Faculteit Bio-ingenieurswetenschappen

Academiejaar 2011-2012

Learning Pairwise Relations in Bioinformatics: Three Case Studies

Michiel Stock
Promotoren: Prof. Dr. Bernard De Baets en Dr. Willem Waegeman
Tutor: Dr. Willem Waegeman

Masterproef voorgedragen tot het behalen van de graad van
Master in de bio-ingenieurswetenschappen: cel-en genbiotechnologie

Copyright

De auteur en promotor geven de toelating deze scriptie voor consultatie beschikbaar te stellen en delen ervan te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting uitdrukkelijk de bron te vermelden bij het aanhalen van resultaten uit deze scriptie.

The author and promoter give the permission to use this thesis for consultation and to copy parts of it for personal use. Every other use is subject to the copyright laws, more specifically the source must be extensively specified when using from this thesis.

Gent, Juni 2012

De promotor	De co-promotor	De auteur
Prof. dr. Bernard De Baets	dr. Willem Waegeman	Michiel Stock

Woord vooraf

Een thesis is een afsluiter van vijf jaar studeren, plantjes verzamelen, differentiaalvergelijkingen oplossen, titreren, tabakseiwitten opzuiveren en particle trackers schrijven. Sectie 5.2.1 gebruikt principes voor het eerst gezien in de lessen chemie van het eerste bachelor, terwijl Figuur 6.1 geïnspireerd is op een cursus waarvan ik gisteren een examen van heb afgelegd. Deze masterproef gaat over veel dingen die mij er vijf jaar geleden toe hebben aangezet om bio-ingenieurswetenschappen te studeren en mij nog altijd fascineren.

Zonder de hulp van heel wat mensen zou deze thesis er niet gekomen zijn (zie daarom ook Figuur 1). Ik zou hier dan ook de ruimte willen nemen deze personen te bedanken.

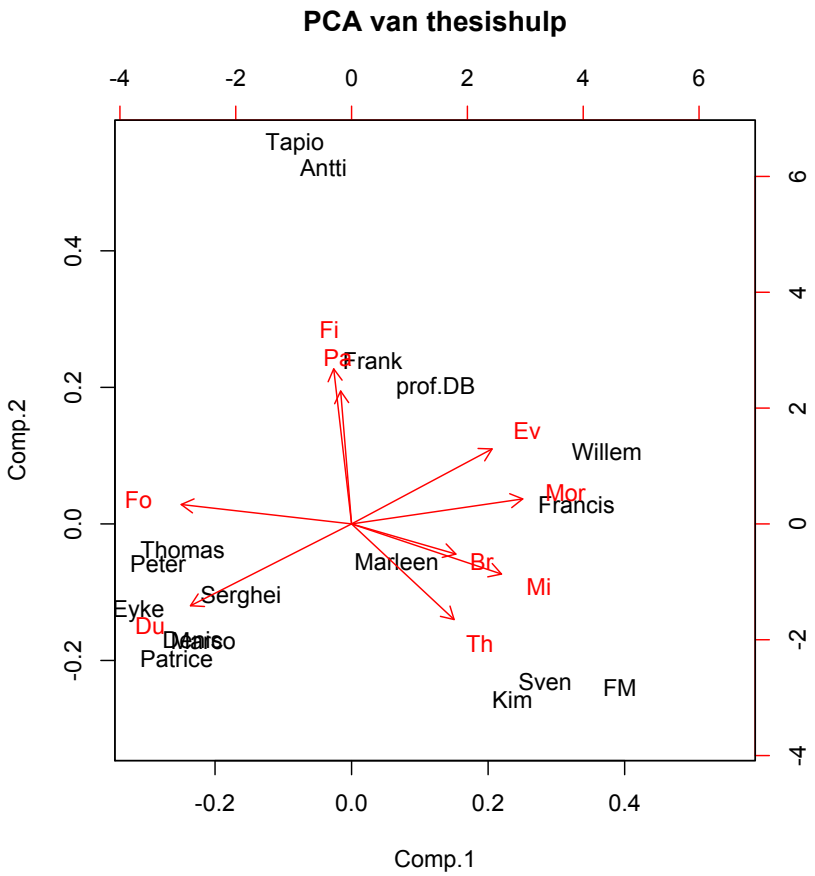


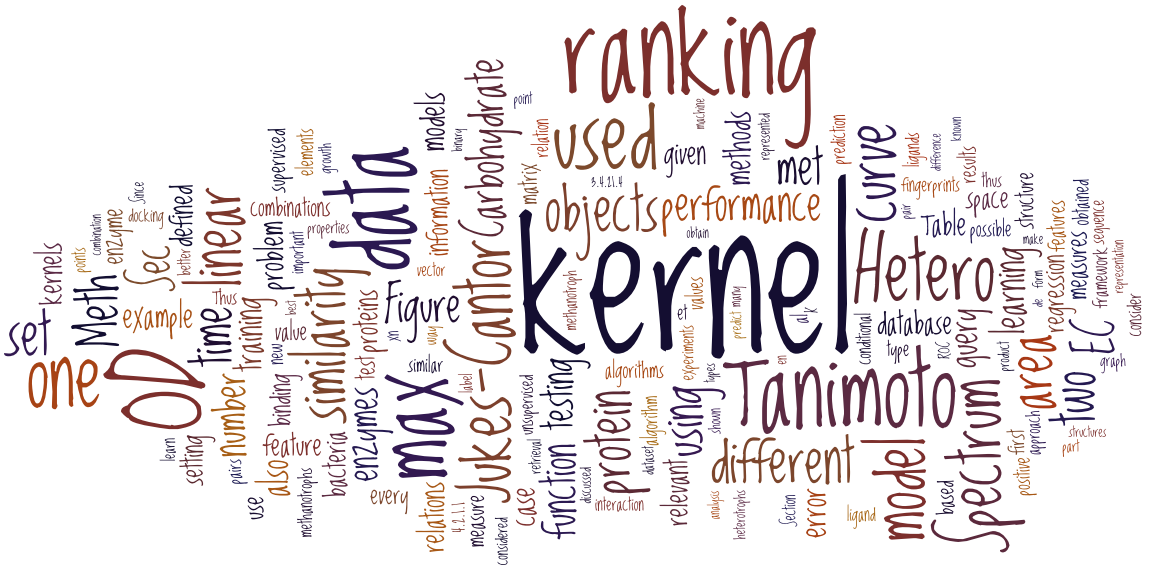
Figure 1: Principale componenten van de mensen betrokken bij deze masterproef.

Eerst en vooral ben ik veel dank aan Willem verschuldigd, mijn co-promotor en tutor, die enorm veel tijd heeft vrijgemaakt om samen te brainstormen, altijd beschikbaar was voor raad en steun en mij de mogelijkheid te geven een maand in Turku aan mijn thesis te werken. Ook ben ik dankbaar om me te pushen al wat resultaten publiek te maken via conferenties en een paper. Daarbij aansluitend dank ik ook prof. De Baets voor zijn raad en hulp.

Kiitos, Tapio en Antti om mij op te vangen en te begeleiden in het verrassend warme Finland vorige zomer.

Zonder data is een machine learning thesis vrij droog en theoretisch, daarom wil ik ook de mensen bedanken die geholpen hebben met het verkrijgen van data en door de nuttige discussies over de mogelijke analyses. FM, Kim en Sven om het mogelijk te maken zelf een nuttig experimentje op poten te zetten en de mensen van Malburg: Denis, Thomas, Marco, Eyke, Peter, Patrice en Serghei voor de eiwit-gerelateerde data, alsook diegene die geholpen hebben met het schrijven van de paper, *dank*.

Finaal zou ik ook nog Francis willen bedanken voor alles, alsook mijn ouders die er de laatste 23 jaar altijd voor mij geweest zijn.



Contents

Chapter 1

Introduction

> I can't be as confident about computer science as I can about biology. Biology easily has 500 years of exciting problems to work on. It's at that level. *Donald Knuth*

In many domains one wants to infer properties of pairs of objects. This is particularly true for biological problems which will be our main sphere of interest. We present a broad framework that is suited for dealing with these kinds of problems, and we try to make the link with bioinformatics where possible. Furthermore, we investigate how to incorporate some ideas from information retrieval to produce results that are more directly useful for biologists.

Our methods are based on generating a joint feature representation of pairs of objects by using the Kronecker product pairwise kernel (KPPK). In the most simple case, methods such as regularized least squares can then be used to make predictions on pairs of objects. It can be shown that the KPPK can learn any arbitrary relation [109], though it is not guaranteed to be the most optimal kernel. By using kernels we can deal with complex data structures such as sequences, graphs and trees, which are frequently encountered in bioinformatics. Our framework is thus more suitable for these kinds of problems than, for example, neural networks or tree based models.

When the ranking error is optimized, the above framework can be used for conditional ranking. Here a set of objects are ranked, conditioned on another object, the query. This means we are now treating the problem as a kind of information retrieval setting where we want to find the most relevant objects. We explore several performance measures such as the ranking error, mean average precision, ROC and CROC curves, to find the most meaningful evaluation criterium for a particular problem.

We test our ideas on some real-world datasets to prove their relevance. As a first application we consider the problem of ranking a database of proteins according to their catalytic similarity to a query protein. We used five state-of-the-art similarity measures as features for the catalytic site of enzymes. We have obtained very promising results for this problem, strongly outperforming the baseline predictor. We showed that our model can give a significant boost compared to using an inferior, but less computationally demanding similarity measure.

In our second similar, but more advanced, application we also use a protein query, not against a database of proteins but for finding ligands that can bind the protein with high

affinity. When using relevant features for these two types of molecules, this framework could serve as a post-processing method for docking algorithms, greatly aiding drug design. From a technical standpoint, this problem differs from the previous one, as this is a dyadic setting, since we consider two different types of objects. We explored how the testing scheme can have a big influence on the performance and practical use of the models.

A third application originates from the field of microbial ecology, where it is known that bacteria form complex ecological networks, exchanging metabolites, nutrients and information. More specifically, the interaction between methanotrophic bacteria and heterotrophic bacteria forms a key interest for biologists. The question we are interested in is how different kinds of each group influence each others growth. In wet lab experiments different combinations of both groups are incubated and their mutual growth density is measured. We constructed a model that ranks the bacteria of one group for their predicted cooperation with our reference bacteria from the other functional group. As features we used a metabolic and phylogenetic representation of the different organisms. This model could give environmental technologists a powerful tool for synthetic ecology.

Chapter 2

Kernel methods for bioinformatics

2.1 Introduction

One could make the very bold statement that bioinformatics is all about comparing things. Take for example the commonly encountered problem of comparing two types of the most important biological sequences, nucleic acids and polypeptides. Many methods have been developed to compare a pair of these sequences by global alignment [65], local alignment [91], extremely fast alignments [2] and aligning multiple sequences at once [66]. These alignments are then used to solve practical biological problems such as protein structure prediction [85], phylogenetic tree construction [83, 33], identification of conserved motifs and domains [31], protein function prediction [23] etc.

The concept behind bioinformatics of extracting useful information from data shows a clear and strong link to machine learning. Indeed, it could be stated that many bioinformatics problems can be posed as machine learning problems [88], as illustrated in Figure 2.1 for some typical problems in proteomics. Larranaga (2006) reviews how machine learning and pattern recognition are applied in real-world problems in six important topics in computational biology: genomics, proteomics, microarrays, systems biology, evolution and text mining. Figure 2.2 shows the interwoven nature of these different fields.

Through the remainder of this master thesis we will present kernel methods as extremely useful tools for computational biology. Kernel methods are naturally suited for dealing

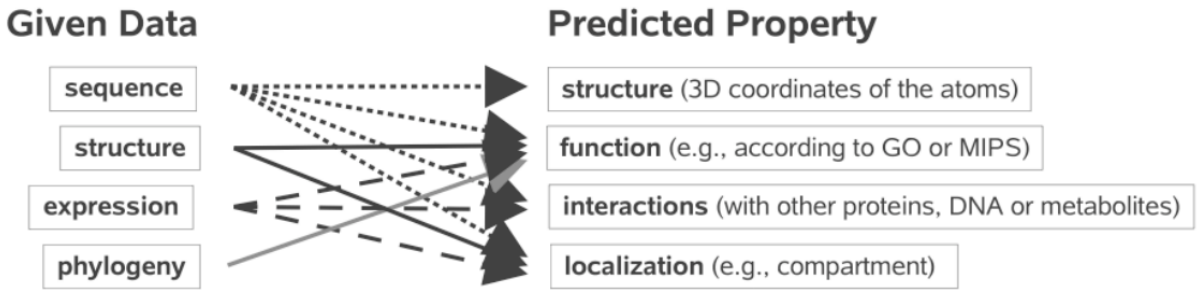


Figure 2.1: Some typical issues in proteomics represented as machine learning problems. [88]

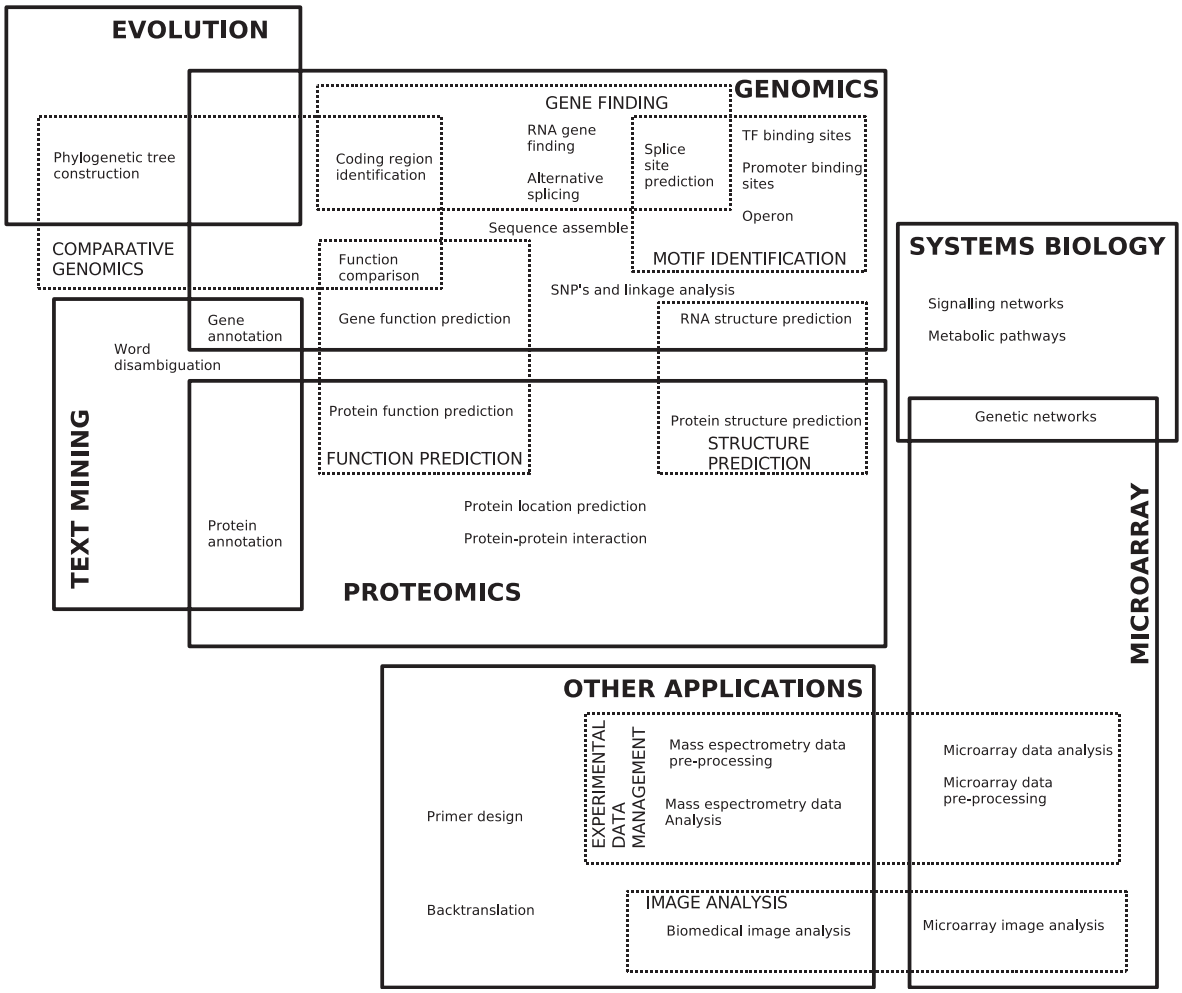


Figure 2.2: Overview of the intersections between bioinformatics and machine learning. [53]

with typical data typed encountered in bioinformatics. Important issues of this data being [88]:

- in large dimension (e.g., microarrays or proteomics data);
- structured (e.g., gene sequences, small molecules, interaction networks, phylogenetic trees...);
- heterogeneous (e.g., vectors, sequences, graphs to describe the same protein);
- in large quantities (e.g., more than 10^6 known protein sequences).

Since kernels can be defined over structured objects and information from different kernel representations can easily be combined, for example by summing the different kernel matrices, they can elegantly deal with these topics.

In the next section the general ideas behind kernel methods will be explained. Section 2.3 gives an overview of some useful kernel-based algorithms. To illustrate the use of these methods in computational biology, a couple of kernels defined over some typical biological objects are discussed in Section 2.4. The reader should be reminded that this chapter

only attempts to give an extremely brief overview of the possibilities of these methods. Much more applications and techniques of kernels and machine learning in general for computational biology are found in the relevant literature.

Armed with this information about kernels over biological objects we are ready to tackle chapter 3, where it is discussed how these techniques can be used to model relations of objects.

2.2 The principle of the kernel trick

Suppose we have a set of N objects $\mathcal{S} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ which we want to use in our learning algorithms. Each object $\mathbf{x}$ is an element of the set $\mathcal{X}$ (and thus $\mathcal{S} \subseteq \mathcal{X}$) which contains all possible objects of this kind. For example, think of $\mathbf{x}$ as ribulose-1,5-bisphosphate carboxylase oxygenase in the set of all proteins involved in photosynthesis. To make an inference about an object $\mathbf{x}$ a feature representation $\phi(\mathbf{x}) \in \mathcal{F}$, with $\mathcal{F}$ a high-dimensional feature space, has to be cast for each possible object $\mathbf{x} \in \mathcal{X}$. Thus the data set $\mathcal{S}$ is represented as a set of individual object representations $\phi(\mathcal{S}) = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N))$. Most data analysis methods outside kernel methods then use this feature mapping in an algorithm to do a prediction.

Kernel methods are different in the sense that instead of the mapping $\phi : \mathcal{X} \to \mathcal{F}$, a real-valued comparison function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is used and hence the data set $\mathcal{S}$ is represented by an $N \times N$ matrix of pairwise comparisons. Such a kernel function $k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ is defined as follows:

Definition 1. *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel iff it is symmetric, that is, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ for any two objects $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, and positive semi-definite, that is,*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for any $N > 0$, any choice of N objects $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$, and any choice of real numbers $c_1, \ldots, c_N \in \mathbb{R}$.

This way of representing the data set as a kernel matrix has the very important property of modularity between the function k on the one hand and the algorithm to process this data representation on the other hand. It also has the advantage that it is often more easy to define comparisons between objects than to construct a meaningful feature representation, especially with complex biological objects such as strings or graphs.

The power of the kernel is that it can be seen as a dot product of the feature representations of two objects:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}'). \tag{2.1}$$

More general, a result by Aronszajn (1950) shows that for an infinitely-dimensional Hilbert space the following theorem applies:

Theorem 1. *For any kernel k on a space $\mathcal{X}$, there exists a Hilbert space $\mathcal{F}$ and a mapping $\phi : \mathcal{X} \to \mathcal{F}$ such that*

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \quad \text{for any } \mathbf{x}, \mathbf{x}' \in \mathcal{X},$$

where $\langle u, v \rangle$ represents the dot product in the Hilbert space between any two points $u, v \in \mathcal{F}$.

This theorem is represented in a more visual way in Figure 2.3.

Since we now have a elegant way to perform a dot product in the Hilbert space, the kernel trick simply states:

Proposition 1. *Any algorithm for vectorial data that can be expressed only in terms of dot products between vectors, can be performed implicitly in the feature space associated with any kernel, by replacing the dot product by a kernel evaluation.*

Thus it is possible to transform linear methods, such as for example linear discriminant analysis into nonlinear method by replacing the dot product by a kernel function. This way we can execute simple linear learning algorithms in a high dimensional Hilbert space with no extra computation costs, save for computing the kernel values[1].

For example, suppose one wants to perform clustering in the feature space [30]. Most cluster algorithms require a distance between the objects, usually the Euclidian distance. This distance in the Hilbert space between points $\mathbf{x}$ and $\mathbf{x}'$ is defined as:

$$d(\mathbf{x}, \mathbf{x}') := ||\phi(\mathbf{x}) - \phi(\mathbf{x}')|| \,. \tag{2.2}$$

Using the following equality, Equation 2.2 can be expressed only as dot products:

$$||\phi(\mathbf{x}) - \phi(\mathbf{x}')||^2 = \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle + \langle \phi(\mathbf{x}'), \phi(\mathbf{x}') \rangle - 2\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \tag{2.3}$$

which allows us to compute the distance only in terms of the kernel:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}')}. \tag{2.4}$$

By these means a distance matrix can be constructed which can be plugged in standard cluster algorithms or more advanced tools, such as self-organizing maps or the k-nearest neighbor algorithm.

2.3 Some important kernel algorithms

2.3.1 Support vector machines

Support vector machines (SVMs) are very popular and powerful tools for regression and classification. Their convex and sparse nature makes them very computationally efficient to train and use. We will present a quick overview how SVMs are derived for binary classification. We want to solve a binary classification problem by using the following linear model in the Hilbert space:

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b, \tag{2.5}$$

with $\mathbf{w}$ and b parameters. The data set contains N samples $\mathbf{x}_1 \ldots \mathbf{x}_N$ with corresponding target values $t_1 \ldots t_N$, where $t_n \in \{-1, 1\}$ and new data points are classified according to the sign of $y(\mathbf{x})$.

[1] Though Section 2.4 may indicate that calculating some kernel values is not always so straightforward...

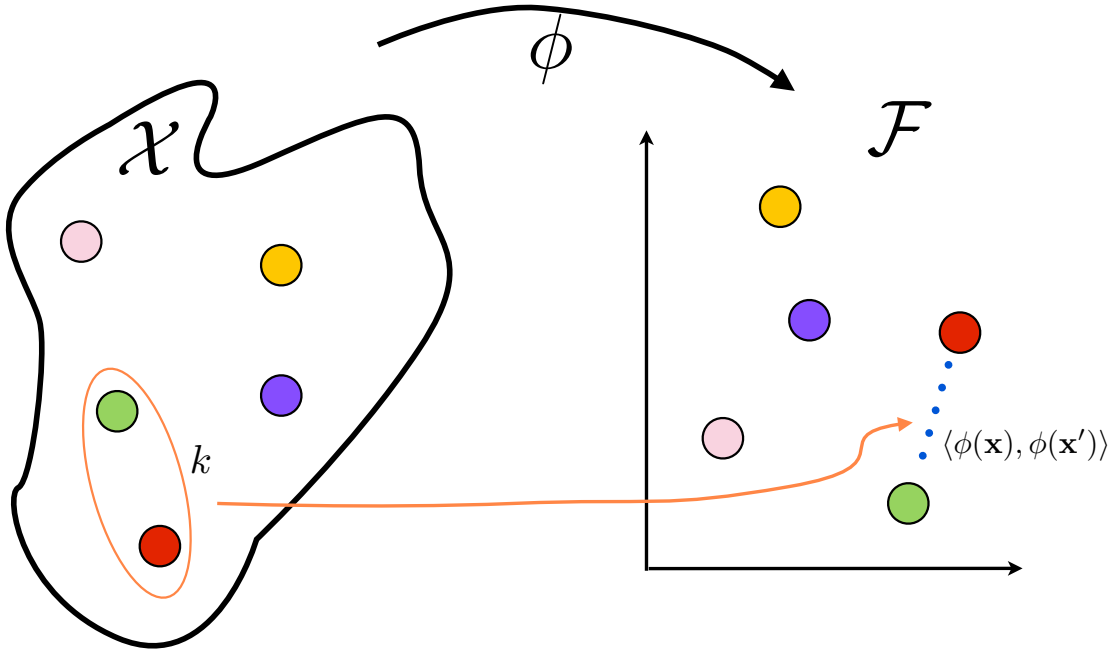


Figure 2.3: Applying a kernel function to two objects (in orange) is equivalent to taking the inner product after the space $\mathcal{X}$ is mapped to a Hilbert space $\mathcal{F}$. Image after [88].

Assuming that the data is linearly separable in the Hilbert space, Equation 4.3 is optimized to separate the classes and maximize the margin. The margin is defined as the perpendicular distance between the decision boundary and the closed data points. This is shown in Figure 2.4.

It can be shown that the maximum margin solution is found by solving:

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n(\mathbf{w}^T \phi(\mathbf{x}) + b) \right] \right\} \tag{2.6}$$

It can be shown that this problem can be cast as a quadratic programming problem, which can be solved using Lagrange multipliers. When cast in the dual representation we have to maximize:

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m), \tag{2.7}$$

with $\mathbf{a} = (a_1, \ldots, a_N)^T$ a vector of Lagrange multipliers, which are subjected to the constraints

$$a_n \geq 0, \quad n = 1, \ldots, N, \tag{2.8}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{2.9}$$

Equation 4.3 can also be written by using kernels:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b, \tag{2.10}$$

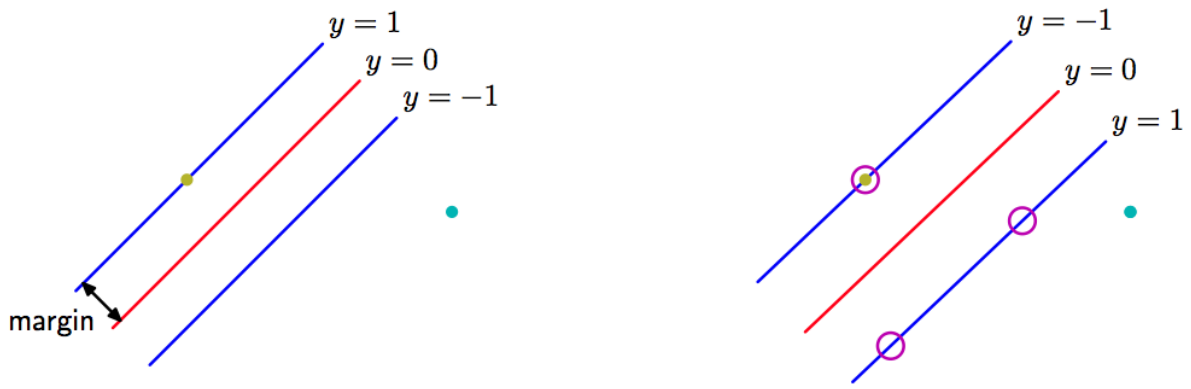


Figure 2.4: The margin is defined as the perpendicular distance between the decision boundary and the closed data points. In the right Figure the margin is maximized leading to a particular choice of decision boundary. The location of this boundary is determined by only a subset of the data points, denoted as the support vectors (encircled). [11]

where this type of constrained optimization must satisfy the Karush-Kuhn-Tucker conditions:

$$a_n \geq 0, \tag{2.11}$$
$$t_n y(\mathbf{x}_n) - 1 \geq 0, \tag{2.12}$$
$$a_n(t_n y(\mathbf{x}_n) - 1) = 0. \tag{2.13}$$

Thus for every data point, either $a_n = 0$ or $t_n y(\mathbf{x}_n) = 1$. Any data point for which $a_n = 0$ will not appear in Equation 2.10 and hence will not have any influence in the predictions for new data points. This sparsity in the use of the data is a very interesting property from a computational point of view as it usually allows us to discard a large part of the data matrix after training the model, while we only retain the data points of the margin, the support vectors. This is especially useful when calculating the kernel matrix is a large part of the computation cost, as often the case for the kernels discussed in Section 2.4. An example for synthetic data is shown in Figure 2.5.

Because in practice the class distributions often overlap, so called slack variables are introduced, which allow the points to be misclassified to a certain degree. An extra parameter C is introduced to regularize the extent to which misclassifications are penalized. Though this complicates the equations somewhat, it does not change anything to the general reasoning.

A very similar idea is applied to construct an SVM framework for regression. Here the line one wants to learn is enclosed by a tube, only points on and outside this tube have an influence on the shape of this line. This is conceptually represented in Figure 2.6.

2.3.2 Regularized least squares

Regularized least squares (RLS) can be seen as a more simple version of support vector machines. It is exactly what's on the tin: performing a least squares regression in the

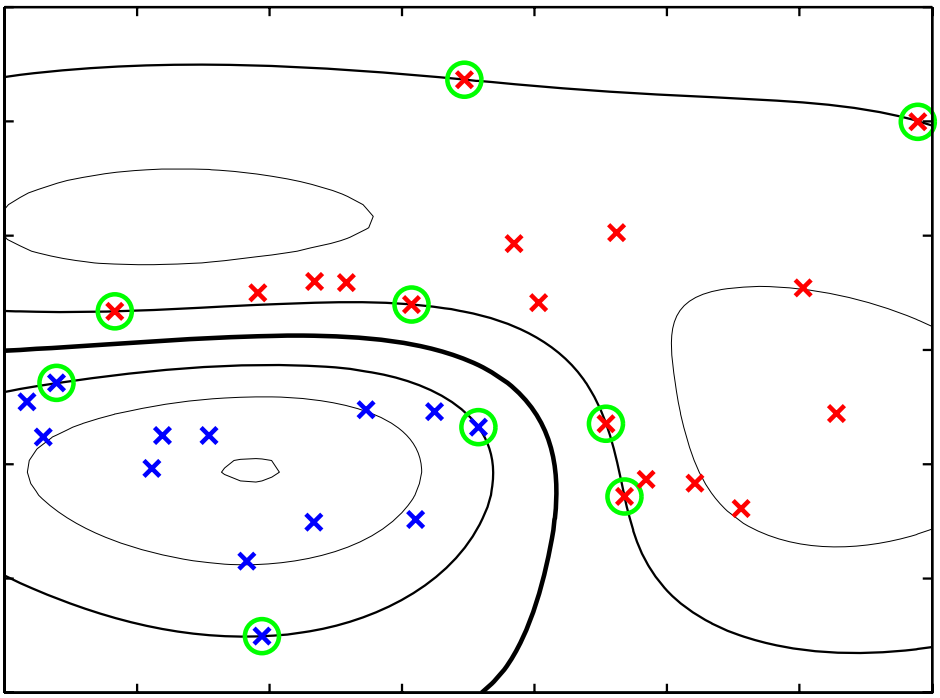


Figure 2.5: An example of the support vector machine on a synthetic data set with two classes and a Gaussian kernel function. The decision boundary, the margin boundaries and the support vectors are also shown. [11]

Hilbert space while using some kind of regularization. Thus the function to be minimized is given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}, \tag{2.14}$$

where the regularization parameter $\lambda \geq 0$. This is a natural setting when performing regression (t_n is real), but can also be used for binary classification. In this case the loss function 'makes no sense', but works great. If the gradient of $J(\mathbf{w})$ is set to zero, it is seen that the solution for $\mathbf{w}$ takes the form of a linear combination of the vectors $\phi(\mathbf{x}_n)$, with coefficients that are functions of $\mathbf{w}$, of the form

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}\phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) = \mathbf{\Phi}^T\mathbf{a} \tag{2.15}$$

where $\mathbf{\Phi}$ is the design matrix whose n^{th} row is given by $\phi(\mathbf{x}_n)^T$. Here the vector $\mathbf{a} = (a_1, \ldots, a_N)^T$, and we have defined

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}. \tag{2.16}$$

If the Gram matrix with all the kernel values is defined as $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T$ and Equation 2.15 is used to eliminate $\mathbf{w}$ from Equation 2.16 and solving the least squares for $\mathbf{a}$ we obtain:

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}. \tag{2.17}$$

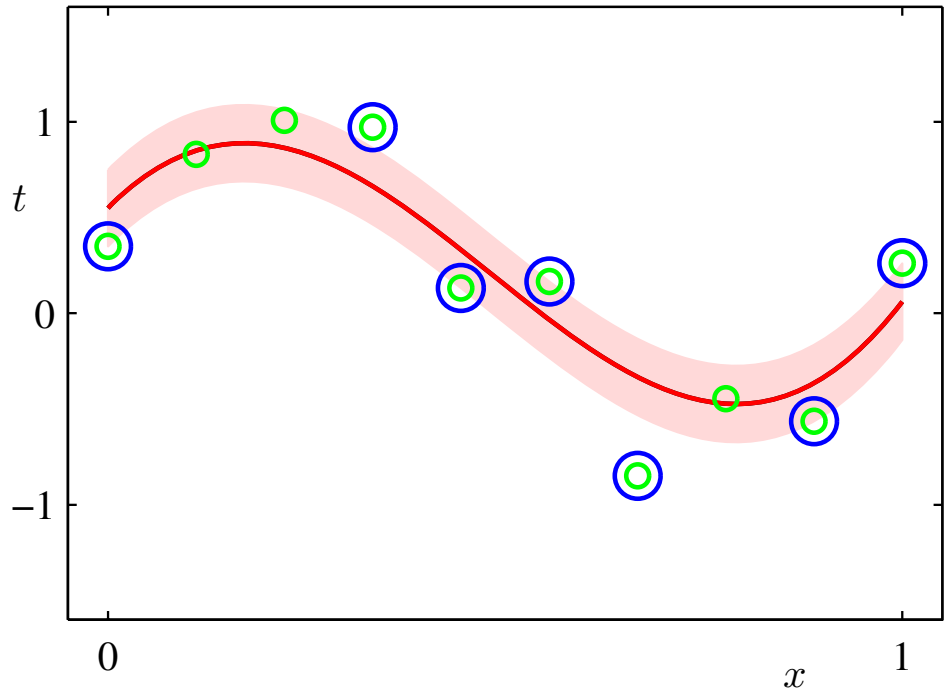


Figure 2.6: Conceptual representation of the support vector machine in a regression setting. The predicted regression curve is shown in red and the tube corresponds to the shaded region, support vectors are encircled in blue. [11]

When making a new prediction we can use the following model:

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) = \mathbf{a}^T\mathbf{\Phi}\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}, \tag{2.18}$$

where the vector $\mathbf{k}(\mathbf{x})$ is defined with the elements $k_n = k(\mathbf{x}_n, \mathbf{x})$.

It is not advised to calculate explicitly he inverse of a matrix, as implied in Equation 2.17, when one can obtain the solution of interest (the vector $\mathbf{a}$) by just solving a set of linear equations.

The simplicity of this algorithm compared to that of support vector machines may led one to believe it is inferior in performance. The performance of RLS is comparable though and some very efficient schemes exist for building these models, as it is possible to perform extensive cross validation without rebuilding the model in each round [70]. It is also possible to implement a RLS algorithm in one line of Matlab or (preferably) Python code.

In this light it may be of interest to show this nice quote about numeric computing:

> You should be asking how the answers will be used and what is really needed from the computation. Time and time again someone will ask for the inverse of a matrix when all that is needed is the solution of a linear system; for an interpolating polynomial when all that is needed is its values at some point; for the solution of an ODE at a sequence of points when all that is needed is the limiting, steady-state value. A common complaint is that least

squares curve fitting couldn't possibly work on this data set and some more complicated method is needed; in almost all such cases, least squares curve-fitting will work just fine because it is so very robust
Leader, Numerical Analysis and Scientic Computation

2.3.3 Kernel principal component analysis

Principal component analysis (PCA) is a commonly used technique for data exploring, feature extraction, dimensionality reduction and data visualization. There are two common ways to look at PCA: it can be viewed as the orthogonal projection of the data onto a lower dimensional linear subspace, the principal subspace, such that the variance is maximized. Or, equivalently, it can be defined as the linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projection [11]. Kernel PCA is simply these principles executed in the Hilbert space by putting the kernel trick to use. This way PCA is not restricted to analyzing linear patterns.

We derive kernel PCA without losing ourselves in too much detail for regular PCA. Using the notation of above, a covariance matrix $\mathbf{C}$ in the feature space of the data set $\mathcal{S} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ can be constructed:

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T. \tag{2.19}$$

In this equation we assume that these feature representations have a zero mean, thus $\sum_n \phi(\mathbf{x}_n) = 0$. We will deal with this problem in a moment. The eigenvector expansion that is needed is defined as:

$$\mathbf{C}\mathbf{v}_i = \eta_i \mathbf{v}_i. \tag{2.20}$$

We want to perform this expansion without working explicitly in the feature space. From Equation 2.19 we can rewrite Equation 2.20 as:

$$\frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^T \mathbf{v}_i\} = \eta_i \mathbf{v}_i, \tag{2.21}$$

which allows us to see that we can write the vector $\mathbf{v}_i$ as a linear combination of the $\phi(\mathbf{x}_n)$. Thus it can be written in the form:

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n). \tag{2.22}$$

This allows us to rewrite the eigenvector equation as:

$$\frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_m) = \eta_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n). \tag{2.23}$$

By multiplying with $\phi(\mathbf{x}_l)$ everything can be expressed in the form of a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Writing this in compact matrix form we obtain:

$$\mathbf{K}^2\mathbf{a}_i = \eta_i N \mathbf{K}\mathbf{a}_i, \tag{2.24}$$

with $\mathbf{a}_i$ an N-dimensional column vector with elements a_{in} for $n = 1, \ldots, N$. And because a positive definite kernel matrix is always invertible:

$$\mathbf{K}\mathbf{a}_i = \eta_i N \mathbf{a}_i. \tag{2.25}$$

Having solved the eigenvector problem, the resulting principal components projections can be cast in terms of the kernel functions, so that the projection of a point $\mathbf{x}$ onto eigenvector i is given by

$$y_i(\mathbf{x}) = \phi(\mathbf{x})^T v_i = \sum_{n=1}^{N} a_{in} \phi(\mathbf{x})^T \phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n). \tag{2.26}$$

This leaves us with the issue of centralized feature vectors. This is simply solved by using a 'centralized' kernel matrix $\tilde{\mathbf{K}}$ for the calculations, which we present without proof as:

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N \mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N \mathbf{K} \mathbf{1}_N, \tag{2.27}$$

where $\mathbf{1}_N$ denotes the $N \times N$ matrix of which every element has the value $1/N$.
Note that when using a linear kernel the standard PCA algorithm is recovered, making kernel PCA actually the more general case.

2.4 Some useful kernel functions

2.4.1 Kernels for sequences

As strings are probably the most important data types in bioinformatics, it is of no surprise that many types of kernels are defined to allow learning properties of (*in casu* protein or nucleic acid) sequences. The kernels discussed here are used for discriminative models, in contrast to the generative models, based on, for example, hidden Markov models [20].

We discuss three relatively simple kernels based on substrings and see how these can be combined to form the homology kernel [24]. The spectrum kernel can compare two sequences by considering exact matches of substrings. The sparse kernel and the groupings kernel in contrast are defined over the substrings, rather than the complete sequence. The former is an extension in the sense that an inexact match between two substrings is quantified while the latter incorporates the notion that the different 'characters' that compose the sequence can be divided in different groups. By using the idea of the spectrum kernel of comparing strings by matching substrings, while comparing those substrings is based on the sparse and grouping kernel, one obtains the homology kernel that processes hierarchical information of the sequences.

The first string kernel is called the spectrum kernel [57]. Suppose that each sequence is denoted $s_i \in \Sigma^*$ where Σ is the alphabet of amino acids or nucleotides, depending of the application. A k-mer, $a \in \Sigma^k$ is a sequence of length k. The sequence s_i contains a iff $s_i = uav$. Let $N(a, s_i)$ be the number of times a appears in sequence s_i. With this

notation in mind the spectrum kernel is defined as:

$$SK_k(s_i, s_j) = \sum_{m \in \Sigma^k} N(m, s_i) * N(m, s_j). \tag{2.28}$$

This kernel is biased toward sequences that contain multiple instances of the same k-mer. This can easily be seen when considering a pair of sequences that contain the k-mer twice they gain a score of 4 while having two different k-mers in common only gives a score of 2. For this reason a normalized spectrum kernel is defined as:

$$NSK_k(s_i, s_j) = \sum_{m \in \Sigma^k} \min(N(m, s_i), N(m, s_j)). \tag{2.29}$$

The normalized kernel has an explicitly defined mapping $\phi(s)$. If the maximum length of the sequences is n, then $\phi(s)$ defines a map of $s \in \Sigma^n$ to a feature space of dimension $\Sigma^k n$ where each dimension is indexed by a k-mer a and an integer $1 \geq i \geq n$. The mapping is as follows:

$$\phi_{w,i}(s) = \begin{cases} 1 & \text{if } k\text{-mer } w \text{ appears at least } i \text{ times in } s, \\ 0 & \text{otherwise.} \end{cases} \tag{2.30}$$

The sparse kernel, designed for comparing the substrings a en b, is somewhat more flexible as it allows inexact matches. For two substrings the sparse kernel is defined as

$$SpK(a, b) = \alpha^{d_H(a,b)}, \tag{2.31}$$

with $d_H(a, b)$ the Hamming distance between k-mers a and b and α a parameter between 0 and 1. With a low value of α substrings with few mismatches are heavily penalized while the opposite is true for high values of this parameter. To understand the feature mapping we have to expand the alphabet Σ with a "wildcard" character denoted by "$*$" which matches every character $c \in \Sigma$. Let $\Sigma' = \Sigma \bigcup \{*\}$, then the feature mapping of a substring a is given by

$$\phi_w(a) = \begin{cases} \sqrt{\alpha^l (1-\alpha)^{(k-l)}} & \text{if } w \in \Sigma'^k \text{ matches } a \in \Sigma^k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.32}$$

Here l denotes the number of wildcards in w.

The third kernel to be discussed is the groupings kernel, also to be used for comparing substrings. This kernel incorporates the notion that some 'letters' of the alphabet share similar properties. For example in DNA adenine mutates more easily in guanine (transition) than in cysteine (transversion) or serine is chemically more similar to threonine than proline. Thus these elements that compose the sequences can be grouped according to a relevant property. The contribution between two k-mers a and b from different sequences is defined to be

$$SpKG(a, b) = \alpha^{d_M(a,b)} \beta^{d_G(a,b)}, \tag{2.33}$$

with $d_G(a, b)$ the number of mismatches between a and b within a group and $d_M(a, b)$ is the number of mismatches between groups and $0 < \alpha < \beta < 1$. For a set of groups G, the

augmented alphabet Σ'' is defined: $\Sigma'' = \Sigma \bigcup \{*\} \bigcup G$. The feature mapping performed by the groupings kernel is given by:

$$\phi_w(a) = \begin{cases} \sqrt{\alpha^M(\beta-\alpha)^G(1-\beta)^{k-M-G}} & \text{if } k\text{-mer } a \text{ matches } w, \\ 0 & \text{otherwise.} \end{cases} \tag{2.34}$$

were M is the number of wildcards in w and G is the number of group characters in w.

The homology kernel is build by not only embedding the the substrings in a sequence but in a alignment of homologue sequences obtained by BLAST over a large database. This kernel is defined as the normalized sparse kernel with groupings applied to this substrings.

2.4.2 Kernels for protein structures

Though protein sequence implies protein structure, in many cases it is desirable to learn directly from the structures if they are available. Indeed, the increasing number of structures in the UniProt database and the vast improvements in homology structure prediction make such approaches possible. There exists kernels that can deal with protein structures, but there are also many relevant similarity measures that can either directly serve as kernels or be modified to become valid kernels.

It is possible to compare global fold similarity, though this is not necessarily the most relevant approach. The function of many proteins seems to be determined by a limited number of specific amino acid residues at the catalytic site [61]. Thus, it is probably more fruitful to compare only the part of the protein most important for its function. For most proteins it is possible to determine the active site as the largest cleft on the surface of the protein [54], which will be used for subsequent analysis. This will prove to be of particular relevance in two of our case studies: enzyme function prediction and protein docking (Chapters 4 and 5).

The relevant binding site is usually represented as a labeled point cloud (LPC). This is a finite set of points, where each point is not only associated with a position in the three-dimensional space, but also with a discrete class label that represents a specific property. Thus the points could represent atoms, amino acid residues or other functional abstractions, such as hydrophobic, positive or aliphatic groups. To construct a similarity two of these LPCs have to be aligned: the optimal translation and rotation has to be found, usually to minimize some squared deviation between the clouds.

There are multiple ways to compare these LPCs, Hoffmann et al. (2010) consider the convolution of the clouds to directly obtain a valid kernel function. This and similar methods are called the geometric approach, as the location of the points is matched in space. This is contrasted with graph-based approaches, such as proposed bySchmitt et al. (2002). The cavity is represented as a set of pseudocenters, labeled with one of five properties relevant for binding a ligand. These pseucenters are structured as the nodes of a graph where the edges represent the distances between the nodes. The problem is then reformulated as finding a similarity between two graphs. These methods are visualized in Figure 2.7. Fober et al. (2011) propose another similar method based on fuzzy equivalence, which is a compromise between the two above approaches.

These kernels and similarities have proven to be very powerful: they are able to detect

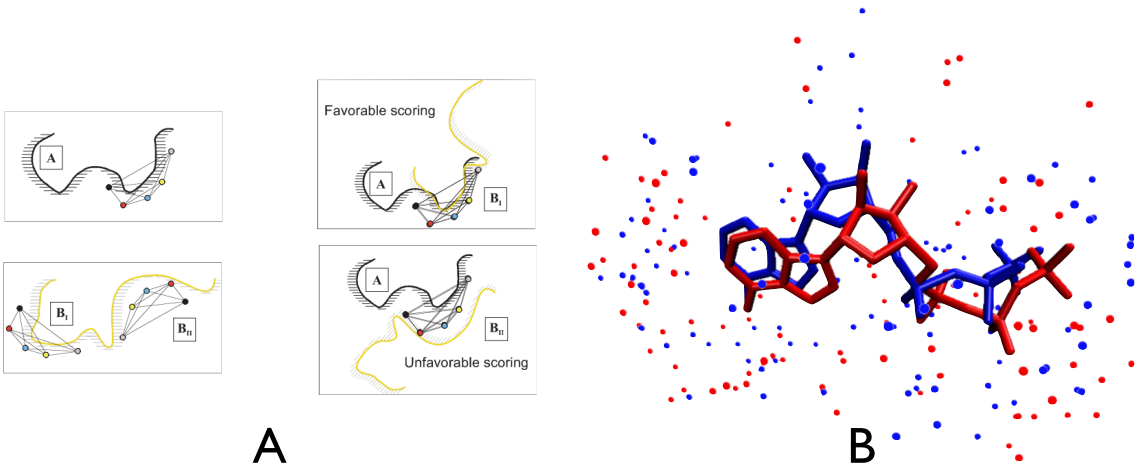


Figure 2.7: Calculating the similarity between protein cavities when they are represented as labeled point clouds. Two broad classes of methods are of particular exist: (A) the graph based approach [86] and (B) the geometric approach [37].

functional similarities when the proteins share not sequence or overall structure similarities. In practice they are very hard to compute due to they computational demanding nature and the complex nature of working with protein strucures.

2.4.3 Kernels for graphs

Graphs are extremely important data structures in bioinformatics. Graphs can be used to describe the relations between objects, something which will be discussed in more detail in Chapter 3. Protein-protein interactions, organic molecules [78], interacting amino acids within a protein [4] etc. can typically represented a graphs. For example many authors consider the problem of using machine learning methods to try to predict protein interaction or gene regulatory networks [56, 98, 113, 100, 29].

Rather than trying to infer a network, we consider the setting where the data used for building a model is presented as a network. Two different cases can be distinguished:

- The objects are nodes of a graph. For example when one wants to make predictions about properties of enzymes when a metabolic network is given. The goal is to construct a kernel that can quantify the 'distance' *between nodes within a graph*.
- One object is represented as a graph. For example in chemoinformatics, a molecule is often represented as a graph or when one wants to compare protein networks across different species or tissues. The goal is in this case to have a kernel that can quantify the similarity *between graphs*.

When learning with objects within one graph one has to quantify the degree in which two nodes are connected. For example when two nodes are not connected, but if a lot of (strongly weighted in case of a weighted graph) edges are leading from one node to another this should still result in a high kernel value. This is embedded in the diffusion kernel [51]. This kernel is based on the matrix exponential of the Laplacian matrix of the

graph, which is inspired by the diffusion equation[2]. This way long-range relationships between the vertices are captured, a parameter that has to be tuned can be used to focus of closer or farther nodes. Thus, informally, this kernel quantifies how quickly 'heat' can spread through a graph.

Kernels that can be used to compare complete graphs share a similar reasoning [105]: given a pair of graphs, perform random walks on both, and count the number of matching walks. Thus it is possible to capture (dis)similarities in topology. As a special, but biologically relevant case of comparing graphsVert et al. (2002), consider tree kernels for using phylogenetic trees as input.

Most kernels for graphs are very computationally demanding [105, 88]. There is a great body of research for calculating these kernel efficiently. For the special cases such as symmetric graphs or trees, computational shortcuts are derived.

2.4.4 Kernels for fingerprints

Sometimes an object is represented as a long string of binary values: fingerprints. For example, chemical compounds are often represented as a series of fingerprints, which correspond to depth-first searches taken in the molecular graph. These fingerprints can in principe be processed with a linear, Gaussian or polynomial kernel, thus just treating them as regular feature vectors.

Swamidass et al. (2005) present some kernels functions specifically for fingerprints. The reasoning is that two fingerprints, represented as the binary vectors $\mathbf{x}_m$ and $\mathbf{x}_n$, are sets which have to be compared. First they consider the *fingerprint similarity* defined as the number of common elements in the set:

$$K_d(\mathbf{x}_m, \mathbf{x}_n) = \langle \mathbf{x}_m, \mathbf{x}_n \rangle, \tag{2.35}$$

which is nothing more than a new name for a linear kernel. This kernel is not normalized in the sense that two sets both with many elements will score higher than two sets with few elements while the fraction of common elements is the same. This does not have to be a negative property but it is something to keep in mind.

Building on this fingerprint similarity the *Tanimoto kernel* (also called Jaccard index) is defined as the number of elements in the intersection between the two sets divided by the number of elements in the union of the sets:

$$K_d^t(\mathbf{x}_m, \mathbf{x}_n) = \frac{K_d(\mathbf{x}_m, \mathbf{x}_n)}{K_d(\mathbf{x}_m, \mathbf{x}_m) + K_d(\mathbf{x}_n, \mathbf{x}_n) - K_d(\mathbf{x}_m, \mathbf{x}_n)}. \tag{2.36}$$

The authors have shown that this kernel could be used for state-of-the-art prediction of mutagenicity, toxicity and anti-cancer activity of small compounds.

[2] $\frac{\partial}{\partial t}\psi = \mu\Delta\psi$

Chapter 3

Learning relations between objects

3.1 Using pairs of objects as instances

In the previous chapter we attempted to show that kernel methods are very flexible learning algorithms that can deal with a variarity of types of objects. Often though, in bioinformatics one does not only want to predict properties of objects, but of pairs of objects, for example whether two proteins are interacting [113, 104]. Examples outside the field of bioinformatics are also numerous: text mining [114], social network analysis [96] etc.

Relational learning is equivalent to inferring labels on edges between vertices of a graph. When considering relations between two objects one can define monadic and dyadic relations. The former only deals with objects of the same 'type', for example inferring interactions between proteins, as depicted in Figure 3.1. The dyadic framework on the other hand treats prediction about a pair of objects of a different type, for example whether a small molecule can bind a protein. This setting can be viewed in Figure 3.2.

Many types of relations have been defined and the framework of this section can be applied to most of them. Let $Q(v, v')$ denote a binary relation on an object space $\mathcal{V}$, then one can distinguish the following basic settings [109]:

- Crisp relations, when $Q : \mathcal{V} \to \{0, 1\}$, this corresponds to a binary classification with the pair of objects as input.
- $[0, 1]$-valued relations when the relation is of the form $Q : \mathcal{V}^2 \to [0, 1]$, thus resulting in a regression type of learning setting. The restriction to the interval $[0, 1]$ is largely historical to provide a link with fuzzy set theory and decision theory. Real values can be obtained by performing a scaling, but in most of the applications in this thesis we will not even bother with this and just regress the unrestricted label.
- Ordinal-valued relations: when a certain order can be derived, but the actual values or differences between the values do not matter.

Assuming the data is structured as a graph $G = (\mathcal{V}, \mathcal{E}, Q)$ where $\mathcal{V}$ corresponds to the set of nodes v and $\mathcal{E} \subseteq \mathcal{V}^2$ represents the set of edges e, for which training labels are provided in terms of relations. These relations are represented by training weights y_e on the edges, generated by the unknown underlying relation $Q : \mathcal{V}^2 \to [0, 1]$. Historically these relations are confined to the interval $[0, 1]$, this will be of importance for some properties discussed

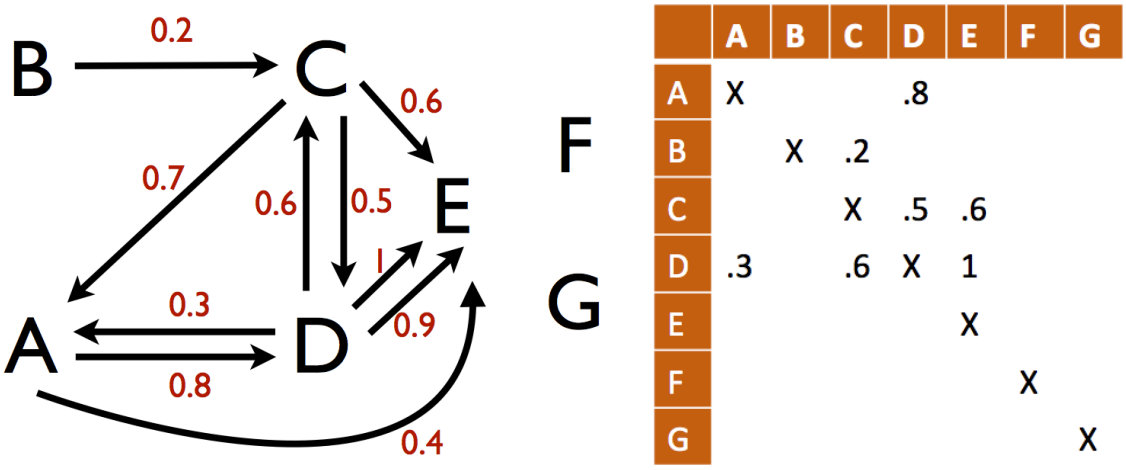


Figure 3.1: Relational learning presented as learning the labels on a graph (in this case a real value between 0 and 1). Since the machine learning framework is based on features of the individual objects, a generalization can be made to make predictions of pairs where one (for example $\{A, F\}$) or both (for example $\{F, G\}$) of the objects do not appear in the training set. After [109].

in Section 3.2. It is alway possible, when desired, to make an extension to real-valued relations $h : \mathcal{V}^2 \to \mathbb{R}$ by using an increasing mapping $\sigma : \mathbb{R} \to [0, 1]$ such that

$$Q(v, v') = \sigma(h(v, v')), \quad \forall (v, v') \in \mathcal{V}^2. \tag{3.1}$$

The learning problem is formulated as the selection of a suitable function $h \in \mathcal{H}$, with $\mathcal{H}$ a certain hypothesis space, in particular a reproducing kernel Hilbert space (RKHS). Thus the hypothesis $h : \mathcal{V}^2 \to \mathbb{R}$ is denoted as

$$h(e) = \mathbf{w}^T \Phi(e), \tag{3.2}$$

with $\mathbf{w}$ a vector of parameter to be estimated from the training data and Φ a joint feature mapping for edges in the graph. Since we have discussed how to learn these types of models in the previous chapter, we would like to focus on how to build a meaningful feature representation for the edges using kernels.

To derive a relevant kernel for pairs of objects, this feature mapping based on the Kronecker product is proposed to express pairwise interactions between features of nodes:

$$\Phi(e) = \Phi(v, v') = \phi(v) \otimes \phi(v'), \tag{3.3}$$

where ϕ denoted the feature making of the individual nodes. This feature mapping is shown[1] to correspond to the the tensor product pairwise kernel [9]

$$K^{\Phi}_{\otimes}(e, \bar{e}) = K^{\Phi}_{\otimes}(v, v', \bar{v}, \bar{v}') = K^{\phi}(v, \bar{v}) K^{\phi}(v', \bar{v}'), \tag{3.4}$$

with K^{ϕ} the kernel corresponding to ϕ or, stated differently, the node kernel.

[1]Remember the mixed-product property for the Kronecker product: $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$

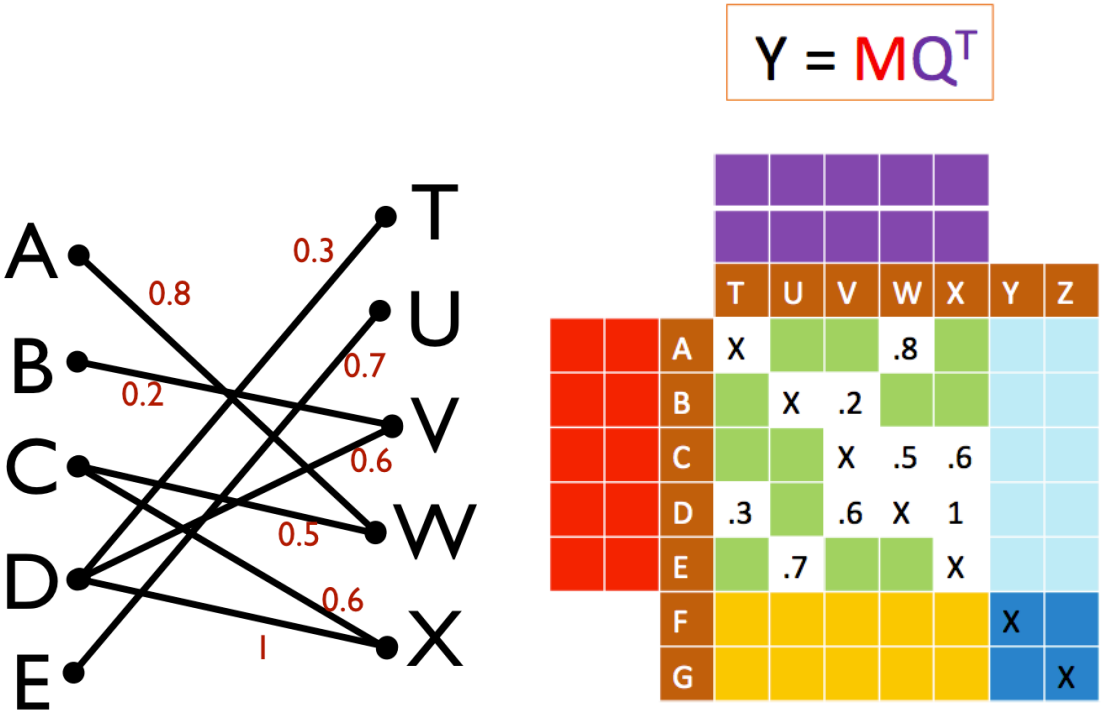


Figure 3.2: Relational learning presented as learning the labels on a graph (in this case a real value between 0 and 1) in the case of dyadic data. Features of the individual objects are used to make predictions about the labels. This problem can also be treated as a matrix factorization problem when only with an incomplete matrix, as seen in the green part of the right figure. In the most simple case, one can predict latent variables of both types of object (purple and red) of which the matrix product is an approximation of the missing values [94].

This is a very beautiful and efficient way to generalize kernel methods to deal with pairs of objects. By simply taking the Kronecker product of the kernel or kernels for the nodes one obtains a kernel representing the edges that can simply be plugged in your favorite kernel algorithm to learn the label of the edges. Waegeman et al. (2012) prove that the Kronecker product pairwise kernel is indeed universal and can be used to learn arbitrary relations.

Theorem 2. *Let us assume that the space of nodes $\mathcal{V}$ is a compact metric space. If a continuous kernel K^{ϕ} is universal on $\mathcal{V}$, then $K^{\Phi}_{\otimes}$ defines a universal kernel on $\mathcal{E}$.*

This is by no means equivalent to saying that this kernel is the most optimal kernel for learning relations. This theorem only states that the Kronecker product pairwise kernel can be used to approximate any relation, given enough suitable data to train the algorithm.

Vert et al. (2007) have made the interesting distinction between direct and indirect inference of networks (or relations). Let us reuse the example of predicting protein-protein interactions. For the direct inference, one could state that our confidence in the correctness of the pair $A - B$ increased if the individual objects show a relevant similarity *to each other*. For example, if a yeast two-hybrid assay shows that they always occur in

the same compartments of the cell [74] or if they show similar evolutionary patterns this could be an indication of possible interaction [79]. From a bioinformatics point of view, this is perhaps the most logical setting. The indirect inference on the other hand relies upon similarities between *pairs of objects*. To quote the authors: "[the] confidence in A - B increases if we find some other, high-confidence edge C - D such that the pair {A, B} resembles {C, D} in some meaningful fashion". In the protein example, if you have a pair of proteins with a high confidence of interaction and one has a certain sequence motif and the second one has some other motif than if there is an other pair with one of the proteins containing the one motif while the second one has the other, it could be deduced that these are likely to interact. To try to make this distinction even more clear, the direct approach is like puzzling: to determine if two pieces fit together one has to learn to recognize complementary shape. Indirect inference may be more similar to learning organic chemistry, if you have to determine whether two molecules could form a chemical reaction, one tries to recognize functional groups on the molecules, which are the same of the series of examples that one has taken the trouble to memorize.

These two approaches for inference of networks can be represented in different pairwise kernels. Take the tensor product pairwise kernel for example [9]:

$$K_{TPPK}(v, v', \bar{v}, \bar{v}') = K^{\phi}(v, \bar{v})K^{\phi}(v', \bar{v}') + K^{\phi}(v, \bar{v}')K^{\phi}(v', \bar{v}). \tag{3.5}$$

The rational behind this kernel is that one compares the elements of the first pair (v, v') with those of the second pair $(\bar{v}, \bar{v}')$ by considering the combinations of the different elements between the two pairs. This kernel[2] thus embeds the indirect method of inference.

The authors contrast this with an other kernel of their own design, the metric learning pairwise kernel (MLPK):

$$K_{MLPK}(v, v', \bar{v}, \bar{v}') = (K^{\phi}(v, \bar{v}) - K^{\phi}(v, \bar{v}') - K^{\phi}(v', \bar{v}) + K^{\phi}(v', \bar{v}'))^2. \tag{3.6}$$

This formula seems less intuitive than the definition of the TPPK (3.5), its meaning becomes more clear when written explicitly as a product of feature vectors:

$$K_{MLPK}(v, v', \bar{v}, \bar{v}') = ((\phi(v) - \phi(v'))^T(\phi(\bar{v}) - \phi(\bar{v}')))^2. \tag{3.7}$$

This equation shows that the MLPK is the squared product of the difference of the feature vectors of the elements of the pairs. This kernel encodes the dissimilarity between objects within a pair, making it a tool for direct inference of relations.

3.2 Symmetry, transitivity and other issues with relations

Relations can have many properties which can be of importance in the learning process. In this section we will only discuss some of them very briefly. First let us consider symmetric relations. These occur when the relation between v and v' is equivalent to the relation between v' and v. Or more formally:

[2]This kernel is also equivalent to the symmetric kernel to be discussed in Section 3.2.

Definition 2. *A binary relation $h : \mathcal{V}^2 \to \mathbb{R}$ is called a symmetric relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $h(v, v') = h(v', v)$.*

For symmetric relations, edges in multi-graphs like Figure 3.1 become undirected. To learn these types of relations, the following feature mapping was proposed:

$$\Phi_S(e) = \Phi_S(v, v') = \Psi(v, v') + \Psi(v', v). \tag{3.8}$$

This representation gives rise to the symmetric Kronecker product pairwise kernel:

$$K^{\Phi}_{\otimes S}(e, \bar{e}) = K^{\Phi}_{\otimes S}(v, v', \bar{v}, \bar{v}') = 2(K^{\phi}(v, \bar{v})K^{\phi}(v', \bar{v}') + K^{\phi}(v, \bar{v}')K^{\phi}(v', \bar{v})). \tag{3.9}$$

This kernel can be used to efficiently learn symmetric relations, as asymmetries in the training data will be considered as noise. It can be proven that (3.9) can be used for learning any symmetric relation.

Next to be considered are reciprocal or antisymmetric relations. These arise often in domains such as preference learning, game theory and bioinformatics when modeling preferences, choice or winning probabilities, gene relations etc. Formally a reciprocal relation is defined as follows:

Definition 3. *A binary relation $Q : \mathcal{V}^2 \to [0, 1]$ is called a reciprocal relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $Q(v, v') = 1 - Q(v', v)$.*

Thus every edge in a graph induces a complementary edge in the opposite direction. Reciprocal relations can, for example, be used to represent a chess tournament where edges between players represent the probability that player v wins from player v'. Similar, antisymmetric relations are defined:

Definition 4. *A binary relation $h : \mathcal{V}^2 \to \mathbb{R}$ is called an antisymmetric relation if for all $(v, v') \in \mathcal{V}^2$ it holds that $h(v, v') = -h(v', v)$.*

Using an appropriate scaling, reciprocal and antisymmetric relations are equivalent. The feature mapping to learn reciproprocal relations is:

$$\Phi_R(e) = \Phi_R(v, v') = \Psi(v, v') - \Psi(v', v). \tag{3.10}$$

It can be easily understood that this feature representation is suitable to learn antisymmetric relations. The pairwise kernel that hence rises is:

$$K^{\Phi}_{\otimes R}(e, \bar{e}) = K^{\Phi}_{\otimes R}(v, v', \bar{v}, \bar{v}') = 2(K^{\phi}(v, \bar{v})K^{\phi}(v', \bar{v}') - K^{\phi}(v, \bar{v}')K^{\phi}(v', \bar{v})). \tag{3.11}$$

Again, it can be shown that this kernel can represent any reciprocal or antisymmetric relation.

When one does not only consider a relation between two objects, but a more broader network of multiple agents interacting the notion transitivity becomes important. Transitivity is basically the property if object v_i is 'better' than object v_j and object v_j is better than object v_k then it follows that v_i is better than v_k. More formally one defines weak and strong stochastical transitivity as follows:

Definition 5. *A binary relation $Q : \mathcal{V}^2 \to [0, 1]$ is called weak stochastical transitive if for any $(v_i, v_j, v_k) \in \mathcal{V}^3$ it holds that*

$$(Q(v_i, v_j) \geq 1/2 \wedge Q(v_j, v_k) \geq 1/2) \Rightarrow Q(v_i, v_k) \geq 1/2 \tag{3.12}$$

Definition 6. *A binary relation $Q : \mathcal{V}^2 \to [0,1]$ is called strong stochastical transitive if for any $(v_i, v_j, v_k) \in \mathcal{V}^3$ it holds that*

$$(Q(v_i, v_j) \geq 1/2 \wedge Q(v_j, v_k) \geq 1/2) \Rightarrow Q(v_i, v_k) \geq \max(Q(v_i, v_j), Q(v_j, v_k)) \tag{3.13}$$

Informally one could state there is some latent value that the objects posses, which can be used to rank the objects 'from good to bad'.

The Definitions 5 and 6 of transitivity are sometimes violated, which is denoted as an intransitive relation. Consider the difference between transitive and intransitive relations as follows: suppose one wants to construct a model to predict the winner in a sports game. For running this might be relatively simple, make some measurements of the athlete and try to predict a metric to quantify his 'running potential'. The higher this metric, the more likely he is to win, as running is mostly transitive. This is compared to a sport where strategy is more important, for example tennis. Strategy A might fare well against strategy B but not so good against strategy C[3]. One can thus only do a good prediction by considering a pair of players. These intransitive relations have been studied in many fields, from psychology to game theory to physics to biology. For example intransitive relations of bacterial populations are studied excessively [48, 80, 46, 95]. Allesina et al. (2011) even proposed a framework based on intransitivity to explain the coexistence of different species, a problem known in ecology as the plankton paradox [39].

The question remains how these intransitive relations influence the learning process. When considering decision making two main types of models can be distinguished [69, 108]:

- *Scoring methods*: these methods construct a continuous function of the form $f : \mathcal{V} \to \mathbb{R}$ which is used to rank the objects according to preference.
- *Pairwise preference models*: here the preference judgments are modeled by one (or more) valued relations $Q : \mathcal{V}^2 \to [0,1]$ that express whether v should be preferred over v'.

Pahikkala et al. (2010) showed that only the latter can be used for learning intransitive relations. This is luckily the framework which is predominantly described in this chapter. It may be of some interest to note that similarly to Equation 3.4, one can define a kernel which can only be used to learn transitive relations:

$$K_T^{\Phi}(e, \bar{e}) = K_T^{\Phi}(v, v', \bar{v}, \bar{v}') = K^{\phi}(v, \bar{v}). \tag{3.14}$$

This kernel treats all intransitivity's in the training data as noise and follows the principle of scoring methods to learn relations. This kernel does not consider the relations of the objects, but only focusses on the 'qualities' of the first objects of the pairs.

3.3 Conditional ranking

Let us give an example to explain the use of conditional ranking. Suppose a company wants to design a new drug to cure a certain disease. They have a set of promising organic

[3]Though it is not because transitivity is violated that transitivity is totally absent from the system. Following the example, a very talented player will probably beat a newcomer no matter which strategy they use.

molecules which they believe some of them could be a suitable drug. From experiments a target protein is found on which the drug should bind and act as an inhibitor. To minimize side effects, there is also a set of proteins that are similar in function, but not involved in the disease for which the compound preferably should have a low affinity. Given a dataset of known protein ligand interactions, it is possible, using the methods described above, to construct a model to predict a value quantifying binding of a ligand to a protein. This model could be used to search for the best compounds in the database which can be further researched for a clinical trail.

Though this may sound very reasonable, one can see this may not be the best approach for this problem. The model constructed is used to predict a binding value, but this is not something one is directly interested in. The researcher wants an ordered list of the best or worst binding compounds for a given protein. The model is optimized to predict an accurate value (usually by minimizing the squared residuals), not to be able to perform an optimal ranking. Thus the appropriate solution for this problem would be to construct a model that can rank a database of molecules according to their binding properties conditioned on a target protein.

The conditional ranking framework is very close to information retrieval which found its origins in document retrieval [90], but is now widely applied in a range of settings. For example, BLAST could be considered as information retrieval where a protein query is used to find homologues in the database ranked according to their E-value. Conditional ranking is broader applicable than information retrieval as the kernel framework allows a generalization to many types of relations, and can be learned from data.

Let us use the same notation as before, with the data structured as a graph $G = (V, E, Q)$, where $V \subseteq \mathcal{V}$ represents the set of nodes, with $\mathcal{V}$ the space of nodes and $E \subseteq 2^{V^2}$ corresponds to the set of edges e, for which we have labels in terms of the relations. These labels are again given by the (unknown) relation $Q : \mathcal{V}^2 \to [0, 1]$. Again, an extension to real values $h : \mathcal{V}^2 \to \mathbb{R}$ can be realized with a simple monotonous mapping $\sigma : \mathbb{R} \to [0, 1]$ such that

$$Q(v, v') = \sigma(h(v, v')), \quad \forall (v, v') \in \mathcal{V}^2. \tag{3.15}$$

Thus, we need a suitable function $h \in \mathcal{H}$, with $\mathcal{H}$ a reproducing Kernel space. We consider the hypothesis $h(e) =< \mathbf{w}, \Phi(e) >$ with $\mathbf{w}$ a vector of parameters to be estimated from the training data. If the training set of cardinality $q = |E|$ is denoted as the set $T = \{(e, y_e | e \in E\}$ of output pairs, then the problem of finding the optimal hypothesis can be formally defined as:

$$\mathcal{A}(T) = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \, \mathcal{L}(h, T) + \lambda \|h\|^2_{\mathcal{H}}, \tag{3.16}$$

with $\mathcal{L}$ an appropriate loss function and λ a regularization parameter to prevent overfitting.

According to the representer theorem [87], the algorithm (3.16) admits a dual representation

$$h(e) =< \mathbf{w}, \Phi(e) >= \sum_{e \in E} K^{\Phi}(e, \bar{e}), \tag{3.17}$$

where $K^{\Phi}(e, \bar{e})$ is a kernel function defined over two edges.

Given the relations $Q(v, v')$ and $Q(v, v'')$ we compose the ranking of v' and v'' conditioned on v as:

$$v' \succeq_v v'' \Leftrightarrow Q(v, v') \geq Q(v, v''). \tag{3.18}$$

To obtain a model that can correctly rank the nodes, it is desirable to use a loss function that punishes error in the ranking, the ranking loss

$$\mathcal{L}(h, T) = \sum_{v \in V} \sum_{e, \bar{e} \in E_v : y_e < y_{\bar{e}}} I(h(e) - h(\bar{e})), \tag{3.19}$$

with $I(x)$ the Heaviside function, returning one when its argument is positive, zero when its argument is negative en 1/2 when its argument is zero. E_v denotes the set of all edges starting from, or the set of all edges ending at the node v, depending on the specific task. Since Equation 3.19 is neither convex nor differentiable, we use an approximation of the ranking loss

$$\mathcal{L}(h, T) = \sum_{v \in V} \sum_{e, \bar{e} \in E_v} (y_e - y_{\bar{e}} - h(e) + h(\bar{e}))^2. \tag{3.20}$$

Thus the mean difference between conditional ranking and a regression-based framework is the use of a more relevant loss function. This framework was described by Pahikkala et al. (2010) and it can be implemented efficiently and is scalable for thousands to millions of nodes.

A ranking can be seen as something in between classification and regression. The output can be represented as some sort of distribution, which can be easily interpreted thanks to our collective experience with search engines such as Google.

Note that the conditional ranking framework is compatible with the notions of intransitivity described in Section 3.2. The model considers the relation between the query and the database object, thus a different query can induce a totally different ranking of the same database.

3.4 Performance measures for ranking

In this section some performance measures for ranking are discussed. Finding the most relevant measure is strongly dependent on the problem one is dealing with. For example, for webpage retrieval one typically is only interested in the top scoring pages, while when trying to rank chess players according to their expected strength for designing a tournament one wants the middle and bottom of the list to be as reliable as the top. The metrics that mainly focus on the top of the ranking are called *user-oriented* metrics, while those that evaluate the overall quality of a list are described as *system oriented*.

Yilmaz et. al. (2009) argue that the relation between the metric that should be optimal and the metric that is actually used in the training phase is not so straightforward as one would think. By using for example a loss function that only considers the first part of the ranking, information is lost from lower ranked elements, resulting in a suboptimal ranking algorithm.

Another desirable property is having no or few parameters that have to be tweaked, being able to deal with ordinal labels and being interpretative. Table 3.1 summarizes the performance measures discussed below for their properties.

Table 3.1: Overview of some desirable properties of the different performance measures for ranking to be discussed. For each metric it is given whether it can deal with ordinal labels or only distinguishes positive from negative instances, whether parameters have to be chosen, if the output has a probabilistic meaning and whether the metric can be used to focus on only a part of the ranking.

Metric	Ordinal label?	Parameters?	Probabilistic?	User defined?
RE	yes	no	yes	no
Precision and recall	no	no	yes	no
AveP	no	no	yes	no
ROC	no	no	yes	no
CROC	no	yes	no	yes
nDCG	yes	yes	no	yes

3.4.1 The ranking error

The ranking error (RE) is probably the most general performance measure for ranking. When obtaining an ordered list from an algorithm, the ranking error is equal to the chance that two randomly chosen objects of this list are in the correct order based on their ordinal label. Using the notation from the previous sections, it is defined as

$$\mathrm{RE} = \frac{1}{|E_v : y_e < y_{\bar{e}}|} \sum_{e,\bar{e} \in E_v : y_e < y_{\bar{e}}} I(h(e) - h(\bar{e})), \tag{3.21}$$

with $I(x)$ the Heaviside function, returning one when its argument is positive, zero when its argument is negative en 1/2 when its argument is zero.

The conditional ranking framework minimizes a loss function which is an approximation of the ranking error (see Equation 3.19). The ranking error can deal with real or ordinal labels and could be considered as an extension to the area under the curve (Section 3.4.4). It has the drawback that every part of the ranking is treated as equally important, making it less desirable for many information retrieval applications where the main interest is in the top ranked elements. A small example of the application of the ranking error is given in Table 3.2.

Table 3.2: Six labeled objects are given. An algorithm has provided a value for ranking the objects. The predicted order is then: B, A, C, E, D, F. Of fifteen possible comparisons, two pairs are in the wrong order, thus the ranking error equals 2/15.

Object	Label	Prediction
A	10	0.59
B	9	0.72
C	8	0.41
D	6	-0.13
E	3	-0.02
F	1	-1.7

3.4.2 Precision and recall

In information retrieval settings, objects, such as text files, are often only denoted as relevant or not relevant. Of a collection $\mathcal{C}$, a fraction π of the elements is considered relevant. Our algorithm selects a fraction $t \in [0, 1]$ of the elements from $\mathcal{C}$, of which a smaller fraction $h(t) \in [0, t]$ is truly relevant. This is shown conceptually in Figure 3.3.

In this framework, when performing a selection of objects that are thought to be relevant based on a value that was predicted by an algorithm, we have two important performance measures: recall and precision. Recall is defined as the probability of detecting an item, given that it is relevant while precision is defined as the probability that an item is relevant, given that it is detected by the algorithm. Based on the given notation the precision and recall can both be defined based on the fraction of selected objects t:

$$r(t) = \frac{h(t)}{\pi}, \tag{3.22}$$

$$p(t) = \frac{h(t)}{t}. \tag{3.23}$$

Typically, $r(t)$ will increase with t, while $p(t)$ decreases. It is desirable to both have a high recall as well as a high precision, though both are entwined in a relation called the recall-precision trade-off:

$$p(t) = \frac{\pi r(t)}{t}, \tag{3.24}$$

which can be shown using some elementary Bayesian probability. This shows that an algorithm cannot optimize both precision and recall by changing the number of selected objects.

Though precision and recall are some of the most important notions in information retrieval, they are usually not directly used as performance measures, as two values have to be taken in account and they are dependent on the fraction of relevant objects and the number of withhold elements. Instead, they are used to build other performance measures.

3.4.3 Average precision

The trade-off between recall and precision means that both of them must be considered simultaneously for evaluating and comparing ranking algorithms. The average precision (AveP) is a popular measure that takes both in account. It is defined as

$$\text{AveP} = \int_0^1 p(r)dr. \tag{3.25}$$

In practice one uses its discrete counterpart:

$$\text{AveP} = \sum_{k=1}^{n} p(r)\Delta r, \tag{3.26}$$

with k the rank in the sequence of the n objects and Δr the change in recall. Average precision is perhaps somewhat less intuitive. An algorithm that can perfectly distinguish relevant from irrelevant objects will have AveP=1, while random selection has AveP=π.

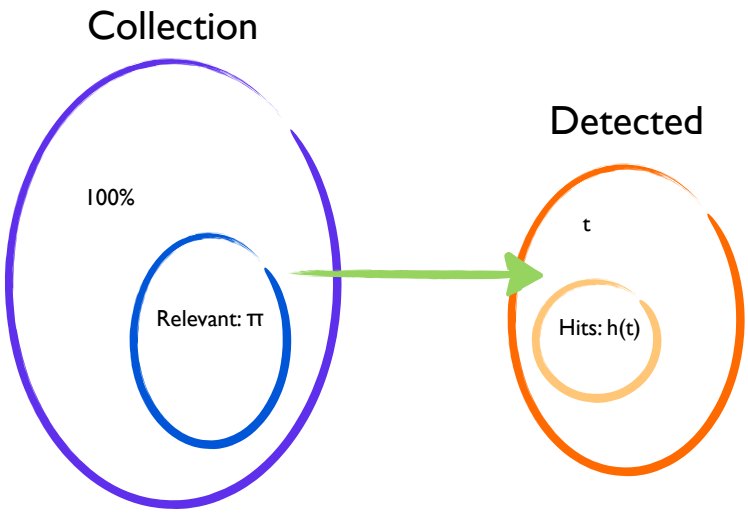


Figure 3.3: A collection $\mathcal{C}$ contains a fraction π of relevant elements. The algorithm selects a fraction $t \in [0, 1]$ of the elements from $\mathcal{C}$, of which only a smaller fraction $h(t) \in [0, t]$ is truly relevant. After [116].

3.4.4 ROC and CROC curves

Receiver Operating Characteristics (ROC) curves [25] are widely used for evaluating the performance of binary classification or ranking. A ROC curve plots the true positive rate in function of the false positive rate. The true positive rate is defined as the number of positive relevant objects selected by the algorithm divided by the total number of positive instances and is equal to the recall. Likewise, the false positive rate is the number of relevant objects that were selected divided by the total number of negative objects. Discrete classifiers directly return a label appear as point on this plot, while algorithms that return a real value that is used for ranking the objects or classification with a certain threshold form the curves in question. For example, a random classifier would appear as the first bisector, while any realistic ranker or classifier would lie above this line. The ROC curve is, in contrast to a precision-recall curve, insensitive to the class distribution, making it suitable for many problems with class skew.

A statistic that can be derived from the ROC curve is the area under the ROC (AUC), of which the ranking error can be seen as an extension for ordinal labels. Since any ROC curve lies within a unity square, the AUC is a real value from the interval [0,1]. Any ranker better than random has an AUC > 0.5. The AUC has also a probabilistic interpretation as it is equal to the probability that the classier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. It is also equivalent to the test statistic for the Wilcoxon test of ranks. It is also very related to the average precision and the Gini coefficient.

Swamidass et al. (2010) make an interesting addition to the ROC curve, called, the concentrated ROC or CROC curve. Since only the first part of the ranking is considered in many applications, only the first part of the ROC curve is relevant. Sections of interest are 'magnified' by performing a nonlinear transformation to the x-axis, usually with a

concave down function[4] with one parameter to tune. Consequently they also define the area under the CROC curve, though it should be noted that the probabilistic intuition is lost.

3.4.5 Discounted cumulative gain

The normalized discounted cumulative gain (nDCG) is a measure that can deal with the ordinal nature of the label and puts more relevance on the objects higher in the ranking. The drawback of this measure is that we have to set a parameter p, the position for which the rank is calculated. The discounted cumulative gain at position p (DCG_p) is defined as:

$$\mathrm{DCG}_p = rel(1) + \sum_{i=2}^{p} \frac{rel_q(i)}{\log_2 i}, \tag{3.27}$$

where $rel(i)$ denotes the relevance (usually the ordinal label). This measure is scaled using an ideal DCG, IDCG_p to become the nDCG:

$$\mathrm{nDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}. \tag{3.28}$$

The nDCG is a value of the interval [0,1] and can be used to compare different queries.

3.5 Cross validations and testing in relational learning

A correct testing procedure is a very important part of the learning phase, it assesses the ability of the model to generalize to new data and allows for an optimal model selection. Though testing is by no means a simple matter when dealing with inferences about single objects [34], in this section we will discuss some important issues when evaluating models for relations of objects.

In a classical machine learning scheme one usually wants to build a model using a dataset of N instances where each instance has p features and a label. Thus the data can be represented as an $N \times (p+1)$ matrix. Since the *training error* is an underestimation of the *test error* or *generalization error* an independent data set is needed to evaluate the model. The most simple case is to randomly withhold a part of the data, for example a quarter, for testing while the remainder is used for training a model as indicated in Figure 3.4. A more advanced setting is K-fold cross validation, where the data is divided in K parts, for which every part is used for testing a model built using the $K-1$ remaining parts. This guarantees that every instance is used for both training and testing. The logical extreme of K-fold cross validation is leave-one-out cross validation, where each of the N instances is withheld once for testing. Though this is usually computationally intensive, it is a very good estimation for the test error of a model built with the whole data set.

[4] A function is concave down over a certain interval iff its derivative function is monotonically decreasing on that interval.

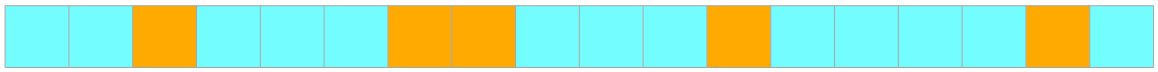


Figure 3.4: The most simple way to correctly test the performance of a model is to randomly withhold a part of the data set (orange) for evaluating a model build using the remainder of the data (blue).

Testing is somewhat more complicated when dealing with relations between objects. Let us consider the case of inferring a relation between two different types of objects, the data set contains N_1 objects of the first kind and N_2 objects of the second kind and labels are known for all combinations. The labels can thus be more naturally represented as a matrix rather than a vector, as was the case when learning properties about individual objects. Depending on the way the test set is sampled, we consider four cases, which we denote, somewhat lazily, as α, β, γ and δ:

- α: both objects in the testing instance were encountered in the training phase but never as a pair. This model is tested for its ability to make predictions about new combinations of known objects.
- β: the first kind of object is already used for training the data but the second one is previously unseen. This is expected to be a harder problem to learn than α.
- γ: similar as β but now only the second kind of objects is new during testing. If the problem is monadic, e.g. there is only one type of objects, β and γ are equivalent.
- δ: both objects used for testing were not encountered during training. There are two reasons why this is probably the hardest case to learn. Firstly, the model has to deal with two objects that are new and, secondly, because a relatively large part of the data has to be omitted. This is because the pairs that have one element in common with either the training or test set cannot be used.

These cases are depicted in Figure 3.5. A practical use of a model will likely have to deal with a combination of the above settings, but for some applications it might be of use to consider the one that is of most relevance. For example, when a researcher is investigating the effect of some pharmaceutical compounds on different types of cell cultures, she may choose to test only a subset of the possible combinations, using a model to infer the value of the other combinations. Here the setting α is of the most importance. On the other hand, if she would like to use her model to predict the effect of novel compounds discovered by the chemistry department, setting β will be of more importance.

Things complicate even further when dealing with conditional ranking. Testing can also be classified in the four settings of above, but for the dyadic setting it could also make a big difference which of the two objects are used as query. As a rule, it is clear which type is used to query and which serves as database objects, but the conditional ranking framework described in Section 3.3 is symmetric in the sense that it can always be used in both ways. This is the reason why the user should be warned that there might be a big difference in performance depending on the choice of query type.

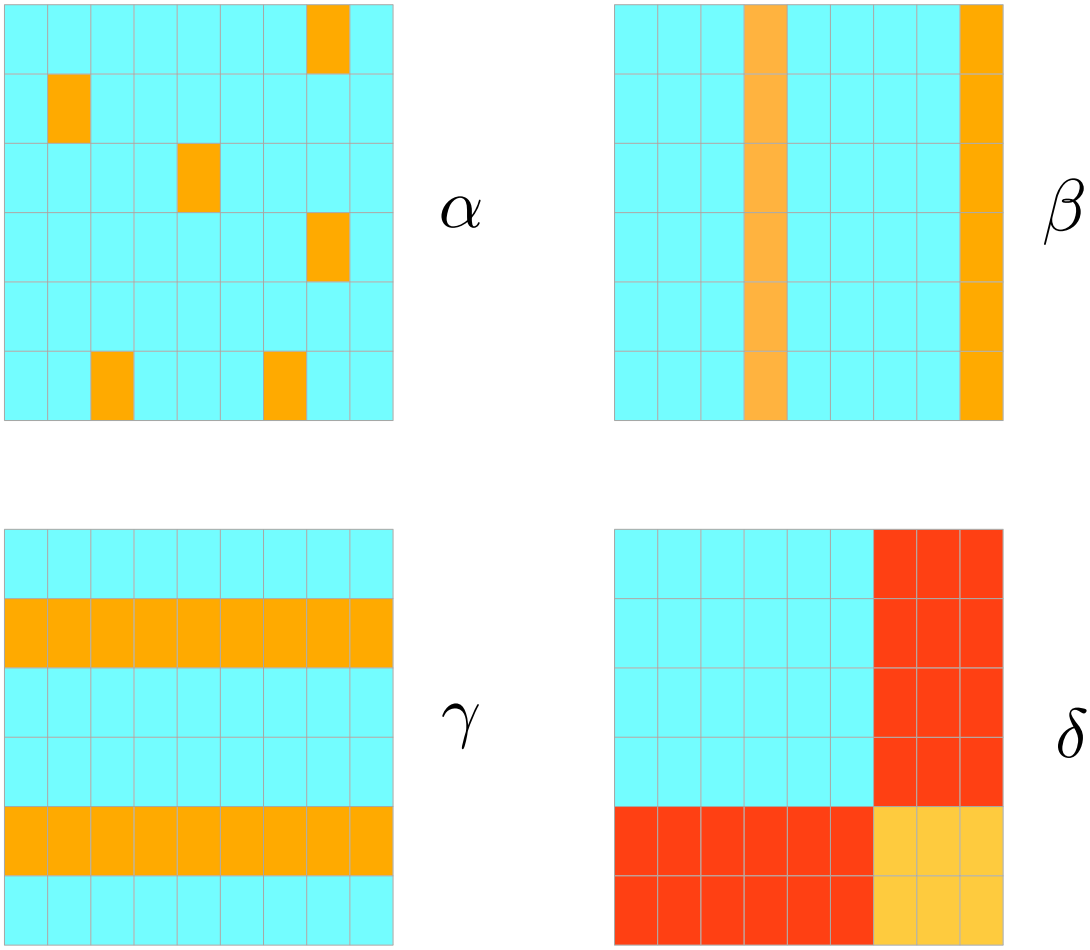


Figure 3.5: The four cases of sampling a test set. Blue stands for training pairs, orange are testing pairs and pairs in red cannot be used in this setting.

Chapter 4

Functional ranking of enzymes

4.1 Introduction

Enzymes are proteins that catalyze almost all chemical reactions in a cell, thus making life possible. They play key roles in all molecular functions of an organism from anabolism to catabolism, signal transduction and cell death. Therefore, they are important subjects of interest in medicine, biotechnology and industry for disease detection, drug design and optimization of biochemical processes. Knowing the enzymes of an organism is understanding a large part of its chemistry. Despite it being generally accepted that the structure of an enzyme determines its function [97], predicting the biological function of enzymes remains extremely challenging [19].

The Enzyme Commission (EC) functional classification is commonly used to subdivide enzymes into functional classes. EC numbers adopt a four-label hierarchical structure, representing different levels of catalytic detail. Importantly, this representation focuses on the chemical reactions that are performed and not on the enzymes themselves, since it is possible for two very different enzymes to have the same EC number, or for two similar enzymes to have a completely different EC number [99]. In addition to the EC number, many other classifications for enzyme function exist, such as the Gene Ontology [7], the Transporter Classification [82] and classifications derived from known databases like KEGG [44] and EcoCyc [47]. These classifications are not considered in this article, but our methodology could be easily extended to such settings as well.

Roughly speaking, existing methods for automated assignment of EC numbers fall in two main categories: unsupervised and supervised approaches. Unsupervised approaches heavily rely on the notion of similarity, defined at the sequence level or structure level of enzymes. A lot of effort has been spent in the last decade on defining similarity measures that reflect the function of enzymes. At the sequence level, one can apply tools depending on global sequence homology (such as BLAST and PSI-BLAST) or local sequence homology to detect motifs indicative for function. If the enzyme structure is available, one might expect to obtain more information with respect to the function of enzymes, resulting in better but more complex similarity measures [52, 22, 23, 67].

Moreover, substrate or ligand recognition is tightly connected with the shape of the enzyme's active site, which is commonly found as a deep and large cleft in a protein's

surface [55]. Hence, emerging tools for protein comparison use structural information of binding sites. These methods are of particular interest in the research domain of drug discovery [75]. They are designed to detect similarities that cannot be found using traditional sequence- and fold-based methods, e.g. where physicochemical recognition features of a binding pocket are conserved despite low sequence similarity. Approaches based on binding sites not only provide a complementary notion of protein family [111], but are also able to detect similarities of binding sites for unrelated proteins [110]. In this chapter we will focus on five state-of-the-art structure-based similarity measures, described more thoroughly in Section 4.2.1.

Once a similarity measure has been computed to compare and discriminate enzymes, one can construct a basic enzyme retrieval system by returning for a given enzyme query a ranking of enzymes in the database, ranging from the most similar enzyme to increasingly dissimilar enzymes. As similarity measures are often defined independent of the EC numbers, we call this type of ranking the unsupervised approach. Hence it does not guarantee retrieval of enzymes having a similar function with respect to the enzyme query. Therefore, supervised learning algorithms have been applied for automatic EC number assignment. If the hierarchical information in the EC number is ignored, one arrives at a standard multi-class classification problem, for which a wide variety of algorithms can be employed - see e.g. [17]. Additionally, more specific hierarchical classification methods can take hierarchical information into account – see e.g. [81, 5]. Similar to multi-class classification methods, they are not capable of detecting new enzyme functions because predictions are restricted to those occurring in the training dataset [12].

In our framework we circumvent this bottleneck by reformulating EC number assignment as an information retrieval problem where enzymes can be interpreted as search queries – see e.g. [112]. Instead of predicting an EC number for enzymes with unknown function, we rather intend to find enzymes in a database that have the same or similar function as the query enzyme. This has the advantage that we can obtain meaningful results even if an enzyme with a novel function is encountered. Moreover, a ranking provides end users with an easily understandable view of the results. We consider specialized ranking algorithms for inferring such rankings in a supervised manner.

4.2 Material and methods

4.2.1 Similarity measures for enzymes

Though enzyme structure determines its function, one believes that the exact catalytic properties of enzymes are characterized by a few specific amino acid residues in the active center rather than the global fold [61]. Consequently, in order to model enzyme function, features that describe the catalytic site are needed. Such features are calculated on the largest cleft of the enzyme. Our work builds upon CavBase [86], a database system for the fully automated detection and extraction of protein binding pockets from experimentally determined protein structures (available in the database PDB). In CavBase, labeled points in the 3-D space are used as a first approximation to describe a binding pocket. The database currently contains 113,718 hypothetical binding pockets that have been extracted from 23,780 publicly available protein structures using the LigSite algorithm [36].

The geometrical arrangement of the pocket and its physicochemical properties are first represented by predefined pseudocenters, i.e. spatial points that represent the center of a particular property. Pseudocenters can be regarded as compressed representations of areas on the cavity surface where certain protein-ligand interactions are experienced. The type and the spatial position of the centers depend on the amino acids that border the binding pocket and expose their functional groups. They are derived from the protein structure using a set of predefined rules [86]. Possible types of pseudocenters considered here are hydrogen-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic and aromatic properties. The following five similarity measures were analyzed in our work.

Labeled point cloud superposition (lpcs) [26]. This measure operates on labeled point clouds, hence the CavBase data can be used directly without a need for transforming it into another representation. Intuitively, two labeled point clouds are similar if they can be spatially superimposed. By fixing the first and moving the second one in a proper way (as a whole, without changing the internal arrangement of points), an approximate superposition of the two structures is obtained. More specifically, we will say that two point clouds are well superimposed if, for each point in one of the structures, there exists a point in the other cloud which is spatially close and has the same label. This concept is used to define a fitness function which is maximized using a direct search approach [10]. The obtained maximal fitness is taken as the similarity between two labeled point clouds. A similar measure was also proposed in Hoffmann et al. (2010), though here a convolution is considered to obtain similarities between the point clouds.

Maximum common subgraph (mcs) [13] Using this measure, the original representation in the form of a labeled point cloud must be transformed into a node-labeled and edge-weighted graph in a preliminary step. Each pseudocenter is becoming a node labeled with the corresponding physicochemical property. To capture geometry, a complete graph is considered, where each edge is weighted with the Euclidean distance between the two pseudocenters it is adjacent to. Measuring similarity between protein binding sites then boils down to measuring similarity between graphs. A well-known approach here is to search for the maximum common subgraph (mcs) of the two input graphs and to define similarity as the ratio of size of mcs and the size of the larger graph. In case of noisy data, a threshold ϵ is required, defining two edges as equal if their weight differs as most by ϵ. In this case study, is the parameter set to 0.2 Å.

Cavbase (cb) similarity [86]. CavBase is also employing an algorithm for the detection of common subgraphs. Here the 100 largest common subgraphs are considered, instead of considering the largest common subgraph, as done in the case of mcs. Each common subgraph is defining a superposition of the complete protein binding sites. For each of the 100 superpositions, the surface points are taken into consideration and used to define a similarity value in a post-processing step, leading to a much finer model. Eventually, a set of 100 similarity values is obtained, from which the highest value is returned as similarity between the two protein binding sites.

Fingerprints (fp) Fingerprints are a well-known concept and were already used successfully for comparison of protein binding sites (cf. Fober et al. (2009)). Here a new definition of fingerprint is used and is described as follows: All possible triangles (complete node-labeled and edge-weighted graphs of size 3) are derived from a protein binding

site so that the edge-weight giving the distance between the adjacent pseudocenters is not shorter than 3 Å and not longer than 14 Å. In addition, the triangle inequality has to be fulfilled. Decoded as a hash-key with the respective physicochemical property, perimeter and enclosed area, this leads to a set of 14,280 features, thus, to a binary vector of this size for each protein binding site. On these fingerprints a kernel matrix was constructed by means of the Tanimoto kernel [58].

Weighted fingerprints (wfp) To make the fingerprints more rich in information, individual bits were rescaled using a detailed statistic comprising the whole information stored in the CavBase. This leads to higher weights for rare physico chemical property combinations and lower weights for the bits of very common physicochemical properties. For comparison, still the Tanimoto kernel is used. To obtain the (weighted) bit representing a triplet (p_1, p_2, p_3), the following procedure is applied: First $n_{1,2,3}$ is determined, giving the number of occurrences of the triplet (p_1, p_2, p_3) in the binding site under consideration. Moreover, let $N_{1,2,3}$ denote the number of occurrences of that triplet, and let N be the number of all triplets in CavBase. The bit is then subsequently obtained as $n_{1,2,3}(1 - N_{1,2,3}/N)$.

4.2.2 Unsupervised ranking

We benchmark our algorithm against a simple unsupervised procedure for retrieval of enzymes. Given a specific enzyme query and one of the above similarity measures, a ranking is constructed by computing the similarity between the query and all other enzymes in the database. Enzymes having a high similarity to the query appear on top of the ranking, those exhibiting a low similarity end up at the bottom. More formally, let us represent the similarity between two enzymes by $K : \mathcal{V}^2 \to \mathbb{R}$, where $\mathcal{V}$ represents the set of all potential enzymes. Given the similarities $K(v, v')$ and $K(v, v'')$ we compose the ranking of v' and v'' conditioned on the query v as:

$$v' \succeq_v v'' \Leftrightarrow K(v, v') \geq K(v, v''). \tag{4.1}$$

This approach follows in principle the same methodology as a nearest neighbor classifier, but rather a ranking than a class label should be seen as the output of the algorithm.

The quality of the obtained ranking can be evaluated by comparison with a *ground truth* ranking that is based on the EC numbers of the enzymes in the ranking (provided that their EC numbers are known). This ground truth ranking can be deduced from the catalytic similarity (i.e. ground truth similarity) between the query and all database enzymes. To this end, we count the number of successive correspondences from left to right, starting from the first digit in the EC label of the query and the database enzymes, and stopping as soon as the first mismatch occurs. For example, an enzyme query with number EC 2.4.2.23 has a similarity of two with a database enzyme labeled EC 2.4.99.12, since both enzymes belong to the family of glycosyl transferases. The same query manifests a similarity of one with an enzyme labeled EC 2.8.2.23. Both enzymes are transferases in this case, but they show no further similarity in the chemistry of the reactions to be catalyzed. Figure 4.1 visualizes a few more examples. Thus, when returning a ranking, it is expected that the top of the results contains proteins having all four numbers in common with the query, followed by those for which only the first three are equal, etc.

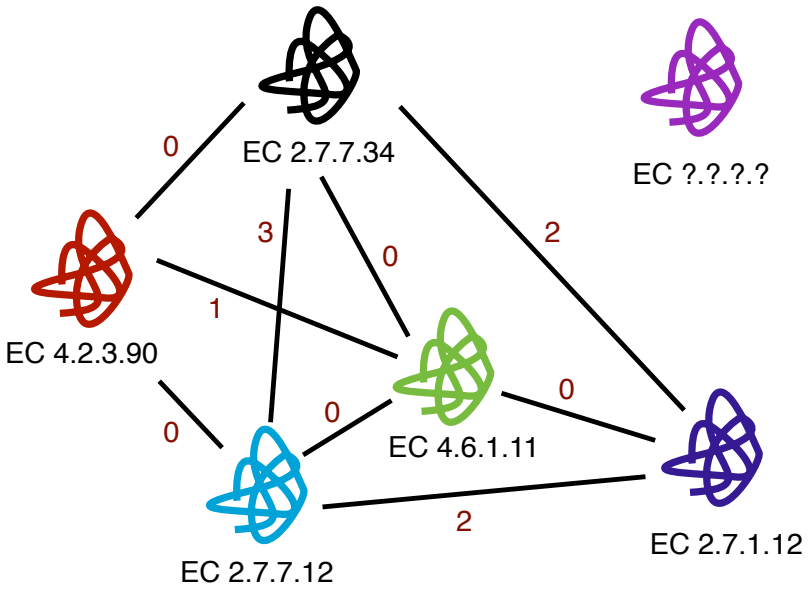


Figure 4.1: Six enzyme structures are shown, five of which have a known EC number. For these their catalytic similarity relation Q can be calculated as depicted on the edges of the graph. Each of the five labeled enzymes can be used for the conditional ranking of the other four. The model we present allows us to infer the ranking of the labeled enzyme conditioned on the unlabeled enzyme based on structural similarity.

More formally, let us represent the catalytic similarity (i.e., ground truth similarity) between two enzymes by a relation $Q : \mathcal{V}^2 \to \{0, 1, 2, 3, 4\}$. Given the relations $Q(v, v')$ and $Q(v, v'')$ we compose similar to (4.1) the ground-truth ranking of v' and v'' conditioned on the query v as:

$$v' \succeq_v v'' \Leftrightarrow Q(v, v') \geq Q(v, v''). \tag{4.2}$$

As a result, an entire ground truth ranking of database enzymes with known EC numbers can be constructed. The ranking obtained with unsupervised or supervised learning algorithms can be compared to the ground truth ranking by applying performance measures that are commonly used in information retrieval. In our experiments we consider four well-known performance measures, as discussed in Section 4.3.

4.2.3 Supervised ranking

In contrast to unsupervised ranking approaches, supervised algorithms do take ground-truth information into account during a training phase. We perform experiments with conditional ranking algorithms, see section 3.3, using the RankRLS implementation [71]. Let us introduce the short-hand notation $e = (v, v')$ to denote a couple consisting of a query v and a database enzyme v'. RankRLS produces a linear basis function model of type:

$$h(e) = h(v, v') = \langle \mathbf{w}, \Phi(e) \rangle, \tag{4.3}$$

in which $\mathbf{w}$ denotes a vector of parameters and $\Phi(e)$ an unspecified feature representation for the couple $e = (v, v')$. RankRLS differs from more conventional kernel-based methods because it optimizes a convex and differentiable approximation of the rank loss (i.e., area under the ROC curve) instead of zero-one loss. Together with the standard $L2$ regularization term on the parameter vector $\mathbf{w}$, the following loss is minimized:

$$\mathcal{L}(h, T) = \sum_{v \in V} \sum_{e, \bar{e} \in E_v} (Q_e - Q_{\bar{e}} - h(e) + h(\bar{e}))^2. \tag{4.4}$$

given a training set $T = \{(e, Q_e) | e \in E\}$, where $Q_e = Q(v, v')$ denotes the ground truth similarity as defined above, E the set of training query-object couples for which ground-truth information is available and E_v the subset of E containing results for the query v. The outer sum in Equation 4.4 takes all queries into account, the inner sum analyzes all pairwise differences between the ranked results for a given query. Pahikkala et al. [73] show how this loss can be minimized in a computationally efficient manner, using analytic shortcuts and gradient-based methods.

According to the representer theorem [87], one can rewrite equation (4.3) in the following dual form:

$$h(e) = \langle \mathbf{w}, \Phi(e) \rangle = \sum_{e \in E} K^{\Phi}(e, \bar{e}). \tag{4.5}$$

with $K^{\Phi}(e, \bar{e})$ a kernel function defined over four enzymes. We adopt the Kronecker product feature mapping containing information on couples of enzymes:

$$\Phi(e) = \Phi(v, v') = \phi(v) \otimes \phi(v'), \tag{4.6}$$

where $\phi(v)$ is a feature mapping of an individual enzyme. One can easily show that this pairwise feature mapping yields the Kronecker product pairwise kernel in the dual representation:

$$K^{\Phi}(e, \bar{e}) = K^{\Phi}(v, v', \bar{v}, \bar{v}') = K^{\phi}(v, \bar{v}) K^{\phi}(v', \bar{v}'), \tag{4.7}$$

with K^{ϕ} a traditional kernel defined over enzymes. Specifying a universal kernel for K^{ϕ} leads to a universal kernel for K^{Φ} [109], meaning that in principle this type of kernels allow modeling arbitrary binary relations. This kernel has been introduced by [9] for modeling protein-protein interactions.

Using the above construction, all similarity measures discussed in Section 4.2.1 can be converted into kernels of type K^{ϕ}, provided that they are symmetric and positive definite. We simply enforced symmetry by averaging the similarity matrix with its transpose. Subsequently, we made the similarity matrix positive definite by performing eigenvalue decomposition and setting all negative eigenvalues to zero. Since this procedure was performed on the whole dataset, one arrives at a so-called transductive learning setting. Minor adjustments would obtain a more traditional inductive learning setting. Remark that overfitting is prevented when applying this procedure, since the EC numbers of the enzymes in the dataset are ignored.

4.2.4 Experimental setup

Our models were build and tested using a dataset of 1730 enzymes with known protein structures. All the enzyme structures had a resolution of at least 2.5 Å, they had a

binding site volume between 350 and 3500 $Å^3$, and they were fully EC annotated. For evaluation purposes our database contained at least 20 observations for every EC number, leading to a total of 21 different EC numbers comprising members of all 6 top level codes. Abundances are summarized in Table 4.1 and a heat map of the catalytic similarity of the enzymes is given in figure 4.2. It can be seen that we are covering a wide variety of metabolic functions. Of all the combinations of enzymes the similarities described in Section 4.2.1 were calculated.

The dataset was randomized and split in four equal parts. Each part was withheld as a test set while the other three parts of the dataset were used for training and model selection. This process was repeated for each part so that every instance was used for training and testing (thus, four-fold outer cross-validation). In addition, a 10-fold inner cross validation loop was implemented for estimating the optimal regularization parameter λ, as recommended by [101]. The value of the hyperparameter was selected from a grid containing all the powers of 10 from 10^{-4} to 10^5. The final model was trained using the whole training set and the median of the best hyperparameter values over the ten folds. We used the implementation RLScore in Python to train the models.

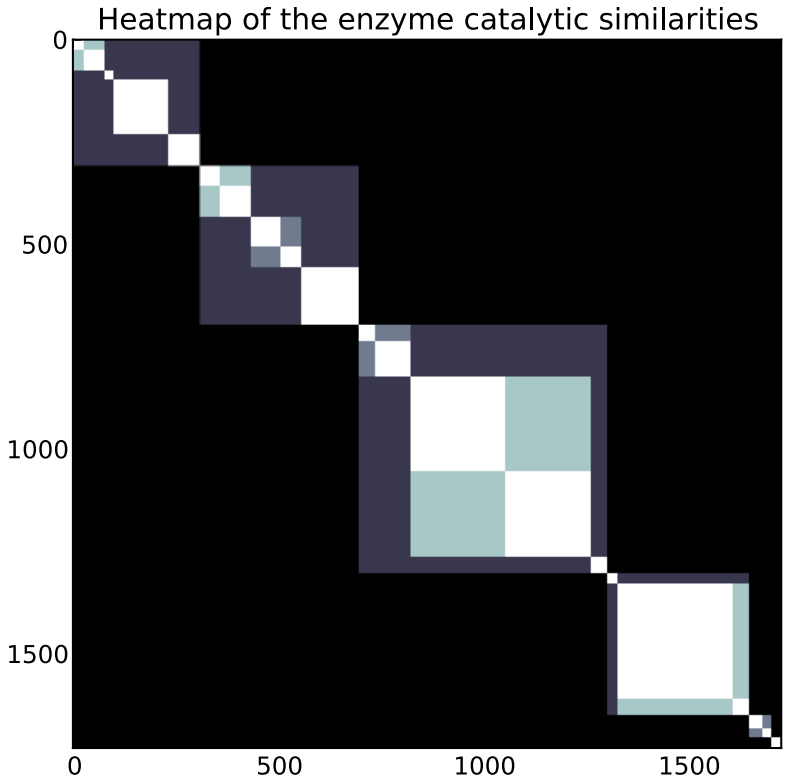


Figure 4.2: Heatmap of the catalytic similarity between the enzymes in the analyzed dataset. Rows and columns are ordered as in Table 1

Table 4.1: List of the 21 EC numbers with their accepted name and how many of each class are used.

EC number	accepted name	#
EC 1.1.1.1	alcohol dehydrogenase	28
EC 1.1.1.21	aldehyde reductase	50
EC 1.5.1.3	dihydrofolate reductase	24
EC 1.11.1.5	cytochrome-c peroxidase	131
EC 1.14.15.1	camphor 5-monooxygenase	78
EC 2.1.1.45	thymidylate synthase	48
EC 2.1.1.98	diphthine synthase	77
EC 2.4.1.1	phosphorylase	72
EC 2.4.2.29	tRNA-guanine transglycosylase	49
EC 2.7.11.1	non-specific serine/threonine enzyme kinase	142
EC 3.1.1.7	acetylcholinesterase	39
EC 3.1.3.48	enzyme-tyrosine-phosphatase	86
EC 3.4.21.4	trypsin	229
EC 3.4.21.5	thrombin	210
EC 3.5.2.6	β-lactamase	39
EC 4.1.2.13	fructose-bisphosphate aldolase	25
EC 4.2.1.1	carbonate dehydratase	280
EC 4.2.1.20	tryptophan synthase	40
EC 5.3.1.5	xylose isomerase	32
EC 5.3.3.1	steroid Δ-isomerase	22
EC 6.3.2.1	pantoate-β-alanine ligase	29

4.3 Results and Discussion

Table 4.2 gives a global summary of the results obtained for the unsupervised and the supervised ranking approach, respectively. All models score relatively well for all the performance measures. After performing some very brief data exploration we discuss the reasons for the performance gains obtained with supervised ranking and the influence of the kernel function and the performance measure in this regard.

4.3.1 The power of the rough data

To the credit of the the calculated similarities, unsupervised ranking is already quite powerful in its own right! To quickly assess the information hidden in these kernels we performed a kernel principal component analysis (PCA) and a hierarchical clustering in the kernel space.

We compare the kernel PCA of the CavBase similarity, the best unsupervised performer, with the weighted fingerprints kernel, the worst unsupervised ranking measure. This is given in Figures 4.3 and 4.4. The other three measures gave rise to clearly different plots, implying that the different kernels where not equivalent. It is clear that even in a low number of dimensions the broadest classes of enzymes form clearly distinguishable groups. In contrast, the weighted fingerprints form somewhat more messy clusters, but it is clear that there is a pattern relevant to the EC number in the data. This explains why

Table 4.2: A summary of the results obtained for unsupervised and supervised ranking (above and below double horizontal line, respectively). For each combination of model, kernel and performance measure, the average of the value over the different queries and folds is given by the standard deviation between parentheses. In every row the best ranking model is marked in bold, while the worst model is indicated in italic.

	cb	fp	wfp	NC	lpcs
RA	**0.9062** (0.0603)	0.8815 (0.0689)	*0.8467* (0.0884)	0.8923 (0.0692)	0.8877 (0.0607)
MAP	**0.9321** (0.1531)	0.7207 (0.235)	*0.2836* (0.2132)	0.8846 (0.1578)	0.7339 (0.2074)
AUC	**0.9636** (0.0795)	0.8655 (0.1387)	*0.7527* (0.1468)	0.9393 (0.0919)	0.8794 (0.1126)
nDCG	**0.9922** (0.0329)	0.9349 (0.1424)	*0.3202* (0.3282)	0.9812 (0.0498)	0.9471 (0.1112)
RA	0.9951 (0.017)	0.995 (0.015)	**0.9981** (0.01)	*0.9944* (0.0112)	0.9952 (0.0156)
MAP	**0.9991** (0.0092)	0.9954 (0.0432)	0.9981 (0.0167)	0.9989 (0.0076)	*0.9835* (0.0797)
AUC	**0.9976** (0.0005)	0.9967 (0.0184)	0.9975 (0.0016)	0.9975 (0.0024)	*0.9934* (0.0368)
nDCG	**0.9968** (0.0171)	0.9942 (0.0424)	0.9979 (0.0173)	0.987 (0.0398)	*0.9812* (0.0673)

using supervised learning can make the weighted fingerprints a very valuable similarity!

In Figures 4.5 and 4.6 a clustering in the Hilbert space is given for 75 randomly selected enzymes. Both measures could not only separate the large groups but also seem to cluster some finer catalytic similarities together.

4.3.2 The benefits of supervised ranking

One can observe that supervised ranking outperforms unsupervised ranking for all four performance measures and all five kernels. The statistical significance of the differences was confirmed by a paired Wilcoxon test and a conservative Bonferroni correction for multiple hypotheses testing ($p < 10^{-6}$). Moreover, for all kernels and performance measures, supervised ranking decreases the standard deviation of the error, implying that the models become more stable.

Three important reasons can be put forward for explaining the improvement in performance. First of all, the traditional benefit of supervised learning plays an important role. One can expect that supervised ranking methods outperform unsupervised ranking methods, because they take ground-truth rankings into account during the training phase to steer towards retrieval of enzymes with a similar EC number. Conversely, unsupervised methods solely rely on the characterization of a meaningful similarity measure between enzymes, while ignoring EC numbers.

Second, we also advocate that supervised ranking methods have the ability to preserve the hierarchical structure of EC numbers in their predicted rankings. figure 4.7 supports this claim. It summarizes the values used for ranking one fold of the test set obtained by the different models. So, for supervised ranking it visualizes the values $h(v, v')$, for unsupervised ranking it visualizes $K(v, v')$. Each row of the heatmap corresponds to one query. For the supervised models one notices a much better correspondence with the ground truth given in figure 4.2. Furthermore, the different levels of catalytic similarity can be better distinguished[1]. In addition, an example of the distributions of the predicted

[1]Note that all the unsupervised heatmaps are symmetric, because they visualize a subset of the distance matrices. Conversely, the supervised heatmaps are approximately symmetric, since rankings

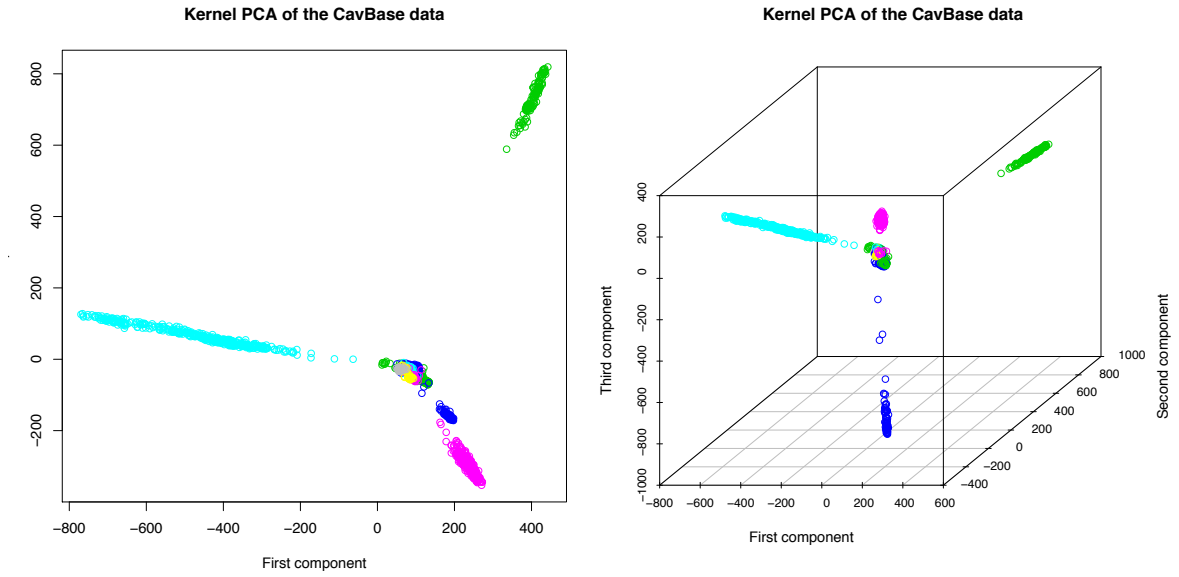


Figure 4.3: Kernel principal component analysis of the CavBase similarity of the enzymes given as a biplot and a triplot. Enzymes, depicted as the points, are colored according to the first digit of their EC number.

values within one query is visualized in figure 4.8 by means of box plots, illustrating again that supervised models establish a better separation of the different levels of similarity to the query enzyme. In this example no quartiles are overlapping in any supervised model, unlike the unsupervised approach, which only detects a good ranking for exact matches (i.e., enzymes having an EC number identical to the query).

A third reason for improvement by the supervised ranking method can be found in the exploitation of dependencies between different similarity values. Roughly speaking, if one is interested in the similarity between enzymes v and v', one can try to compute the similarity in a direct way, or derive it from the similarity with a third enzyme v''. In the context of inferring protein-protein interaction and signal transduction networks, both methods are known as the direct and indirect approach, respectively [104, 29]. We argue that unsupervised ranking boils down to a direct approach, while supervised ranking should be interpreted as indirect. Especially when the kernel matrix contains noisy values, one can expect that the indirect approach allows for detecting the *back bone* entries and correcting the noisy ones.

4.3.3 Differences between kernels

In the unsupervised case both measures based on fingerprints show the least structure, the CavBase similarity seems to have the clearest structure while lpcs and mcs are in between. The good performance of CavBase can be explained easily since CavBase is using additional information which leads a a much finer model of the protein binding site. Measures as mcs and lpcs must perform calculations without having this informa-

are inferred row by row.

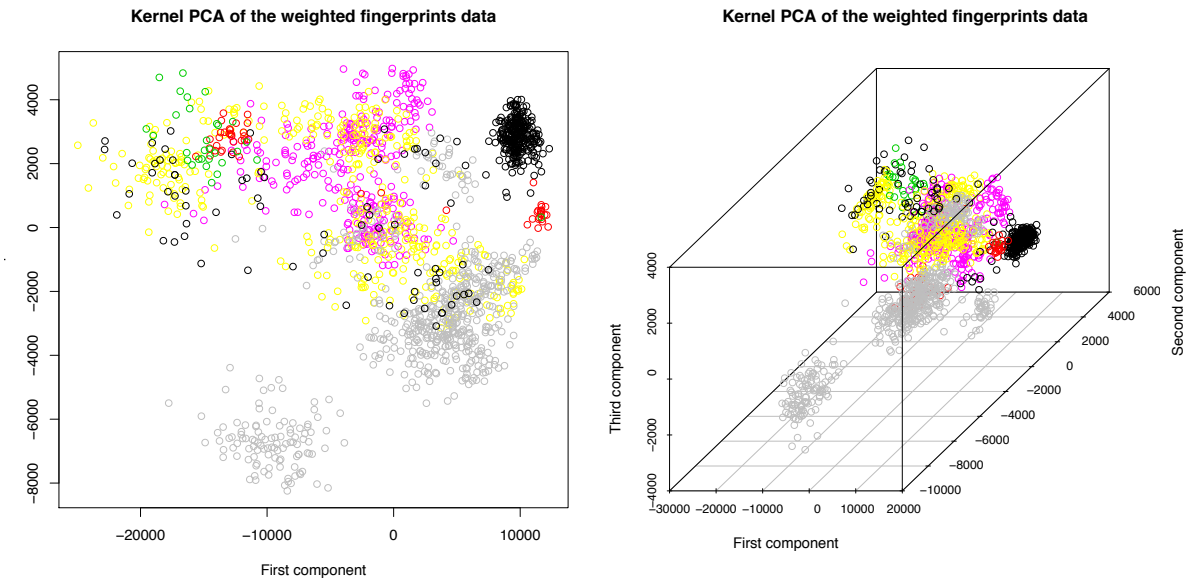


Figure 4.4: Kernel principal component analysis of the weighted fingerprints similarity of the enzymes given as a biplot and a triplot. Enzymes, depicted as the points, are colored according to the first digit of their EC number.

tion. However, they are still able to produce meaningful similarities. The measures based on fingerprints cause the highest loss of information, since only a set of local patterns is considered, whereas information about the global structure is lost. Therefore, these measures lead to the poorest result.
It is interesting to note that the weighted fingerprints produce one of the better models in the supervised case despite it being by far the worst datatype for using for unsupervised ranking. This suggests the weighted fingerprints are rich in information but need training (basically a rescaling) to be useful for ranking. The labeled point cloud superposition on the other hand is one of the worst models supervised, while it was a relatively good measure for the unsupervised ranking. To test whether there was a statistical difference between the different models a paired Wilcoxon signed rank test was used on all the combinations of supervised rankers for the ranking accuracy. Using a conservative Bonferroni correction for multiple testing we found a significant difference between all the models except for the fingerprints and labeled point cloud superposition.

4.3.4 Differences between performance measures

Since no consensus concerning performance measures exists in the information retrieval literature, we evaluated our algorithms on four well-known performance measures, each having different properties that are relevant in some way. First of all, we considered the ranking accuracy (RA), which is defined as follows:

$$\mathrm{RA} = 1 - \frac{1}{|V|} \sum_{v \in V} \frac{1}{|\{E_v : y_e < y_{\bar{e}}\}|} \sum_{e,\bar{e} \in E_v : y_e < y_{\bar{e}}} I(h(e) - h(\bar{e})), \tag{4.8}$$

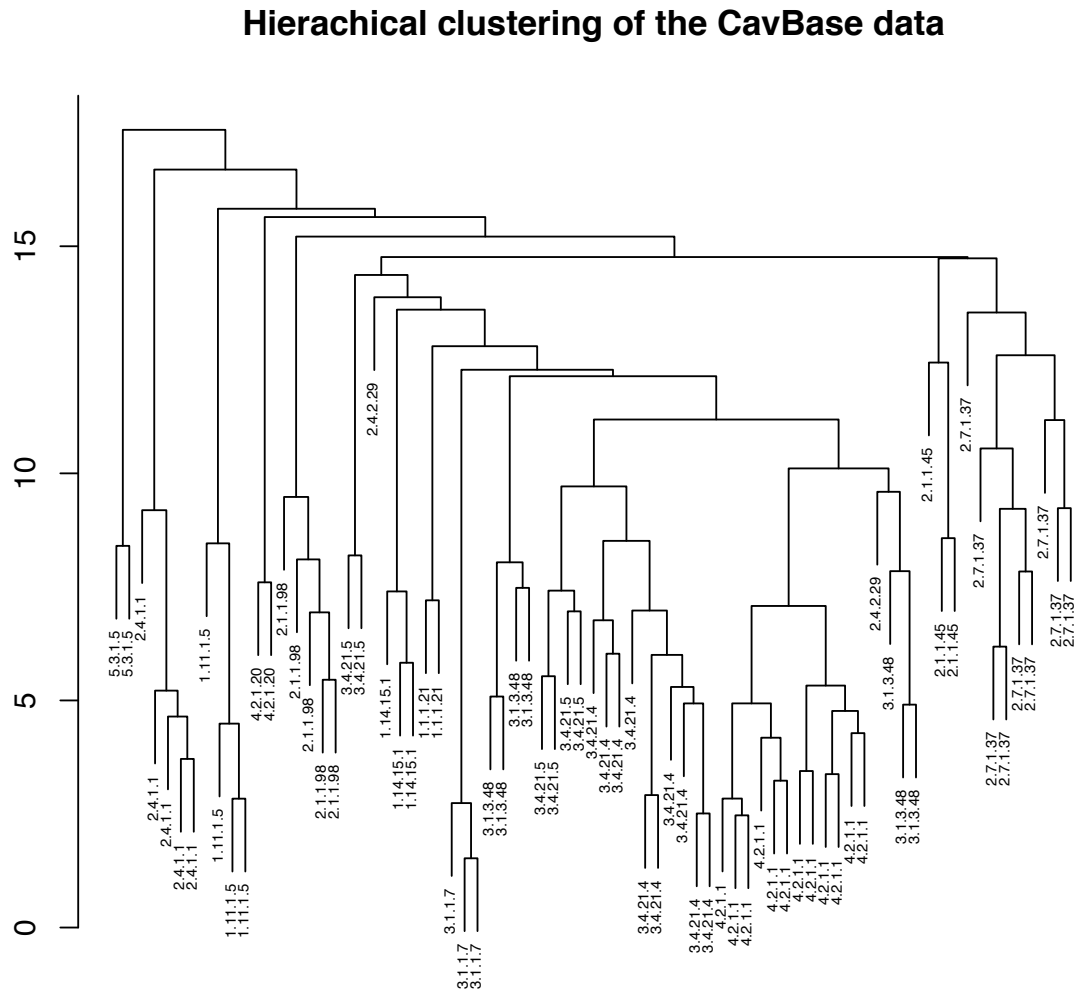


Figure 4.5: Hierarchical kernel clustering of 75 randomly chosen enzymes using the kernels based on the CavBase similarity.

with $I(x)$ the Heaviside function, returning one when its argument is positive, zero when its argument is negative and $1/2$ when its argument is zero. Recall that the ranking accuracy can be considered as a generalization of the area under the ROC curve for more than two ordered classes [107].

Our interest in the ranking accuracy is fueled by two reasons. Unlike the other measures we consider, it analyzes all levels of the EC hierarchy to determine the performance of different algorithms. In our setting predicted rankings can be interpreted as layered or multipartite rankings – see e.g. [106, 28]. Enzymes with exactly the same EC number as the query should appear on top of the predicted ranking, followed by enzymes that share with the query the first three parts of the EC number, and so on. In this way, five layers are maintained in the most general case for a ranking, ranging from all four parts of the EC number identical to none of the four parts identical. The ranking accuracy preserves this hierarchical structure by counting all pairwise comparisons.

As the second reason of interest, the ranking accuracy is also optimized by the RankRLS

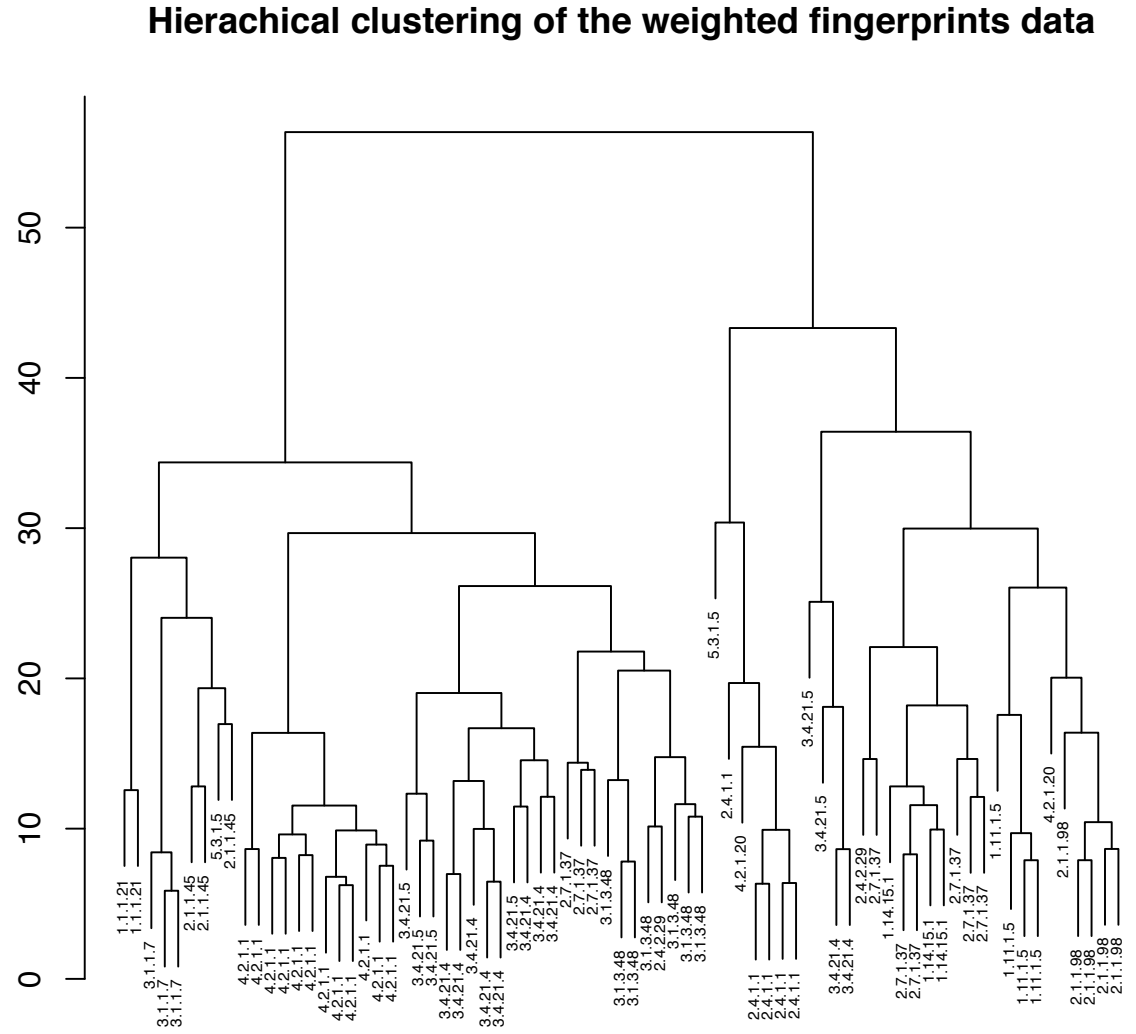


Figure 4.6: Hierarchical kernel clustering of 75 randomly chosen enzymes using the kernels based on the weighted fingerprints similarity.

software, using the convex and differentiable approximation given in equation 4.4. This loss function forms one of the main building blocks of our supervised ranking approach, and it characterizes the most important difference with other kernel-based algorithms for biological network inference, resulting in an information retrieval setting instead of a more traditional network inference setting.

In general one cannot expect that a single learning algorithm can be optimal for different performance measures at the same time. Since our approach is designed for maximizing the ranking accuracy, it can be dominated by other algorithms if other performance measures are analyzed. However, since the ranking accuracy is not generally known in bioinformatics, we also evaluated our algorithms using three more conventional performance measures that are commonly considered for bipartite rankings (i.e., rankings containing relevant versus irrelevant enzymes). These three measures are the area under the ROC curve (AUC), mean average precision (MAP) and normalized discounted cumulative gain

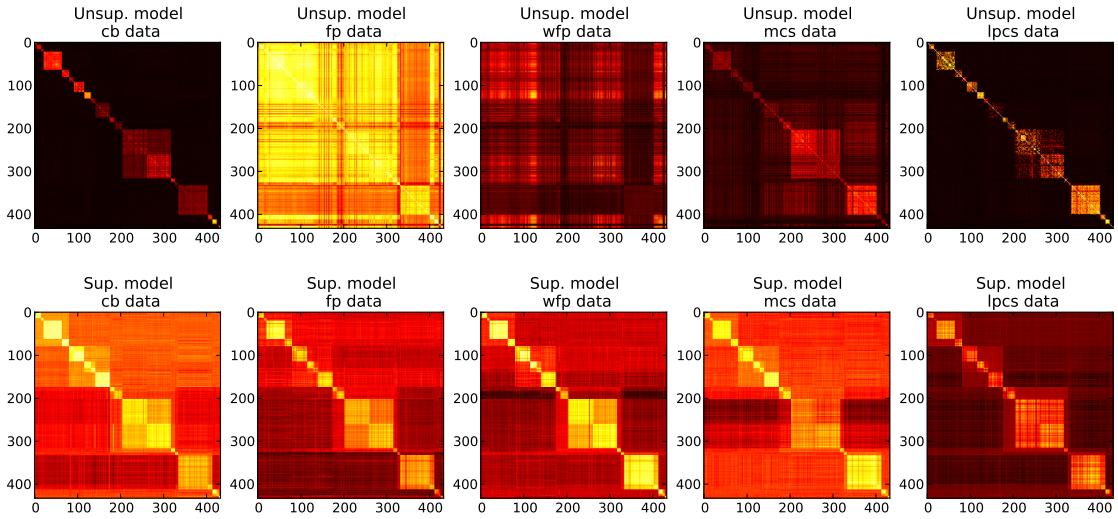


Figure 4.7: Heat maps of the values used for ranking the database during one fold in the testing phase. Each row of the heat map corresponds to one query. This figure can be compared with the ground truth of Figure 4.2

(nDCG). For AUC and MAP all ground-truth rankings had to be converted to bipartite rankings, leading to a decrease in granularity in performance estimation. We chose a cut-off threshold of three in ground-truth similarity: a retrieved enzyme is relevant to the enzyme query if at least the first three parts of its EC number are identical to the query. Figure 4.9 shows the ROC curves that are obtained by applying this cut-off threshold. Again a clear improvement is observed for the supervised methods, resulting in AUCs that almost reach the theoratical optimum of one. This can be motivated by the fact that the AUC is strongly related to the ranking accuracy (similar to ranking accuracies, AUCs were computed as an average over all queries). Moreover, even in the unsupervised case, the AUCs are relatively high. Such a result is in fact not very surprising, given that all analyzed kernels have been introduced in contexts where similar performance measures were considered.

Nonetheless, one can observe that different kernels end up as winner for different performance measures. Differences become most visible when taking a closer look at MAP and nDCG (Table 4.2). Especially for the weighted fingerprints combined with the unsupervised method, the performance strongly decreases. As two key advantages, nDCG can deal with the hierarchical structure of EC an number and it puts more relevance on the objects higher in the ranking. The drawback of this measure is that one needs to set a parameter specifying the position for which the rank is calculated. In this case this position was set equal to ten, meaning that we are particularly interested in the ten highest ranked enzymes.

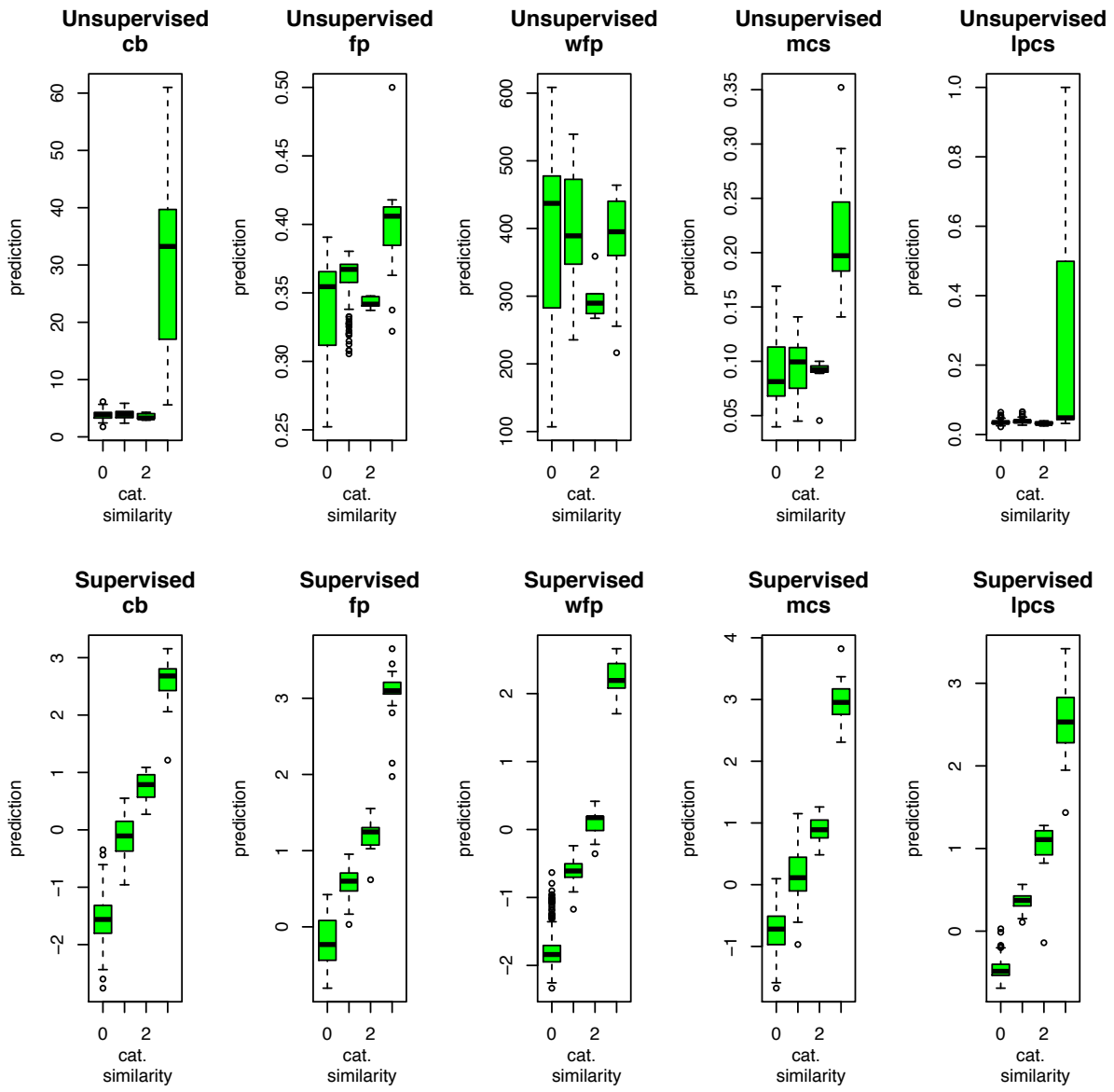


Figure 4.8: The values used for ranking the database for one random chosen enzyme query. The top five plots show the unsupervised model, while the bottom five plots show the supervised model. The predicted values are divided in populations according to the similarity label with the query enzyme.

4.4 Conclusion

In this chapter we reformulated EC number assignment for enzymes as an information retrieval problem. Instead of assigning an EC number to enzymes with an unknown function, our approach predicts a ranking of enzymes that are expected to have a similar function as the target enzyme query. In this way our approach is capable of extracting meaningful information for novel enzymes that can exhibit rare or new functions.

We showed that retrieval of enzymes w.r.t. functionality can be substantially improved by

applying a supervised ranking method that takes advantage of ground-truth EC numbers during the training phase. Supervised ranking outperformed unsupervised ranking in a consistent manner for all kernels and performance measures we considered. While the unsupervised approach succeeded quite well in returning exact matches to a query, the hierarchical structure of EC numbers was better preserved in the rankings predicted by the supervised approach. As such, supervised ranking can be interpreted as a correction mechanism for existing unsupervised methods that rely on the notion of similarity.

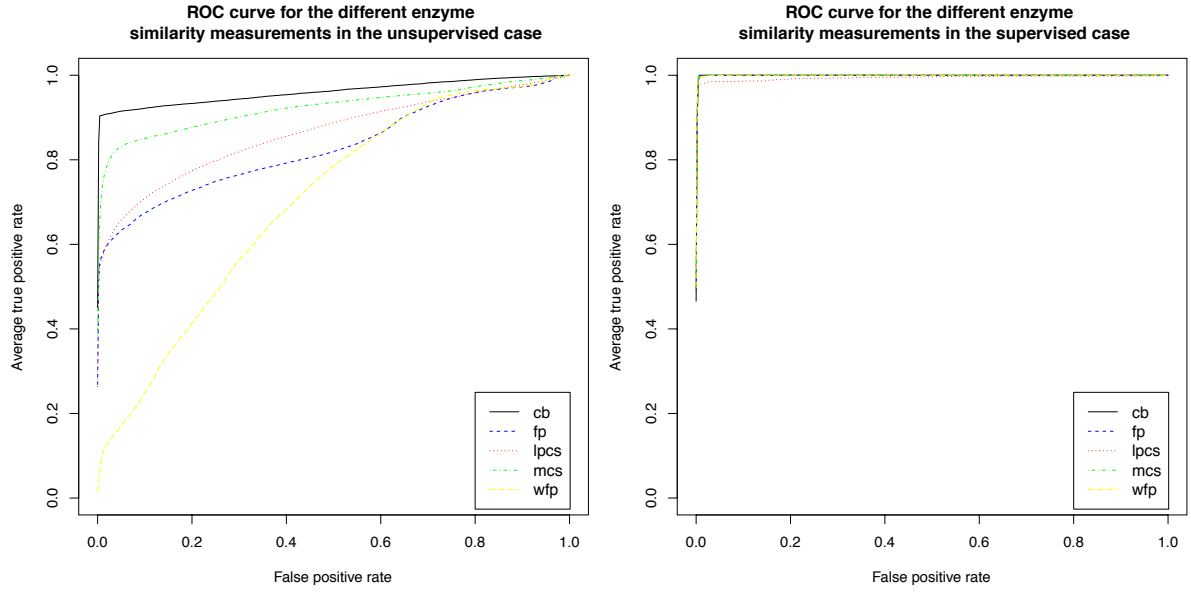


Figure 4.9: ROC curves obtained by means of unsupervised (left) and supervised (right) ranking. An enzyme is considered positive if the first three parts of its EC number are identical to those of the query.

Chapter 5

Protein ligand interactions

5.1 Introduction

The development of new drugs is not a process that only happens in laboratories these days. Computational methods permeate all aspects of drug discovery [42]. Bioinformatics is excessively used to determine potential targets for drugs to work on [89] and algorithms are used for screen databases of compounds in search for interesting leads which can be used for subsequent research. This is usually done by docking, where one attempts to estimate the binding energy between a protein and a chemical compound [60]. Since it rarely happens that an unmodified natural compound or single chemical structure could end up as a drug, subtle modifications have to be added to optimize the pharmacokinetic properties, aided by computational chemistry.

Techniques from machine learning and artificial intelligence are already widely applied in the pharmaceutical sciences [18], for example to predict ADME properties[1] of molecules [84]. In this chapter we will apply some techniques from Chapters 2 and 3 to a problem related to drug design and docking. We will use binary classification and conditional ranking for this task. Our goal is to learn a model that can be used for docking, allowing us to screen a protein database with a ligand or a ligand database with a protein. We have features for proteins and ligands and a docking set at our disposal as input for our algorithm. One of the questions that arise is how a statistical model can compete with or complement a classical docking.

In this case study we will focus on kinases, a diverse family of proteins that can transfer phosphate groups to molecules and other proteins. Kinases are involved in many biological processes, mainly by allowing communication in and between cells and thus form very important targets for drugs [59]. Studies have shown that when a compound can bind to the active site of a kinase, it often shows an inhibitory effect [3]. This shows that kinases are not only attractive targets due to their importance in specific processes, but also because they can potentially be turned off. Our framework could also prove to be very useful because issues such as specificity and selectivity of the compound is naturally incorporated in the framework.

[1] ADME stands for absorption, distribution, metabolism and excretion and are the most important physiochemical properties that determine whether a molecule could show any bioactivity *in vivo*.

5.2 Material and methods

5.2.1 The Karaman dataset

The Karaman dataset [45] is a large benchmark for studying kinase inhibitor selectivity. This data set contains information of the interactions of 38 known kinase inhibitors with 317 kinase targets. With more than 50% of the human kinome covered, it is expected to be very useful in pharmaceutical research.

Of the possible 12046 protein-ligand combinations, 3175 (26.4%) showed a significant binding interaction. The interation is quantified as the disociation coefficient K_d in nM for the following reaction:

$$\mathrm{PL} \rightleftharpoons \mathrm{P} + \mathrm{L}, \tag{5.1}$$

with P the protein, L the ligand and PL the complex of the kinase and its inhibitor. K_d is then defined as:

$$K_d = \frac{[\mathrm{P}][\mathrm{L}]}{[\mathrm{PL}]}. \tag{5.2}$$

Thus larger values for K_d indicate a weaker interaction. Figure 5.1 summarizes the data set. It can be seen that there is a wealth of different patterns of selectivity of the compounds. Some kinase inhibitors, such as staurosporine, show strong binding to almost all the kinases, while for example SB-202190 binds to a select few kinases but also from different families, while lapatinib and GW-2580 are examples of compounds that only interact with a very limited number of kinases. Clearly this dataset shows some interesting variety. To build a predictive model features where needed for both the proteins as well as the ligands, as to be discussed in the next sections. Because of only a subset of all the kinases in the data set only a subset had relevant protein structure available, we could only use 127 kinases in the modeling phase. Since it was possible to construct features for all the ligands, we could keep these combinations with all the kinases. Thus the Karaman data set was reduced to 127×38 interactions.

The dissociation constants range from values in the order of 1 nM (a very strong binding) to plus infinity (no binding to the protein). Both to obtain a more meaningful and easy to understand range of values and to be (expected to) better processed with our algorithms we perform a scaling. To justify this scaling recall the Van 't Hoff equation for isotherm reactions:

$$\ln K_d = -\frac{\Delta G_d}{RT}, \tag{5.3}$$

with ΔG_d the change in Gibbs free energy[2] of dissociation, R the gas constant and T the temperature. The logarithm of the dissociation constant is thus proportional to the negative of the Gibbs free energy of dissociation (of which the change of enthalpy is positive as it costs energy to dissociate the complex and the change of entropy is positive as more free molecules are formed). We propose to scale the data by using

$$y = \log_{10}(1/K_d + 1). \tag{5.4}$$

This new label y is equal to zero when there is no binding interaction between a protein and a ligand and it is roughly proportional to the binding free energy, larger values indi-

[2]Gibbs free energy is defined as the enthalpy minus the entropy times the temperature: $G = H - TS$

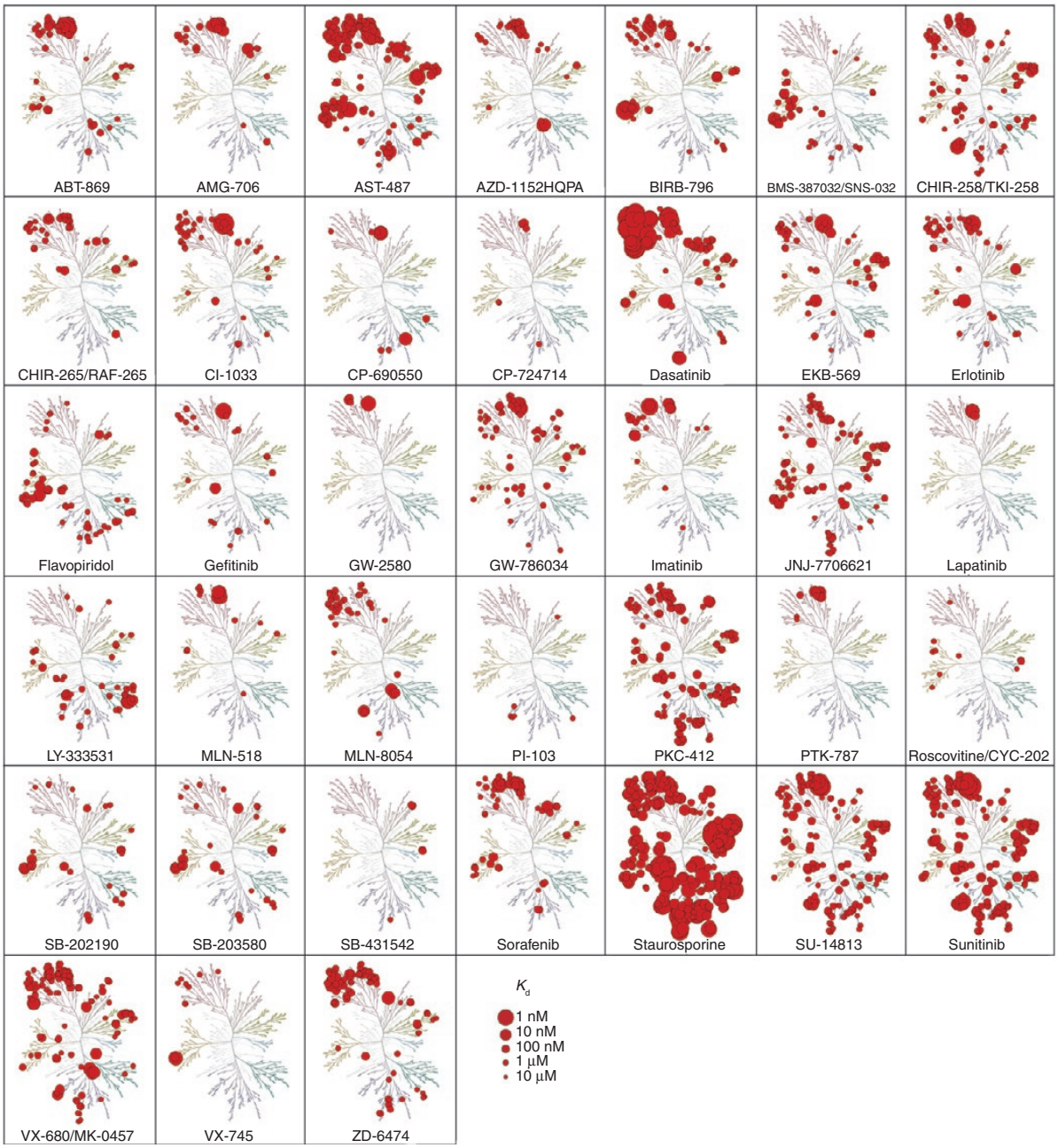


Figure 5.1: Small molecule-kinase interaction maps for the 38 kinase inhibitors. Kinases that are found to bind are marked as red circles, where the size corresponds to binding affinity. [45]

cate a stronger binding interaction. Since this is a monotonic scaling it has no influence on our ranking experiments. Figure 5.2 shows a heatmap of the final data.

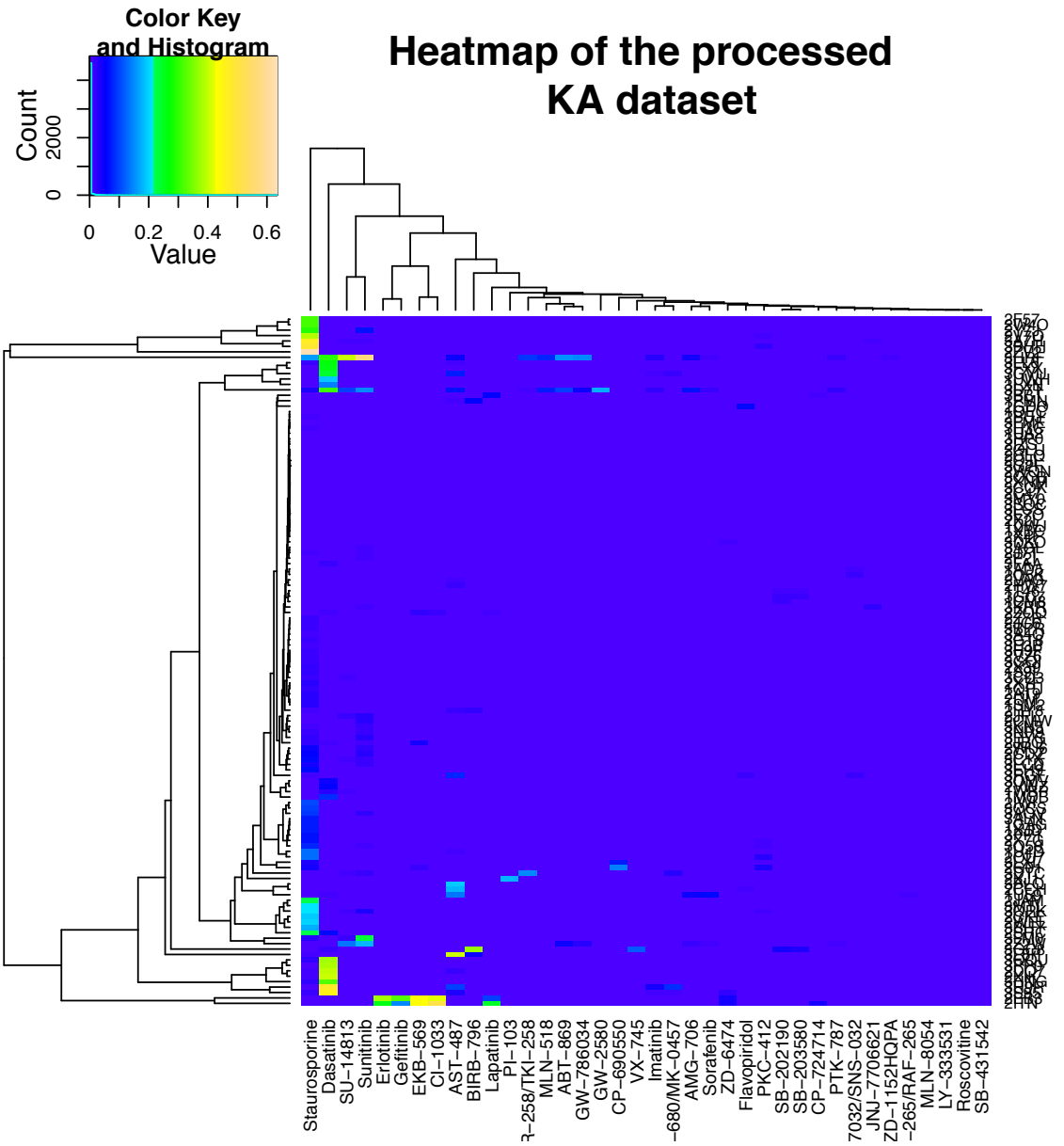


Figure 5.2: Heat map of the Karaman data set after withholding only the proteins of which a valid structure could be obtained and after scaling of the data.

5.2.2 The proteins

Of the 317 proteins, 127 had valid crystal structures which could be used to construct a feature representation. These were obtained from the Protein Data Bank. Many proteins had multiple crystal structures which could be used due to the different possible conformation changes. When multiple structures correspond to one protein, we chose the crystal structure in agreement with the docking results as described in Section 5.2.4. Using the CavBase similarity [86] a similarity matrix was constructed for the predicted active site (more information can be found in Section 2.7). Due to heuristics this matrix had to be made symmetric by averaging this matrix with its transpose. Though technically not hundred percent correct, this similarity matrix is treated as kernel matrix for the

subsequent experiments, as there was only a minor violation for the positive definiteness requirement. A heatmap of this kernel matrix can be found in Figure 5.3.

5.2.3 The ligands

The 38 ligands were processed in Molecular Operating Environment (MOE) to compute features derived from the graph representation of the molecules, given in the form of nine vectors of binary fingerprints. These were concatenated together and the kernel matrix was constructed by applying the Tanimoto similarity (see Section 2.4.4), visualized in Figure 5.4.

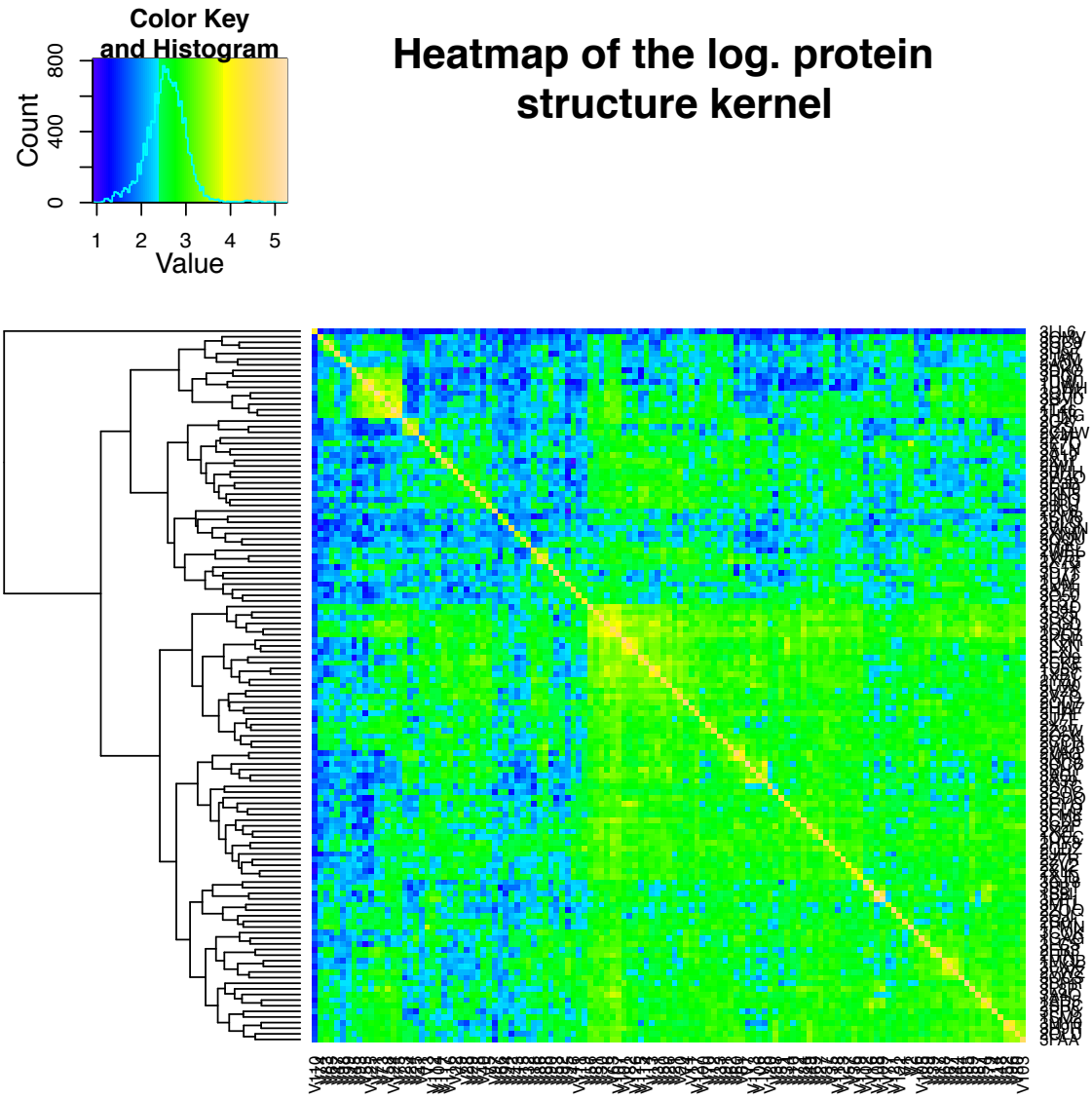


Figure 5.3: Heat map of the kernel of the protein structures that were used for the Karaman data set.

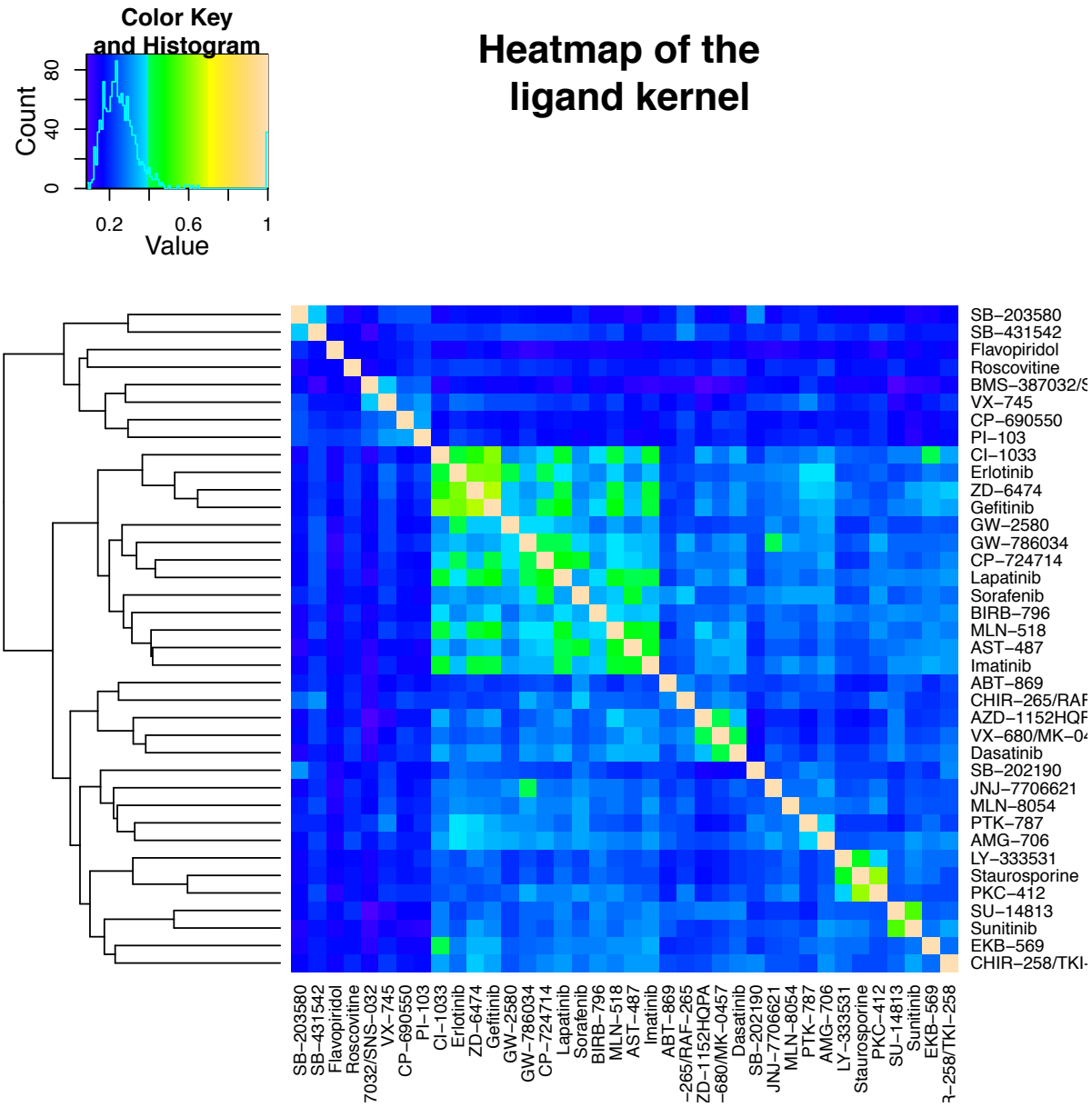


Figure 5.4: Heat map of the kernel of the ligands used in the Karaman data set.

5.2.4 Docking results

The third kind of data available for predicting the interaction between the proteins and the ligands is a virtual docking of the 650 structures of the 127 proteins withhold with the 38 ligands. For every combination of structure and ligand, a value between minus infinity and infinity that is approximately proportional to the binding energy was computed. Since our goal is producing a statistical model to predict interaction, this data can be seen as a noisy and unscaled (values are only comparable for the same protein) solution to our problem.

Since there are on average about 5 crystal structures for each protein the question arises which structure should be used to represent which protein. We followed the reasoning that only one of the crystal structures is the most relevant for binding all the ligands,

we withhold the structure which scores the best for each kinase inhibitor. To be more specific, for each protein we selected the structure that had the best (lowest) docking value for the highest number of ligands. This is expected to be the most relevant structure for predicting binding interaction and was subsequently used for both the kernel derived from docking and the kernel for the individual proteins as described in Section 5.2.2. The docking data is shown as a heatmap in Figure 5.5.

To construct a kernel matrix from the docking data, each score was treated as a feature vector of length one of a tuple containing a protein and a ligand. A pairwise kernel is then constructed as a linear kernel of the 'features' of two pairs. More formally, if the docking score of the protein p and the ligand l is referenced to as $D(p, l)$, then the docking kernel is defined as:

$$K^D((p, l), (p', l')) = D(p, l) * D(p', l'). \tag{5.5}$$

5.3 Modeling and results

In this section we discuss the statistical models we used to try to predict the interactions between the kinases and the kinase inhibitors. We use the data discussed above to construct relational models based on kernels as described in Chapter 3. By combining a kernel directly derived from docking results with a kernel that combines features of the proteins and the ligands, we hope to obtain a powerful method as we combine a mechanistic model with an empirical one. A similar setup, but without docking results as features, was performed by Jacob et al. (2008)..

The kernel K^{PL} that combines the features of the proteins with those of the ligands is simply obtained by taking the Kronecker product of the kernels of the objects in question. These kernels were combined by simply taking a linear combination:

$$K = \omega K^D + (1 - \omega) K^{PL}, \tag{5.6}$$

where we tested all the models of this section for $\omega = \{0, 0.2, 0.4, \ldots, 1\}$.

Unfortunately we found no useful results in the docking data. When only using K^{PL} results where always respectable, but increasing the weight ω was similar to adding more and more noise to the data to the point that with $\omega = 1$ (using only only the docking results) the models performed no better than a random classifier or ranker. We could not detect any correlation between the docking results and the Karaman data set, leading us to believe that the former is somehow corrupted.

Needless to say, this leaves us with some mixed feelings. At the one hand, we are very glad we could build a working model using the features for the proteins and the ligands, something which was though as difficult. On the other hand, it is a letdown that we could not test if these could be used to amplify the performance of docking, thus potentially allowing a state-of-the-art prediction. Since the docking kernel could not be used, any models with $\omega > 0$ will not be discussed in the next sections.

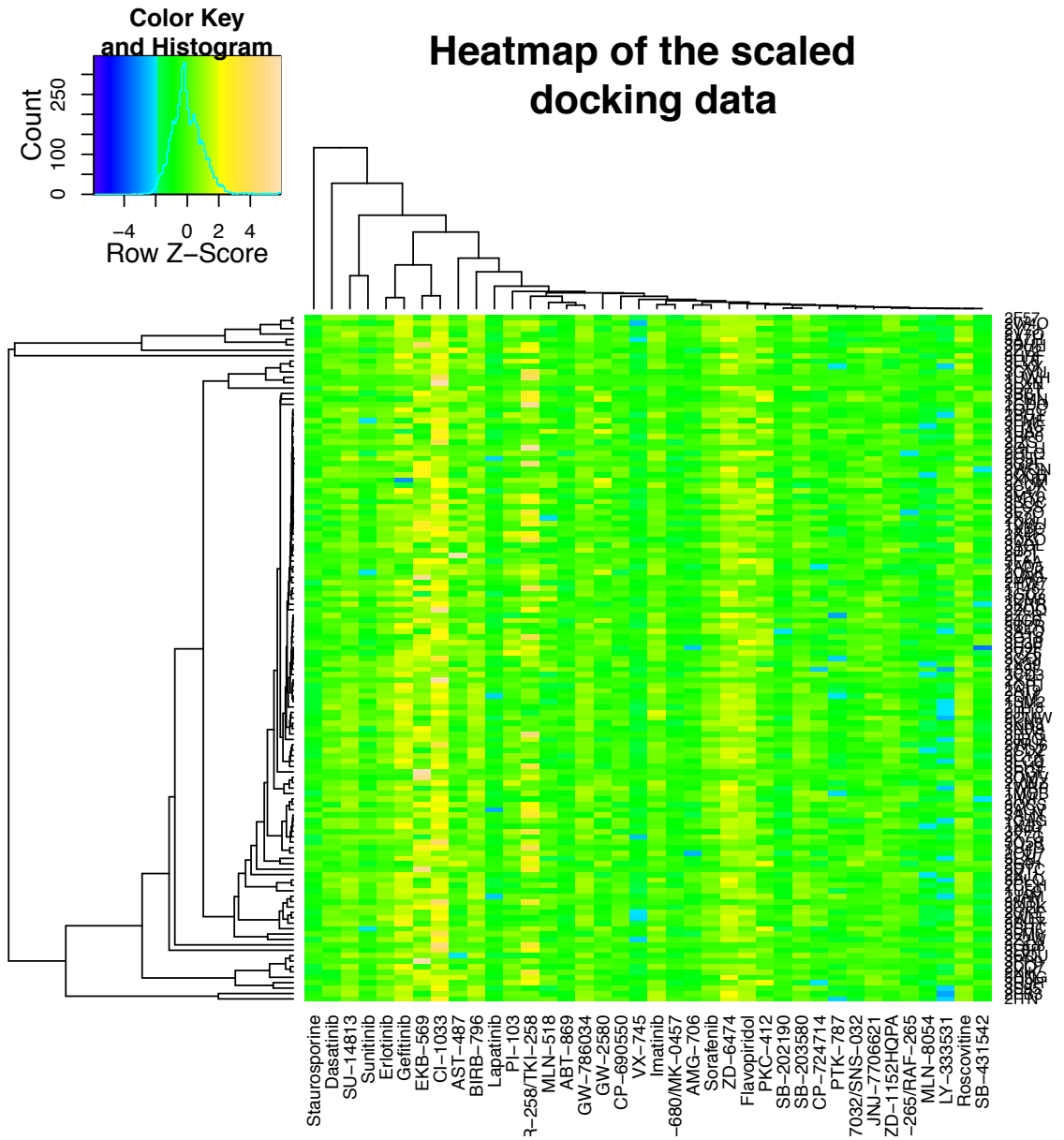


Figure 5.5: Heatmap of the docking results after withholding only the most relevant structure for each protein. Values are scaled using a Z-score. The dendrograms are calculated on the original Karaman dataset allowing easy comparison with Figure 5.2.

5.3.1 Classification

In this section we treat the problem as a binary classification setting by trying to predict whether a protein will bind to a ligand or not. To this end we apply a cutoff of either 1000 or 10000 nM to the Karaman data set, values below this cutoff will be considered as positive instances, the others as negative ones. Since the vast majority of the instances are negative we have chosen the area under the ROC curve (AUC) as a performance measure, rather than the misclassification error, as the former deals with the unbalanced nature of the data set.

The pairwise model was obtained by using regularized least squares. The optimal regu-

larization parameter λ was chosen from a grid of $\{2^{-10}, 2^{-8}, 2^{-6}, \ldots, 2^{10}, 2^{12}\}$ by using a six-fold cross validation within the training set.

As discussed in Section 3.5, multiple schemes of how the test set is sampled can be distinguished. We will again denote these schemes as α, β, γ and δ.

- α: this is the setting where testing is done by using new combinations of objects in which both the protein and the ligand were used before in the training set. This was performed by randomizing all the pairs and using a five-fold cross validation to test the model.
- β: this setting considers the case when performing testing with a new ligand and a familiar protein. The testing is performed by withholding each ligand once for testing.
- γ: is similar to β, with the difference that each protein is withheld once for testing, thus assessing how the model can deal with a combination of a new protein and a known ligand.
- δ: here it is tested how the model performs on pairs where both the protein and the ligand are previously unseen in the testing phase. In each round all the combinations of one fifth of the proteins and one fifth of the ligands are selected and used for testing while the combinations of the remaining proteins and ligands were used for building the model when both the training and the test set where not homogenous. This procedure was repeated 250 times.

The performances of these models are given in Table 5.1. All models seem to perform comfortably better than random. A first thing that can be noticed is that the problem seems to be easier to learn with a higher threshold (thus using more positive instances and making the data set more balanced). Setting γ seems to be the exception on this rule, though the small difference with large standard deviations make probably this less relevant.

Though setting α was expected to be the most easy case, as both objects of the test-pairs are familiar from training, it seems to be overshadowed by setting γ where only the ligand in the test pair is previously seen. It should be remembered that these are

Table 5.1: Performance of the models for inferring interaction by means of binary classification. Only the kernel based of the features of the proteins and ligands is used. For each performance the cutoff and the test sampling method is given.

Test sampling	Cutoff $[nM]$	AUC
α	1000	0.724240 (0.019201)
α	10000	0.758080 (0.023105)
β	1000	0.621584 (0.104163)
β	10000	0.653330 (0.107727)
γ	1000	0.812184 (0.185627)
γ	10000	0.801310 (0.157205)
δ	1000	0.636947 (0.037966)
δ	10000	0.676178 (0.036850)

actually quite different problems: the test set of α contains pairs of all the possible objects while γ considers all the combinations of ligands with one particular (unseen) protein. Another factor that could contribute to this difference in performance is that the former model is always trained with 80% of the data while the latter uses 99.2% of the data for training each model. Consequently, testing is performed using 20% and 0.8% of the data respectively. Considering the similarity between the settings β and γ, it is somewhat strange that these have the greatest difference in performance. This might indicate that our framework could generalize better to a new protein than to a new ligand. It can be noted that the CavBase similarity used for the former is state of the art and based on features that are highly relevant to the current problem setting. Our experience with the enzyme case study also suggests that this similarity is on the whole very reliable. In contrast, the features of the ligands are more general graph-based fingerprints that might not describe them in particulary relevant way for our problem.

When considering the standard deviations calculated on the different performances, one obtains a measure for the reliability of the models. Setting α and δ have a low standard deviation, while β and γ have a very high standard deviation. This is perfectly within expectations when one considers the sampling used for generating the test set. In the first case, elements are randomly selected from the complete data set, thus every testing problem is expected to be equally hard as all test sets originate from the same distribution. The higher standard deviation of δ is simply due to the higher sample size compared to α. In the test sets of the second case one object of each pair is fixed. There will almost certainly be sets that are very easy to predict (because for example the protein acts very similar to one that was in the test set) or have sets that are very difficult to perform well on. The lower variance of β may again be due to the larger sample size.

5.3.2 Conditional ranking

Rather than trying to predict whether a compound will bind or not, it is probably more meaningful to obtain a ranking of the compounds according to their (expected) binding potential to a certain protein or a ranking of the proteins conditioned on a ligand. To this end we use the framework for conditional ranking from Section 3.3 to produce such a model.

We always consider both the ligands as the proteins as queries as both approaches are meaningful. Estimation of the regularization parameter was done by using a leave-one-query-out approach on a grid with $\lambda = \{2^{-10}, 2^{-8}, 2^{-6}, \ldots, 2^{10}, 2^{12}\}$.

Again it is possible to define multiple sampling methods for testing, but due to the sparseness of the dataset not all of them are possible. Ranking is only possible when there is enough variability in the dataset, e.g. there has to be at least one label different from the other in a database. Since we have seen in Section 5.2.1 that the Karaman data set contains multiple inhibitors that only bind to a limited number of proteins and the other way around, we will only consider only two of the four possible settings as we feel these are he only we can compute statistical founded. We will denote these cases as κ and σ to avoid confusion with the settings of the classification experiments:

- κ: in this setting both the query and the database elements used for testing were encountered during the training phase, but the query was not used for ranking the

database in question. The performance was assessed by withholding for each query a quarter of the database elements for testing and repeating this three times for every query. Notice that for all the other queries not used for testing were used with all the database elements during training.

- σ: in this setting it is tested how the model performs with a new query against a familiar data base. The performance is in this case obtained by simply using a leaving each query out for testing and averaging over these queries.

The results for the conditional ranking with these different settings are given in Table 5.2. Since for all the settings we obtain a ranking error of roughly 35%, we can conclude that there is at least some signal in the data and we are able to learn from the data.

It is interesting to note that setting κ seems to be harder to learn than σ, especially for the case when using ligands as queries. This is counter intuitive as the former used more data for training and should be expected to be easier. It is expected that this is due to sampling problems. In Figure 5.6 the number of binding partners for the ligands and proteins are plotted as a histogram. Since the ligands have about 30 interactions and the proteins 9 binding partners, on average. This low number of partners, especially for the proteins, give rise to high variances for the ranking error, as for setting κ only a quarter of the database corresponding to a query was used for testing. Thus, the test set is likely to contain only a very limited number of positive interactions and might here not be the most efficient or unbiased estimator of the true ranking error. When a test set contained no positive instances, it could not be used as no ranking error could be calculated. The problem is probably in the testing phase, as for training setting κ the same data is used for training σ with the addition of a set of database elements whitin the test-query.

For the test setting σ, given the relatively high standard deviation, using ligands or proteins as queries makes hardly any difference and the ranking error is about 32.5%. There is some difference in the standard deviation between the two types, using a protein query seems to be more reliable than using a ligand query, as indicated by the standard deviation (though both are more reliable than the models of setting κ). The higher uncertainty of the ranking error when using ligands as a query is probably due to a greater difference in the number of interacting partners compared to the proteins, as indicated again by the histograms of Figure 5.6.

Table 5.2: Results of the conditional ranking ranking experiments for inferring interaction between the kinases and kinase inhibitors. For each experiment the test sampling method and the query type is given.

Test sampling	Query type	Ranking Error
κ	Ligand	0.381124 (0.177515)
κ	Protein	0.330250 (0.207463)
σ	Ligand	0.324000 (0.129307)
σ	Protein	0.32799 (0.088344)

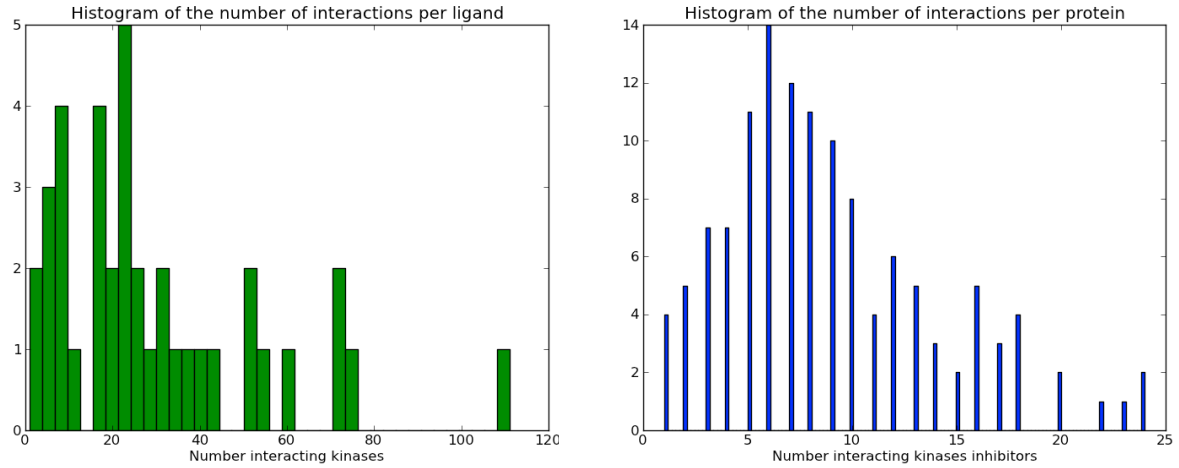


Figure 5.6: Number of interaction partners by type.

5.4 Conclusion

The goal of this chapter was to apply some of the techniques from relational learning to try to predict a binding interaction between a set of kinases and kinase inhibitors. This was approached by both learning binary classification and conditional ranking. Using a feature representation of the proteins and ligands we succeeded in building different models with reasonable predictive power, given the expected difficulty of this problem. Though we could not test this, we believe that our results could be greatly improved by using (successful) docking data. A better model could probably also obtained by using a better feature representation for the ligands.

The performance and reliability of the models were found to be dependent of the way the model was used. This has a great influence in the real-life applications of our models, as one has to keep in mind for what the algorithms are to be used, depending on the degree of new proteins or ligands used.

Using conditional ranking is a promising way to tackle this docking problem as a ranking, since it presents the results in a way that is relevant for the user (an ordered list of the most promising elements) moreover it can also deal with some important issues in drug design such as selectivity and specificity. We found that our ranking models are roughly symmetric with respect to the queries, e.g. it matters less whether a protein or ligand query is used. The model could also generalize well to new queries.

Chapter 6

Microbial ecology

6.1 Introduction

Bacteria are of the utmost importance in both natural and manmade systems. They form complex networks that play key roles in the biogeochemical cycles of the earth and human health, but are also indispensable in waste water treatment, composting and, more recently, in bio-electrochemical systems [77]. Microbial resource management is the term used for utilizing and steering these ecosystems to humanity's advantage [102].

To be able to effectively implement actions to influence these systems, a quantitative understanding is required. Though these global networks are still poorly understood, modern techniques in molecular analysis and bioinformatics allow large scale analysis of microbial networks, such as done for example by Chaffron et al. (2010). A lot of interesting models for microbial interaction have been constructed, usually based on the principle of exchanging metabolites across species [15, 43, 49, 50]. These models are as a rule deductive in the sense that they are based on models for the individual species.

In this chapter we want to construct a statistical model which can be of aid to an environmental engineer when applying these ideas of synthetic ecology or microbial resource management. *In casu*, we want to develop some type of information retrieval algorithm which allows the engineer to obtain a list of useful microbial partners he can add to his culture to make the system more robust, faster or allow a better performance. The data that is used for this model should be obtained in a series of simple, quick and relatively high throughput experiments.

The system in question is the interaction between methanotrophic and heterotrophic bacteria. These potential partners are being used in several environmental processes, such as biodegradation of toxic chemicals [32, 38] and in the production of bioplastics [35]. Methanotrophs also play an important role in the methane cycle (and may thus be of relevance in dealing with global warming) and forming cooperations with higher organisms. The ultimate goal of our model is to be able to predict one or more desirable heterotrophic partners for a given methanotrophic bacterium species.

6.2 Material and methods

6.2.1 The methanotrophs

Methanotrophic bacteria use methane as their sole carbon and energy source. They are a subgroup of the methylotrophic bacteria that use one-carbon compounds more reduced than formic acid. Methane is oxidized to formaldehyde which can be assimilated and enter the metabolism of the organism or can be further degraded and serve as an energy and electron source [32]. These pathways are shown in Figure 6.1. The key enzyme in this process is methane monooxygenase (MMO) which oxidizes methane to methanol, but can also play a role in certain reactions in bioremediation. This enzyme can occur in two forms: a soluble or cytoplasmic methane monooxygenase (sMMO) synthesized by low concentrations of copper in the medium while higher concentrations of copper induce a membrane-associated or particulate methane monooxygenase (pMMO) [63]. The methanotrophs are both of a great taxonomic diversity as well as of great ecological importance. In our experiments we use ten different species to represent this diversity. These are listed in Table 6.1.

Of these species we have the DNA sequences of their methane monooxygenase subunit alpha. It is our hope and expectations that we can extract features of these sequences, which reflect both the functional differences in their MMO gene as well as the taxonomical relation between the different bacteria. With this goal in mind we construct two different kernel matrices of these sequences.

Our first similarity measure is based on multiple sequence alignment. It is known that this method can shed light on the evolution of a protein family and it is an indisposable tool for studying phylogeny, structure, conserved motifs and domains of protein families [66]. For the alignment we use Muscle, a faster and more accurate multiple sequence alignment tool than the better known ClustalW [21]. Of this MSA the Jukes-Cantor similarity measure was calculated. Stricktly spreaking this is not a valid kernel as the condition of positive definiteness is not necessarily fulfilled, but as we did in the previous case studies we can use this similarity matrix anyway without any grave consequences.

The second kernel matrix is generated by applying the spectrum kernel [57] on all the pairs of the DNA sequences. The normalized form is used and length of the k-mers is

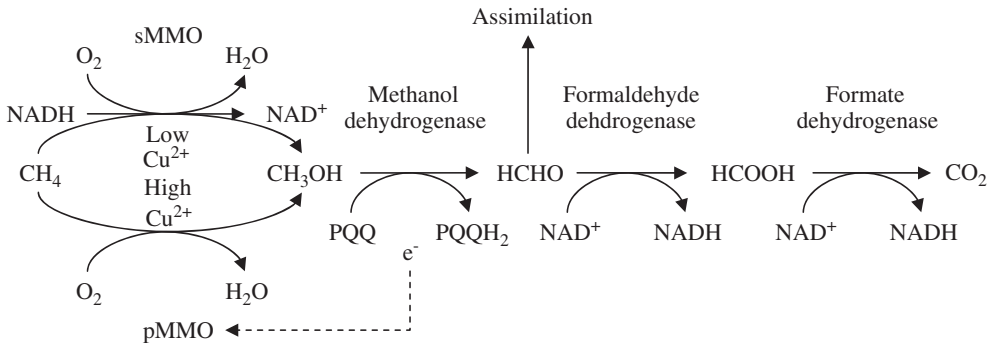


Figure 6.1: Basic metabolism of methane oxidation in methanotrophic bacteria [62].

Table 6.1: Table of the different methanotropic bacteria used in this setup.

ID	Strain number	Species name
1	R-45363	*Methylomonas* sp.
2	R-45364	*Methylomonas* sp.
3	R-45374	*Methylomonas* sp.
4	R-45378	*Methylomonas* sp.
5	R-45383	*Methylomonas* sp.
6	R-45379	*Methylomonas* sp.
7	DSM 18500T	*Methylocystis hirsuta*
8	NCIMB 11130T	*Methylomonas methanica*
9	NCIMB 11129T	*Methylocystis parvus*
10	DSM 13736T	*Methylosarcina fibrata*

chosen to be three, to enable the codons to be compared. More information about the spectrum kernel and its properties can be found in Section 2.4.1.

A comparison between the kernels and the phylogeny can be found in Figure 6.2. The phylogenetic trees were drawn using *Phylogeny.fr* [16]. These trees show that the main structure between the different kernels is very similar, thus both represent some phylogenetic and functional information about both the gene homology as well as the species. It should be mentioned that constructing a phylogenetic tree is not equivalent to performing clustering as the former requires an evolutionary model.

6.2.2 The heterotrophs

The heterotrophs we will study use more complex carbon sources as building blocks and energy source. These microorganisms are listed in Table 6.2. This collection contains 26 prokaryotic bacteria and one yeast: *Pichia pastoris* which is used in the wet labs experiments but is not used for building the mathematical models in later sections due to its difference with the rest of the instances. All these micro organisms have a completely sequenced genome.

When cocultivating these organisms together with the methanotrophic bacteria we expect that there will be a mutual influence on each others growth pattern. Since the organisms are grown in a medium with methane as a sole carbon source, the only way the heterotrophs can utilize carbon is through the methanotrophs. This can be because the latter might leak carbon compounds or because the heterotrophs feed on the necromass of the methanotrophs. A carbon flux derived from methane across different species was observed using stable isotope probing [64, 76]. The methanotrophs could potentially be influenced by the heterotrophs if the latter produce useful vitamins or harmful antibiotical substances. The exchange of cobalamin (also known as vitamin B12) from the former to the latter has been observed by Iguchi et. al. using using a setup somewhat similar to ours.

Since the exact mechanism how the heterotrophs influence the methanotrophs is unknown we select features that represent their carbohydrate metabolism and the part of the metabolism dealing with cofactors, vitamins, prosthetic groups and pigments. This

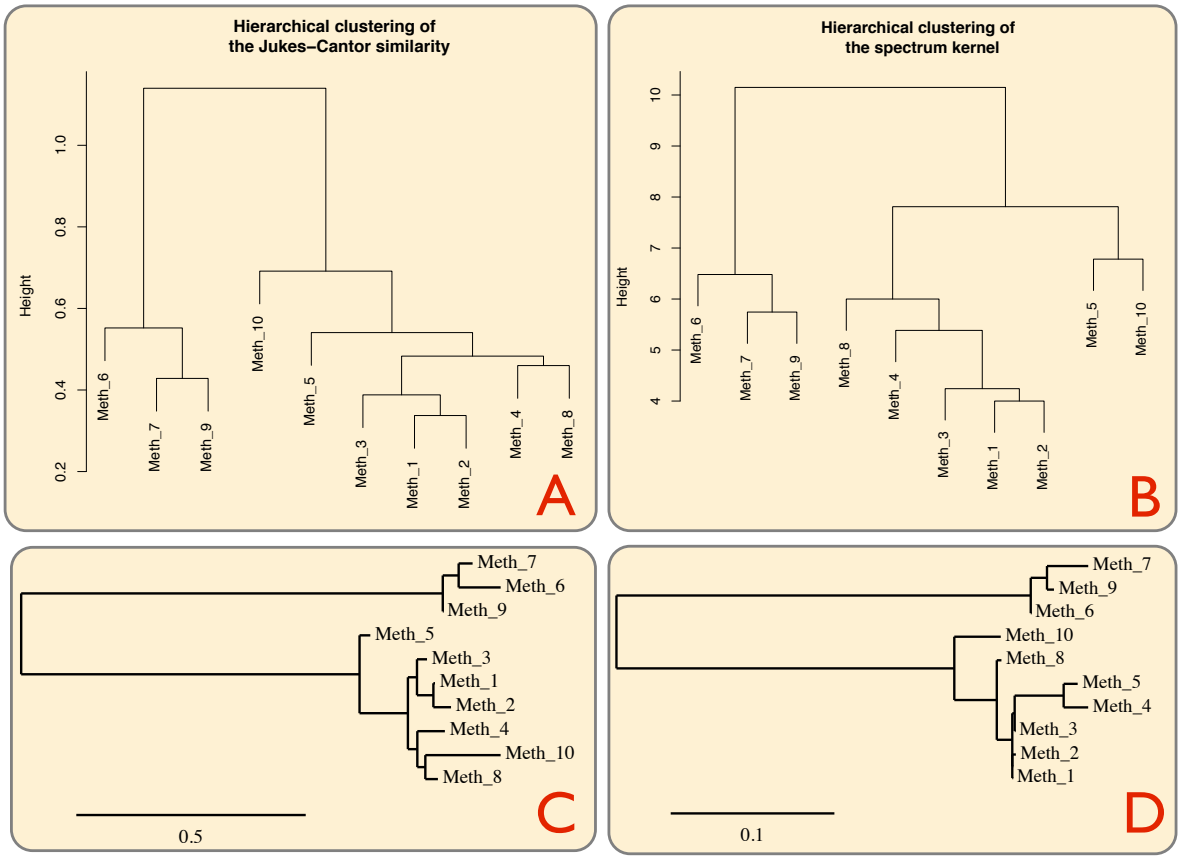


Figure 6.2: (A) Hierarchical clustering of the methanotrophs in the kernel space derived from the Jukes-Cantor similarity. (B) Hierarchical clustering of the methanotrophs in the kernel space derived from the Jukes-Cantor similarity. (C) Phylogenetic tree drawn from the different MMO genes. (D) Phylogenetic tree drawn from the ribosomal DNA of the different species.

data is obtained from the SEED-Viewer [68]. Here a list of all the genes that an organism possesses for a particular biological function can be generated. The genomes of the bacteria with IDs 6, 11, 14, 15 and 16 were manually submitted to the automatic annotation tool RAST [8] so that the features of these organisms also became available. Since *Pichia pastoris* was not used for building models, no features were derived for this yeast. In total a list of 1029 genes involved in carbohydrate metabolism and 1712 genes dealing with the secondary metabolism is used to describe these organisms.

Thus, every bacterium is represented by two binary strings of features: each bit indicates whether a particular gene of the carbohydrate metabolism or secondary metabolism is present in this particular organism or not. Of these strings we can build two types of kernels: the linear and the Tanimoto kernel. The linear kernel is obtained by taking the dot product of the feature vectors, or stated otherwise, the kernel value of two bacteria is the number of relevant genes both species have in common. Note that this measure is not normalized in the sense that two species with many genes will have a higher value than two with less genes if the fraction of common genes is the same. A kernel that represents the two types of metabolic systems is obtained by summing the two kernels

Table 6.2: Table of the different heterotrophic bacteria used in this setup. The strain number refers to the number used in the Laboratory of Microbiology culture collection.

ID	Species name	Strain ID
1	*Acinetobacter baumannii*	LMG 1025
2	*Bacillus azotoformans*	LMG 9581T
3	*Bacillus bataviensis*	LMG 21833T
4	*Bacillus licheniformis*	LMG 6933
5	*Bacillus vireti*	LMG 21834T
6	*Cupriavidus metallidurans*	LMG 1195
7	*Cupriavidus taiwanensis*	LMG 19424
8	*Escherichia coli*	R-23895
9	*Escherichia coli*	R-23891
10	*Escherichia coli*	R-23894
11	*Escherichia fergusonii*	LMG 7866
12	*Flavobacterium johnsoniae*	LMG 1341
13	*Methylibium petroleiphilum*	LMG 22953
14	*Methylobacterium nodulans*	R-7055
15	*Methylobacterium radiotolerans*	LMG 2269
16	*Ochrobactrum anthropi*	LMG 2134
17	*Pseudomonas aeruginosa*	LMG 12228
18	*Pseudomonas putida*	LMG 24210
19	*Pseudomonas putida*	R-17801
20	*Rhizobium radiobacter*	LMG 287
21	*Ensifer (Sinorhizobium) meliloti*	R-20688
22	*Rhodobacter sphaeroides*	LMG 2827
23	*Roseobacter denitricans*	LMG 19751
24	*Shigella flexneri*	R-23896
25	*Staphylococcus aureus*	R-23700
26	*Staphylococcus aureus subsp. aureus*	R-23902
27	*Pichia pastoris*	-

for each system. For the linear kernel this is the same as taking a linear kernel of the concatenated feature strings of both organisms.

We have also built kernels using the Tanimoto similariy discussed in more detail in Section 2.4.4. This kernel also uses the size of the intersection of the two gene sets to be compared but is normalized by dividing by the union of the two sets. Thus the Tanimoto similarity of an object with itself is always equal to one. The two Tanimoto kernels for the different systems are summed to form a third kernel that encloses information about both the carbohydrate and the secondary metabolism. Note that due to the non-linear nature of this similarity this kernel does not equal to the kernel value of the concatenated feature vectors. The resulting six kernel matrices are represented as heatmaps in Figure 6.3.

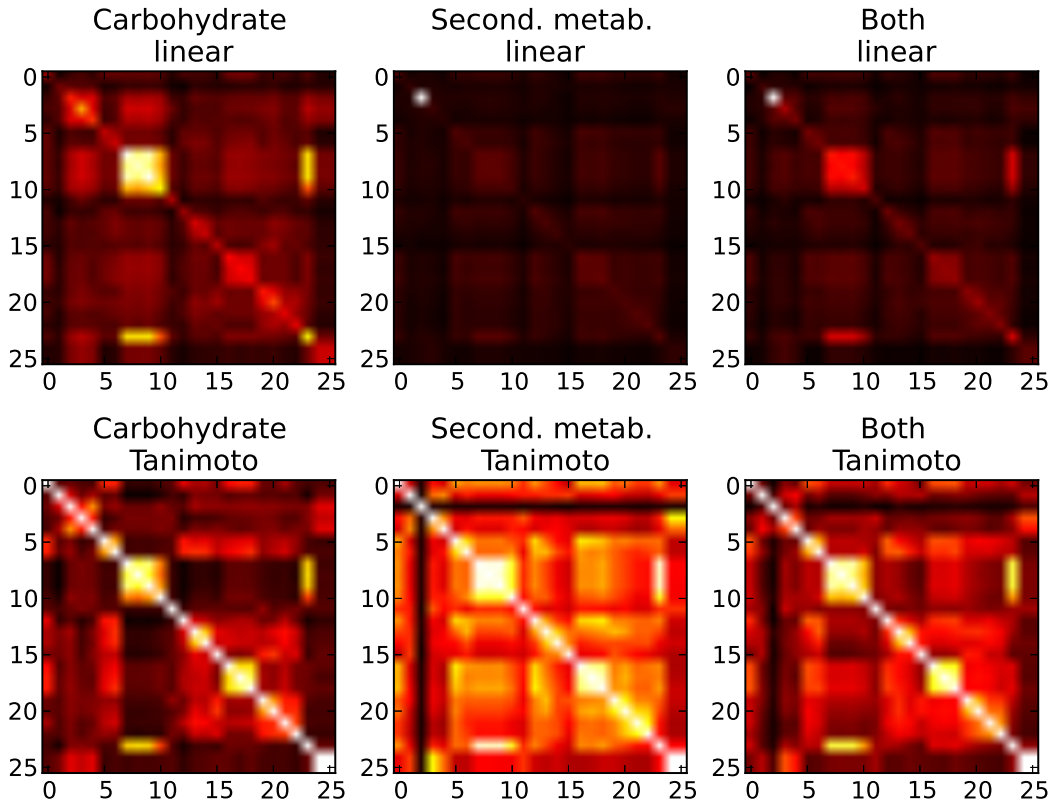


Figure 6.3: Heatmaps of the kernel matrices constructed for the heterotrophs. The columns represent the different kinds of data used while the two rows indicate different kernel functions to represent this data. All kernels look sufficiently different to assume they encode different information about the bacteria.

6.2.3 Experimental setup

The goal of the experiments that were performed was to obtain data that could be used to build a statistical model which can predict the interaction between the two types of micro organisms. For this reason we need to cultivate the combinations of methanotrophs and heterotrophs, preferably with replications, and also grow them individually. We also need as many measurements of their (collective) growth to obtain a clear image of how the influence each other.

These microorganisms were grown in 96 well plates in nitrate mineral salts (NMS) medium under a methane atmosphere. The NMS medium contains a nitrogen source, minerals and a buffer but no carbon source. Thus growth is only directly possible for the methanotrophs that can metabolize this gas. In this experiment we use ten methanotrophs plus one control containing only medium, 27 heterotrophs plus one control containing only medium. Every combination is performed in triplicate and for every plate nine blanco wells are needed for calibrating the measurement. Thus there were eleven 96 well plates, one for every type of methanotroph (including on for the combinations without methanotroph) where the heterotrophs are repeated. The layout of one plate is then given in Table 6.3. These plates were stocked in airtight jars with a part of the atmosphere re-

placed by methane. These in turn were placed in a incubation room. Figure 6.4 shows some photos of the experiment.

Before the experiment was started the bacteria were grown on petri dishes. At the beginning of the experiment the wells were inoculated with a cell density which was just over the detection limit. On the first day and every two days for two weeks a sample of 30 μL was taken from each of the wells and cell density was determined by measuring optical density (OD) at 600 nm. Thus, each pair of bacteria could be given eight OD measurements at different times spaced over fourteen days.

We would like to state that some wells were hard to measure due to some combinations form excessive amounts of slime after about a week, which made precise pippetation sometimes impossible. In some cases the slime could not be dissolved in the dilution for OD measurement, giving results that were above the maximum detection limit of the machine. This is a flaw in the experimental setup and could probably be circumvented with better in a more sophisticated setup, but as the goal was mainly to create a benchmark dataset in a high throughput fashion as a proof of concept for the use of our models, these issues shall be largely ignored. OD values that were above the upper detection limit were set to the maximum value encountered in the dataset.

6.2.4 Experimental results and analysis

Of all the different combinations of bacteria, a growth curve could be constructed for each of the three replicates. Though the different combinations showed growth curves with range of different shapes, the curves within a repetition were, as a rule, remarkably consistent. Four of these growth curves are given in Figure 6.5 as an example.

For every combination of bacteria we have data in the form of a time series. However, for our algorithms, both regression and ranking, we need a real value. This is dealt with by deriving three relevant statistics that can be processed by the algorithms of these plots. In each case the median of this statistic is taken across the repetitions as a robust estimator.

For this first measure we take the maximum OD density obtained over the time interval.

Table 6.3: Layout of one 96 well plate. All wells are also inoculated with one type of methanotroph, except for those denoted as empty, which only contain the NMS medium. The heterotrophs are not in exactly the same order as in Table 6.2 due to a moment of weakness on part of the student who performed the experiments.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
A	Hetero 1			Empty			Hetero 17			Hetero 25		
B	Hetero 2			Hetero 10			Hetero 18			Hetero 26		
C	Hetero 3			Hetero 11			Hetero 19			Hetero 27		
D	Hetero 4			Hetero 12			Hetero 20			NMS medium		
E	Hetero 5			Hetero 13			Hetero 21			Hetero 9		
F	Hetero 6			Hetero 14			Hetero 22			Empty		
G	Hetero 7			Hetero 15			Hetero 23			Empty		
H	Hetero 8			Hetero 16			Hetero 24			Empty		

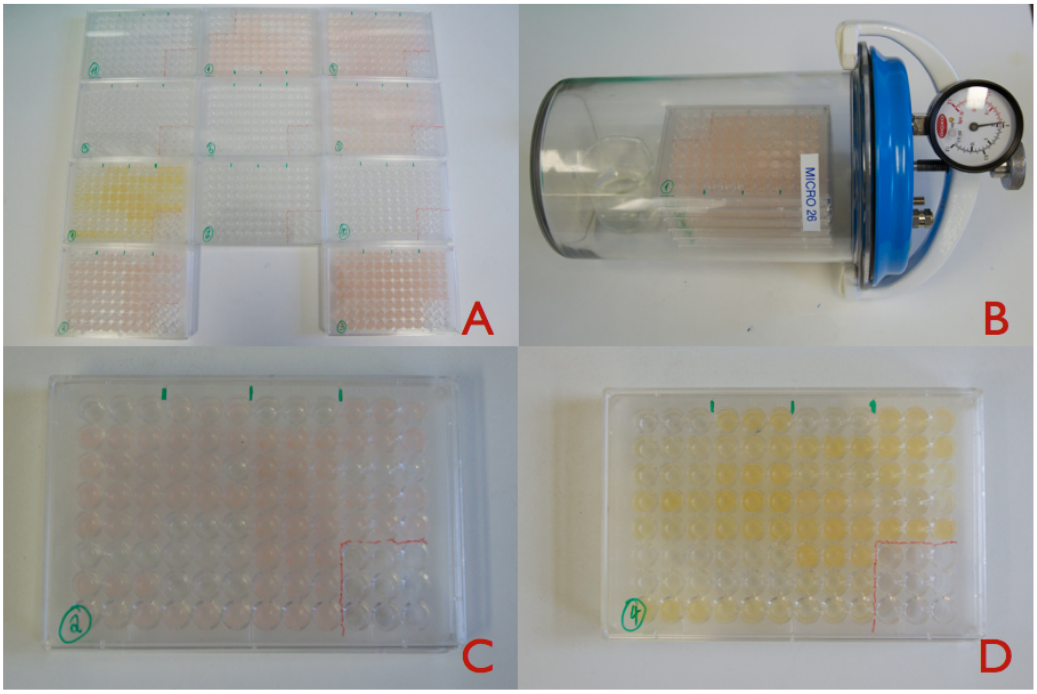


Figure 6.4: (A) The eleven 96 well plates after one week of incubation. Some methanotrophs produce a pigment which makes it possible to examine their growth on sight. (B) The plates are stocked in an airtight jar of which part of the atmosphere is replaced with methane to allow as a nutrition source for the methanotrophs. A little beaker of water is placed in the jar to control the humidity and prevent the wells from evaporating too much water. (C) and (D). After one week clear patterns that are consequent within replicates emerge.

This is an important statistic as it indicates the maximum cell density before cell growth stops due to limitation of nutrients or accumulation of toxic wastes and net cell starvation starts to kick in. A rise in maximum OD with respect to a pure methanotrophic culture may be due to an enhanced growth of the methanotroph, the heterotroph or both. A heatmap of the median of this measure is given in Figure 6.6.

The second statistic is the time (in days after incubation) when the maximum OD was reached. This is more indicative for the dynamics of the (diauxic) growth rather than the potential for maximum growth. It is important to note that this value is discrete rather than continuous, which will have some impact on the regression and ranking experiments that were performed. The median of this statistic is given in the heatmap of Figure 6.7.

As a final statistic we take the area under the growth curve, which is proportional with the time averaged OD. Combinations that can optimize their resources will probably have a high area under the curve. This measure is shown in Figure 6.8 for the different combinations.

The heatmaps show that these statistics are not necessarily coupled. An engineer may be interested in different statistics, for example if he wants to optimize long term cooperation,

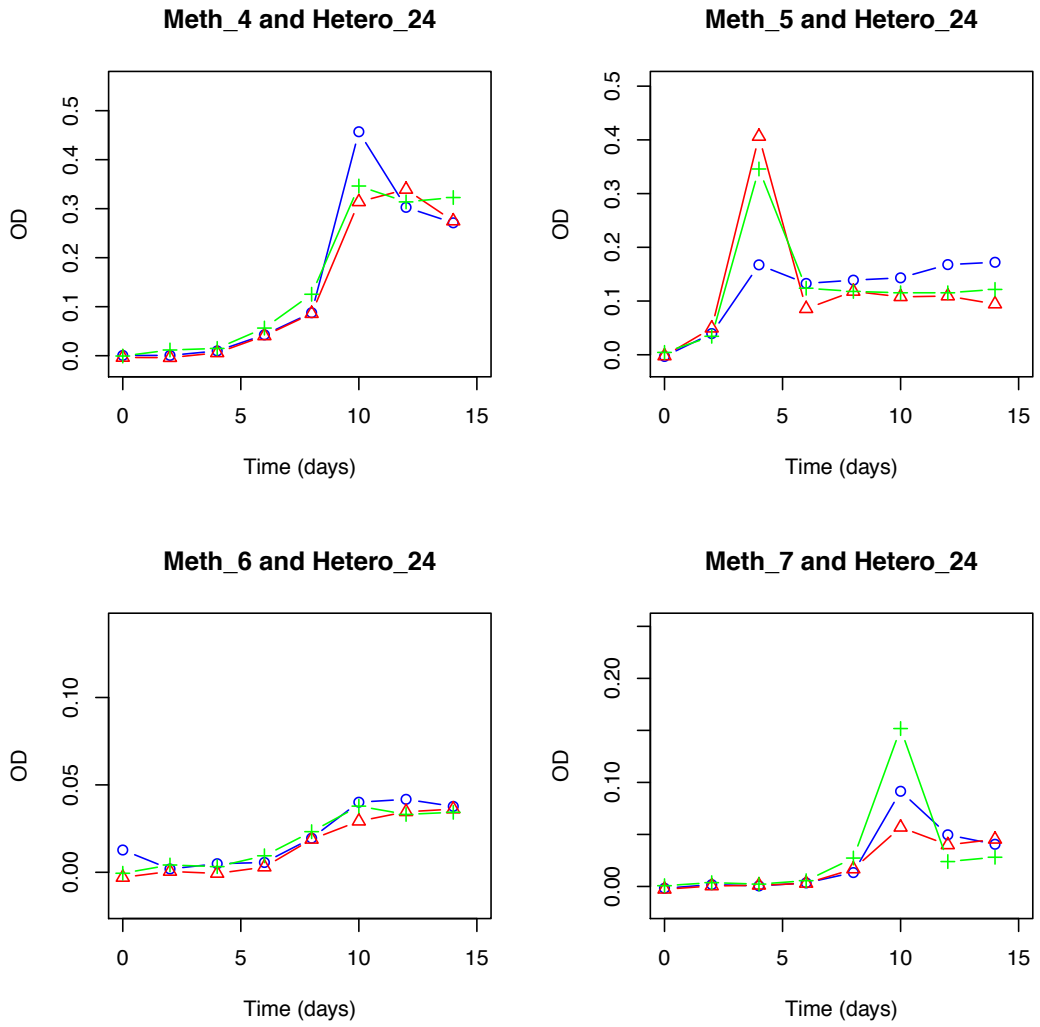


Figure 6.5: Growth curves for four combinations of methanotrophs and heterotrophs. Of every combination three repetitions were performed to evaluate to consistency of the experiments.

he is probably most interested in adding cultures that are predicted to increase the area under the OD curve. On the other hand, if he wants to try to decrease the start up time for a bioreactor the time when maximal OD was reached in the experiments will be more relevant.

6.3 Modeling and results

Before we view the models that were built for regression and conditional ranking it is probably advised to discuss the differences and similarities of both approaches and what their practical use is.

In the regression setting we will try to predict the labels derived from the OD measure-

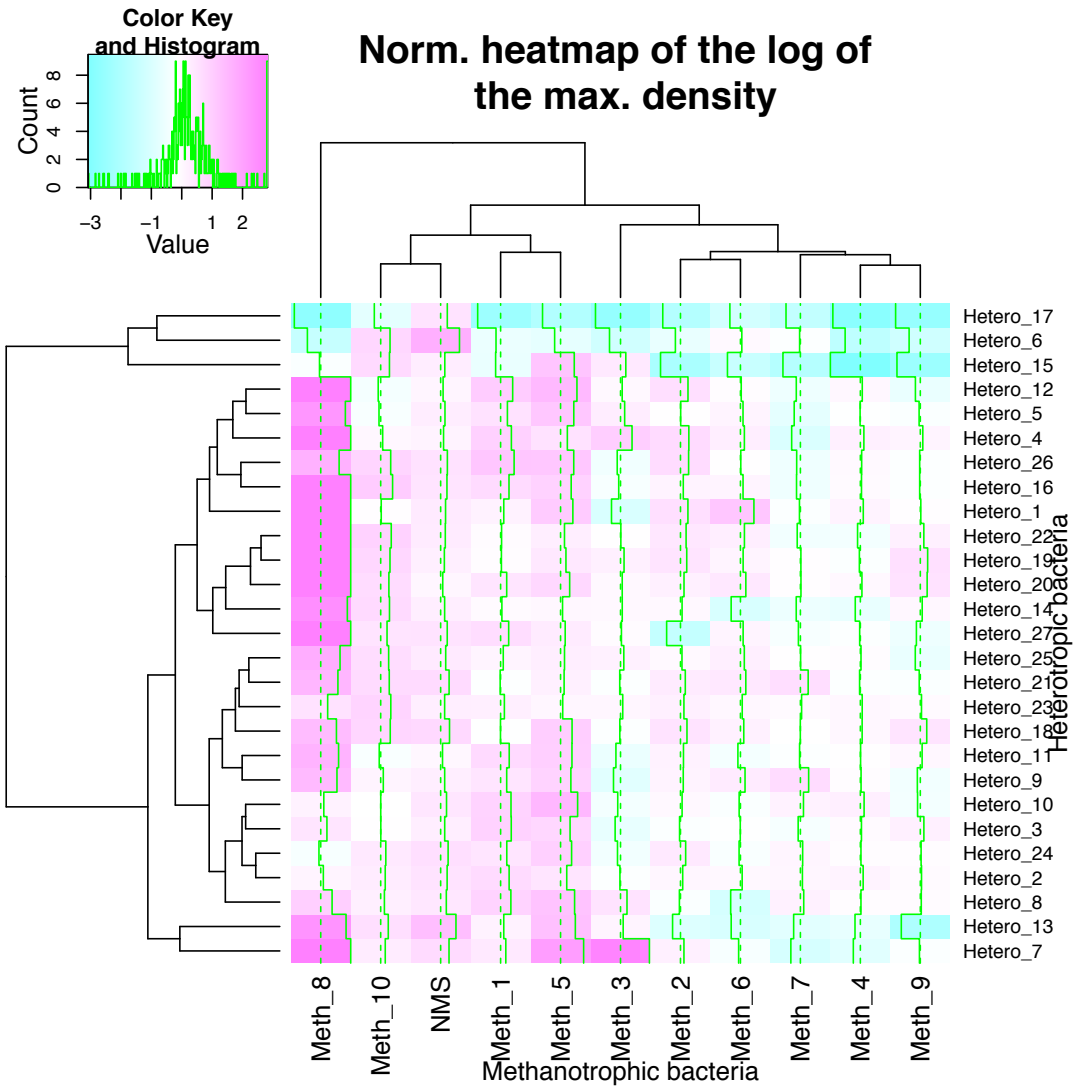


Figure 6.6: Median of the logarithm of the maximum optical density for the different combinations of bacteria. Results are corrected for the methanotroph, meaning that for every combination of bacteria, the value of the methanotroph of the pair grown with NMS was subtracted. This way only the effect of the heterotroph is shown. The clustering and dendrogram is calculated based on the maximum optical density.

ments, given a feature representation of the two bacterial types. We do not consider the methanotrophs more relevant than the heterotrophs or vice versa. This means that the labels to be predicted are not normalized using the metric of the single methanotroph, nor of the single heterotroph. For example, in heatmaps of Figures 6.6, 6.7 and 6.8 the metrics of each combination were subtracted with the value of the corresponding methanotroph with NMS medium, thus assessing the effect of the heterotroph on the growth. Though this is an interesting setting on its own, in our regression experiments we will treat both partners equal, thus performing no normalization in either way.

In conditional ranking one naturally enforces an asymmetry to the data, a one has to choose a data type as a query while the other serves as database elements. The methan-

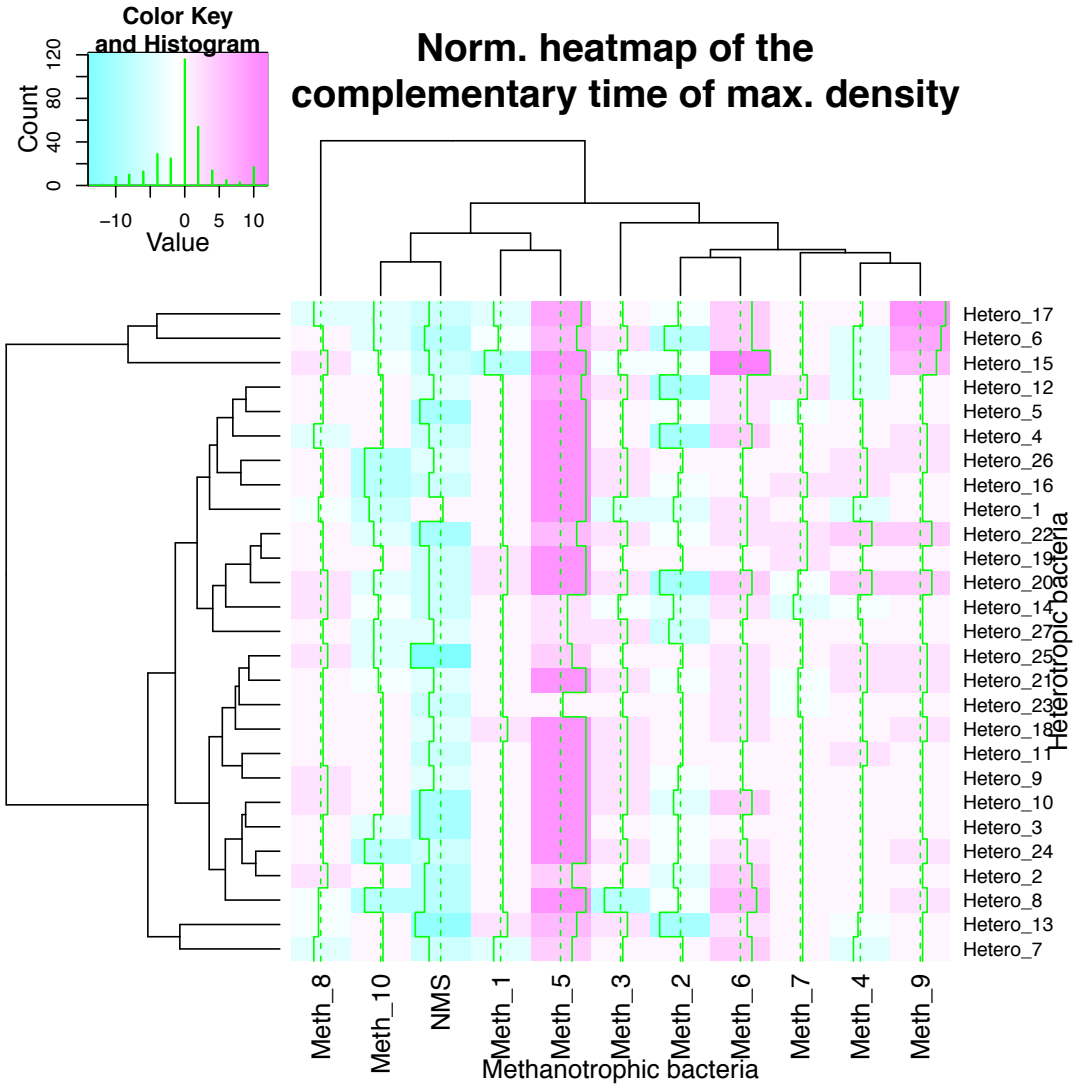


Figure 6.7: Median of the time until day fourteen when the maximum optical density for the different combinations of bacteria was reached. Results are corrected for the methanotroph, meaning that for every combination of bacteria, the value of the methanotroph of the pair grown with NMS was subtracted. This way only the effect of the heterotroph is shown. The clustering and dendrogram is calculated based on the maximum optical density, allowing comparison with the other heatmaps.

otrophs will be chosen as queries for both technical and biological reasons. Their low number of instances make them statistically and computational less feasible, while their ability to capture carbon in the system is an argument to put them as the center of our focus. Thus the problem that is assessed in the conditional ranking could be stated as follows: given a methanotroph, rank the heterotrophs according to their ability to influence their combined growth from most positive to most negative. In this setting normalizing as described above is meaningless as one only considers the metric of a pair of bacteria relative to each other with the methanotroph fixed for each ranking.

Both regression as conditional ranking are considered as applications standing on their

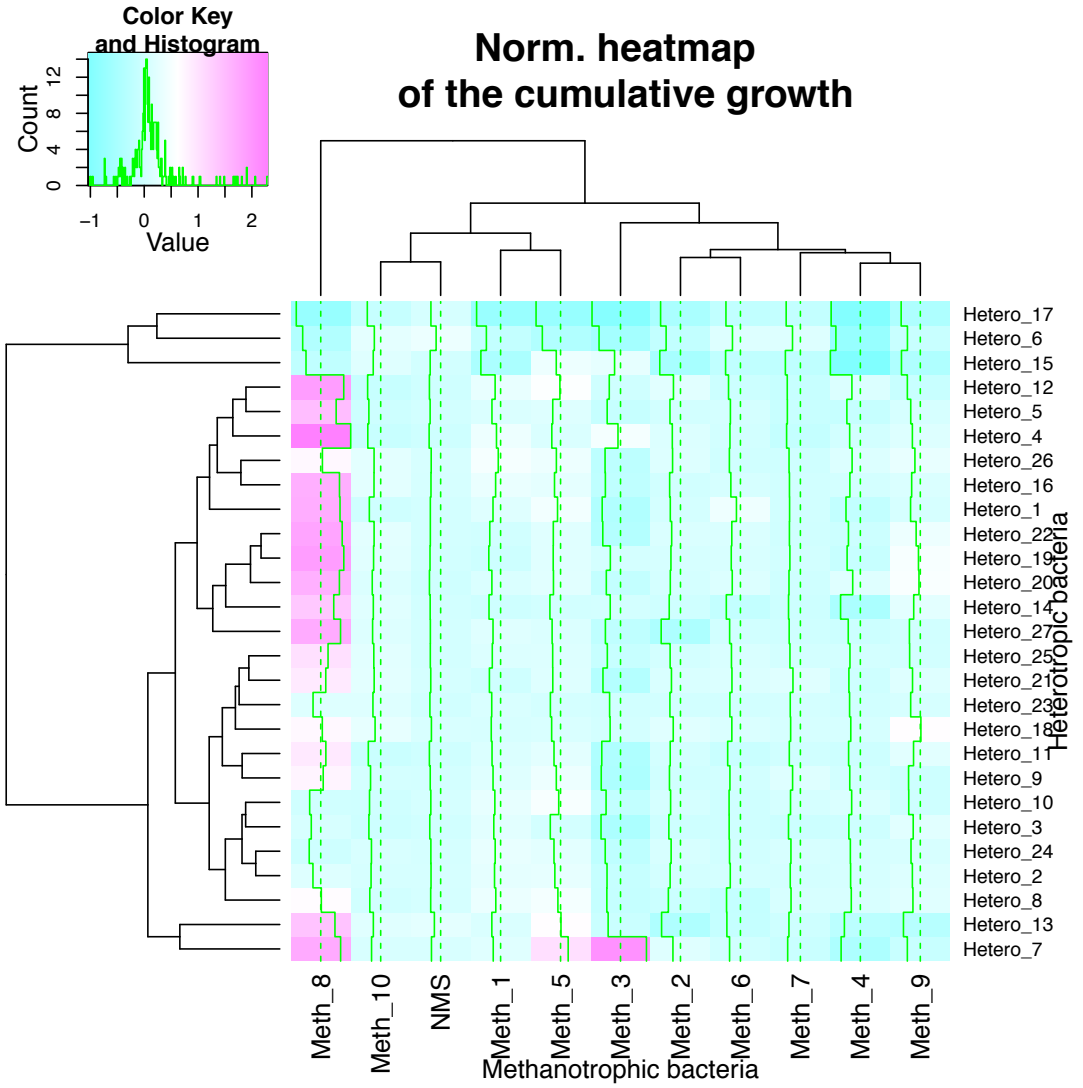


Figure 6.8: Median of the area under the growth curve for the different combinations of bacteria. Results are corrected for the methanotroph, meaning that for every combination of bacteria, the value of the methanotroph of the pair grown with NMS was subtracted. This way only the effect of the heterotroph is shown. The clustering and dendrogram is calculated based on the maximum optical density, allowing comparison with the other heatmaps.

own. All testing schemes where designed to research how a model with this particular data would perform in a real life setting. That is why only small amounts of test instances were withheld, to allow training with an approximately complete data set. From a machine learning point of view it is tempting to compare regression with conditional ranking to find the 'better' model, especially since for both the ranking error was (among others) calculated as a performance error. Such comparisons are meaningless as we have indicated above, since both approaches serve for different problems. A different setup could be constructed to specifically asses the difference in regression and conditional ranking from a statistical perspective. As with the previous two chapters, our focus is here on the application, as our data sets are rather novel and to answer these questions a well

characterized benchmark data set would probably more suitable.

We consider the performance of many different models using different data, different labels, for ranking or for regression and using different testing schemes. In general, it is not possible to determine the 'best' model as this is heavily dependent on the application the user has in mind.

6.3.1 Regression

In a first series of experiments it is investigated if regression can be used to predict the OD value of the different combinations of methanotrophic and heterotrophic bacteria. The values to be inferred are the maximum of the OD, the time when this maximum was reached and the cumulative growth, as described in Section 6.2.4. To build this model we used the features of the heterotrophs and methanotrophs, thus having the Jukes-Cantor similarity and the spectrum kernel for the former and six kernel matrices derived from fingerprints dealing with the carbohydrate and secondary metabolism. Using the Kronecker product pairwise kernel method, described in detail in Chapter 3, we could thus generate twelve different kernel matrices.

We would like to stress the particular difficulty of the problem on which we release our algorithm. Any scientist who has ever dealt with living matter in the laboratory could tell that microorganisms are not deterministically described by a sequence of DNA or a list of genes. This is why the goal of these regression experiments is to have some predictive power that could be indicative for microbiologists or environmental engineers, rather than predicting the cell density to an unrealistic accuracy.

The regression models are built using support vector machines from the Kernlab package in the programming language R. The optimal regularization parameter was estimated using a tenfold cross validation of a grid, which contained all integer powers of two from -2 to 7. Because of the difficulty of this problem, we consider multiple performance measures to evaluate the regression. The first is the well-known mean squared residual error (MSE). But since squaring the residual error puts a big emphasis on the largest error, we also consider the mean absolute error (MAE), which is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|e_i|, \tag{6.1}$$

where e_i is the residual error of testing instance i. By taking the absolute value rather than the square of the residuals we can obtain a more robust performance measure. As a final performance measure we consider the ranking error (RE), thus only evaluating whether the algorithm could be used to predict which combinations could obtain higher cell densities relative to other combinations. It is of great importance to note that this is not equivalent to the condition ranking experiments as to be discussed in Section 6.3.2. In that case a model is trained with a specific ranking loss function, while the case discussed here is optimized for regression.

Because things were not quite as complicated as could be, we also distinguish four cases of model testing, as discussed in Section 3.5. Thus for each of these situations 36 models are trained and evaluated. Because it is difficult to asses whether a regression model

performs better than random, Table 6.4 gives the relevant statistics that can be used to compare the MSE and MAE of each model.

In the first scheme each model is tested by withholding ten arbitrary combinations of bacteria for testing. This is repeated 26 times to obtain a good estimation of the performance statistics. This setting mimics the case when one wants to build a good model but lack the time or resources to perform measurements for all the possible combinations. We expect that this setting is the easiest to learn as in general every testing instance exists out of a methanotroph and a heterotroph which both have been used to train the model, but never together. Thus, no new objects are encountered during testing. The results are summarized in Table 6.5. In general one cannot say that a combination of features is the best for all label types and performance measures. This may indicate that different properties of the bacteria may be of more or less importance for inferring different properties of their collective system. The ranking error shows that always some pattern could be learned, as every model performed better than the baseline, also is the error always better than the corresponding variance or mean absolute deviation.

In the second scheme a model is tested by withholding one methanotroph and all its heterotrophic partners for testing when training the model. This procedure is repeated for all ten of the methanotrophic bacteria. These results are presented in Table 6.6 and can almost always considered better than random, but not by a very large margin. This seems to be a harder problem than the first, because in every testing example new objects are encountered. By means of example we present a bar plot of the residuals for the regression of the area under the growth curve with the spectrum kernel of the MMO genes and the sum of the two Tanimoto kernel of the genetic fingerprints in Figure 6.9. This illustrates nicely that certain pairs were relatively easy to model while other are probably biased due to unmodeled factors.

The third testing scheme, of which the results are given in Table 6.7, is very similar to the previous one with the difference that the heterotrophs are withheld for testing. This seems to be a slightly more easy problem to learn than withholding methanotrophs.

As the final setting we consider the case of using completely unseen combinations as testing instances. This is a very harsh condition as we ignore a large part of the data for both training and testing. To still have a dataset comparable in size with the previous setting we perform testing by withholding a pair of a heterotroph and methanotroph for testing while training the model with the remainder of the data, excluding any combination which contains any of the test-partners. By using this type of leave-one-out cross validation we can construct relevant test statistics. Note that this comes at a cost as we can no longer construct the ranking error. These results are presented in Table 6.8, notice

Table 6.4: Variance and mean absolute deviation of the three different labels. A model with some predictive power has a mean squared error lower than the variance or a mean absolute error lower than the mean absolute deviation.

Label type	Variance	Mean absolute deviation
max OD	1.495875	0.9539107
time max OD	11.06718	2.892308
area under curve	0.3213721	0.4234429

Table 6.5: Results of the regression of the microbial data. Performance is calculated by withholding ten random chosen combinations for testing and repeating this procedure 26 times. The best performance for every combination of regression label and performance measure is put in boldface. Compare the performance with the references in Table 6.4.

Meth. kernel	Hetero. kernel	Label type	MSE	MAE	RE
Spectrum kernel	Carbohydrate Tanimoto	max OD	0.7012	0.6034	0.1885
Jukes-Cantor	Carbohydrate Tanimoto	max OD	0.6878	0.5963	**0.1697**
Spectrum kernel	Carbohydrate linear	max OD	0.719	0.6226	0.1765
Jukes-Cantor	Carbohydrate linear	max OD	0.7542	0.6255	0.1867
Spectrum kernel	Sec. met. Tanimoto	max OD	0.6856	0.5784	0.1937
Jukes-Cantor	Sec. met. Tanimoto	max OD	0.7375	0.6071	0.1998
Spectrum kernel	Sec. met. linear	max OD	0.9661	0.7358	0.2204
Jukes-Cantor	Sec. met. linear	max OD	0.7095	0.5884	0.192
Spectrum kernel	Both Tanimoto	max OD	**0.6527**	**0.5745**	0.1893
Jukes-Cantor	Both Tanimoto	max OD	0.6852	0.5866	0.1964
Spectrum kernel	Both linear	max OD	0.7684	0.6391	0.1938
Jukes-Cantor	Both linear	max OD	0.7128	0.5953	0.1851
Spectrum kernel	Carbohydrate Tanimoto	time max OD	**7.1099**	**1.873**	0.2195
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	7.3392	1.9723	0.2305
Spectrum kernel	Carbohydrate linear	time max OD	9.5245	2.2907	0.2709
Jukes-Cantor	Carbohydrate linear	time max OD	7.7664	1.9998	0.2283
Spectrum kernel	Sec. met. Tanimoto	time max OD	7.4006	1.8899	0.233
Jukes-Cantor	Sec. met. Tanimoto	time max OD	7.4942	2.0278	0.2424
Spectrum kernel	Sec. met. linear	time max OD	10.8084	2.4715	0.2926
Jukes-Cantor	Sec. met. linear	time max OD	7.5891	1.9337	0.2269
Spectrum kernel	Both Tanimoto	time max OD	7.2858	1.89	**0.219**
Jukes-Cantor	Both Tanimoto	time max OD	7.2528	1.9344	0.2259
Spectrum kernel	Both linear	time max OD	9.2916	2.266	0.2536
Jukes-Cantor	Both linear	time max OD	7.6274	1.9659	0.2307
Spectrum kernel	Carbohydrate Tanimoto	area under curve	0.1305	0.2485	0.2034
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.1354	0.25	0.1974
Spectrum kernel	Carbohydrate linear	area under curve	0.1321	0.2543	0.1966
Jukes-Cantor	Carbohydrate linear	area under curve	0.1341	0.2573	0.1889
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.1332	0.2405	0.2
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.1335	**0.2341**	0.1812
Spectrum kernel	Sec. met. linear	area under curve	0.1724	0.2803	0.2128
Jukes-Cantor	Sec. met. linear	area under curve	0.1408	0.2443	0.1829
Spectrum kernel	Both Tanimoto	area under curve	**0.128**	0.2402	0.194
Jukes-Cantor	Both Tanimoto	area under curve	0.1323	0.2359	**0.1752**
Spectrum kernel	Both linear	area under curve	0.1409	0.2583	0.2034
Jukes-Cantor	Both linear	area under curve	0.1459	0.2588	0.1932

that the model is no better than the baseline predicted for the maximum OD label. As expected this seems to be the most difficult problem to learn.

Table 6.6: Results of the regression of the microbial data. Performance is calculated by withholding one methanotroph with all its heterotrophic partners for testing and repeating this procedure for every methanotroph. The best performance for every combination of regression label and performance measure is put in boldface. Compare the performance with the references in Table 6.4.

Meth. kernel	Hetero. kernel	Label type	MSE	MAE	RE
Spectrum kernel	Carbohydrate Tanimoto	max OD	1.2567	0.8277	0.4062
Jukes-Cantor	Carbohydrate Tanimoto	max OD	1.5448	0.9279	0.419
Spectrum kernel	Carbohydrate linear	max OD	**1.2314**	**0.8243**	0.4053
Jukes-Cantor	Carbohydrate linear	max OD	1.5678	0.9292	**0.403**
Spectrum kernel	Sec. met. Tanimoto	max OD	1.2563	0.8232	0.419
Jukes-Cantor	Sec. met. Tanimoto	max OD	1.5207	0.915	0.433
Spectrum kernel	Sec. met. linear	max OD	1.2723	0.8353	0.409
Jukes-Cantor	Sec. met. linear	max OD	1.5544	0.9198	0.4148
Spectrum kernel	Both Tanimoto	max OD	1.2613	0.8253	0.4055
Jukes-Cantor	Both Tanimoto	max OD	1.5302	0.9161	0.4198
Spectrum kernel	Both linear	max OD	1.241	0.826	0.4046
Jukes-Cantor	Both linear	max OD	1.5568	0.9226	0.4098
Spectrum kernel	Carbohydrate Tanimoto	time max OD	8.8919	**2.3212**	**0.3991**
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	9.078	2.3902	0.4243
Spectrum kernel	Carbohydrate linear	time max OD	9.9154	2.4898	0.414
Jukes-Cantor	Carbohydrate linear	time max OD	8.9267	2.3682	0.421
Spectrum kernel	Sec. met. Tanimoto	time max OD	8.8657	2.3202	0.4065
Jukes-Cantor	Sec. met. Tanimoto	time max OD	9.1077	2.4356	0.4591
Spectrum kernel	Sec. met. linear	time max OD	9.9825	2.5005	0.4137
Jukes-Cantor	Sec. met. linear	time max OD	**8.7689**	2.3296	0.4154
Spectrum kernel	Both Tanimoto	time max OD	8.9454	2.3326	0.4113
Jukes-Cantor	Both Tanimoto	time max OD	9.1532	2.4123	0.4337
Spectrum kernel	Both linear	time max OD	9.9454	2.4922	0.4139
Jukes-Cantor	Both linear	time max OD	8.868	2.3552	0.4199
Spectrum kernel	Carbohydrate Tanimoto	area under curve	0.2379	0.3431	0.3917
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.2635	0.3553	0.4018
Spectrum kernel	Carbohydrate linear	area under curve	0.2344	0.3425	0.3877
Jukes-Cantor	Carbohydrate linear	area under curve	0.2796	0.3675	0.4022
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.2345	0.3362	0.3969
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.2572	0.3503	0.4025
Spectrum kernel	Sec. met. linear	area under curve	0.2389	0.3438	0.3902
Jukes-Cantor	Sec. met. linear	area under curve	0.2653	0.3556	0.3982
Spectrum kernel	Both Tanimoto	area under curve	**0.2343**	**0.3387**	0.3898
Jukes-Cantor	Both Tanimoto	area under curve	0.2721	0.3599	0.4046
Spectrum kernel	Both linear	area under curve	0.2372	0.3433	**0.3871**
Jukes-Cantor	Both linear	area under curve	0.2726	0.3638	0.4052

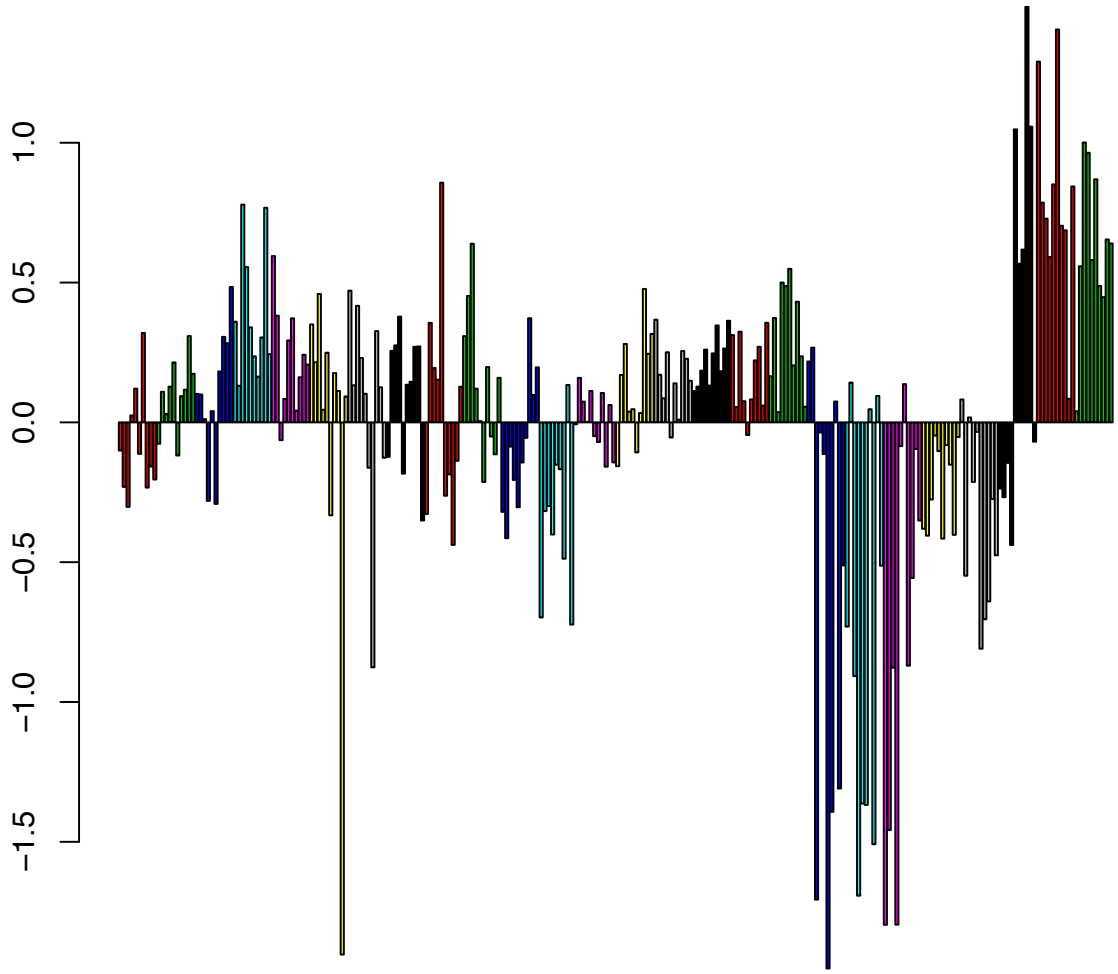


Figure 6.9: Bar plots of the residuals of the regression of the area under the growth curve with the spectrum kernel of the MMO genes and the sum of the two Tanimoto kernel of the genetic fingerprints. The different methanotrophs, and consequentially in this case the different folds for testing, are depicted in different colors.

6.3.2 Conditional ranking

Due to the relatively small number of methanotrophs in the dataset, these bacteria will always be used as queries. Note that it is also perfectly reasonable to consider the heterotrophs as queries, but the low number of methanotrophs that could serve as database objects does not allow a adequate testing procedure, in the author's opinion. Also, for the heterotrophs, the linear kernels will not be considered to reduce some of the complexity of our results. The four schemes for model evaluation will be used for the ranking experiments as well. Only the ranking error will be considered as a performance measure and because sampling is used, we will also give the standard deviation of this statistic.

The first test setting considers the case of evaluating the model when both the query and the database objects are encountered in the training phase. Thus for every methanotroph ten of its heterotrophic partners are randomly sampled and withheld as a testing database

Table 6.7: Results of the regression of the microbial data. Performance is calculated by withholding one heterotroph with all its methanotrophic partners for testing and repeating this procedure for every heterotroph. The best performance for every combination of regression label and performance measure is put in boldface. Compare the performance with the references in Table 6.4.

Meth. kernel	Hetero. kernel	Label type	MSE	MAE	RE
Spectrum kernel	Carbohydrate Tanimoto	max OD	1.2889	0.7643	**0.1582**
Jukes-Cantor	Carbohydrate Tanimoto	max OD	1.2912	0.749	0.1668
Spectrum kernel	Carbohydrate linear	max OD	1.3117	0.7827	0.1625
Jukes-Cantor	Carbohydrate linear	max OD	1.3193	0.7564	**0.1582**
Spectrum kernel	Sec. met. Tanimoto	max OD	1.5956	0.8612	0.1736
Jukes-Cantor	Sec. met. Tanimoto	max OD	1.4772	0.8263	0.1727
Spectrum kernel	Sec. met. linear	max OD	2.1404	1.0338	0.1984
Jukes-Cantor	Sec. met. linear	max OD	1.7443	0.8999	0.1753
Spectrum kernel	Both Tanimoto	max OD	1.3077	0.7608	0.1642
Jukes-Cantor	Both Tanimoto	max OD	**1.282**	**0.741**	0.1693
Spectrum kernel	Both linear	max OD	1.4815	0.8281	0.1719
Jukes-Cantor	Both linear	max OD	1.3886	0.7726	0.1702
Spectrum kernel	Carbohydrate Tanimoto	time max OD	**6.7203**	**1.8153**	0.1829
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	6.8361	1.8824	0.1899
Spectrum kernel	Carbohydrate linear	time max OD	7.8597	2.051	0.2301
Jukes-Cantor	Carbohydrate linear	time max OD	7.0486	1.8534	**0.187**
Spectrum kernel	Sec. met. Tanimoto	time max OD	7.3339	1.9203	0.1891
Jukes-Cantor	Sec. met. Tanimoto	time max OD	7.45	1.9853	0.1941
Spectrum kernel	Sec. met. linear	time max OD	9.0941	2.1854	0.247
Jukes-Cantor	Sec. met. linear	time max OD	7.3186	1.8874	0.1982
Spectrum kernel	Both Tanimoto	time max OD	6.8813	1.8435	0.1921
Jukes-Cantor	Both Tanimoto	time max OD	7.003	1.9035	0.1911
Spectrum kernel	Both linear	time max OD	8.0348	2.0708	0.2408
Jukes-Cantor	Both linear	time max OD	7.2044	1.8699	0.1881
Spectrum kernel	Carbohydrate Tanimoto	area under curve	0.1688	0.2495	**0.1359**
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	**0.1621**	**0.2435**	0.1479
Spectrum kernel	Carbohydrate linear	area under curve	0.1639	0.2557	0.1462
Jukes-Cantor	Carbohydrate linear	area under curve	0.1625	0.2496	0.153
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.193	0.269	0.153
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.1828	0.2596	0.1487
Spectrum kernel	Sec. met. linear	area under curve	0.2718	0.3136	0.159
Jukes-Cantor	Sec. met. linear	area under curve	0.221	0.2854	0.1556
Spectrum kernel	Both Tanimoto	area under curve	0.1638	0.2466	0.147
Jukes-Cantor	Both Tanimoto	area under curve	0.1689	0.2473	0.1487
Spectrum kernel	Both linear	area under curve	0.1835	0.2655	0.1453
Jukes-Cantor	Both linear	area under curve	0.181	0.2615	0.1564

while the remainder of the data is used for training. This sampling of the database is repeated five times for each query (methanotroph), providing 5×10 (the number of

Table 6.8: Results of the regression of the microbial data. Performance is calculated by withholding one combination of heterotrophic and methanotrophic bacteria for testing while training the model with the remainder of the data minus all the combinations that contain one of the bacteria used for testing. This procedure is repeated for every combination. The best performance for every combination of regression label and performance measure is put in boldface. Compare the performance with the references in Table 6.4.

Meth. kernel	Hetero. kernel	Label type	MSE	MAE
Spectrum kernel	Carbohydrate Tanimoto	max OD	1.9127	1.0202
Jukes-Cantor	Carbohydrate Tanimoto	max OD	2.1718	1.0969
Spectrum kernel	Carbohydrate linear	max OD	**1.8756**	**1.0177**
Jukes-Cantor	Carbohydrate linear	max OD	2.1539	1.0926
Spectrum kernel	Sec. met. Tanimoto	max OD	2.185	1.0707
Jukes-Cantor	Sec. met. Tanimoto	max OD	2.3317	1.1138
Spectrum kernel	Sec. met. linear	max OD	2.5633	1.1877
Jukes-Cantor	Sec. met. linear	max OD	2.6312	1.1942
Spectrum kernel	Both Tanimoto	max OD	1.9458	1.0103
Jukes-Cantor	Both Tanimoto	max OD	2.187	1.0805
Spectrum kernel	Both linear	max OD	2.0298	1.0449
Jukes-Cantor	Both linear	max OD	2.2799	1.0989
Spectrum kernel	Carbohydrate Tanimoto	time max OD	**8.4652**	**2.2797**
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	8.6588	2.3633
Spectrum kernel	Carbohydrate linear	time max OD	8.8055	2.3783
Jukes-Cantor	Carbohydrate linear	time max OD	8.6337	2.3305
Spectrum kernel	Sec. met. Tanimoto	time max OD	8.8074	2.3347
Jukes-Cantor	Sec. met. Tanimoto	time max OD	9.002	2.4224
Spectrum kernel	Sec. met. linear	time max OD	9.5516	2.4617
Jukes-Cantor	Sec. met. linear	time max OD	8.5549	2.3155
Spectrum kernel	Both Tanimoto	time max OD	8.5814	2.2976
Jukes-Cantor	Both Tanimoto	time max OD	8.7022	2.373
Spectrum kernel	Both linear	time max OD	8.9614	2.3735
Jukes-Cantor	Both linear	time max OD	8.505	2.3159
Spectrum kernel	Carbohydrate Tanimoto	area under curve	0.2825	0.3752
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.305	0.3902
Spectrum kernel	Carbohydrate linear	area under curve	0.2762	0.3782
Jukes-Cantor	Carbohydrate linear	area under curve	0.3061	0.393
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.3043	0.3918
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.3191	0.3975
Spectrum kernel	Sec. met. linear	area under curve	0.3426	0.4212
Jukes-Cantor	Sec. met. linear	area under curve	0.3534	0.4232
Spectrum kernel	Both Tanimoto	area under curve	**0.2809**	**0.3744**
Jukes-Cantor	Both Tanimoto	area under curve	0.3037	0.3865
Spectrum kernel	Both linear	area under curve	0.2867	0.3836
Jukes-Cantor	Both linear	area under curve	0.3179	0.3988

methanotrophs) samples used for calculating the performance. The means and standard deviations of the ranking error for each model are presented in Table 6.9. Even though this is expected to be the most easy case we consider, the ranking errors are quite bad, even compared to what we became for regression in Table 6.5! It should also be noted that the standard deviations are always quite high. Nearly always it seems that it is easiest to rank according to the area under the growth curve, perhaps because this is a global metric.

When we withheld every query methanotroph for testing we obtained the results of Table 6.10. This case seems to be somewhat harder to learn than the previous one. Again, ranking the area under the curve always performed best for all combinations of features. Note the lower standard deviation.

To test how the models perform on new heterotrophs as database objects, we obtain Table 6.11. In this case ten randomly selected heterotrophs were withheld for all combinations with the methanotrophs. For testing, all the methanotrophs were used as query for this list of heterotrophs. This was always repeated 50 times to obtain a relevant statistic. Note that we did not do this for the case with the time of maximal OD due to the discrete nature of this label which conflicted with our sampling methods. A ranking error cannot be calculated when all the labels are equal. This problem seems to be harder than the previous one, but also has a reasonably low standard deviation.

Table 6.9: Results of the ranking of the microbial data. The methanotrophs were used as queries, while their heterotrophic partners served as the database elements. In this testing scheme for every methanotroph, ten of its partners were withhold for testing and this was repeated five times for every model. The best ranking error is put in bold face while the worst is in italic.

Meth. kernel	Hetero. kernel	Label type	Ranking error
Spectrum kernel	Carbohydrate Tanimoto	area under curve	**0.3504** (0.1169)
Spectrum kernel	Carbohydrate Tanimoto	time max OD	0.4491 (0.1812)
Spectrum kernel	Carbohydrate Tanimoto	max OD	0.375 (0.1384)
Spectrum kernel	Both Tanimoto	area under curve	0.3702 (0.1007)
Spectrum kernel	Both Tanimoto	time max OD	0.4424 (0.1896)
Spectrum kernel	Both Tanimoto	max OD	0.3847 (0.1118)
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.396 (0.1474)
Spectrum kernel	Sec. met. Tanimoto	time max OD	*0.4767* (0.1782)
Spectrum kernel	Sec. met. Tanimoto	max OD	0.4118 (0.1286)
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.3524 (0.1236)
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	0.4553 (0.1621)
Jukes-Cantor	Carbohydrate Tanimoto	max OD	0.3798 (0.1135)
Jukes-Cantor	Both Tanimoto	area under curve	0.4038 (0.1254)
Jukes-Cantor	Both Tanimoto	time max OD	0.4193 (0.1719)
Jukes-Cantor	Both Tanimoto	max OD	0.3915 (0.1209)
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.4058 (0.1298)
Jukes-Cantor	Sec. met. Tanimoto	time max OD	0.4362 (0.1566)
Jukes-Cantor	Sec. met. Tanimoto	max OD	0.4092 (0.1129)

Table 6.10: Results of the ranking of the microbial data. The methanotrophs were used as queries, while their heterotrophic partners served as the database elements. The queries are withhold for testing. The best ranking error is put in bold face while the worst is in italic.

Meth. kernel	Hetero. kernel	Label type	Ranking error
Spectrum kernel	Carbohydrate Tanimoto	area under curve	**0.3971** (0.0553)
Spectrum kernel	Carbohydrate Tanimoto	time max OD	0.4513 (0.0862)
Spectrum kernel	Carbohydrate Tanimoto	max OD	0.4172 (0.0698)
Spectrum kernel	Both Tanimoto	area under curve	0.4002 (0.0565)
Spectrum kernel	Both Tanimoto	time max OD	0.4424 (0.0891)
Spectrum kernel	Both Tanimoto	max OD	0.4161 (0.0719)
Spectrum kernel	Sec. met. Tanimoto	area under curve	**0.3971** (0.0463)
Spectrum kernel	Sec. met. Tanimoto	time max OD	0.4436 (0.0925)
Spectrum kernel	Sec. met. Tanimoto	max OD	0.4194 (0.073)
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.4058 (0.0498)
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	*0.4613* (0.0978)
Jukes-Cantor	Carbohydrate Tanimoto	max OD	0.4214 (0.0525)
Jukes-Cantor	Both Tanimoto	area under curve	0.3995 (0.0498)
Jukes-Cantor	Both Tanimoto	time max OD	0.4593 (0.0958)
Jukes-Cantor	Both Tanimoto	max OD	0.4144 (0.0433)
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.412 (0.0543)
Jukes-Cantor	Sec. met. Tanimoto	time max OD	0.4514 (0.0757)
Jukes-Cantor	Sec. met. Tanimoto	max OD	0.4282 (0.0607)

Table 6.11: Results of the ranking of the microbial data. The methanotrophs were used as queries, while their heterotrophic partners served as the database elements. The best ranking error is put in bold face while the worst is in italic.

Meth. kernel	Hetero. kernel	Label type	Ranking error
Spectrum kernel	Carbohydrate Tanimoto	area under curve	*0.4369* (0.0548)
Spectrum kernel	Carbohydrate Tanimoto	max OD	0.4621 (0.0532)
Spectrum kernel	Both Tanimoto	area under curve	0.4539 (0.0511)
Spectrum kernel	Both Tanimoto	max OD	0.4517 (0.0447)
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.473 (0.054)
Spectrum kernel	Sec. met. Tanimoto	max OD	0.4798 (0.0603)
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.4671 (0.075)
Jukes-Cantor	Carbohydrate Tanimoto	max OD	0.4423 (0.0509)
Jukes-Cantor	Both Tanimoto	area under curve	0.4867 (0.0671)
Jukes-Cantor	Both Tanimoto	max OD	0.4622 (0.0464)
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.5057 (0.0524)
Jukes-Cantor	Sec. met. Tanimoto	max OD	*0.4779* (0.0438)

Finally we present the case of using an unseen query to rank a new database. For every methanotroph, ten heterotrophs were withheld, while using all other pairs with no common elements for training and repeating this 50 times for every query. A shown in Table 6.12, results are approximately random, our algorithms could not deal with this

problem.

In conclusion we can state that conditional ranking was possible though we apparently obtained a worse performance compared to the regression of Section 6.3.1. This can be due to multiple reasons. Firstly, for regression we used support vector machines while the framework for ranking is based on regularized least squared. Since the former is based on a more robust ϵ-regression rather than squared error, a difference in performance might be expected. Given the many very high residuals, the loss function may have an influence on the performance. Secondly, when comparing the ranking errors of regression and conditional ranking, the setting may be a bit more strict for conditional ranking. For regression we predicted the values for combinations, but we have seen on heatmaps that the methanotrophs are the most important to determine the growth[1]! This is a reasonable assumption as all the carbon uptake in the system passes the methanotrophic bacteria. However this makes it a more difficult (but interesting) problem for ranking, as we only consider the ranking of a group of heterotrophic bacteria with a methanotroph in common! Stated differently, in regression the knowledge which methanotroph you are dealing with is already a very good baseline for doing predictions, while in the case of conditional ranking you have to make a prediction relative to this bacterium. We quickly redid the regression experiments but this time using normalized data, obtained by subtracting the

[1]The heatmaps presented in this chapter are already corrected for this fact, so this cannot be deduced from figures 6.6, 6.7 and 6.8.

Table 6.12: Results of the ranking of the microbial data. The methanotrophs were used as queries, while their heterotrophic partners served as the database elements. Testing was done on completely unseen combinations. The best ranking error is put in bold face while the worst is in italic.

Meth. kernel	Hetero. kernel	Label type	Ranking error
Spectrum kernel	Carbohydrate Tanimoto	area under curve	0.5127 (0.1338)
Spectrum kernel	Carbohydrate Tanimoto	time max OD	**0.4565** (0.1854)
Spectrum kernel	Carbohydrate Tanimoto	max OD	0.5068 (0.1357)
Spectrum kernel	Both Tanimoto	area under curve	0.5253 (0.1277)
Spectrum kernel	Both Tanimoto	time max OD	0.4564 (0.1841)
Spectrum kernel	Both Tanimoto	max OD	0.5097 (0.1295)
Spectrum kernel	Sec. met. Tanimoto	area under curve	0.5326 (0.124)
Spectrum kernel	Sec. met. Tanimoto	time max OD	0.4896 (0.1673)
Spectrum kernel	Sec. met. Tanimoto	max OD	0.5212 (0.1224)
Jukes-Cantor	Carbohydrate Tanimoto	area under curve	0.5419 (0.1217)
Jukes-Cantor	Carbohydrate Tanimoto	time max OD	0.4626 (0.1752)
Jukes-Cantor	Carbohydrate Tanimoto	max OD	0.5239 (0.1359)
Jukes-Cantor	Both Tanimoto	area under curve	0.5459 (0.1215)
Jukes-Cantor	Both Tanimoto	time max OD	0.4752 (0.1886)
Jukes-Cantor	Both Tanimoto	max OD	0.5168 (0.1258)
Jukes-Cantor	Sec. met. Tanimoto	area under curve	0.5421 (0.1283)
Jukes-Cantor	Sec. met. Tanimoto	time max OD	0.4926 (0.1786)
Jukes-Cantor	Sec. met. Tanimoto	max OD	*0.543* (0.1276)

corresponding value of the methanotroph with NMS (data not shown). In general this normalization of the data, and thus only considering the effect of the heterotrophs on the growth, had a negative influence on the performance, thus supporting our hypothesis.

6.4 Conclusion

In this chapter we attempted to build a useful tool for environmental biotechnologists by using data obtained from a series of simple wet lab experiments and some easily obtained bioinformatical features.

The performed laboratory experiments provided data that was consistent within replicates and give rise to some interesting patterns, especially given the limited lab experience of the author. This formed a good data set to test some of our algorithms, but might also prove to be of some value as a benchmark set for other models from machine learning or mathematical ecology.

We showed that with regression and ranking it is possible to construct models that can roughly predict interaction of the species. The interaction is mainly dominated by the methanotroph, something to keep in mind when using these models. Whether a bacterium is previously encountered in the training phase can make a big difference in performance. The best features one has to use depend heavily on de details of the problem.

In summary we can state that we are very happy with the outcome of the experiments in this chapter. We have proven that the methods discussed in the beginning of this thesis can be of use for more complex objects than molecules. From a biological perspective we have shown that machine learning can have its place in bacterial ecology. With this proof of concept we are motivated to encourage performing larger experiments (both in number of bacteria as in the measurements) and more complex feature extractions, making full use of modern bioinformatics.

Bibliography

[1] S. Allesina and J. M. Levine. A competitive network theory of species diversity. *Proceedings of the National Academy of Sciences of the United States of America*, 108(14):5638–5642, 2011.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[3] T. Anastassiadis, S. W. Deacon, K. Devarajan, H. Ma, and J. R. Peterson. Comprehensive assay of kinase catalytic activity reveals features of kinase inhibitor selectivity. *Nature biotechnology*, 29(11):1039–45, Nov. 2011.

[4] M. Antal, C. Böde, and P. Csermely. Perturbation waves in proteins and protein networks: applications of percolation and game theories in signaling and drug design. *Current protein & peptide science*, 10(2):161–72, Apr. 2009.

[5] A. Arakaki, Y. Huang, and J. Skolnick. EFICAz2: enzyme function inference by a combined approach enhanced by machine learning. *BMC Bioinformatics*, 10:107, 2009.

[6] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–449, 1950.

[7] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[8] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko. The RAST Server: rapid annotations using subsystems technology. *BMC genomics*, 9:75, Jan. 2008.

[9] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics (Oxford, England)*, 21 Suppl 1:i38–46, June 2005.

[10] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[11] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2009.

[12] K. M. Borgwardt and H. P. Kriegel. Shortest-path kernels on graphs. In *Proc. International Conference Data Mining*, pages 74–81, Houston, Texas, 2005.

[13] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, 1998.

[14] S. Chaffron, H. Rehrauer, J. Pernthaler, and C. von Mering. A global network of coexisting microbes from environmental and whole-genome sequence data. *Genome research*, 20(7):947–59, July 2010.

[15] N. Christian, T. Handorf, and O. Ebenhöh. Metabolic synergy: increasing biosynthetic capabilities by network cooperation. *Genome informatics. International Conference on Genome Informatics*, 18:320–9, Jan. 2007.

[16] A. Dereeper, V. Guignon, G. Blanc, S. Audic, S. Buffet, F. Chevenet, J.-F. Dufayard, S. Guindon, V. Lefort, M. Lescot, J.-M. Claverie, and O. Gascuel. Phylogeny.fr: robust phylogenetic analysis for the non-specialist. *Nucleic acids research*, 36(Web Server issue):W465–9, July 2008.

[17] P. Dobson and A. Doig. Predicting enzyme class from protein structures without alignments. *Journal of Molecular Biology*, 345:187–199, 2005.

[18] W. Duch, K. Swaminathan, and J. Meller. Artificial intelligence approaches for rational drug design and discovery. *Current Pharmaceutical Design*, 13(14):1497–1508, May 2007.

[19] D. Dunaway-Mariano. Enzyme function discovery. *Structure*, 16(11):1599–1600, 2008.

[20] S. Eddy. Multiple alignment using hidden Markov models. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 114–120, 1995.

[21] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–7, Jan. 2004.

[22] V. Egelhofer, I. Schomburg, and D. Schomburg. Automatic assignment of EC numbers. *PLoS Computational Biology*, 6:1000661, 2010.

[23] S. Erdin, A. M. Lisewski, and O. Lichtarge. Protein function prediction: towards integration of similarity metrics. *Current Opinion in Structural Biology*, 21(2):180–8, Apr. 2011.

[24] E. Eskin and S. Snir. The homology kernel: a biologically motivated sequence embedding into Euclidean space. 2004.

[25] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. *Machine Learning*, pages 1–38, 2004.

[26] T. Fober, S. Glinca, G. Klebe, and E. Hüllermeier. Superposition and alignment of labeled point clouds. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(6):1653–66, 2011.

[27] T. Fober, M. Mernberger, R. Moritz, and E. Hüllermeier. Graph-kernels for the comparative analysis of protein active sites. In *German Conference on Bioinformatics*, pages 21–31, Halle (Saale), Germany, 2009.

[28] J. Fürnkranz, E. Hüllermeier, and S. Vanderlooy. Binary decomposition methods for multipartite ranking. *Lecture Notes in Computer Science*, 5781:359–374, 2009.

[29] P. Geurts, N. Touleimat, M. Dutreix, and F. D'Alché-Buc. Inferring biological networks with output kernel trees. *BMC bioinformatics*, 8 Suppl 2:S4, Jan. 2007.

[30] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002.

[31] M. Gribskov, A. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 84(13):4355–4358, 1990.

[32] R. Hanson. Methanotrophic bacteria. *Microbiological reviews*, 60(2):439–471, 1996.

[33] M. Hasegawa, H. Kishino, and N. Saitou. On the maximum likelihood method in molecular phylogenetics. *Journal of molecular evolution*, 32(5):443–5, May 1991.

[34] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, 2003.

[35] J. Helm, K.-D. Wendlandt, G. Rogge, and U. Kappelmeyer. Characterizing a stable methane-utilizing mixed culture used in the synthesis of a high-quality biopolymer in an open system. *Journal of applied microbiology*, 101(2):387–95, Aug. 2006.

[36] M. Hendlich, F. Rippmann, and G. Barnickel. LIGSITE: Automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15:359–363, 1997.

[37] B. Hoffmann, M. Zaslavskiy, J.-P. P. Vert, and V. Stoven. A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3D: application to ligand prediction. *BMC bioinformatics*, 11(1):99+, 2010.

[38] D. Hršak and A. Begonja. Possible interactions within a methanotrophic-heterotrophic groundwater community able to transform linear alkylbenzenesulfonates. *Applied and environmental microbiology*, 66(10):4433–4439, 2000.

[39] G. E. Hutchinson. The paradox of the plankton. *The American Naturalist*, 95(882):137–145, 1961.

[40] H. Iguchi, H. Yurimoto, and Y. Sakai. Stimulation of methanotrophic growth in cocultures by cobalamin excreted by rhizobia. *Applied and environmental microbiology*, 77(24):8509–15, Dec. 2011.

[41] L. Jacob and J.-P. Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics (Oxford, England)*, 24(19):2149–56, Oct. 2008.

[42] W. Jorgensen. The many roles of computation in drug discovery. *Science*, 303(5665):1813–1818, 2004.

[43] P. Kambam, M. Henson, and L. Sun. Design and mathematical modelling of a synthetic symbiotic ecosystem. *Systems Biology, IET*, 2(1):33–38, 2008.

[44] M. Kanehisa. The KEGG database. *Novartis Found Symp*, 247, 2002.

[45] M. W. Karaman, S. Herrgard, D. K. Treiber, P. Gallant, C. E. Atteridge, B. T. Campbell, K. W. Chan, P. Ciceri, M. I. Davis, P. T. Edeen, R. Faraoni, M. Floyd, J. P. Hunt, D. J. Lockhart, Z. V. Milanov, M. J. Morrison, G. Pallares, H. K. Patel, S. Pritchard, L. M. Wodicka, and P. P. Zarrinkar. A quantitative analysis of kinase inhibitor selectivity. *Nature biotechnology*, 26(1):127–32, Jan. 2008.

[46] G. Károlyi, Z. Neufeld, and I. Scheuring. Rock-scissors-paper game in a chaotic flow: the effect of dispersion on the cyclic competition of microorganisms. *Journal of theoretical biology*, 236(1):12–20, Sept. 2005.

[47] I. M. Keseler, C. Bonavides-Martinez, J. Collado-Vides, S. Gama-Castro, R. P. Gunsalus, D. A. Johnson, Krummenacker M., L. M. Nolan, S. Paley, I. T. Paulsen, M. Peralta-Gil, A. Santos-Zavaleta, A. G. Shearer, and P. D. Karp. EcoCyc: a comprehensive view of Escherichia coli biology. *Nucleic Acids Research*, 37:D464–D470, Jan. 2009.

[48] B. C. Kirkup and M. a. Riley. Antibiotic-mediated antagonism leads to a bacterial game of rock-paper-scissors in vivo. *Nature*, 428(6981):412–4, Mar. 2004.

[49] N. Klitgord and D. Segrè. Environments that induce synthetic microbial ecosystems. *PLoS computational biology*, 6(11), 2010.

[50] N. Klitgord and D. Segrè. Ecosystems biology of microbial metabolism. *Current opinion in biotechnology*, 22(4):541–6, Aug. 2011.

[51] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322. Morgan Kaufmann, 2002.

[52] M. Kotera, Y. Ukura, M. Hattori, and S. Goto. Computational assignment of the EC numbers for genome-scale analysis of enzymatic reactions. *Journal of the American Chemical Society*, 126:1648716498, 2004.

[53] P. Larranaga. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112, Feb. 2006.

[54] R. a. Laskowski, N. M. Luscombe, M. B. Swindells, and J. M. Thornton. Protein clefts in molecular recognition and function. *Protein science : a publication of the Protein Society*, 5(12):2438–52, Dec. 1996.

[55] R. A. Laskowski, N. M. Luscombe, M. B. Swindells, and J. M. Thornton. Protein clefts in molecular recognition and function. *Protein Science*, 5(12):2438–2452, 1996.

[56] S. Lèbre, J. Becq, F. Devaux, M. P. H. Stumpf, and G. Lelandais. Statistical inference of the time-varying structure of gene-regulation networks. *BMC systems biology*, 4:130, Jan. 2010.

[57] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 575:564–75, Jan. 2002.

[58] P. Mahé, N. Ueda, T. Akutsu, J. L. Perret, and J. P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J Chem Inf Model*, 45(4):939–951, 2005.

[59] G. Manning, D. B. Whyte, R. Martinez, T. Hunter, and S. Sudarsanam. The protein kinase complement of the human genome. *Science (New York, N.Y.)*, 298(5600):1912–34, Dec. 2002.

[60] T. J. Marrone, J. M. Briggs, and, and J. A. McCammon. STRUCTURE-BASED DRUG DESIGN: Computational Advances. *Annual Review of Pharmacology and Toxicology*, 37(1):71–90, Apr. 1997.

[61] a. C. Martin, C. a. Orengo, E. G. Hutchinson, S. Jones, M. Karmirantzou, R. a. Laskowski, J. B. Mitchell, C. Taroni, and J. M. Thornton. Protein folds and functions. *Structure*, 6(7):875–84, July 1998.

[62] O. Modin, K. Fukushi, and K. Yamamoto. Denitrification with methane as external carbon source. *Water research*, 41(12):2726–38, June 2007.

[63] J. Morton and K. Hayes. Effect of copper speciation on whole-cell soluble methane monooxygenase activity in Methylosinus trichosporium OB3b. *Applied and environmental*, 66(4):1730–1733, 2000.

[64] J. Murase and P. Frenzel. A methanedriven microbial food web in a wetland rice soil. *Environmental microbiology*, 9(December):3025–3034, 2007.

[65] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–53, Mar. 1970.

[66] C. Notredame. Recent progresses in multiple sequence aligment: a survey. *Pharmacogenomics*, pages 131–144, 2002.

[67] M. Osadchy and R. Kolodny. Maps of protein structure space reveal a fundamental relationship between protein structure and function. *Proceedings of the National Academy of Sciences of the United States of America*, 108(30):12301–12306, 2011.

[68] R. Overbeek, T. Begley, R. M. Butler, J. V. Choudhuri, H.-Y. Chuang, M. Cohoon, V. de Crécy-Lagard, N. Diaz, T. Disz, R. Edwards, M. Fonstein, E. D. Frank, S. Gerdes, E. M. Glass, A. Goesmann, A. Hanson, D. Iwata-Reuyl, R. Jensen, N. Jamshidi, L. Krause, M. Kubal, N. Larsen, B. Linke, A. C. McHardy, F. Meyer, H. Neuweger, G. Olsen, R. Olson, A. Osterman, V. Portnoy, G. D. Pusch, D. a. Rodionov, C. Rückert, J. Steiner, R. Stevens, I. Thiele, O. Vassieva, Y. Ye, O. Zagnitko, and V. Vonstein. The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic acids research*, 33(17):5691–702, Jan. 2005.

[69] M. Öztürk, A. Tsoukiàs, and P. Vincke. Preference modelling. In *State of the Art in Multiple Criteria Decision Analysis*, pages 169–189. 2005.

[70] T. Pahikkala, J. Boberg, and T. Salakoski. Fast n-fold cross-validation for regularized least-squares. In *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence*, pages 83–90, 2006.

[71] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Järvinen, and J. Boberg. An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1):129–165, 2009.

[72] T. Pahikkala, W. Waegeman, A. Airola, T. Salakoski, and B. D. Baets. Conditional ranking on relational data. In *Proceedings of the European Conference on Machine Learning, ser. Lecture Notes in Computer Science*, pages 499–514, 2010.

[73] T. Pahikkala, W. Waegeman, E. Tsivtsivadze, T. Salakoski, and B. De Baets. Learning intransitive reciprocal relations with kernel methods. *European Journal of Operational Research*, 206(3):676–685, Nov. 2010.

[74] F. Pazos and A. Valencia. In silico twohybrid system for the selection of physically interacting protein pairs. *PROTEINS: Structure, Function, and Genetics*, 47(2):219–227, 2002.

[75] S. Pérot, O. Sperandio, M. A. Miteva, A.-C. Camproux, and B. O. Villoutreix. Druggable pockets and binding site centric chemical space: a paradigm shift in drug discovery. *Drug Discov. Today*, 15(15-16):656–667, 2010.

[76] Q. Qiu, R. Conrad, and Y. Lu. Cross-feeding of methane carbon among bacteria on rice roots revealed by DNA-stable isotope probing. *Environmental Microbiology Reports*, 1(5):355–361, 2009.

[77] K. Rabaey and W. Verstraete. Microbial fuel cells: novel biotechnology for energy generation. *Trends in biotechnology*, 23(6):291–8, June 2005.

[78] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural networks : the official journal of the International Neural Network Society*, 18(8):1093–110, Oct. 2005.

[79] A. K. Ramani and E. M. Marcotte. Exploiting the co-evolution of interacting proteins to discover interaction specificity. *Journal of Molecular Biology*, 327(1):273–284, Mar. 2003.

[80] T. Reichenbach, M. Mobilia, and E. Frey. Mobility promotes and jeopardizes biodiversity in rock-paper-scissors games. *Nature*, 448(7157):1046–9, Aug. 2007.

[81] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

[82] M. H. Saier, M. R. Yen, K. Noto, D. G. Tamang, and C. Elkan. The transporter classification database: recent advances. *Nucleic Acids Research*, 37:D274–D278, Nov. 2008.

[83] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–25, July 1987.

[84] Y. Sakiyama. The use of machine learning and nonlinear statistical tools for ADME prediction. *Expert opinion on drug metabolism & toxicology*, 5(2):149–169, 2009.

[85] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, 9(1):56–68, 1991.

[86] S. Schmitt, D. Kuhn, and G. Klebe. A New Method to Detect Related Function Among Proteins Independent of Sequence and Fold Homology. *Journal of Molecular Biology*, 323(2):387–406, Oct. 2002.

[87] B. Schölkopf and A. J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002.

[88] B. Schölkopf, K. Tsuda, and J.-p. Vert. *Kernel Methods in Computational Biology*. 2004.

[89] D. B. Searls. Pharmacophylogenomics: genes, evolution and drug targets. *Nature reviews. Drug discovery*, 2(8):613–23, Aug. 2003.

[90] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, pages 1–9, 2001.

[91] T. F. Smith and M. S. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[92] S. J. Swamidass, C.-A. Azencott, K. Daily, and P. Baldi. A CROC stronger than ROC: measuring, visualizing and optimizing early retrieval. *Bioinformatics*, 26(10):1348–56, May 2010.

[93] S. J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, and P. Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21 Suppl 1(2):i359–68, June 2005.

[94] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Major components of the gravity recommendation system. *ACM SIGKDD Explorations Newsletter*, 9(2):80, Dec. 2007.

[95] Tamás L. Czárán, R. Hoekstra, and L. Pagie. Chemical warfare between microbes promotes biodiversity. *Proceedings of the National Academy of Sciences of the United States of America*, 99(2):786–790, 2002.

[96] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems*. 2004.

[97] J. Thornton, A. Todd, D. Milburn, N. Borkakoti, and C. Orengo. From structure to function: Approaches and limitations. *Nature Structural Biology*, pages 991–994, 2000.

[98] C. C. To and J. Vohradsky. Supervised inference of gene-regulatory networks. *BMC bioinformatics*, 9:2, Jan. 2008.

[99] A. E. Todd, C. A. Orengo, and J. M. Thornton. Evolution of function in protein superfamilies, from a structural perspective. *Journal of molecular biology*, 307(4):1113–43, Apr. 2001.

[100] K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics (Oxford, England)*, 20 Suppl 1:i326–33, Aug. 2004.

[101] S. Varma and R. Simon. Bias in Error Estimation when Using Cross-Validation for Model Selection. *Bioinformatics*, 7:91, 2006.

[102] W. Verstraete. Microbial ecology and environmental biotechnology. *The ISME journal*, 1(1):4–8, May 2007.

[103] J.-P. Vert. A tree kernel to analyse phylogenetic profiles. *Bioinformatics (Oxford, England)*, 18 Suppl 1:S276–84, Jan. 2002.

[104] J.-P. Vert, J. Qiu, and W. S. Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC bioinformatics*, 8 Suppl 10:S8, Jan. 2007.

[105] S. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph Kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[106] W. Waegeman, B. De Baets, and L. Boullart. Learning layered ranking functions with structured support vector machines. *Neural Networks*, 21(10):1511–1523, 2008.

[107] W. Waegeman, B. De Baets, and L. Boullart. ROC analysis in ordinal regression learning. *Pattern Recognition Letters*, 29:1–9, 2008.

[108] W. Waegeman, B. De Baets, and L. Boullart. Kernel-based learning methods for preference aggregation. *4OR*, 7:169–189, 2009.

[109] W. Waegeman, T. Pahikkala, A. Airola, T. Salakoski, and B. De Baets. Learning valued relations from data. In P. Melo-Pinto, P. Couto, C. Serôdio, J. Fodor, and B. De Baets, editors, *Eurofuse 2011*, volume 107 of *Advances in Intelligent and Soft Computing*, pages 257–268. Springer, 2012.

[110] A. Weber, A. Casini, A. Heine, D. Kuhn, C. T. Supuran, A. Scozzafava, and G. Klebe. Unexpected Nanomolar Inhibition of Carbonic Anhydrase by COX-2-selective Celecoxib: New Pharmacological Opportunities due to Related Binding Site Recognition. *Journal of Medical Chemistry*, 47(3):550–557, 2004.

[111] M. Weisel, J. M. Kriegl, and G. Schneider. Architectural repertoire of ligand-binding pockets on protein surfaces. *Chembiochem*, 11(4):556–563, 2010.

[112] J. Weston, A. Eliseeff, D. Zhou, C. Leslie, and W. S. Noble. Protein ranking: from local to global structure in the protein similarity network. *Proceedings of the National Academy of Science*, 101:6559–6563, 2004.

[113] Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics (Oxford, England)*, 20 Suppl 1:i363–70, Aug. 2004.

[114] Y. Yang, N. Bansal, W. Dakka, P. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 34–43, 2009.

[115] E. Yilmaz and S. Robertson. On the choice of effectiveness measures for learning to rank. *Information Retrieval*, 13(3):271–290, Sept. 2009.

[116] M. Zhu. Recall, precision and average precision. pages 1–11, 2004.