicted and experimental stability values with different amounts of training data, averaged over 100 random folds and the standard deviations of the correlations.

7 Predicting stability changes upon single mutations for multiple proteins

In this section we will consider the prediction of stability changes upon single mutations when we have data from multiple proteins. We will introduce the used data sets, model formulation and the obtained results.

7.1 Data sets

The S2648 data set that we use for the prediction of stability effects upon single mutations for multiple proteins, was constructed by Dehouck et al. [17]. It contains only single-site mutations in globular proteins whose experimental structure is available.

Mutations that destabilise the structure by more than 5 kcal/mol and mutations involving a proline were not considered. Also homo-multimeric proteins for which it was unclear if the free energy changes correspond either to the whole protein or to a monomer only were excluded. The stability data was originally obtained from Protherm database and if multiple stability values existed for a mutation, the mutation was assigned a weighted average of those values. Measurements performed with pH close to 7 and temperature near 25 °C were assigned a larger weight. The complete data set and a more detailed description of its construction can be found in the supplementary material of Dehouck et al. [17]. This data set was chosen because of its wide usage [17, 21, 36, 42, 43, 62]. A subset of 350 mutations of this data known as S350, has been commonly used for the testing of stability prediction methods, when the remaining data has been used for the training.

7.2 Model formulation

When our data consists of single mutations for multiple proteins, the formulation used for the kernel matrix in the single protein case is not sufficient anymore. The proteins in the data set have amino acid sequences of different lengths and also their contact maps differ from each other. Therefore they cannot be compared in the same manner as by Equation 63. Also the differences between the proteins are often more significant than the differences caused by the mutations, so it is not sufficient to merely compare the complete structures of the mutated proteins. Here we construct a kernel matrix by estimating how well a residue fits to its neighbourhood before and after a mutation. The fit of an amino acid to its neighbourhood is estimated with amino acid pair-wise contact potentials available from AAindex. A pair-wise contact potential gives a value for all possible naturally occurring amino acid pairs in the same manner as substitution matrices, except that now the values in the resulting 20 × 20 matrix give a score for the interactions of the corresponding amino acids [27].

Using these contact potentials, the fit can be estimated as a probability distribution $p_{fza_i}(x)$ over the values x of a contact potential f. This is done by fitting a Gaussian distribution at each contact potential value x, so that the mean of the Gaussian is centered at that value and the density is estimated by the frequency of the value in the given neighbourhood z. The final distribution is calculated as the sum of these Gaussians. The similarity of two mutations is estimated with a Gaussian kernel of the differences of the wild type and mutant fit distributions $\mathbf{d}_{a_i} = p_{fz_w a_i} - p_{fz_m a_i}$ between the different mutations. The resulting kernel matrix value between proteins a and b is calculated as follows:

$$\mathbf{K}_f(a,b) = e^{-(\mathbf{d}_{a_i} - \mathbf{d}_{b_i})^2 / 2l^2}. \tag{66}$$

For the density estimation we used a bandwidth of 0.02 and for the Gaussian kernel a length scale l=0.15. The constructed kernel can be seen a weighted decomposition kernel, where the selectors s are single residues that are matched only if they are the mutated residue. The neighbourhoods z consist of the contacts between the selector-residues and their neighbours.

For example, if the mutation F17A is introduced to bacillus subtilis cold chock protein B (PDB ID 1CSP), we can estimate how well Alanine (A) fits to that position compared to Phenylalanine (F). Figure 20 shows the residue number 17 and its neighbours. The contacts are marked with the dark lines and each of them is assigned a contact potential value.

(a) (b)

Figure 20: Bacillus subtilus cold shock protein B (PDB ID 1CSP). Residue number 17 is mutated from Phenylalanine (a) to Alanine (b). The residue 17F is coloured with red and 17A with orange and their neighbours with blue and presented by a sphere located at their α-carbon. Contacts between residue 17 and its neighbours are marked with dark lines. Thicker lines indicate greater contact potential values.

When the density estimation is done based on the contact potential values of AAindex3 entry TANS760101 (Statistical contact potential derived from 25 x-ray protein structures) with the native and mutated amino acid, we get the distributions and their difference, that are presented in Figure 21.

There are 47 different contact potentials available from AAindex from which 43 have no missing values. These 43 contact potentials $f \in \{1, 2, \ldots, 43\}$, where used to create 43 different kernels using Equation 66. A combination kernel was formed from these base kernels as follows:

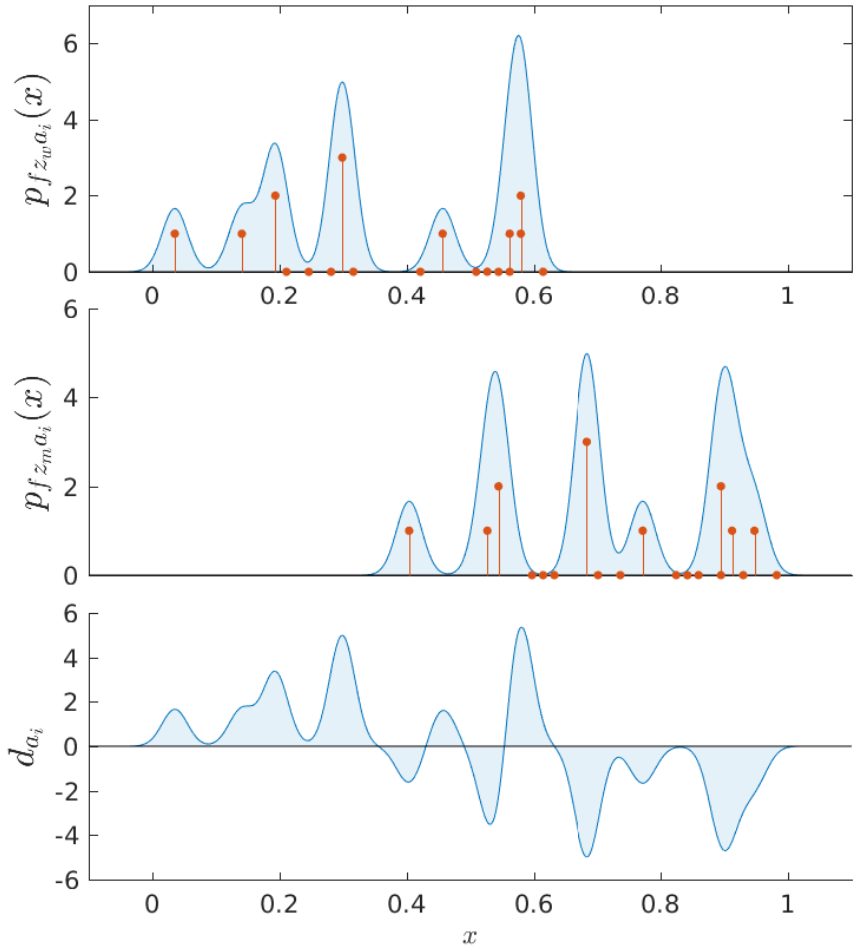


Figure 21: The first two figures represent the estimated probability distributions of the statistical contact potential values between residue 17 and its neighbours from bacillus subtilis cold shock protein B (PDB ID 1CSP). The first figure corresponds to the contact potentials with the native amino acid Phenyalanine (F) and the second with the mutated residue and Alanine (A). The third figure is the difference of these distributions.

$$\mathbf{K} = \sum_f \mu_f \mathbf{K}_f^{\gamma_f} \tag{67}$$

and the values of the hyperparameters μ_f, γ_f and the noise σ_n were optimised using marginal likelihood maximisation.

7.3 Results

The Gaussian process regression model (GPMKL-multi) was then trained using the S2648 data set from which the S350 data set was excluded and the kernel matrix that was calculated using the 43 different contact potentials as described in the previous section. Using this model we obtained a correlation $r = 0.54$ and root-mean-squared error rmse $= 1.32$ kcal/mol. Figure 22 shows the correlation between the experimental and predicted stability values.

Figure 22: The correlation between the experimental ΔΔG-values and the ΔΔG-values predicted with model described in Section 7.2 using the S350 data set is $r = 0.54$ and rmse $= 1.32$.

As can be seen from Table 4, these results are not quite as good as those achieved with some of the existing methods such as NeEMO [21], PoPMuSiC-2.0 [17], DUET [42] and mCSM [43]. However, these results demonstrate that the contact potentials contain relevant information for protein stability predictions. Also, as this model now uses only 43 different kernels, improvements could be achieved by using additional kernels matrices that exploit different information of the mutated proteins.

Table 4: Comparative stability change predictions using the S350 data set

Method	r	rmse
GPMKL-multi	0.54	1.32
NeEMO	0.67	1.16
PoPMuSiC-2.0	0.67	1.16
DUET	0.71	1.13
mCSM	0.73	1.08

8 Conclusions

We introduced two methods based on Gaussian processes for predicting stability changes in proteins upon mutations. We showed that when only data from the protein of interest is used, comparing the complete structures of the protein variants using amino acid features is a good way to estimate the similarity of these protein variants. The predictions can be improved by using different features and MKL for assigning appropriate weights for the kernels created with these features. Alternatively, the use of additional stability data that is generated by a prediction method relying on energy potentials, such as Rosetta, can improve the predictions. Based on the results using MKL with only the experimental stability data, it is likely that the predictions could be further improved if MKL and multiple features were used with the additionally generated stability data as well.

We also demonstrated that a more general model that exploits data from different proteins can be constructed using Gaussian processes and contact potentials between the mutated residues and their neighbours. This model is useful when there exists only

little or no experimental data from the protein of interest. However, more accurate results can be achieved with the first model, if sufficient amount of experimental data is available. The second model is quite simple as it exploits only 43 different contact potentials. Therefore it is likely that it could still be improved by adding kernels that utilize different information sources.

The use of Gaussian processes allowed us to utilise the marginal likelihood maximisation for the MKL. Although not discussed here in detail, the performance of the MKL is dependent on the method chosen for the calculation of step directions of the gradients. The two-metric projection method that was used in Section 6 has shorter running times than the interior point method used in Section 7, but better results can be achieved with the interior point method. This trade-off between the running times and accuracy should be considered when implementing the models presented in this thesis. Also, Gaussian processes provided a convenient way to exploit stability data from different sources, as we could assign different variances to the data depending on its origin and properties.

References

[1] Robert A. Adams and Christopher Essex. *Calculus: A Complete Course*. Pearson, 8 edition, 2013.

[2] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular biology of the cell*. Garland Science, 5 edition, 2007.

[3] Eric V Anslyn and Dennis A Dougherty. *Modern physical organic chemistry*. University Science Books, 2006.

[4] Alexander Benedix, Caroline M Becker, Bert L de Groot, Amedeo Caflisch, and Rainer A Böckmann. Predicting free energy changes using structural ensembles. *Nature methods*, 6(1):3–4, 2009.

[5] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, TN Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[6] Matthew J Betts and Robert B Russell. Amino-acid properties and consequences of substitutions. *Bioinformatics for geneticists*, 2:311–342, 2007.

[7] Andreas S Bommarius, Janna K Blum, and Michael J Abrahamson. Status of protein engineering for biocatalysts: how to design an industrially useful biocatalyst. *Current opinion in chemical biology*, 15(2):194–200, 2011.

[8] Carl Branden and John Tooze. *Introduction to protein structure*. Garland, 2 edition, 1999.

[9] Emidio Capriotti, Piero Fariselli, Remo Calabrese, and Rita Casadio. Predicting protein stability changes from sequences using support vector machines. *Bioinformatics*, 21(suppl 2):ii54–ii58, 2005.

[10] Emidio Capriotti, Piero Fariselli, and Rita Casadio. I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic acids research*, 33(suppl 2):W306–W310, 2005.

[11] Emidio Capriotti, Piero Fariselli, Ivan Rossi, and Rita Casadio. A three-state prediction of single point mutations on protein stability changes. *BMC bioinformatics*, 9(2), 2008.

[12] Chi-Wei Chen, Jerome Lin, and Yen-Wei Chu. iStable: off-the-shelf predictor integration for predicting protein stability changes. *BMC bioinformatics*, 14(2), 2013.

[13] Jianlin Cheng, Arlo Randall, and Pierre Baldi. Prediction of protein stability changes for single-site mutations using support vector machines. *Proteins: Structure, Function, and Bioinformatics*, 62(4):1125–1132, 2006.

[14] Joel R Cherry and Ana L Fidantsef. Directed evolution of industrial enzymes: an update. *Current opinion in biotechnology*, 14(4):438–443, 2003.

[15] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Two-stage learning kernel algorithms. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 239–246, 2010.

[16] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *Advances in Neural Information Processing Systems*, 14(1):367–373, 2002.

[17] Yves Dehouck, Aline Grosfils, Benjamin Folch, Dimitri Gilis, Philippe Bogaerts, and Marianne Rooman. Fast and accurate predictions of protein stability changes upon mutations using statistical potentials and neural networks: PoPMuSiC-2.0. *Bioinformatics*, 25(19):2537–2543, 2009.

[18] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[19] Lukas Folkman, Bela Stantic, and Abdul Sattar. Feature-based multiple models improve classification of mutation-induced stability changes. *BMC genomics*, 15(Suppl 4), 2014.

[20] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[21] Manuel Giollo, Alberto JM Martin, Ian Walsh, Carlo Ferrari, and Silvio CE Tosatto. NeEMO: a method using residue interaction networks to improve prediction of protein stability upon mutation. *BMC genomics*, 15(4):1, 2014.

[22] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[23] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, 2004.

[24] Raphael Guerois, Jens Erik Nielsen, and Luis Serrano. Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. *Journal of molecular biology*, 320(2):369–387, 2002.

[25] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[26] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

[27] Shuichi Kawashima, Piotr Pokarowski, Maria Pokarowska, Andrzej Kolinski, Toshiaki Katayama, and Minoru Kanehisa. AAindex: amino acid index database, progress report 2008. *Nucleic acids research*, 36(suppl 1):D202–D205, 2008.

[28] Romas J Kazlauskas and Uwe T Bornscheuer. Finding better protein engineering strategies. *Nature chemical biology*, 5(8):526–529, 2009.

[29] Elizabeth H Kellogg, Andrew Leaver-Fay, and David Baker. Role of conformational sampling in computing mutation-induced changes in protein structure and stability. *Proteins: Structure, Function, and Bioinformatics*, 79(3):830–838, 2011.

[30] Ole Kirk, Torben Vedel Borchert, and Claus Crone Fuglsang. Industrial enzyme applications. *Current opinion in biotechnology*, 13(4):345–351, 2002.

[31] Andrzeji Kolinski. *Multiscale approaches to protein modeling*. Springer, 2011.

[32] MD Shaji Kumar, K Abdulla Bava, M Michael Gromiha, Ponraj Prabakaran, Koji Kitajima, Hatsuho Uedaira, and Akinori Sarai. ProTherm and ProNIT: thermodynamic databases for proteins and protein–nucleic acid interactions. *Nucleic Acids Research*, 34(suppl 1):D204–D206, 2006.

[33] Andrew Leaver-Fay and Elizabeth Kellogg. Rosetta 3.5 user guide: Documentation for the ddg_monomer application. https://www.rosettacommons.org/manuals/archive/rosetta3.5_user_guide/.

[34] Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, et al. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. *Methods in enzymology*, 487:545, 2011.

[35] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[36] Jianguo Liu and Xianjiang Kang. Grading amino acid properties increased accuracies of single point mutation on protein stability prediction. *BMC bioinformatics*, 13(1):1, 2012.

[37] Sauro Menchetti, Fabrizio Costa, and Paolo Frasconi. Weighted decomposition kernels. In *Proceedings of the 22nd international conference on Machine learning*, pages 585–592. ACM, 2005.

[38] William M Mendenhall and Terry L Sincich. *Statistics for Engineering and the Sciences*. CRC Press, 3 edition, 1992.

[39] Yanay Ofran and Burkhard Rost. Analysing six types of protein–protein interfaces. *Journal of molecular biology*, 325(2):377–387, 2003.

[40] C Nick Pace and J Martin Scholtz. Measuring the conformational stability of a protein. *Protein structure: A practical approach*, 2:299–321, 1997.

[41] C Nick Pace and Kevin L Shaw. Linear extrapolation method of analyzing solvent denaturation curves. *Proteins: Structure, Function, and Bioinformatics*, 41(S4):1–7, 2000.

[42] Douglas EV Pires, David B Ascher, and Tom L Blundell. DUET: a server for predicting effects of mutations on protein stability using an integrated computational approach. *Nucleic acids research*, page gku411, 2014.

[43] Douglas EV Pires, David B Ascher, and Tom L Blundell. mCSM: predicting the effects of mutations in proteins using graph-based signatures. *Bioinformatics*, 30(3):335–342, 2014.

[44] Navin Pokala and Tracy M Handel. Energy functions for protein design: adjustment with protein–protein complex affinities, models for the unfolded state, and negative design of solubility and specificity. *Journal of molecular biology*, 347(1):203–227, 2005.

[45] Vladimir Potapov, Mati Cohen, and Gideon Schreiber. Assessing computational methods for predicting protein stability upon mutation: good on average but not in the details. *Protein Engineering Design and Selection*, 22(9):553–560, 2009.

[46] Ralph Rapley and John M Walker. *Molecular Biology and Biotechnology*. Royal Society of Chemistry, 4 edition, 2000.

[47] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

[48] Gale Rhodes. *Complementary Science: Crystallography Made Crystal Clear*. Academic press, 3 edition, 5 2014.

[49] Andrew D Robertson and Kenneth P Murphy. Protein structure and the energetics of protein stability. *Chemical reviews*, 97(5):1251–1268, 1997.

[50] Rosetta basics: Units in Rosetta. https://www.rosettacommons.org/docs/latest/rosetta_basics/Rosetta-Basics.

[51] Sergio Sanchez and Arnold L Demain. Enzymes and bioconversions of industrial, pharmaceutical, and biotechnological significance. *Organic Process Research & Development*, 15(1):224–230, 2010.

[52] Mark W. Schmidt. minConf: Projection methods for optimization with simple constraints in Matlab. http://www.cs.ubc.ca/~schmidtm/Software/minConf.html, 2008.

[53] Michael Schneider, Xiaoran Fu, and Amy E Keating. X-ray vs. NMR structures as templates for computational protein design. *Proteins: Structure, Function, and Bioinformatics*, 77(1):97–110, 2009.

[54] Bernhard Schölkopf and Smola Alexander J. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[55] LLC Schrödinger. The PyMOL molecular graphics system, version 1.6.

[56] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[57] Simon J Sheather et al. Density estimation. *Statistical Science*, 19(4):588–597, 2004.

[58] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-Lehman graph kernels. *The Journal of Machine Learning Research*, 12:2539–2561, 2011.

[59] Jian Tian, Ningfeng Wu, Xiaoyu Chu, and Yunliu Fan. Predicting changes in protein thermostability brought about by single- or multi-site mutations. *BMC bioinformatics*, 11(1):1, 2010.

[60] Nobuhiko Tokuriki and Dan S Tawfik. Stability effects of mutations and protein evolvability. *Current opinion in structural biology*, 19(5):596–604, 2009.

[61] S. Vichy N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[62] Gilad Wainreb, Lior Wolf, Haim Ashkenazy, Yves Dehouck, and Nir Ben-Tal. Protein stability: a single recorded mutation aids in predicting the effects of other mutations in the same amino acid site. *Bioinformatics*, 27(23):3286–3292, 2011.

[63] Catherine L Worth, Robert Preissner, and Tom L Blundell. SDM – a server for predicting effects of mutations on protein stability and malfunction. *Nucleic acids research*, pages W215–W222, 2011.

[64] Ying Xu, Dong Xu, and Jie Liang. *Computational Methods for Protein Structure Prediction and Modeling: Volume 1: Basic Characterization*, volume 1. Springer Science & Business Media, 2007.

A Stability data sets for mutations in single proteins

The stability data that was used in the predictions of stability changes within single proteins are presented below. Tables A1 and A2 contain the data for bacteriophage T4 lysozyme, Tables A3 and A4 for human lysozyme and tables A5 and A6 for enterobacteria phage m13 gene V protein. There separate tables for single and multiple mutations exist for notational convenience.

Table A1: Stability data for Bacteriophage T4 lysozyme: single mutations

mutation	ΔΔG	mutation	ΔΔG	mutation	ΔΔG	mutation	ΔΔG
I3A	-0.90	S44T	0.01	R96M	-2.60	Q123E	0.27
I3D	-2.50	S44W	0.05	R96F	-4.20	K124G	-0.05
I3C	0.15	S44Y	0.19	R96P	-5.75	E128A	0.16
I3E	-1.62	S44V	0.10	R96S	-2.70	E128K	-1.16
I3G	-1.95	E45A	0.28	R96T	-2.95	A129L	-1.30
I3L	0.84	L46A	-2.32	R96W	-4.35	A129M	-1.90
I3M	-0.60	D47A	-0.61	R96Y	-4.60	A129V	-0.75
I3F	-1.04	K48A	-0.44	R96V	-2.45	A130S	-1.00
I3P	-2.82	A49S	-0.50	A98S	-2.50	V131A	0.28
I3S	-1.80	I50A	-2.00	A98V	-4.90	V131D	0.08
I3T	-2.11	C54T	0.83	L99A	-4.62	V131E	0.20
I3W	-2.80	C54V	-0.70	L99G	-6.85	V131G	-0.68
I3Y	-2.50	N55G	-0.55	L99I	-1.45	V131I	0.16
I3V	-0.50	I58A	-3.20	L99M	-0.57	V131L	0.09
M6A	-1.90	I58T	-3.40	L99F	-0.35	V131M	0.12
M6I	-1.38	T59A	-2.15	L99V	-2.15	V131S	-0.05
M6L	-2.80	T59N	-0.85	I100A	-3.40	V131T	-0.09
L7A	-2.60	T59D	-1.05	I100M	-1.60	N132I	1.20
E11A	1.10	T59G	-1.90	I100V	-0.40	N132M	1.50
E11N	-0.10	T59S	-0.45	N101A	-1.50	N132F	1.30
E11H	0.10	T59V	-2.15	M102L	-0.85	L133A	-4.25
E11M	1.60	K60H	-0.45	M102K	-6.00	L133D	-5.37
E11F	1.70	K60P	0.05	M102V	-3.00	L133M	-0.40
R14K	-0.03	L66A	-3.90	V103A	-2.20	L133F	-0.27
K16E	0.00	F67A	-1.90	V103I	-0.50	A134S	-0.10
I17A	-2.70	N68A	-0.05	V103M	-1.20	K135E	-0.73
D20A	-0.30	Q69P	-3.10	F104A	-3.10	W138Y	-1.71
D20N	1.30	V71A	-1.50	Q105A	-0.75	N144D	0.30
D20S	0.70	D72P	-2.65	Q105E	-0.80	N144E	0.40
D20T	0.90	A73S	-0.40	Q105G	-1.60	N144H	-0.15
E22K	0.57	A74P	-4.95	Q105M	-1.20	A146T	-1.90
T26S	0.57	V75T	-1.30	M106A	-2.30	K147E	-0.43
I27A	-3.10	G77A	0.00	M106I	0.20	V149A	-3.15
G28A	-2.00	I78A	-1.60	M106L	0.50	V149C	-2.10
I29A	-2.60	I78M	-1.50	M106K	-3.40	V149G	-4.90
G30A	0.10	I78V	-0.80	E108V	1.10	V149I	-0.05
G30F	-1.50	R80K	-0.17	T109N	0.05	V149S	-4.40
H31N	-4.00	A82P	0.55	T109D	0.15	V149T	-2.90
L33A	-3.60	A82S	-0.30	V111A	-1.20	T151S	0.39
S38A	-0.77	K83H	-0.45	V111I	-0.77	T152A	-1.50
S38N	-0.05	L84A	-3.90	V111F	-1.56	T152C	-0.50
S38D	0.33	L84M	-1.90	G113A	0.40	T152I	-0.40
L39A	-0.90	K85A	-0.25	G113E	0.30	T152S	-2.30
N40A	0.32	V87A	-1.70	T115A	-0.14	T152V	0.20
N40D	0.44	V87I	-0.30	T115E	-0.10	F153A	-3.55
A41D	0.29	V87T	-1.60	N116A	0.17	F153I	-0.35
A41S	-0.60	D89A	-0.60	N116D	0.36	F153L	0.28
A41V	0.39	S90H	-1.07	S117A	1.27	F153M	-0.67
A42K	-3.70	L91A	-3.10	S117I	1.70	F153V	-1.80
A42S	-2.30	L91M	-0.80	S117F	1.20	R154E	-0.45
K43A	-1.03	D92N	-0.75	S117V	2.00	G156D	-1.85
S44A	0.19	A93P	0.37	L118A	-3.50	T157A	-0.97
S44R	0.24	A93S	-0.20	L118I	-1.20	T157R	-0.68
S44N	-0.14	A93T	0.06	L118M	-0.70	T157N	-0.94
S44D	-0.11	V94A	-1.80	R119A	-0.18	T157D	-1.10
S44C	-0.11	R96A	-2.00	R119E	-0.08	T157C	-1.30
S44Q	0.27	R96N	-3.15	R119H	-0.29	T157E	-1.27
S44E	0.00	R96D	-2.90	R119M	-0.10	T157G	-1.10
S44G	-0.53	R96C	-2.90	M120A	-0.20	T157H	-2.10
S44H	0.04	R96Q	-0.50	M120L	0.50	T157I	-1.95
S44I	0.31	R96E	-2.20	M120K	-1.60	T157L	-1.60
S44L	0.39	R96G	-2.90	M120Y	-0.10	T157F	-2.40
S44K	0.20	R96H	-3.16	L121A	-2.57	T157S	-0.66
S44M	0.33	R96I	-2.85	L121M	-0.80	T157V	-1.53
S44F	0.06	R96L	-3.20	Q122A	-0.24	A160T	-1.65
S44P	-3.03	R96K	-0.25	Q123A	-0.22	N163D	-0.21

Table A2: Stability data for Bacteriophage T4 lysozyme: multiple mutations

mutation	**ΔΔG**
I3L S38D A82P N144D	2.13
I3L S38D A82P V131A N144D	2.41
I3L S38D A41V A82P V131A N144D	2.75
I3L S38D A41V A82P N116D V131A N144D	3.25
M6I R96H	-5.08
L13D K60H	-2.37
R14A K16A I17A K19A T21A E22A	-3.00
K16E K135E	-0.80
K16E K135E K147E	-1.25
K16E R119E	-0.10
K16E R119E K135E K147E	-0.95
K16E R154E	-0.35
Y24A Y25A T26A I27A	-5.50
G28A I29A G30A	-5.50
H31N D70N	-4.00
L32A L33A T34A E108V	-3.00
T34A K35A S36A P37A	-0.70
T34A K35A S36A P37A E128A V131A N132A	-0.20
S38D A82P N144D	1.32
S38D N144D	0.81
N40A S44A E45A D47A K48A D127A E128A V131A N132A	1.75
A41V V131A	0.65
E45A K48A	0.20
N53A N55A V57A	-1.50
N53A N55A V57A E128A V131A N132A	-0.80
C54T C97A	-0.79
C54V C97S	-2.10
R80K R119H	-0.47
K83H A112D	-1.33
K85A R96H	-3.55
D89A R96H	-4.05
S90H Q122D	-1.73
A98V T152S	-4.80
A98V V149C T152S	-4.40
A98V V149I T152S	-4.40
L99A E108V	-3.20
L99A F153A	-7.60
L99G E108V	-5.90
L99F F153L	0.12
L99F M102L	-0.73
L99F M102L F153L	-0.21
L99F M102L V111I	-1.17
L99F M102L V111I F153L	-0.59
L99F V111I	-0.88
L99F V111I F153L	-0.49
M102L V111F	-1.81
V111I F153L	-1.09
T115A R119A	-0.17
T115A S117A	0.95
N116A M120A	0.21
N116D R119M	0.15
S117A N132I	2.00
S117A N132M	1.80
S117A R119A	1.16
S117I N132I	1.40
S117I N132M	2.00
R119A Q123A	-0.17
R119E K135E	-0.80
R119E K135E K147E	-0.95
M120A Q122A	-0.25
L121A A129L	-1.10
L121A A129L L133A F153L	-3.50
L121A A129M	-1.00
L121A A129M F153L	-1.10
L121A A129M V149I	-1.40
L121A A129V L133M F153L	-2.30
L121I A129L L133M F153W	-1.30
L121I A129W L133M	-1.40
L121M A129L L133M V149I F153W	-1.30
L121M L133V F153L	-2.50
W126Y W138Y W158Y	-1.79
D127A E128A	0.24
D127A E128A V131A N132A	0.93
D127A E128A V131A N132A K135A S136A	-0.50
D127A E128A V131A N132A K135A S136A R137A Y139A N140A Q141A	-0.50
D127A E128A V131A N132A L133A	-2.38
E128A V131A	0.42
E128A V131A N132A	0.92
E128A V131A N132A K135A S136A R137A	-0.50
E128A V131A N132A K135A S136A R137A Y139A N140A Q141A	-0.50
E128A V131A N132A L133A	-2.54
A129M F153A	-4.30
V131A N132A	0.59
K135E K147E	-0.90
N144E K147M	0.40

Table A3: Stability data for Human lysozyme: single mutations

mutation	ΔΔG	mutation	ΔΔG	mutation	ΔΔG	mutation	ΔΔG
K1A	-0.60	G37Q	-0.26	D67N	-1.11	V100A	-0.41
K1M	-0.12	Y38A	-2.49	G68A	-0.12	V100F	-1.65
V2A	-1.51	Y38G	-2.32	P71G	-1.60	V100T	-0.29
V2R	-0.38	Y38F	-0.19	G72A	-0.36	D102N	-0.07
V2N	-1.34	Y45F	0.07	V74A	-0.40	P103G	-0.10
V2D	-1.43	A47P	0.10	V74R	-0.07	G105A	-0.62
V2G	-2.29	G48A	0.45	V74N	-0.33	I106A	-0.93
V2I	1.10	D49N	-0.50	V74D	-0.43	I106V	-0.85
V2L	-0.05	R50A	0.43	V74G	-0.22	V110A	0.30
V2M	-0.31	R50G	0.26	V74I	0.45	V110R	0.88
V2F	-0.86	Y54F	-0.96	V74L	0.19	V110N	0.07
V2S	-1.41	I56A	-3.71	V74M	0.65	V110D	0.17
V2Y	-0.36	I56L	-0.10	V74F	-0.29	V110G	0.48
E7Q	-0.76	I56M	-1.77	V74S	-0.38	V110I	0.86
L8T	-3.73	I56F	-4.09	V74Y	-0.24	V110L	0.07
A9S	-0.02	I56T	-3.64	C77A	-4.60	V110M	0.53
G16A	-1.39	I56V	-1.25	H78A	-0.14	V110F	-0.05
D18N	-1.11	Q58A	0.91	H78G	-0.12	V110P	0.50
G19A	-1.77	Q58G	1.86	I89A	-2.70	V110Y	-0.14
Y20F	-0.50	I59A	-1.59	I89V	-0.41	N118A	0.19
R21A	1.31	I59G	-3.83	D91P	-0.40	N118G	0.05
R21G	1.15	I59L	0.00	A92S	0.81	D120N	-0.08
G22A	-1.79	I59M	-1.29	V93A	-0.87	V121A	-1.59
I23A	-2.54	I59F	-0.81	V93T	-0.67	Y124F	-0.36
I23V	-0.39	I59S	-3.59	A96M	0.02	V125A	-1.46
A32L	-0.10	I59T	-2.22	A96S	-1.00	G127A	-0.55
A32S	-0.33	I59Y	-3.78	V99A	-0.94	G129A	0.14
E35L	-0.53	I59V	-1.04	V99T	-0.50	V130A	-0.98
G37A	-0.29	Y63F	-0.24				

Table A4: Stability data for Human lysozyme: multiple mutations

mutation	ΔΔG
P71G P103G	-1.60
C77A C95A	-4.60

Table A5: Stability data for Enterobacteria phage m13 gene V protein: single mutations

mutation	ΔΔG	mutation	ΔΔG	mutation	ΔΔG	mutation	ΔΔG
I6V	-0.70	V35I	-0.65	I47C	-5.25	F68V	-5.00
F13T	-0.70	V35L	-2.70	I47L	-0.65	K69H	-1.25
V19C	-0.30	V35M	-1.05	I47M	-2.15	K69M	0.15
V19T	-0.60	V35F	-3.15	I47F	-1.95	V70C	-3.20
K24V	0.80	V35T	-5.30	I47T	-7.40	V70P	-5.05
Y26R	-0.40	D36N	-1.00	I47V	-2.55	V70T	-3.50
L28V	1.10	D36C	-2.05	T48C	-0.80	F73W	0.80
E30N	-1.10	L37A	-7.70	T48V	0.00	M77A	-2.10
E30M	0.65	L37C	-4.60	L49A	-6.10	M77C	0.00
E30F	2.05	L37I	-1.40	L49C	-4.10	M77I	1.60
L32R	-1.60	L37T	-5.20	L49I	-1.90	M77L	-1.20
L32H	-0.90	L37V	-3.50	L49T	-5.70	M77F	-0.20
L32W	2.80	E40C	-1.60	L49V	-2.90	M77T	-0.80
L32Y	1.00	E40T	-0.40	D50H	-1.55	M77V	1.20
C33A	-0.50	Y41A	-0.40	T62C	-0.70	I78C	-4.40
C33I	-0.90	Y41F	-0.60	T62V	1.30	I78T	-6.60
C33L	-2.60	V43C	-2.10	V63C	-4.10	I78V	-1.30
C33M	-3.50	V43T	-1.60	V63T	-5.00	L81C	-3.70
C33S	-4.25	V45A	-2.10	H64C	0.50	L81T	-5.10
C33T	-4.60	V45C	-0.10	L65P	-1.50	L81V	-0.20
C33V	-0.20	V45L	-3.00	S67C	-3.70	R82C	-1.50
V35A	-2.20	V45T	-3.50	S67T	-1.60	A86T	-0.70
V35C	-1.40	I47A	-7.05	F68L	-4.25	A86V	0.50

Table A6: Stability data for Enterobacteria phage m13 gene V protein: multiple mutations

mutation	ΔΔG
I6V E30F	0.71
I6V M77I	0.34
I6V M77V	-0.04
F13T E30F	1.59
L28V F68L	-3.86
E30F A86T	1.17
E30F A86V	2.05
L32Y R82C	0.15
C33M I47C	-5.73
C33V V35C	-1.57
V35A I47A	-10.80
V35A I47F	-3.63
V35A I47L	-2.94
V35A I47M	-4.46
V35A I47V	-4.46
V35C I47C	-7.15
V35I I47F	-2.04
V35I I47L	-1.14
V35I I47M	-2.83
V35I I47V	-3.06
V35L I47F	-3.97
V35L I47L	-3.54
V35L I47M	-5.36
V35L I47V	-5.05
V35M I47F	-2.33
V35M I47L	-1.65
V35M I47M	-3.56
V35F I47L	-4.16
Y41F F73W	-0.66
V45C R82C	-1.05
H64C F68L	-4.07
L65P F68L	-4.25

B AAindex1 features used by the GPMKL-model

Figure B1 shows which features from the AAindex1 were used in the training of the GPMKL-models that were trained using the stability data of bacteriophage T4 lysozyme (PDB ID 2LZM), human lysozyme (PDB ID 2NWD) and enterobacteria phage m13 gene V protein (PDB ID 1VQB). This figure also shows the fraction of the folds that used these features for each of the three proteins as leave-one-out cross validation was used. Short descriptions of the ids of these features are presented in Table B1. More detailed information from these features and the other features in AAindex1 can be found from http://www.genome.jp/aaindex.

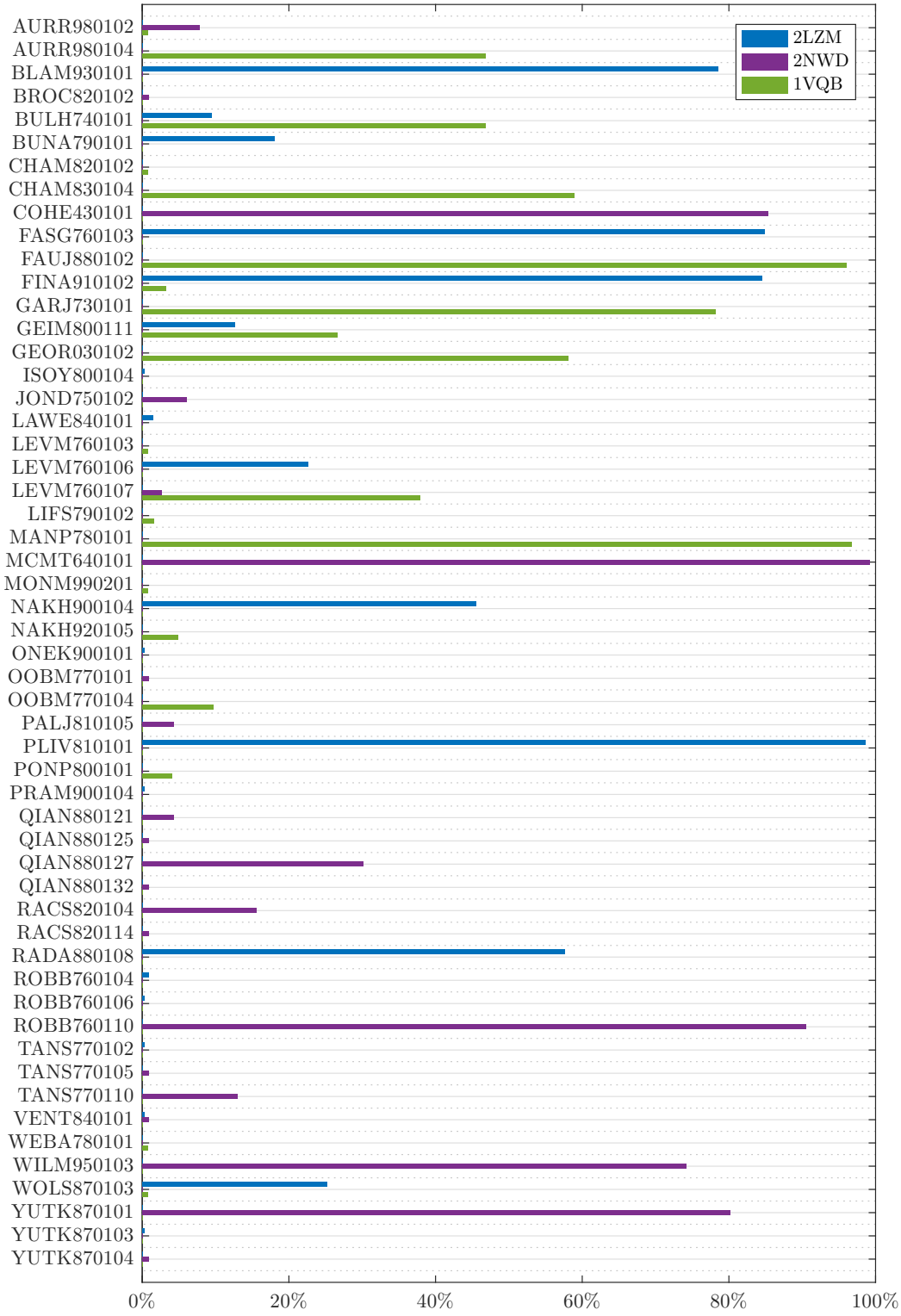


Figure B1: Fractions of folds for 2LZM, 2NWD and 1VQB that used the presented features, when the GPMKL-models were trained using leave-one-out cross validation

Table B1: Ids and descriptions of the AAindex1 features that were used with the GPMKL-model trained with stability data for 2LZM, 2NWD and 1VQB using leave-one-out cross validation.

id	description
AURR980102	Normalized positional residue frequency at helix termini N"' (Aurora-Rose, 1998)
AURR980104	Normalized positional residue frequency at helix termini N' (Aurora-Rose, 1998)
BLAM930101	Alpha helix propensity of position 44 in T4 lysozyme (Blaber et al., 1993)
BROC820102	Retention coefficient in HFBA (Browne et al., 1982)
BULH740101	Transfer free energy to surface (Bull-Breese, 1974)
BUNA790101	alpha-NH chemical shifts (Bundi-Wuthrich, 1979)
CHAM820102	Free energy of solution in water, kcal/mole (Charton-Charton, 1982)
CHAM830104	The number of atoms in the side chain labelled 2+1 (Charton-Charton, 1983)
COHE430101	Partial specific volume (Cohn-Edsall, 1943)
FASG760103	Optical rotation (Fasman, 1976)
FAUJ880102	Smoothed upsilon steric parameter (Fauchere et al., 1988)
FINA910102	Helix initiation parameter at posision i,i+1,i+2 (Finkelstein et al., 1991)
GARJ730101	Partition coefficient (Garel et al., 1973)
GEIM800111	Aperiodic indices for alpha/beta-proteins (Geisow-Roberts, 1980)
GEOR030102	Linker propensity from 1-linker dataset (George-Heringa, 2003)
ISOY800104	Normalized relative frequency of bend R (Isogai et al., 1980)
JOND750102	pK (-COOH) (Jones, 1975)
LAWE840101	Transfer free energy, CHP/water (Lawson et al., 1984)
LEVM760103	Side chain angle theta(AAR) (Levitt, 1976)
LEVM760106	van der Waals parameter R0 (Levitt, 1976)
LEVM760107	van der Waals parameter epsilon (Levitt, 1976)
LIFS790102	Conformational preference for parallel beta-strands (Lifson-Sander, 1979)
MANP780101	Average surrounding hydrophobicity (Manavalan-Ponnuswamy, 1978)
MCMT640101	Refractivity (McMeekin et al., 1964), Cited by Jones (1975)
MONM990201	Averaged turn propensities in a transmembrane helix (Monne et al., 1999)
NAKH900104	Normalized composition of mt-proteins (Nakashima et al., 1990)
NAKH920105	AA composition of MEM of single-spanning proteins (Nakashima-Nishikawa, 1992)
ONEK900101	Delta G values for the peptides extrapolated to 0 M urea (O'Neil-DeGrado, 1990)
OOBM770101	Average non-bonded energy per atom (Oobatake-Ooi, 1977)
OOBM770104	Average non-bonded energy per residue (Oobatake-Ooi, 1977)
PALJ810105	Normalized frequency of turn from LG (Palau et al., 1981)
PLIV810101	Partition coefficient (Pliska et al., 1981)
PONP800101	Surrounding hydrophobicity in folded form (Ponnuswamy et al., 1980)
PRAM900104	Relative frequency in reverse-turn (Prabhakaran, 1990)
QIAN880121	Weights for beta-sheet at the window position of 1 (Qian-Sejnowski, 1988)
QIAN880125	Weights for beta-sheet at the window position of 5 (Qian-Sejnowski, 1988)
QIAN880127	Weights for coil at the window position of -6 (Qian-Sejnowski, 1988)
QIAN880132	Weights for coil at the window position of -1 (Qian-Sejnowski, 1988)
RACS820104	Average relative fractional occurrence in EL(i) (Rackovsky-Scheraga, 1982)
RACS820114	Value of theta(i-1) (Rackovsky-Scheraga, 1982)
RADA880108	Mean polarity (Radzicka-Wolfenden, 1988)
ROBB760104	Information measure for C-terminal helix (Robson-Suzuki, 1976)
ROBB760106	Information measure for pleated-sheet (Robson-Suzuki, 1976)
ROBB760110	Information measure for middle turn (Robson-Suzuki, 1976)
TANS770102	Normalized frequency of isolated helix (Tanaka-Scheraga, 1977)
TANS770105	Normalized frequency of chain reversal S (Tanaka-Scheraga, 1977)
TANS770110	Normalized frequency of chain reversal (Tanaka-Scheraga, 1977)
VENT840101	Bitterness (Venanzi, 1984)
WEBA780101	RF value in high salt chromatography (Weber-Lacey, 1978)
WILM950103	Hydrophobicity coefficient in RP-HPLC, C4 with 0.1%TFA/MeCN/H2O (Wilce et al., 1995)
WOLS870103	Principal property value z3 (Wold et al., 1987)
YUTK870101	Unfolding Gibbs energy in water, pH7.0 (Yutani et al., 1987)
YUTK870103	Activation Gibbs energy of unfolding, pH7.0 (Yutani et al., 1987)
YUTK870104	Activation Gibbs energy of unfolding, pH9.0 (Yutani et al., 1987)