Interpretable Machine Learning Approaches in Computational Biology

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktor der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Dipl.-Bioinf. Sebastian Briesemeister
aus Ilmenau

Tübingen
2011

Tag der mündlichen Qualifikation:	29.02.2012
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr.-Ing. Oliver Kohlbacher
2. Berichterstatter:	Prof. Dr. Jörg Rahnenführer

Abstract

Machine learning has become an essential tool for analyzing, predicting, and understanding biological properties and processes. Machine learning models can substantially support the work of biologists by reducing the number of expensive and time-consuming experiments. They are able to uncover novel properties of biological systems and can be used to guide experiments. Machine learning models have been successfully applied to various tasks ranging from gene prediction to three-dimensional structure prediction of proteins. However, due to their lack of interpretability, many biologists put only little trust in the predictions made by computational models.

In this thesis, we show how to overcome the typical "black box" character of machine learning algorithms by presenting two novel interpretable approaches for classification and regression.

In the first part, we introduce YLoc, an interpretable classification approach for predicting the subcellular localization of proteins. YLoc is able to explain why a prediction was made by identifying the biological properties with the strongest influence on the prediction. We show that interpretable predictions made by YLoc help to understand a protein's localization and, moreover, can assist biologists in engineering the location of proteins. Furthermore, YLoc returns confidence scores, making it possible for biologists to define their level of trust in individual predictions.

In the second part, we show how our two novel confidence estimators, CONFINE and CONFIVE, can improve the interpretability of MHC-I–peptide binding prediction. In contrast to plain affinity values predicted by usual regression models, CONFINE and CONFIVE estimate affinity intervals, which provide a very natural interpretation of confidence. While low confidence predictions exhibit fairly large intervals, reliable predictions yield a very small range of affinities. We show that distinguishing between

reliable and unreliable predictions is important for discovering and engineering reliable epitopes for vaccines.

The interpretable approaches presented in this thesis are a significant step forward towards making machine learning methods more transparent to the users and, thus, towards improving the acceptance of computational methods.

Zusammenfassung

Maschinelles Lernen ist zu einem unverzichtbaren Werkzeug für die Analyse, Vorhersage und für das Verständnis biologischer Merkmale und Prozesse geworden. Als Alternative zu Experimenten im Labor, die oft teuer und zeitintensiv sind, können maschinelle Lernmodelle die Arbeit von Biologen erheblich erleichtern. So können beispielsweise neue Merkmale biologischer Systeme sowie Ansatzpunkte für Experimente gefunden werden. Maschinelles Lernen wurde erfolgreich für verschiedene Aufgaben, von der Genvorhersage bis hin zur Vorhersage der dreidimensionalen Struktur von Proteinen, eingesetzt. Aufgrund der schlechten Interpretierbarkeit von Vorhersagen computergestützter Lernverfahren, haben Biologen jedoch oft nur wenig Vertrauen in diese. Im Rahmen dieser Dissertation entwickeln wir daher neue Ansätze um die Klassifikation und die Regressionsanalyse für biologische Problemstellungen interpretierbarer und damit nachvollziehbarer zu machen.

Im ersten Teil der Dissertation stellen wir YLoc vor, ein neues interpretierbares Klassifikationsverfahren zur Vorhersage der subzellulären Lokalisation von Proteinen. YLoc ist in der Lage Begründungen für eine gemachte Vorhersage zu geben, indem es die biologischen Merkmale mit dem größten Einfluss auf die Vorhersage identifiziert. Interpretierbare Vorhersagen von YLoc können helfen die Lokalisierung von Proteinen besser nachzuvollziehen und ferner Biologen bei der Planung von Experimenten, die Aufschluß über mögliche Änderung der Lokalisation von Proteinen geben sollen, zu unterstützen. Darüber hinaus bewertet YLoc die Zuverlässigkeit einzelner Vorhersagen, wodurch es Biologen möglich ist, das Maß an Vertrauen in Vorhersagen individuell abzuwägen.

Im zweiten Teil dieser Arbeit stellen wir CONFINE und CONFIVE vor, zwei neue Verfahren zur Konfidenzschätzung von Vorhersagen, welche die Interpretierbarkeit von MHC-I-Bindungsvorhersagen entscheidend verbessern können. Im Gegensatz

zu üblichen Regressionsmodellen, welche lediglich Affinitätswerte vorhersagen, können CONFINE und CONFIVE Affinitätsintervalle schätzen. Diese stellen eine intuitive Interpretation von Verlässlichkeit dar. Während weniger verlässliche Vorhersagen durch breite Affinitätsintervalle auffallen, weisen sichere Vorhersagen einen sehr kleinen Bereich möglicher Affinitäten auf. Wir können weiterhin zeigen, dass die Unterscheidung zwischen verlässlichen und unsicheren Vorhersagen wichtig für das Identifizieren und Verbessern von Epitopen in der Impfstoffgewinnung ist.

Die in dieser Arbeit vorgestellten interpretierbaren Vorhersagemethoden stellen einen wichtigen Schritt in der Entwicklung transparenter maschineller Lernmethoden dar und können die Akzeptanz von computergestützte Methoden maßgeblich verbessern.

Acknowledgments

First of all, I want to thank my supervisor Prof. Dr. Oliver Kohlbacher for giving me the opportunity to work on various exciting research topics. I very much appreciate his ideas, support, guidance, and motivation during all phases of this thesis. Furthermore, I would like to thank Prof. Dr. Jörg Rahnenführer for raising interesting questions, participating in fruitful discussions, and supporting my work in the last years. I am also very thankful to Prof. Dr. Klaus Harter for giving me the opportunity to work for a couple of weeks in his lab. In addition, I gratefully acknowledge financial support from LGFG Promotionsverbund "Pflanzliche Sensorhistidinkinasen" and Deutsche Forschungsgemeinschaft (SPP 1335).

I would also like to thank my other cooperation partners: Prof. Dr. Sebastian Böcker, Prof. Dr. Gunnar Klau, Dr. Quang Bao Anh Bui, Anke Truss, Prof. Dr. Hagit Shatkay, Dr. Torsten Blum, Dr. Scott Brady, Yin Lam, Dr. Christina Chaban, Dr. Katharina Caesar, and Janika Witthöf.

I am very grateful to Johannes Junker, Peter Niermann, Nico Pfeifer, and Nora Toussaint for their support and interesting discussions on machine learning. I also thank Magdalena Feldhahn, Sven Nahnsen, Nora Toussaint, and Nico Weber for enjoyable lunch times and Dr. Johannes Fischer, Peter Niermann, and Dr. Lars Nilse for relaxing jogging trips. Many thanks to my room mates Dr. Marc Sturm and Nina Fischer for providing a nice working atmosphere. Moreover, I thank all my colleagues in the group Applied Bioinformatics as well as its former members for a motivating working atmosphere and interesting conversations.

Finally, I want to thank my family for creating a firm foundation for me to stand on. Thanks also to my friends for their patience and support. Last, but definitely not least, I express my deep gratitude to Jule for her inspiration, her support, many interesting discussions, and her enduring patience.

In accordance with the standard scientific protocol, I will use the personal pronoun "we" to indicate the reader and the writer, or my scientific collaborators and myself.

Contents

Chapter 1

Introduction

"You make experiments and I make theories. Do you know the difference? A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it."
– Albert Einstein (personal communication to Herman F. Mark, quoted by Holton [84])

There has always been a controversy between experimental and theoretical science. While experimental scientists work under real world conditions and learn from observation, theoretical scientists study concepts and models based on assumptions that might not have been proven by real world experiments, yet. Despite their differences, they are by necessity closely connected: Experiments are often used to prove theories and ideas that have been established by theoretical scientists. On the other hand, theories often inspire new experiments or yield explanations for unexpected experimental findings.

Despite the importance of theoretical science, genius minds had always been struggling to get their theories accepted in the scientific community. When Darwin first published his theory on evolution in "On the Origin of Species" [46], he received a lot of opposition. Although Darwin's theories have become a central part of modern biology, even today, more than 150 years later, his theories are still not fully accepted. Although society has changed, theories unproven by real world experiments will always lack credibility.

In the present information age, computational biology represents the theoretical counterpart of modern biology. Instead of real word experiments performed in a laboratory, computational biology is based on mathematical theories and algorithms. A

prominent class of algorithms in computational biology are machine learning algorithms. As the name suggests, machine learning aims at learning patterns and coherences from empirical data. Thus, similar to a human, a learning algorithm tries to generalize from its experiences. The learned knowledge serves as basis for a machine learning model, which can be later used to theoretically predict properties of previously unseen examples. A prominent application is stock market prediction, which aims to predict the future behavior of a stock by learning from trading data from the past. Depending on the learning task, we distinguish between classification and regression. In classification, observations are categorized into groups based on distinct attributes. In contrast, regression aims at finding a relationship between attributes that explains a quantitative property. For example, classification algorithms can be applied to predict whether a stock should be purchased or sold, whereas regression models predict the stock value.

Machine learning has become an essential part of modern biology. Machine learning algorithms have been applied to various problems, ranging from gene prediction to three-dimensional structure prediction of proteins. Predictions made by computational models offer an attractive alternative to time-consuming and expensive experiments. Sometimes machine learning methods are even praised as a potential replacement of wet lab experiments.

Unfortunately, the credibility of machine learning models in the biological community is still rather low. Results obtained by real world experiments earn far more trust than predictions made by theoretical models. One major reason is the "black box" character of many state-of-the-art machine learning approaches. While recent work has mainly concentrated on improving the prediction performance of classification and regression models, their interpretability has been mostly neglected. Consequently, when applying state-of-the-art prediction algorithms, two important questions remain unanswered: Why did the model predict this particular outcome? And, how reliable is this individual prediction?

One major reason for non-interpretable predictions is the complexity of state-of-the-art classification and regression algorithms. Kernel-based machine learning algorithms such as support vector machines (SVMs) and support vector regression (SVR) are often able to give high quality predictions although the underlying problem might be very

difficult. However, due to their complexity, users generally have no insight into the prediction process. For an individual prediction, it is often not possible to understand what attribute contributed most to the prediction outcome. Instead, users are left with plain prediction values without any additional information that could support the prediction. Since users are not provided with an understandable and intuitive reasoning, it is not evident why an individual prediction was made. This lack of interpretability prevents users from gaining a deeper understanding of the underlying problem. In addition, users have less trust in predictions that are not interpretable.

To provide predictions that are interpretable for the user, it is often necessary to employ more simple machine learning algorithms. As a consequence, learning methods that provide interpretability yield often a reduced prediction quality. Furthermore, explanations are often rather technical than understandable. Instead of offering understandable problem specific textual explanations, they provide only non-intuitive tables consisting of plain numbers.

Another problem that is often not considered by state-of-the-art machine learning models is the reliability of individual predictions. Machine learning models are often trained on rather limited datasets. While they maintain good performance on closely related datasets, the error may increase drastically when applied to data points far from the training set. For example, a stock prediction model trained in the 1970s is unlikely to give reliable predictions in the time of today's financial crisis.

In classification, there have been some first attempts to estimate the confidence in individual predictions [129]. However, there is often a misunderstanding between the uncertainty between classes and the confidence in the correctness of the prediction. In regression analysis, the concept of confidence estimation was introduced by estimating the applicability of a model [164]. While in the experimental sciences, a measurement is associated with an error, confidence estimators for regression often provide only a non-interpretable qualitative score, which is difficult to relate to an actual error.

Despite the importance of confidence estimation, it has not been applied extensively in the context of computational biology. Instead, most machine learning approaches are totally non-transparent for the user and often lack the explicit underlying model required for error propagation. Since confidence information is hardly ever reported, users are often compelled to assume that the machine learning model performs equally well for all examples, which is not the case. When selecting candidates for expensive

experiments, it is especially desirable to mitigate the risk of accidentally relying on erroneous predictions. To overcome this problems, confidence estimation, which determines the reliability of individual predictions, is desirable.

In this thesis, we aim at filling the gap between experimental sciences and computational methods by developing interpretable machine learning approaches for both classification and regression We are interested in models that elucidate why a particular prediction was made and how reliable the predicted outcome is. Therefore, we study the interpretability of machine learning models on two biological problems. We first introduce an interpretable classification approach for subcellular localization of proteins. In the second part of this thesis, we introduce methods that make regression approaches for immunoinformatics more interpretable. For both biological problems, we show that user can greatly benefit from interpretable predictions made by our introduced machine learning approaches. We demonstrate that interpretable predictions can help biologists to gain deeper insights into the underlying biological problems and can substantially support their experimental work.

The first goal of this thesis is to develop an interpretable classification approach for protein subcellular localization, an important problem of systems biology. Subcellular localization of proteins is a key process in many eukaryotic cells. Correct localization of proteins within a cell is crucial for all organisms since errors in this sorting process can lead to diseases. Hence, it is a major research topic in biology.

After being synthesized in the ribosomes, proteins are transported into the different organelles of a cell depending on their function within the cell. Some proteins are even transported to multiple compartments. Even though not all details of protein sorting are fully understood, it is known that protein localization is often mediated by sorting signals. These sorting signals are short and well-defined subsequences usually located at the N-terminus or C-terminus of the protein. Since organelles of eukaryotic cells are normally separated from each other by membranes, the transport of a protein is often mediated by transmembrane proteins or protein complexes that recognize sorting signals and facilitate transport across the membranes. It is known that the subcellular localization of a protein is highly correlated with its function and is, thus, used to

draw conclusions about its cellular role, interaction partners, and function in biological processes.

There exists various ways to experimentally determine the subcellular location of a protein, for example by fluorescence microscopy, cell fractionation, or by immunolocalization. However, all these techniques are expensive and time-consuming. In the last years, experimental annotation could not keep up with the dramatic growth of sequence data. Hence, many proteins are not yet annotated with their subcellular location. In fact, to determine the locations of all protein sequences by experiments is an almost infeasible task.

Computational prediction methods that predict the subcellular localization of a protein from its amino acid sequence alone, thus, represent an attractive alternative to experimental methods. Predicting the subcellular location of a protein is a typical classification problem that can be solved by using machine learning algorithms. Based on a set of proteins with known location site, a classification model is trained to predict the locations of novel proteins. For this, proteins are usually encoded by a set of informative attributes which, for example, indicate whether certain sorting signals are present in the protein's sequence or not. While early approaches were simply sequence based [9, 62, 131], more recent methods use a wide range of derived attributes including domain information and information on proteins with a similar sequence [8, 37, 71, 123, 158]. Furthermore, the prediction quality has significantly improved over the years, also due to the application of more complex machine learning models [8, 45, 124, 139].

An unfortunate drawback of state-of-the-art subcellular localization predictors is their "black box" character. It is not obvious whether, for example, the presence of a sorting signal influenced the outcome. Hence, from their predictions biologists are not able to gain a deeper understanding of the underlying sorting process nor can predictions be used to guide further experiments. In addition, the lack of confidence estimation makes it difficult for users determine their trust in individual predictions.

To overcome the shortcomings of current state-of-the-art methods, we present YLoc, an interpretable prediction approach for protein subcellular localization. Users are not only provided with the prediction itself, but also with an explanation why this prediction was made. Since YLoc gives explanations in natural language, biologists can derive knowledge why a protein is localized to a certain organelle. For example, YLoc identifies relevant biological properties of a protein contributing to its subcellular

localization, e.g. localization signals or motifs relevant to protein sorting. Hence, YLoc can help to understand subcellular localization processes without conducting expensive experiments. Furthermore, YLoc employs a confidence estimator, which helps biologists to verify whether a prediction is reliable or not. Due to its ability to produce interpretable predictions, YLoc can be used for supervised protein engineering. We show several examples where YLoc is able to predict experimentally validated changes of localization sites by detecting changes in the sorting signal.

The presented interpretable classification approach is an important step to overcome the theoretical character of computational prediction approaches.

In the second part of this thesis, we illustrate how regression analysis can be made more interpretable. A prominent application of regression algorithms can be found in the field of immunoinformatics. In particular, we are interested in the interaction of major histocompatibility complex class I (MHC-I) molecules with peptides, which plays an important role in the adaptive immune system.

The adaptive immune system is a highly specialized system that recognizes, eliminates, and remembers invading pathogens such as bacteria or viruses. MHC-I molecules, as an important element of the adaptive immune system, present pathogenic peptides on the surface of the cell to the immune system. Presented peptides that can trigger an immune response and lead to the death of the infected cell are called epitopes.

The property of the immune system to remember pathogens provides the foundation for vaccines. Traditional vaccines contain a weakened form of the pathogen to stimulate the immunological memory. In contrast, epitope-based vaccines contain only the peptides derived from pathogens that are able to induce an immune response. As a prerequisite for an immune response, epitopes are required to bind to MHC-I molecules. Hence, to design epitope-based vaccines, it is crucial to find a set of peptides with a high binding affinity to MHC-I molecules.

Experimentally determining the binding affinity of peptides to MHC-I molecules is a time-consuming and expensive process. Consequently, various machine learning algorithms have been applied as an alternative to experimental approaches. Predicting binding affinities is a typical regression problem. Based on peptides with known binding affinities to MHC-I molecules, a regression model is trained to predict the affinities of peptides for which no experiments have been conducted. While in the early years,

simple linear algorithms were used [136], current state-of-the-art prediction approaches use more complex algorithms [116].

Unfortunately, predictions of state-of-the-art approaches are not interpretable. It is not clear which amino acid in the peptide exhibits the strongest interaction with the MHC-I molecules, nor which amino acid might prevent the peptide from binding. Most importantly, the confidence of individual predictions is often neglected. In contrast to experiments performed in a laboratory, computational prediction methods do not produce likewise reliable affinity values for all peptides. In fact, if the given training data does not allow to draw conclusions for a novel peptide, their predictions are almost random. The absence of confidence estimation in most MHC-I–peptide binding predictors is certainly a disadvantage for tasks like epitope-based vaccine design where highly reliable predictions are invaluable.

To make MHC-I–peptide binding prediction more interpretable, we developed two novel confidence estimators, CONFINE and CONFIVE, for regression prediction. They estimate the error of individual predictions, while requiring only a small computational overhead. In contrast to a plain predicted binding affinity, both estimators determine a confidence interval which contains the correct affinity with a certain probability. This makes predictions more transparent for users since low confidence predictions can be identified by very broad confidence intervals. We prove that our confidence estimators can identify highly reliable predictions, which yield a considerably lower prediction error than average. In an example application, we illustrate how epitopes for vaccines can be automatically engineered using a genetic algorithm. By combining binding affinity prediction with confidence estimation based on our estimator CONFINE, we optimize the MHC-I-binding affinity by considering only confident predictions. The presented interpretable regression approach represents an important step from plain, non-informative predictions towards transparent MHC-I–peptide binding prediction, which represents a valuable alternative to experimental approaches.

This thesis is structured into five chapters. Following this introduction, we provide the relevant background on machine learning, protein subcellular localization, and immunology in the second chapter. In Chapter 3, we present YLoc, our interpretable classification approach for protein subcellular localization. Further, we show how confidence estimation increases the interpretability of MHC-I–peptide binding prediction in

the fourth chapter. In Chapter 5, we conclude and discuss our findings on interpretable machine learning approaches and show perspectives for further work in this area.

Chapter 2

Background

2.1 Machine Learning

Similar to humans, machine learning algorithms can generalize from their experience and are, thus, able to predict properties of novel data. In this work, we focus on supervised learning algorithms, which aim at learning a prediction function from data with given input and output, called training data. In particular, we are interested how the attributes of the input influence the output. The training dataset is a set of learning examples, also called instances, which contain information on previous observations. Each instance i can be encoded by a vector of k attributes $x_i = (x_{i1}, \ldots, x_{ik})^T$, called input vector or feature vector. The matrix of all feature vectors $X = (x_1, \ldots, x_n)^T$ is often called feature matrix. The provided features describe distinct properties of the corresponding real world objects in a qualitative or quantitative manner. A feature representation that describes the individual aspects of the different instances is essential for most learning algorithms. In addition, each instance is assigned with at least one label y_i, which is the output we aim to predict. Vector $Y = (y_1, \ldots y_n)^T$ is called output vector.

To predict the label $\hat{y}$ of a novel instance, machine learning algorithms usually first have to be trained on the training data $D = \{(x_i, y_i)\}$. However, since the provided data often covers only a small part of the possible input space, the underlying distribution of the data is unknown. Hence, machine learning can only approximate the real distribution by generalizing from the provided training data.

Depending on the values of the labels, we distinguish between classification and

regression. In classification, labels can take only a limited number of values, often called class labels or classes. Classification algorithms aim at assigning unlabeled instances to the most probable class. In contrast, regression algorithms predict continuous values, also called response values. A regression algorithm tries to find a functional relationship between the attributes and the response values.

In the following sections, we introduce the basics of classification and regression together with some important algorithms. Finally, we discuss the interpretability and reliability of predictions made by machine learning models.

2.1.1 Classification

Classification is the problem of assigning instances to a fixed number of $m \geq 2$ categories, i.e. classes. Every instance in the training data is assigned a class label $y_i \in C = \{C_1, \ldots, C_m\}$. If exactly two classes are given, we refer to the problem as binary classification. Otherwise, it is called a multi-class classification problem.

Based on the training data, a classification algorithm aims at partitioning the input space in a way that each separate regions contains only instances from the same class. The different partitions will be later used to classify novel instances by assigning the class label of their corresponding region in the input space. The boundaries of these regions are called decision boundaries. Linear classification models can only learn decision boundaries that are linear in the features. Given a high dimensional input space, this amounts to finding the optimally separating hyperplane. However, often a linear decision boundary is not sufficient to separate the input space into regions of distinct classes. This can be the case if the feature representation is not discriminative enough but also if a nonlinear combination of several features has influence on the category of the instances. Nonlinear models are able to learn more complex decision boundaries and, thus, are often more powerful than simple linear models.

In many real word applications, it is very difficult to find the optimal decision boundary. To obtain at least a good approximation, we require an optimality criterion which helps us to choose the best parameters for our classifier. Usually, the expected value of a loss function, called risk, is used to measure the quality of a classifier. A typical loss function is the 0-1 loss. It equals one if the real class label y_i equals the predicted label $\hat{y}_i$ and zero otherwise. Minimizing the loss on the training data results

in the lowest risk and determines the classifier that is optimal with respect to our criterion.

A plethora of different classification algorithms are in use today. Their formal history probably starts in 1958, when Rosenblatt introduced the perceptron algorithm [149]. It finds an optimal separating hyperplane in the training data, if one exists. Later, other linear classification models such as logistic regression [109] and linear discriminant analysis [67] were introduced. Prominent examples for nonlinear classification models are k-nearest neighbors (kNN) [42], artificial neural networks (ANNs) [87], support vector machines (SVMs) [41], and naïve Bayes classifier [119]. In cases where the data cannot be separated by linear decision boundaries, nonlinear models often perform better than linear classification models. However, their predictions are often even harder to interpret than those of their linear counterparts, which is discussed in Section 2.1.4.

Support Vector Machines

Due to their good generalization performance, SVMs have become popular classification algorithms. An SVM is a binary classification algorithm that tries to find a separating hyperplane between two classes such that the margin between both classes is maximized. It is based on the assumption that the best separating hyperplanes is the one with the largest distance to instances of the different classes (see Figure 2.1). Since an SVM aims at maximizing the margin between two classes, it is also called maximum margin classifier.

The separating hyperplane is defined by all possible vectors x that satisfy

$$f(x) = \langle w, x \rangle + b = 0, \tag{2.1}$$

where $w \in \mathbb{R}^k$ is a weight vector and $b \in \mathbb{R}$ is the offset of our hyperplane. The predicted class label $\hat{y}$ is assigned with the result of the following decision function

$$\operatorname{sgn}(f(x)). \tag{2.2}$$

The margin of the hyperplane is defined by the closest distance of an instance to the hyperplane. To obtain a normalized form of the problem, we require

$$\forall (x_i, y_i) \in D : \min_i |f(x_i)| = 1. \tag{2.3}$$

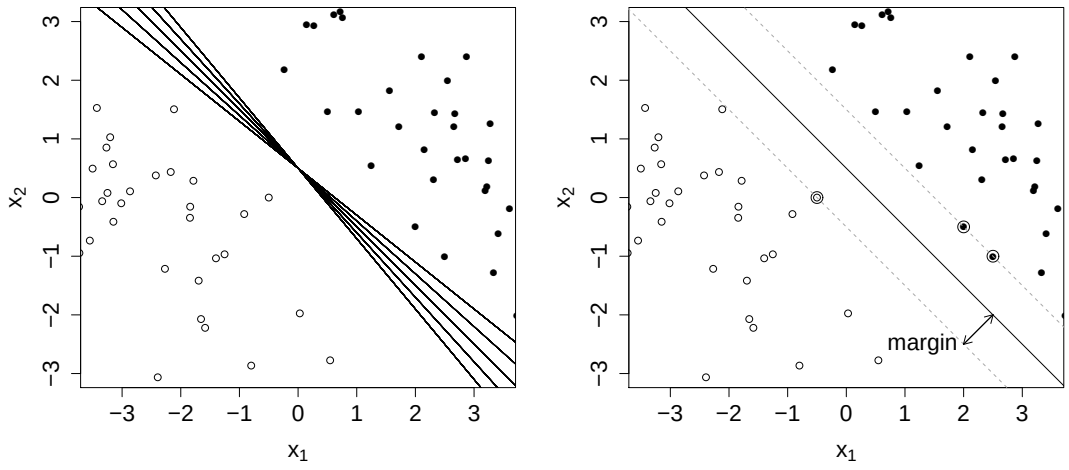

Figure 2.1: The two-dimensional input space of a binary classification problem with one class as white points and a second class as black points. The left-hand plot shows a set of separating hyperplanes (solid lines) which all perfectly separate the instances in the input space. Based on the assumption that the best separating hyperplane is the one which induces the largest margin between both classes, an SVM chooses the optimal hyperplane displayed as solid line in the right-hand plot. The encircled instances on the borders of the margin (dashed gray lines) are called support vectors.

Instances with $|f(x_i)| = 1$ are called support vectors since they support the borders of the margin. The resulting hyperplane is a canonical hyperplane with a margin of $1/||w||$, where $||w||$ is the Euclidean norm of vector w. Finding the hyperplane with the largest margin is equivalent to maximizing the margin of the canonical hyperplane by minimizing $||w||$. Since minimizing $\frac{1}{2}||w||^2$ yields the same solution, we can define the margin maximization as a quadratic programming optimization problem as follows:

$$\begin{aligned} &\min_{w \in \mathbb{R}^k, b \in \mathbb{R}} \ \frac{1}{2}||w||^2, \\ &\text{subject to } \ y_i(\langle w, x_i \rangle + b) \geq 1 \qquad \forall (x_i, y_i) \in D. \end{aligned} \tag{2.4}$$

The above optimization problem can be solved only if the training data is perfectly separable by a linear hyperplane. However, in many real world applications, classes overlap in the input space making a perfect separation impossible. To overcome this shortcoming, Cortes and Vapnik [41] introduced a soft margin SVM. In contrast to the

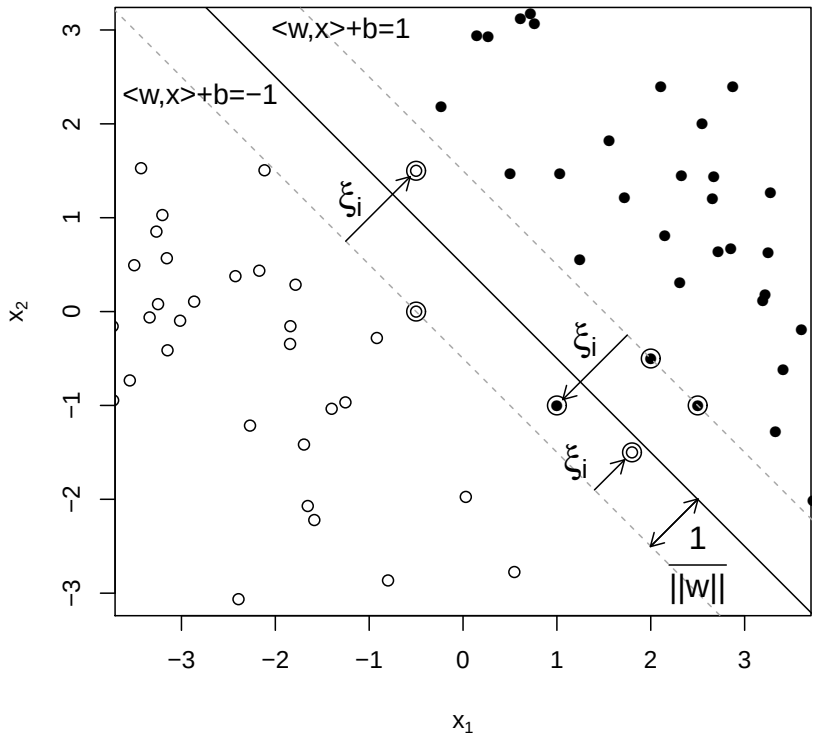

Figure 2.2: The canonical hyperplane (solid line) of a soft-margin SVM. The encircled instances are the corresponding support vectors. For instances on the margin of the hyperplane (dashed gray lines), $|\langle w, x\rangle + b| = 1$ holds. Instances on the wrong side of the margin are penalized by a slack variable ξ_i.

hard margin SVM presented above, it allows for a certain degree of imperfect separation. Every instance is assigned a slack variable $\xi_i \geq 0$, which measures the distance of an instance lying on the wrong side of the separating hyperplane to the borders of the margin (see Figure 2.2). To find a good trade-off between a large margin and small loss, we define the following optimization problem

$$\begin{aligned} \min_{w\in\mathbb{R}^k,\, b\in\mathbb{R},\, \xi\in\mathbb{R}_0^{+n}} \; & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i, \\ \text{subject to } & y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \quad \forall (x_i, y_i) \in D, \end{aligned} \tag{2.5}$$

where C is the trade-off factor, called cost parameter. In addition to instances on the borders of the margin, instances with a slack variable ≥ 0 are now also called support

vectors.

We can extend linear SVMs by introducing a dual representation of the classification problem. The separating hyperplane of an SVM can also be described as a linear combination of support vectors:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i, \tag{2.6}$$

where $\alpha_i \geq 0$ are Lagrange multipliers. Using this representation, maximizing the margin results in the dual optimization problem:

$$\begin{aligned} &\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ &\text{subject to} \sum_{i=1}^{n} \alpha_i y_i = 0, \, \alpha_i \geq 0 \quad \forall i = 1, \ldots, n. \end{aligned} \tag{2.7}$$

See the appendix of Cortes and Vapnik [41] for a detailed derivation of the dual optimization problem and the corresponding optimization problem for a soft-margin SVM. Using the dual representation of w, the predicted class label $\hat{y}$ is determined by the following decision function:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \right). \tag{2.8}$$

The dual representation of an SVM has an important advantage. Since the hyperplane is only defined by inner products $\langle x_i, x_j \rangle$, we can replace them by a so-called kernel function or kernel

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle, \tag{2.9}$$

where function ϕ maps the input space into a higher dimensional inner product space. An explicit mapping function ϕ is not even necessary as long as the kernel maps the data into an inner product space. This mapping is often referred to as the kernel trick. Hence, if the data is not linearly separable, we can implicitly map the data into a higher-dimensional space where a linearly separating hyperplane exist without actually transforming the data. Consequently, the decision boundary in the original space is no longer linear in the input variables. Usually, the actual decision boundary cannot be obtained since the explicit mapping function is often unknown.

There exist various kernel functions with different properties. Very popular kernels are the polynomial kernel and the Radial Basis Function (RBF). Furthermore, various kernels based on strings, graphs, and other input types have been developed to tackle problems in computational biology [4, 162].

Although basic SVMs can only perform binary classification, they can also be applied to multi-class classification problems by combining multiple SVMs. In the one-vs-one approach, each SVM discriminates between a pair of classes, whereas in the one-vs-all approach, each SVM discriminates between one class and all other classes. The final prediction is obtained by combining the individual prediction results using majority vote. In addition, recent progress has shown that multi-class classification can also be formulated as a single optimization problem [43].

An SVM is a very powerful learning tool that can be adapted to various learning tasks by the use of kernels. However, SVMs often provide no explanation that answers why a prediction was made. Usually it remains unknown how features contribute to a particular outcome. Consequently, predictions made by SVMs are often not considered as interpretable. In the following section, we introduce another nonlinear classification algorithm that, in contrast to an SVM, can create interpretable predictions.

Naïve Bayes

Naïve Bayes is a very simple and robust probabilistic classification model based on Bayes' Theorem. Given a set of k features $F = \{F_1, \ldots, F_k\}$ and a set of m class labels $y \in C = \{C_1, \ldots, C_m\}$, we can apply Bayes' theorem to the classification problem and obtain

$$P(C_j|F) = \frac{P(C_j)P(F|C_j)}{P(F)}. \tag{2.10}$$

Since the distribution of the data in the input space is unknown, $P(F)$ can often not be determined. In addition, $P(F)$ is independent of the actual class C_j and has the same influence on the probability of all classes. As a consequence, it is not considered in practice.

Furthermore, naïve Bayes assumes features to be conditionally independent of each other, leading to $P(F|C_j) = \prod_{h=1}^{k} P(F_h|C_j)$. This drastic assumption makes the model "naive" since, in practice, conditional independence is often not the case. However,

it has been shown that naïve Bayes is still surprisingly effective in cases where the independency assumption is violated [147].

As a consequence of the above assumptions, the probability estimation of naïve Bayes simplifies to

$$P(C_j|F) \propto P(C_j) \prod_{h=1}^{k} P(F_h|C_j). \tag{2.11}$$

The final probabilities are obtained by normalizing the posteriori probability $P(C_j|F)$ such that the sum of all posteriori probabilities is one. Obviously, the class with the largest posteriori probability $C_{max} = \arg\max_{C_j} P(C_j|F)$ will be predicted as class label $\hat{y}$.

For training a naïve Bayes classifier, we are required to estimate the underlying class and feature distributions. This is usually done in a very time-efficient manner. The prior probability of a class $P(C_j)$ is estimated via maximum likelihood estimation by simply assigning the relative frequency of the class in the training data. Often a Laplace estimate is applied, which adds a pseudo instance to every class:

$$P(y = C_j) = \frac{n(C_j) + 1}{n + m}, \tag{2.12}$$

where $n(C_j)$ is the number of training examples in class C_j.

Estimating $P(F_h|C_j)$ from the training data is often performed by discretizing the data. For this purpose, every feature is divided into a set of p intervals $\{(a_i, b_i] \mid i \in \{1, \ldots, p\},\ b_i = a_{i+1}\}$, also called bins. If a feature value matches an interval $x \in (a_i, b_i]$, we approximate $P(F_h = x|C_j)$ by $P(a_i < x \leq b_i|C_j)$. The probabilities of an interval are estimated by the relative frequency of instances from class C_j within the interval. To avoid zero probabilities, we use a Laplace estimate and approximate

$$P(F_h = x|C_j) = P(a_i < x \leq b_i|C_j) = \frac{n_i(C_j) + 1}{n(C_j) + p}, \tag{2.13}$$

where $n_i(C_j)$ is the number of instances from class C_j within the i-th interval.

There exist various methods for feature discretization. Very simple discretization methods do not consider the actual class distributions. For example, equal-width discretization creates intervals of equal size $b_i - a_i$, whereas equal-frequency discretization chooses interval borders such that each bin contains an equal amount of training examples [196]. More involved discretization methods consider the different class affiliations, e.g., by adjusting interval borders to decrease the entropy of class labels in each

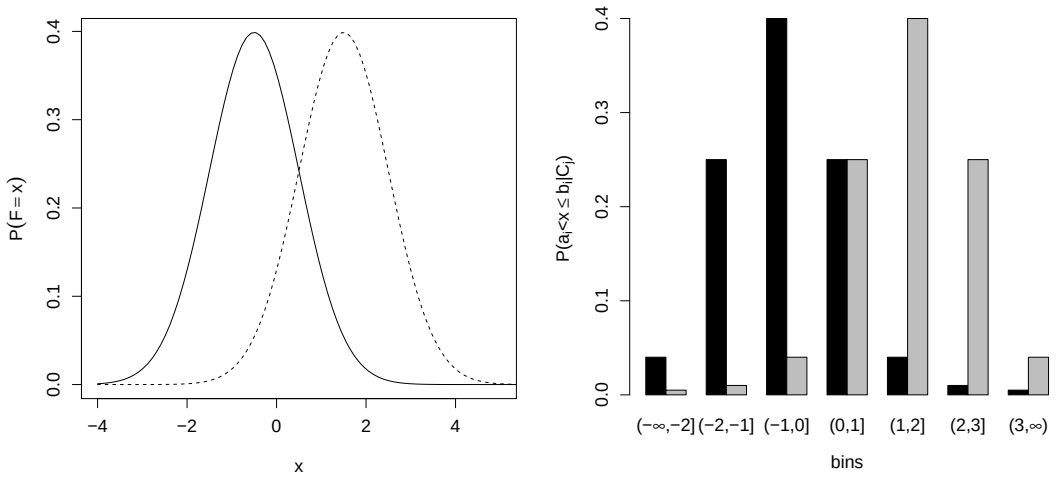

Figure 2.3: Example discretization and probability estimation of naïve Bayes. In the left-hand plot, an example distribution of a feature for two classes is displayed by a solid line and a dashed line. The right-hand plot shows the corresponding discretized distributions obtained by equal-width discretization in black and gray, respectively.

bin [65]. See Figure 2.3 for an example discretization and the corresponding estimated probabilities. An important aspect of using naïve Bayes with discrete features is its interpretability. It is easy to understand why and how an individual feature contributed to a prediction outcome. If an instance exhibits a certain feature value, it is obvious whether the feature is typical for a class or not. For example in Figure 2.3, a feature value of -1.5 is ten times more likely for an instance of the first class than for the second class and is, thus, very typical for the first class.

Although naïve Bayes does not consider feature combinations, it is not a linear model. Different values of a feature can have different impact on the prediction outcome, which is not the case for a linear model. For example, in linear SVMs, features are multiplied by a constant coefficient of the weight vector of the model and, thus, have a constant influence on the prediction outcome. Thus, although naïve Bayes is a very simple model, it can learn nonlinear dependencies between features and output. Furthermore, due to its simplicity, the contributions of features can be easily assessed by users.

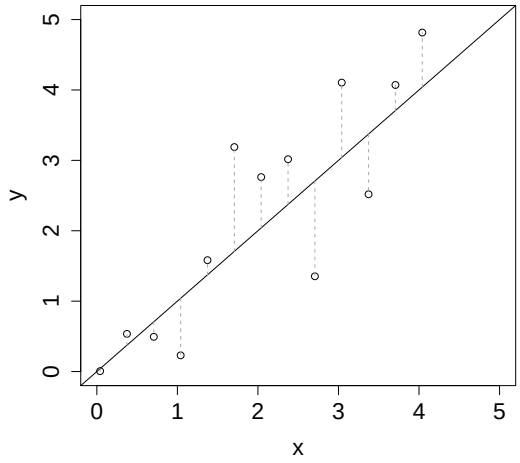

Figure 2.4: Example of a one-dimensional linear regression function. A regression function (solid line) is fitted to some data points with the corresponding residuals indicated by dashed gray lines.

2.1.2 Regression

Regression analysis aims at finding a regression function $f(x)$ that provides the best explanation of the relationship between a set of independent input variables $x_1, \ldots, x_k$ and an output variable $y \in \mathbb{R}$. This process is also often called fitting since we try fit a regression function to the data points.

Similar to classification models, regression models can only approximate the true underlying function. In an optimization process, we try to find the regression function that fits the training data best. To assess the quality of a fit, we require a formalized quality measure. Typically, the difference between the predicted response and the real response $\hat{\epsilon} = y_i - \hat{y}_i$, also referred to as residuals or prediction errors, are used by a loss function (see Figure 2.4). A simple loss function is the sum of squared errors (SSE):

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{2.14}$$

Finding the regression function that minimizes the SSE is also called least squares.

Around 1800, Gauss and Legendre were the first to perform a least squares regression analysis. Their work laid the basis for modern regression algorithms. Today, we distinguish between linear regression algorithms and nonlinear regression algorithms.

Linear regression algorithms can model a linear function of the input variables. Often used linear algorithms are least squares regression, ridge regression [181], and principal component regression [89]. In contrast, nonlinear regression algorithms can model also functions that are a nonlinear combination of the input variables. Commonly used nonlinear regression approaches are locally weighted regression [2], ANNs [174], Gaussian processes [126], and support vector regression (SVR) [41].

Linear Least Squares Regression

Linear regression is often used as a synonym for linear least squares regression. It assumes an approximately linear relationship between the input variables $x_{i1}, \ldots, x_{ik}$ and the output variable y_i such that

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \epsilon_i, \tag{2.15}$$

where β is the parameter vector containing the coefficients of our linear function and ϵ is the corresponding vector of residuals. If we define $x' = (1, x_1, \ldots, x_n)^T$ to be the feature vector with a leading constant and X' to be the corresponding feature matrix, we receive

$$y = X'\beta + \epsilon. \tag{2.16}$$

For simplicity, let X denote X' in the following.

Finding the linear regression model that explains the data best is achieved by performing ordinary least squares on the training data. Hence, we aim at finding a vector β that minimizes

$$SSE = (y - X\beta)^T (y - X\beta). \tag{2.17}$$

The above optimization function has a unique global optimum if the features are independent. The optimal parameters of the linear regression models could in theory be calculated by

$$\hat{\beta} = (X^T X)^{-1} X^T y. \tag{2.18}$$

The regression function used to predict the response value $\hat{y}$ of an instance x can be summarized as

$$f(x) = x^T \hat{\beta}. \tag{2.19}$$

A key assumption of linear regression – and regression analysis in general – is the independence of the features. If two features are perfectly collinear, the feature matrix

X does not have full column rank, which makes β not identifiable. Also, if features are highly correlated, parameters can take meaningless large values. To penalize large parameter estimates, we introduce a regularization term, for example a Tikhonov regularizer [181]. In the resulting optimization problem, we aim at minimizing

$$(y - X\beta)^T(y - X\beta) + \lambda||\beta||, \tag{2.20}$$

where λ is the ridge penalty. This regression approach is also called ridge regression. Its parameters can be determined by

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y, \tag{2.21}$$

where I denotes the identity matrix.

Linear regression models show optimal results regarding the SSE if the relationship between input and output data is almost linear. However, a linear model fails if non-linear relationships occur in the dataset. Nevertheless, a significant advantage of linear models is that the influence of the different features on the prediction can be easily assessed.

Support Vector Regression

SVR is a generalization of the SVM [41] introduced above. It aims at finding a regression function with small error that is sparse at the same time. Following the principle of Occam's razor [52], a sparse model with the same explanatory power is preferred over a more complex model.

A linear SVR uses the same regression function as linear least squares regression

$$f(x) = \langle w, x \rangle + b, \tag{2.22}$$

where $w \in \mathbb{R}^k$ corresponds to $(\beta_1, \ldots, \beta_k)$ and $b \in \mathbb{R}$ equals β_0, the offset.

To find the optimal regression function, we place an ϵ-insensitive band around the regression function. Thus, absolute errors smaller than ϵ are not penalized, whereas instances i with an absolute error larger than ϵ are penalized by slack variables $\xi_i \in \mathbb{R}$ and $\xi_i^* \in \mathbb{R}$, respectively (see Figure 2.5). Our regression function is sparse if it depends on very few features, leading to a small norm of vector w. Hence, in SVR, we minimize

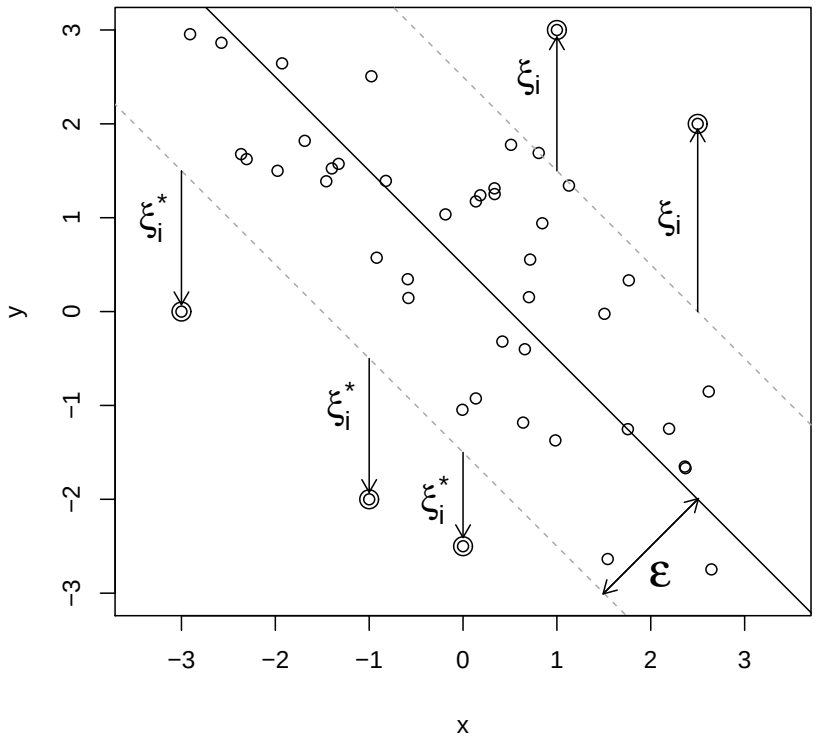

Figure 2.5: A one-dimensional linear SVR with an ϵ-insensitive band. The circled instances are the corresponding support vectors outside the ϵ-insensitive band. They have an absolute error larger than ϵ and are penalized by slack variables ξ_i and ξ_i^*.

$||w||$ and the error at the same time. We have to solve the following optimization problem to train an ϵ-SVR:

$$\begin{aligned} &\min_{w\in\mathbb{R}^k,\, b\in\mathbb{R},\, \xi\in\mathbb{R}_0^{+n},\, \xi^*\in\mathbb{R}_0^{+n}} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*), \\ &\text{subject to } (\langle w, x_i\rangle + b) - y_i \leq \epsilon + \xi_i \quad \forall (x_i, y_i) \in D \\ &\qquad\qquad y_i - (\langle w, x_i\rangle + b) \leq \epsilon + \xi_i^* \quad \forall (x_i, y_i) \in D, \end{aligned} \tag{2.23}$$

where C is the cost parameter determining the trade-off between minimizing $||w||$ and the error.

Note that the value of ϵ has to be given in advance. Using a ν-SVR, the optimal

value of ϵ is automatically determined during the optimization. See Schölkopf *et al.* [153] for details.

Similar to SVMs, a regression function can also be described by a linear combination of support vectors. In case of SVR, support vectors are instances that yield an absolute error larger than ϵ. Considering only instances outside of the ϵ-insensitive band gives us a sparse definition of our regression function:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b, \tag{2.24}$$

where α and α^* are Lagrange multipliers. See Smola and Schölkopf [172] for the corresponding dual optimization problem. The sparseness of the solution in terms of $||w||$ can now be described by the number of support vectors required to describe $f(x)$.

The linear SVR introduced above is only able to model a regression function that is linear in the features. However, by replacing the inner product $\langle x_i, x_j \rangle$ by a kernel function, SVR can also model nonlinear responses, see also Section 2.1.1. As a drawback, the actual nonlinear regression function can often not be determined explicitly. Consequently, predictions are not interpretable since the contribution of individual features to the outcome cannot be determined.

2.1.3 Model Evaluation and Selection

The performance of machine learning models depends strongly on parameter settings and the used feature set. To select the model that yields the highest prediction quality, a model has to be evaluated with different parameter settings and different feature sets. Moreover, the evaluation of a model is very important to get an unbiased impression of its predictive power.

The prediction quality of a machine learning model can be estimated on a test dataset with known labels, i.e. $T = \{(x_i, y_i)\}$. It is required that T is independent of the training dataset D, that is $T \cap D = \emptyset$. If both sets are not independent, we would test whether the model memorizes the training data but not whether the model is able to generalize. Hence, we might overestimate the performance of the model. The performance of a model is estimated by comparing the predicted labels with the real labels of the test dataset. For this, various statistical performance measures for classification and regression exist.

Classification Measures

The most common quality measure for classification is the accuracy (ACC), which calculates the proportion of correctly classified instances. Although the accuracy is a widely used measure, it has a major drawback. Given a dataset where the instances are not uniformly distributed among classes, also called a skewed dataset, the accuracy is biased towards the majority class.

An alternative classification measure is based on the precision and the recall of a class C_j defined as

$$\begin{aligned} \mathrm{PRE}_j &= \frac{\mathrm{TP}_j}{\mathrm{TP}_j + \mathrm{FP}_j} \quad \text{and} \\ \mathrm{REC}_j &= \frac{\mathrm{TP}_j}{\mathrm{TP}_j + \mathrm{FN}_j}, \end{aligned} \tag{2.25}$$

respectively. TP_j, FP_j, and FN_j are the number of true positives, false positives, and false negatives of class C_j, respectively. Precision is a measures of exactness and equals the fraction of instances that were correctly assigned to a class. Recall is a measure of completeness and equals the fraction of instances of a class that were correctly classified. Precision and recall are complementary measures and can be combined in the F_1 score (F1) of a class C_j by calculating their harmonic mean

$$\mathrm{F1}_j = 2 \cdot \frac{\mathrm{PRE}_j \cdot \mathrm{REC}_j}{\mathrm{PRE}_j + \mathrm{REC}_j}. \tag{2.26}$$

The F_1 score is a balanced measure of the classification accuracy of one particular class. To obtain an overall measure, the F_1 score is usually averaged over all classes.

Regression Measures

As introduced in Section 2.1.2, the sum of squared error (SSE) is often used as a performance measure for regression. Since the SSE is not comparable across datasets, the coefficient of determination, also called R^2, has been introduced. It normalizes the SSE by the sample variance

$$R^2 = 1 - \frac{\mathrm{SSE}}{\sum_i (y_i - \bar{y})^2}, \tag{2.27}$$

where $\bar{y}$ is the average response value in the test dataset. The R^2 is close to one if the prediction error is very small.

Another often used performance measure is the Pearson product-moment correlation coefficient, often referred to as correlation coefficient. It is defined as the covariance of the response vector and the predicted response vector divided by the standard deviations of both vectors:

$$\rho = \frac{\mathrm{cov}(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}}. \tag{2.28}$$

The correlation coefficient equals one if the predicted responses are perfectly correlated with the real responses. On the other hand, random predictions yield a correlation coefficient around zero.

Cross-Validation

In many real world scenarios, an independent test dataset is not available. As an alternative, an evaluation technique called cross-validation can be employed. In k-fold cross-validation, the training dataset is partitioned into k subsets of equal size. One subset is retained as test set, while $k-1$ subsets are used as training data. This evaluation procedure is repeated k times, called folds, resulting in k independent training and test datasets. The model's performance is then estimated by averaging the model's performance over all folds (see Figure 2.6).

A cross-validation on a training dataset of another cross-validation is called inner cross-validation (see Figure 2.6). Nesting cross-validations is often performed to evaluate a model selection, as explained in the following.

Model Selection

The performance of most machine learning models depends on its parameters and the chosen feature set. Different parameter values and different features can lead to different prediction behavior. Finding the optimal parameter values and the optimal feature set is done by a model selection prior to the actual training of the machine learning model.

Model selection is based on a search strategy that returns a candidate model defined by its parameter values and used feature set. Such a search strategy could range from greedy search to an exhaustive search. To assess the prediction performance of a candidate model, we perform a cross-validation on the training dataset. The quality of the parameter setting and the selected feature set is assessed by averaging the prediction

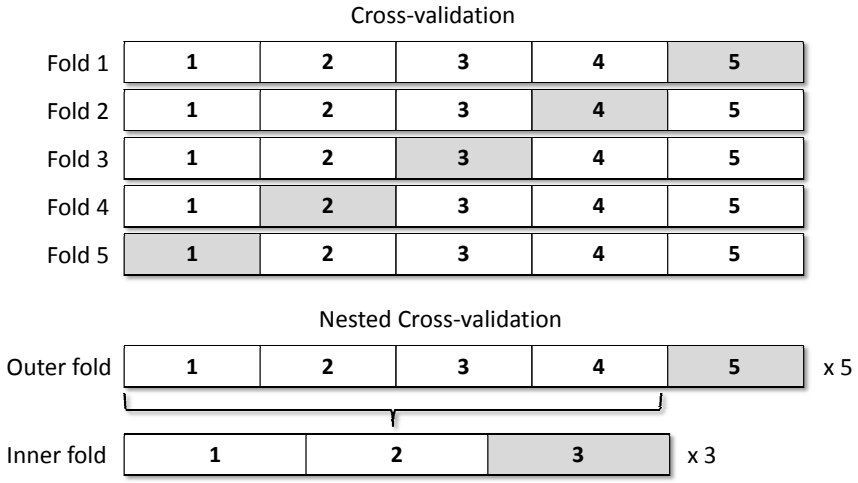

Figure 2.6: The concept of a five-fold cross-validation and five-times nested three-fold cross-validation. Each block represents the given dataset and its five subsets. The subset with the gray background is the corresponding test set of each fold. In nested cross-validation, we perform an additional inner cross-validation on the training dataset of the outer fold (white background). For this, the dataset is again divided into k subsets including one test set. In this example, we choose to set k to three, resulting in $5 \times 3 = 15$ train-test scenarios in the inner cross-validation.

quality over all folds. The settings yielding the highest prediction quality are then used for future predictions.

To evaluate the selected model, we have to apply it to a test dataset. However, independent test data is often not available. To circumvent the lack of test data, we can perform a nested cross-validation. Therefore, we run the above described model selection based on repeated cross-validations on the training data of each outer fold. The selected model is then used to predict the test set of the outer fold (see Figure 2.6). Using this procedure, we do not evaluate one distinct model, but evaluate the quality of a model obtained from model selection. Depending on the number of parameters and number of initial features, model selection and its evaluation can be a very time-consuming process.

2.1.4 Interpretability of Machine Learning Models

In the context of this work, interpretability describes whether a prediction can be explained in understandable terms. Experiments and their outcomes are often inter-

pretable. In most cases, it is known how the experiment works and why a particular outcome is obtained. In addition, by performing experimental replicates or estimating error propagation, we know whether a result can be trusted or not. Unfortunately, machine learning models and their outputs are often not interpretable. Usually, they are presented to the user as a "black box", making it impossible to understand how they work and how predictions are obtained. To understand, in detail, how a machine learning model prediction was obtained, it is often a prerequisite to understand how a model works, which requires basic or even advanced knowledge in statistics. Furthermore, the output of a machine learning model is a plain value, i.e. a class label or a response value. Additional error information is often not provided. As a consequence, computational predictions are often not trusted by biologists. Furthermore, the "black box" character of many machine learning models prevents biologists from gaining a deeper insight into the modeled problem.

Nevertheless, providing interpretable predictions using machine learning models is not impossible. We require an interpretable machine learning model to answer two important questions in an understandable fashion: Why did the model predict this particular outcome? How reliable is this prediction?

To understand why a particular prediction was made, it is necessary to realize how the individual features contributed to the outcome. In most cases, only a few features have a strong impact on the prediction, while all other features are almost irrelevant [110]. Thus, an overview of the most important features might be already sufficient to understand why a particular prediction was obtained. This obviously also requires a set of interpretable features that can be associated to a real world interpretation [175]. Discovering features that are relevant for a particular prediction might help to gain knowledge on the modeled biological problem. Hence, using a selected subset of important and interpretable features can support the interpretability of predictions. Moreover, understanding how certain features influence the output can help users to understand why a particular prediction was made, see Figure 2.7 for an example. A common scenario where the influence of individual features is of central importance is inverse engineering [75]. In inverse engineering, users are interested in modifications of the input instance that result in changes of the prediction outcome and, in turn, in an improved real world object.

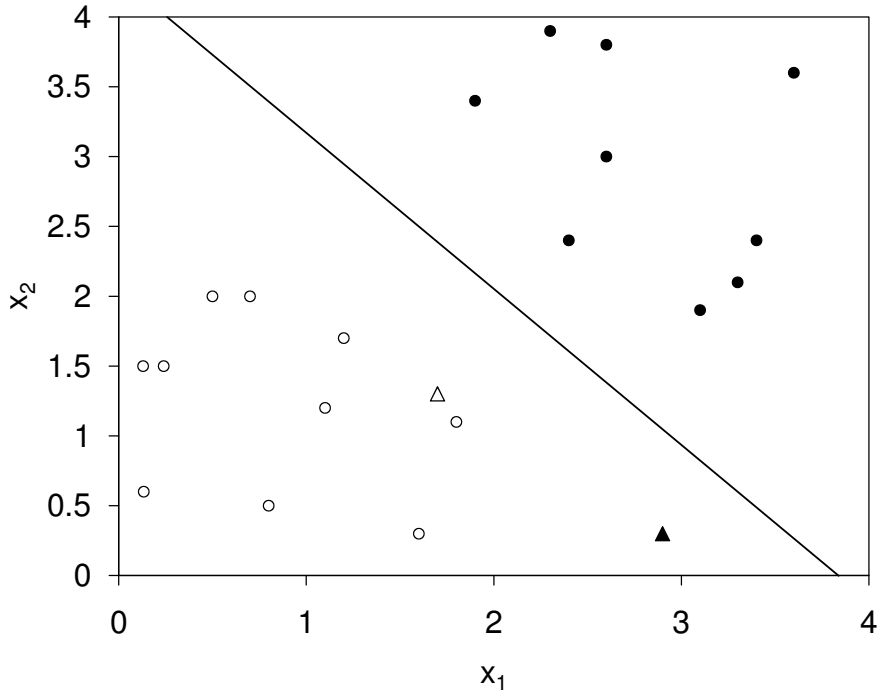

Figure 2.7: The two-dimensional input space of a binary classification problem solved by a linear classifier. Instances from the two classes of the training dataset are displayed as points in white or black. Two novel instances that we aim to classify are depicted as triangles. The first instance (white triangle) can be easily classified as a member of the white class since its features are very typical for that class. In contrast, the second instance (black triangle) exhibits a typical features for both classes, x_1 is large whereas x_2 is considerably low. Using an interpretable classification model it is possible to understand the differences of both predictions and manually reveal contradictions. In addition, the location in the input space of the second test instance is very untypical for the training dataset. In this part of the input space the classifier has to extrapolate the class distributions and, hence, gives less reliable predictions. Although the distance to the separating decision boundary (solid line) is almost the same for both test instances, they cannot be predicted with the same confidence. A confidence estimation would alert the user for unreliable predictions as in the second case.

While nonlinear machine learning algorithms perform usually better than linear ones, their predictions are often not interpretable [176]. The major reason lies in the complexity of nonlinear methods. Since the output is a nonlinear combination of the input, there is often no simple and easily understandable explanation for a particular

prediction. Hence, one has to employ heuristics to get a rough estimation of the influence of the individual features [76, 188]. If support vector algorithms such as SVMs and SVRs are used with high-dimensional kernels, it is usually impossible to determine an explicit output function. Although some special kernel allows to determine the exact contributions of individual features [173], this is not possible for most other high-dimensional kernels. As a consequence, we are required to find a trade-off between performance and interpretability. Using linear or simple nonlinear machine learning algorithms might result in a slight performance decrease but will considerably improve the interpretability of predictions [75].

Another important aspect of interpretability is the reliability of individual predictions. Computational predictions are often prone to errors. Although we estimated the model's prediction quality in a performance evaluation, we cannot assume this prediction quality for all predictions [13]. Machine learning models yield different errors for different inputs and do not perform equally well for all instances [73]. Knowing how much confidence we can put into individual predictions can increase the applicability of machine learning models. For this, the level of confidence has to be formalized into an intuitive measure.

In classification, most algorithms provide a measure of uncertainty, for example the distance to the separating hyperplane [129] of an SVM and the posteriori of naïve Bayes. While the output values indicate a preference for a class, they do not indicate how good this estimation is, see Figure 2.7 for an example. Hence, we have to make a clear distinction between the output value and a confidence score, which estimates the error of the output [17].

In regression, the output is often only a plain response value without additional confidence information [12]. A first step towards confidence estimation has been done in quantitative structure-activity relationship (QSAR) prediction. For example, properties of the input instance have been used to give a qualitative estimate whether the model can be applied to this particular instance or not [96]. However, most confidence estimation algorithms provide no quantitative measure of confidence or only provide non-interpretable confidence scores that cannot be related to an actual error [12]. A confidence interval as an estimation of a possible error would be a powerful addition to plain predicted values. Developing confidence estimation methods for individual pre-

dictions is an important step to make computational predictions more transparent and understandable for users with a biological background.

2.2 Subcellular Localization of Proteins

Eukaryotic cells are organized into membrane-enclosed subcellular compartments, also called organelles (see Figure 2.8). Each organelle is a functional subunit of the cell. Its function and its structure is closely related to the set of molecules it contains. Probably the most important macromolecules of organelles are proteins. They participate in virtually every process within a cell by fulfilling various important functions. For example, enzymes catalyze chemical reactions. Integral membrane proteins can act as pumps or channels for small molecules. Other proteins are involved in signal cascades and cell cycle control. In Section 2.3, we discuss proteins important for immune response. Moreover, proteins can have structural or mechanical functions to support movement and the stability of the cytoskeleton.

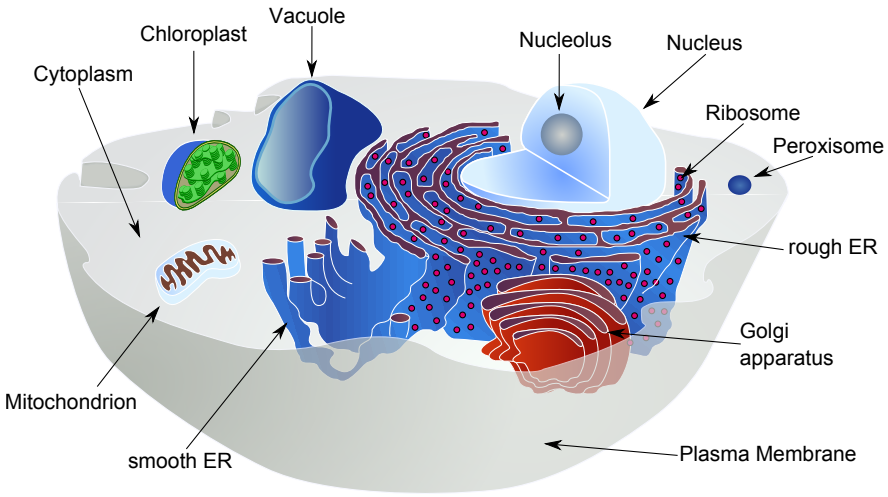

Figure 2.8: A plant cell with its most important organelles. The Figure has been adapted from Wikimedia Commons [193, 194].

On one hand, every organelle requires a distinct set of proteins to function. On the other hand, each protein needs to be located within the correct organelle to fulfill its function. In fact, an incorrectly localized protein can lead to diseases. Only in the correct subcellular compartment, proteins can find the required environmental conditions and their interaction partners. However, the subcellular compartments of an eukaryotic cell are surrounded by intracellular membranes, which are mostly impermeable to proteins. To transport proteins to their place of function, each organelle has to employ import and export mechanisms.

In the following sections, we discuss how proteins are localized to their cellular compartments. Further, we show how this process is mediated by sorting signals in the protein.

2.2.1 Protein Transport

The presence of a nucleus distinguishes a complex eukaryotic cell from simple prokaryotic cells. The nucleus contains most of the genetic information of a eukaryotic cell in form of deoxyribonucleic acid (DNA) and is responsible for synthesis of messenger ribonucleic acid (mRNA). Transcribed mRNA is transported to the surrounding cytoplasm where it is translated by ribosomes into a specific amino acid chain that later folds into a protein. Some proteins are even directly synthesized into the endoplasmic reticulum (ER). Each protein type has distinct properties defined by its unique sequence of amino acids. Each of these amino acids exhibits different physicochemical properties such as charge, hydrophobicity, and size. To fulfill their particular functions, proteins are subsequently transported to their target organelle.

We distinguish three different forms of protein transport (see Figure 2.9). The first form is the gated transport, which can be found between the nucleus and the cytoplasm. Nuclear pore complexes are selective gates that allow specific proteins to pass the nuclear envelope. Smaller molecules can enter the nucleus by free diffusion.

The second type of transport is transmembrane transport. It is based on membrane-embedded transporter proteins, also called translocators. To be transported, proteins usually have to unfold to fit through the translocator. This form of transport can be found between the cytoplasm and mitochondrion, chloroplast, peroxisome, and the ER. The mitochondrion and the chloroplast, which is present only in plants, are responsible for energy production of the cell. The peroxisome is mainly responsible for the breakdown of long-chain fatty acids by oxidative reactions. The ER produces most of the cells lipid and is a major storage of Ca^{2+}. In addition, the ER is responsible for the synthesis and modification of both soluble and transmembrane proteins of the secretory pathway.

Finally, vesicular transport by membrane-enclosed vesicles is the major transport form along the secretory pathway. Vesicles are loaded with cargo by pinching off from the membrane of the starting compartment. Transmembrane proteins are inserted into

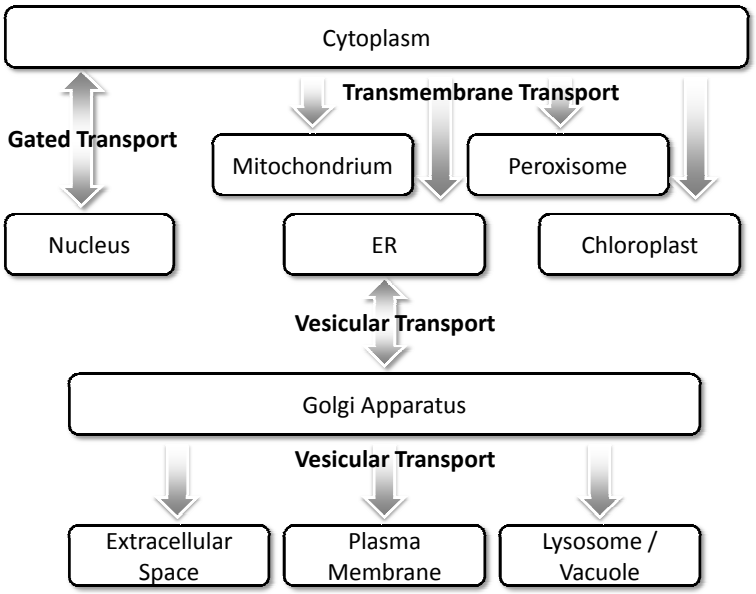

Figure 2.9: The three different forms of protein transport.

the membrane of the vesicle, whereas soluble proteins are transported inside. When fusing with the target organelle's membrane, the vesicle discharges its cargo molecules into the lumen of the organelle. The secretory pathway consist of a series of compartments that are used by the cell to secrete proteins. From the ER, proteins can be transported via vesicles to the Golgi apparatus, where they undergo covalent modification. The Golgi apparatus is then responsible for further transport until the proteins are eventually secreted, giving the pathway its name. For example, proteins for intracellular digestion, such as hydrolases, are transported via the endosome to the lysosome. In cases of plants, these proteins are transported to the vacuole, which also functions as a major storage. Furthermore, proteins in the Golgi apparatus can be transported to the plasma membrane, which separates the cytoplasm from the extracellular space. Proteins in the plasma membrane are often involved in cell signaling, for example as part of the immune system. Finally, vesicles can undergo exocytosis, in which proteins are secreted into the extracellular space.

2.2.2 Sorting Signals

All three forms of transport share one important property: they are guided by so-called sorting signals. Sorting signals are recognized by receptor proteins or directly by

translocator proteins. Each sorting signal directs the protein to a particular compartment of the cell. We distinguish between two types of sorting signals: Signal sequences are continuous subsequences of the protein's amino acid sequence. They are typically between 15 and 60 residues long. Often signal sequences are located at the ends of the protein, the N-terminus or the C-terminus. Terminal sequences are frequently cleaved from the protein once the transport has been accomplished. In contrast, signal patches are distinct three-dimensional structures at the surface of a protein. The residues involved are not necessarily adjacent to each other in the sequence of the protein. This makes them difficult to identify if no structural information is available. In most cases, the exact amino acid sequence of sorting signals is not relevant for recognition. Instead, sorting signals are defined by physicochemical properties of the amino acids such as hydrophobicity, charge, or size. See Figure 2.10 for an overview of the most important sorting signals.

Transport Between Nucleus and Cytosol

Macromolecules such as proteins and RNA are imported and exported from the nucleus via nuclear pore complexes. Before they can pass through pores of the nuclear envelope, their sorting signals need to be recognized by nuclear import receptors or nuclear export receptors.

Proteins imported to the nucleus contain a nuclear localization signal (NLS) in their amino acid sequences. One can distinguish monpartite NLSs, which are short continuous signal sequences, and bipartite NLSs, which are two signal sequences separated by a spacer of about 10 residues. Both types of NLSs are rich in positively charged lysines and arginines and can be located almost anywhere in the protein sequence. Although various NLSs have been observed, it is difficult to create a universal consensus sequence for NLSs.

To be targeted for export to the cytosol, proteins are required to have a nuclear export signal (NES). A typical NES consists of four hydrophobic residues, often leucines, each separated by 1–3 other amino acids.

In addition, there exist sorting signals that target proteins to sub-nuclear locations like the nucleolus, nuclear lamina, chromatin, and the nucleoplasm [49].

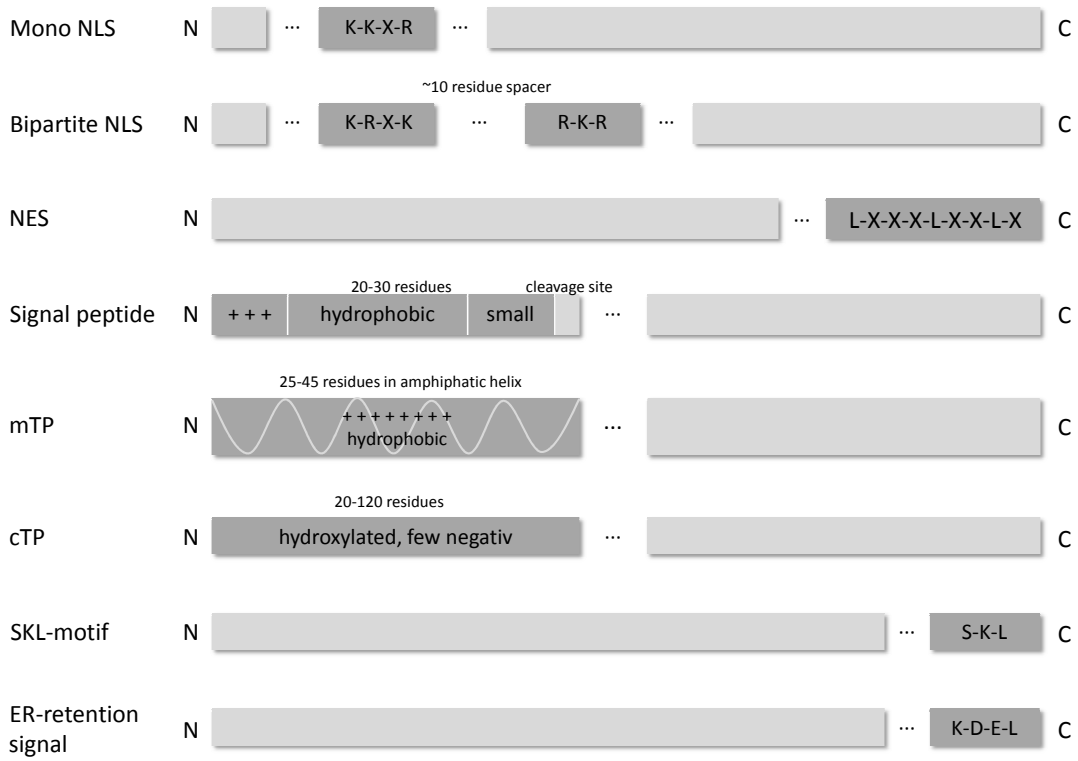

Figure 2.10: An overview of the most important sorting signals. Amino acids are shown in one letter code, while an X can be any amino acid. For the monopartite NLS and the bipartite NLS, an explicit example sequence is given [51]. The NES shown is a consensus motif obtained by La Cour *et al.* [106].

Transport into Mitochondria and Chloroplasts

Protein localization into mitochondria and chloroplasts is mediated by multi-protein complexes. These integral membrane proteins are responsible for translocating proteins from the cytosol into the lumen, the outer membrane, and the inner membrane of the organelle. Transport is initiated by recognition of a mitochondrial targeting peptide (mTP) or a chloroplast targeting peptide (cTP) by a receptor on the organelle's surface. Proteins have to unfold to be transported by the corresponding transporter proteins. After transport, the targeting peptide is removed and the protein folds into its mature form.

The mTP is a 25 to 45 amino acid long N-terminal signal sequence. It often forms an amphiphatic α-helix containing positively charged amino acids on one side and un-

charged, hydrophobic residues on the other side. Moreover, negatively charged residues are rare (see Figure 2.10).

cTPs are also N-terminal signal sequences containing only very few negatively charged residues. However, they show, with 20 to 120 residues, a wider variation in length and are rich in hydroxylated amino acids, in particular serines (see Figure 2.10).

Transport to sub-locations such as different membranes in case of mitochondria or the thylakoid lumen in case of chloroplasts have also been studied [152, 165, 167].

Transport into Peroxisomes

The import of proteins into the peroxisome is still poorly understood. It is known that more than 20 proteins, called peroxins, are responsible for translocation [22]. In contrast to mitochondrial and chloroplast import, proteins do not necessarily have to unfold for transportation. The best studied peroxisomal signal sequence consists of three amino acids, Ser-Lys-Leu, located at the very C-terminus, also known as SKL motif (see Figure 2.10). However, various other potential signal sequences have been observed, mostly located at the C-terminus or the N-terminus of proteins [20].

Intracellular Vesicular Transport in the Secretory Pathway

The transport of proteins along the secretory pathway is performed via vesicular transport. Translocation into the ER is guided by an N-terminal sorting sequence called signal peptide or secretory pathway signal [190]. It is a very hydrophobic N-terminal segment consisting of 20 to 30 amino acids with slightly positively charged residues at the very N-terminus and mostly small and polar residues at the C-terminal part of the signal peptide (see Figure 2.10). Proteins containing a signal peptide are usually synthesized by ribosomes attached to the membrane of the rough ER. The synthesized polypeptide is directly translated into the ER lumen if it contains a cleavage site or stays in the ER membrane if it contains a stop-transfer signal anchor. Imported proteins are usually automatically transported to the Golgi apparatus via transport vesicles unless they contain some ER-retention signal. This signal sequence is located at the very C-terminus and usually consists of four amino acid, KDEL in case of soluble proteins and KKXX in case of membrane proteins, where X stands for any amino acid.

From the Golgi apparatus, proteins are transported to their final destination. In the different parts of the Golgi apparatus, most proteins are covalently modified. Some of

these modifications are sorting signals for further transport, while proteins that reside in the Golgi apparatus are not modified. Special signal patches of lysosomal proteins are recognized in the cis Golgi-network leading to phosphorylation of an attached sugar molecule. The phosphorylated sugar is a sorting signal responsible for vesicular transport to the lysosome [105]. It is assumed that a similar signal patch is responsible for secreting proteins via exocytosis into the extracellular space [142]. Another sorting signal are very long hydrophobic transmembrane regions, which trigger transport to the plasma membrane. Vesicles intended for the plasma membrane are rich in cholesterol resulting in thicker membranes. Since proteins have to included in the vesicular membrane for transportation, only transmembrane proteins that span a membrane with around 20–25 amino acids can be transported to the plasma membrane.

Transport to Multiple Locations

It is assumed that more than one-third of all eukaryotic proteins are transported to multiple compartments [118, 197]. For some specialized proteins, such as messenger proteins, moving between compartments is essential for their function. There are various reasons why proteins localize to more than one organelle. For example, for some proteins, multiple isoforms exist caused by multiple transcriptional or translational initiation sites [170]. In addition, ambiguous sorting signals can guide proteins to multiple organelles. This is often the case in plants, because the cTP and the mTP show a similar overall amino acid composition [120]. Moreover, weak sorting signal that exhibit a reduced binding affinity to the corresponding receptor may cause a certain fraction of proteins to remain in the source organelle. Finally, also multiple sorting signals in the protein sequence can lead to multiple targeting of proteins [99, 150].

2.2.3 Determining Subcellular Localization

There exist various experimental approaches to determine the subcellular location of a protein. A key technique is fluorescence microscopy, where fusion proteins of the protein of interest and a fluorescent protein are constructed. An fluorescence microscope is then able to detect light emitted by the fluorescent fusion protein after excitation of the fluorophor, making it possible to determine the location of the protein within the cell. Other prominent experimental approaches are cell fractionation [29], immunolocalization [169], and western blot [86]. Usually, experimental approaches are very accurate

in determining the location of a protein. However, they are very time-consuming and can be expensive.

Predicting the subcellular location of a protein from its primary sequence is an attractive alternative to labor-intensive experiments. Since this is a typical classification problem, many machine learning approaches have been developed for this task.

2.3 The Immune System

Immunity, the resistance to an infection or a disease, was first observed in ancient Greece. More than 2,000 years later, in 1796, Edward Jenner was the first to develop a vaccine. In the 1880s, Louis Pasteur extended Jenner's findings by successfully developing vaccines against cholera in chicken, rabies, and anthrax based on weakened forms of the corresponding pathogens. His discoveries revolutionized medicine and led to a new research discipline, called immunology. Since then, immunology has made numerous important advances that uncovered the principles of the immune system leading to novel approaches in immunotherapy.

The immune system is a biological system that protects and defends an organism against pathogenic microorganisms and tumor cells. It is not only able to recognize invading pathogens such as bacteria, fungi, and viruses, but also to fight possible infections by an immune response. Due to its ability to adapt, the immune system can provide better protection against future infections. Recognition and response involve a variety of specialized cells that act together in a complex network.

The immune system can be divided into the innate immune system and the adaptive immune system. The following sections introduce today's knowledge of the immune system and its relationship to modern immunotherapy.

2.3.1 The Innate Immune System

The innate immune system is the less specific part of the immune system. It provides natural resistance that is present from birth, called innate immunity. It comprises cells and mechanisms that are able to detect molecules not specific to a particular pathogen but molecules common to many pathogens.

Anatomic and physiological barriers, such as skin and pH, are the first frontier for invading microorganisms. When pathogens succeed in entering an organism, specialized immune cells are able to provide a second barrier of defense. For example, in a process called phagocytosis, macrophages are able to recognize, engulf, and digest pathogens. Further, chemical mediators such as cytokines can trigger an inflammatory response in which vascular fluid is released in the affected region and other cells involved in an immune response, such as neutrophils, blood monocytes, and macrophages, are attracted.

In a later step of the inflammatory response, also lymphocytes of the adaptive immune system can be summoned.

2.3.2 The Adaptive Immune System

The adaptive immune system is the more specific component of the immune system. In contrast to the innate immune system, it is not uniform in all members of a species. Its highly specialized immune cells yield no broad reactivity, but are able to recognize, eliminate, and remember specific pathogens. Due to its ability to create a huge diversity in its recognition molecules, the adaptive immune system is able to recognize a large number of different structures. At the same time it does not recognize structures of the host as foreign. The other important feature of the adaptive immune system is its ability to memorize previously seen foreign substances. The immunological memory enables a stronger and faster immune response if the pathogen is encountered a second time. The ability to adapt to invading pathogens makes the adaptive immune system so valuable and effective.

An adaptive immune response involves two types of lymphocytes, B lymphocytes (B cells) and T lymphocytes (T cells), which are both a special type of white blood cells produced in the bone marrow. Lymphocytes are able to recognize foreign substances and molecules, often refered to as antigens, by expressing a diverse set of antigen-binding receptors on their cell surface.

B cells can directly recognize antigens in their native form. An immune response mediated by B cells is called humoral immune response since immunity is provided by secreted antibodies in the body fluids (Latin: humor). After activation, B cells differentiate into plasma cells and memory B cells (see Figure 2.11).

In contrast, T cells can recognize immunogenic peptides bound to major histocompatibility (MHC) molecules by a so-called T-cell receptor (TCR). Depending on the expression of co-receptor CD4 or co-receptor CD8 on their surface, we distinguish between T helper (Th) cells and T cytotoxic (Tc) cells, respectively. Th cells can recognize peptides bound to MHC class II (MHC-II) molecules and release effector molecules, called cytokines, which can activate other immune cells. In contrast, Tc cells recognize peptides bound to MHC class I (MHC-I–peptide) and can induce apoptosis of the infected cell. The corresponding immune response is also called cellular immune response since immunization is associated with cells (see Figure 2.11). If T cells are activated by

binding an MHC–peptide complex, they rapidly duplicate and differentiate into effector cells. After the infection, a certain number of effector T cells persist as memory cells, supporting an effective response to future infections.

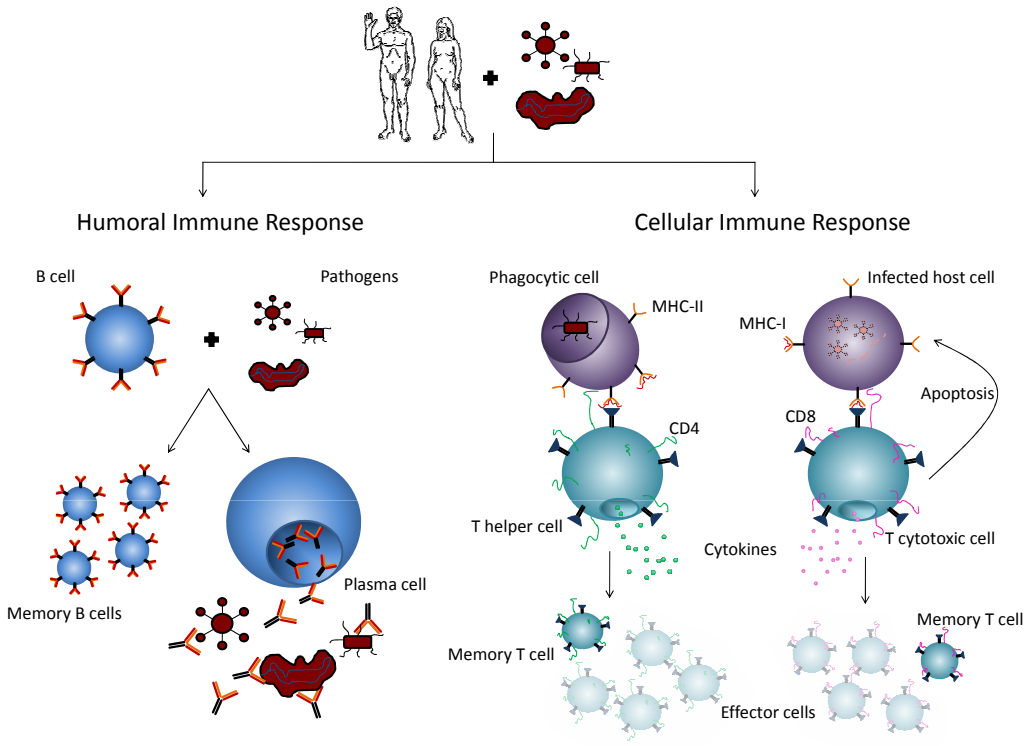

Figure 2.11: An overview of an adaptive immune response. On the left, a humoral immune response mediated by B cells is shown. On the right, a cellular immune response mediated by T helper cells and cytotoxic T cells is shown. Both types of lymphocytes, B cells and T cells, can exhibit immunological memory by retaining memory cells.

The Major Histocompatibility Complex

The major histocompatibility complex (MHC) is a highly polymorphic gene cluster that encodes for MHC molecules. In human, MHC molecules are also called human leukocyte antigens (HLAs).

Although they share many structural features, they fulfill different functions. MHC class I (MHC-I) molecules present peptides on the surface of almost all nucleated cells, including peptides from pathogenic proteins. Peptides from degraded proteins can bind

to MHC-I molecules in the ER. From there, bound peptides are transported to the cell surface. By displaying selected peptides on the cell surface, MHC-I molecules present a "fingerprint" of the cells proteome to the immune system. In contrast, MHC-II molecules can be found only on antigen-presenting cells. They bind protein fragments derived from previously endocytosed material. Bound peptides are then transported to the cell surface and presented to the immune system.

MHC-I and MHC-II molecules exhibit one major structural difference in their binding cleft. While the binding cleft of MHC-I molecules is closed at its ends, it is open in case of MHC-II molecules. Consequently, MHC-I molecules can bind only peptides of a well-defined length, usually 8 to 12 amino acids, whereas MHC-II molecules can also bind peptide consisting of more than twenty amino acids [30]. See Figure 2.12 for an example binding pocket of an MHC-I molecule.

Every type of MHC-I and MHC-II molecule can bind only a certain set of peptides. Thus, each MHC molecule is only able to present a limited spectrum of pathogenic peptides to the immune system. However, since every human expresses up to six different MHC-I types and up to 12 different types of MHC-II molecules, the adaptive immune system is able to recognize an enormous range of pathogenic structures.

The binding of peptides to MHC molecules can be analyzed with various experimental approaches such as competitive binding assays, mass spectrometry, Edman degradation, and X-ray crystallography [143]. In competitive binding assays, it is measured which concentration of a radiolabeled or fluorescence-labeled peptides is necessary to inhibit the binding of a reference peptide. The actual binding affinity is often given as the half-maximal inhibitory concentration (IC_{50}), i.e. the concentration of peptide required to displace half of the reference peptide. Note that the $\log(IC_{50})$ of a peptide is proportional to its binding free energy ΔG. Peptides that bind to an MHC molecule yield an IC_{50} value lower than 500 nM. Peptides yielding an IC_{50} value even lower than 50 nM are often called strong binders. Experiments in a laboratory can help to understand how peptides bind to MHC and which peptides have the ability to bind. Unfortunately, a large part of the binding spectrum of MHC molecules is still unknown since experimental approaches are expensive and time-consuming. As an attractive alternative to experimental approaches, computational MHC–peptide binding prediction methods have been introduced [54, 74, 116, 132, 187]. Current state-of-the-art approaches are based on multi-layer neural networks [128] or kernel-based SVMs [95].

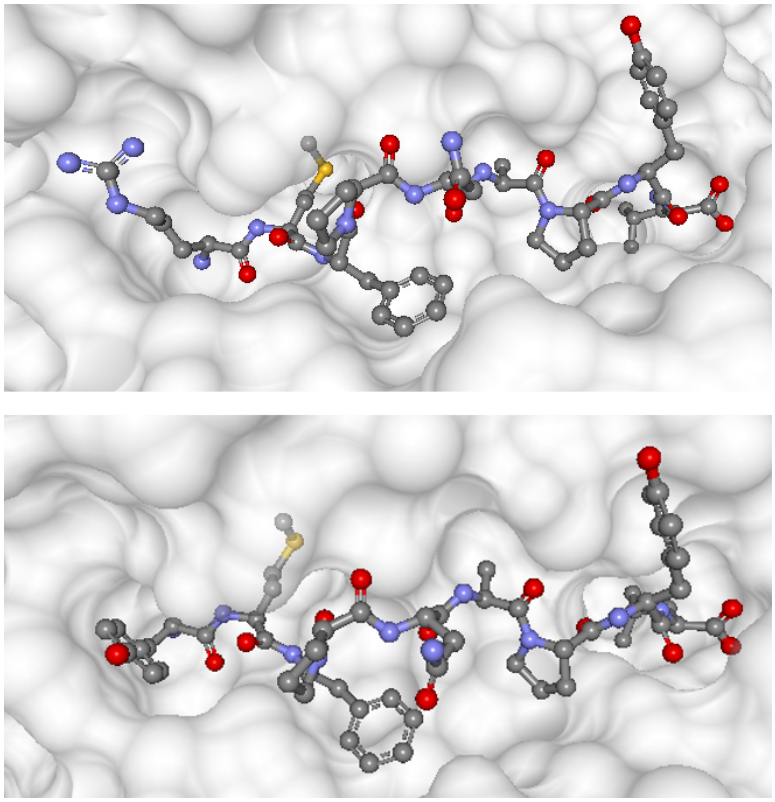

Figure 2.12: The binding pocket of MHC-I HLA-A*02:01 molecules bound to two different peptides. In the upper image, the MHC-I molecule is bound to peptide RMFPNAPYL of the Wilms Tumor 1 transcription factor. The lower image shows a MHC-I molecule bound to the heteroclitic peptide YMFPNAPYL, a variant of the above peptide that exhibits a stronger binding affinity than the wild type peptide [10]. This figure has been created with BALLView [121] and structural data from Protein Data Bank (PDB) [5] entries 3MYJ and 3HPJ.

While yielding a good prediction performance, their predictions are not interpretable. It is not possible to understand why a certain affinity was predicted and, most importantly, how reliable the obtained affinity is.

2.3.3 Epitope-Based Vaccines

The ability of the immune system to memorize encountered antigens provides the foundation for developing vaccines. A traditional vaccine often contains a weakened or killed form of the pathogen. Although the patient is not getting sick, the invader is recognized and "remembered" by the immune system. This form of vaccination has often been

applied in medicine even before the details of the underlying immune mechanisms were known.

To develop effective vaccines, more rational approaches such as epitope-based vaccines have attracted attention [159]. Epitopes are peptides that can trigger an immune response. Vaccines based on epitopes consist of peptides from pathogens that are recognized and, consequently, remembered by the immune system. Injecting only epitopes instead of the whole pathogen makes epitope-based vaccines safe, easy to produce, and cheap. However, discovering potential epitopes for a given pathogen is difficult since it requires a lot of experimental effort. Moreover, since every human expresses different types of MHC-I and MHC-II molecules, we are even required to find a set of epitopes that covers large parts of a population [185]. See Figure 2.13 for an overview of the design process of an epitope-based vaccine.

Figure 2.13: Design process of an epitope-based vaccine. Given a set of antigens, candidate epitopes are determined based on a given target population. Suitable epitopes candidates, e.g. epitopes with a strong T-cell reactivity approximated by their MHC–peptide affinity, are selected. In a final step, the epitope-based vaccine is assembled by combining the selected epitopes into a polypeptide. This figure and its caption were adapted from Toussaint and Kohlbacher [184].

To discover potential epitopes, computational MHC–peptide binding predictions are often used as an approximation of T-cell reactivity [184]. Furthermore, *in silico* methods can be used to find peptides that are similar to known epitopes but induce a strong or even stronger immune response than the wild type epitope [6, 90]. These peptides are called heteroclitic peptides. The idea of epitope engineering results from the fact that a stronger immune response usually leads to a better immunization. The drawback of using computational models for epitope discovery and epitope engineering lies in the "black box" character of most MHC–peptide binding predictors. Prediction methods cannot guide epitope engineering since it is not evident, which amino acid has the strongest impact on the binding affinity nor whether a specific amino acids prevents

the peptide from binding. Most importantly, the reliability of individual predictions cannot be obtained. Since *in silico* prediction methods are prone to errors, we might rely on low confidence predictions. In particular for important tasks like vaccine design, considering only reliable predictions is vital.

Chapter 3

YLoc – An Interpretable Classification Approach

Subcellular protein localization is a key process in most eukaryotic cells. The location of a protein within the cell is highly correlated with its function and is, thus, often used to draw conclusions about its cellular role, interaction partners, and function in biological processes. During the last decade a huge number of novel proteins were discovered in the context of large-scale sequencing projects. Unfortunately, for the majority of these proteins their function and subcellular localization is unknown. Experimentally determining the localization of a protein is expensive and time-consuming. Computational classification approaches that predict subcellular localization from the amino acid sequence represent an attractive alternative to experimental methods. However, their predictions are often not interpretable. This chapter shows how to overcome the "black box" character of computational subcellular localization prediction. The classification approach we introduce, YLoc, is able to give interpretable predictions while being as accurate as state-of-the-art methods. In addition, it rates the confidence of individual predictions, making it a powerful tool for biologists. Parts of this chapter have been previously published in Briesemeister *et al.* [17] and Briesemeister *et al.* [18].

3.1 Introduction

In 1992, Nakai and Kanehisa [125] laid the foundation for computational subcellular localization prediction by introducing PSORT. It is a simple rule-based prediction ap-

proach that uses basic knowledge on protein sorting signals. At this time, the authors could exploit only a small set of protein sequences annotated with their subcellular location. Since then, more and more annotated sequence data has become available. In addition, more complex classification models such as SVMs [8, 124, 139] and ANNs [62] as well as combinations of multiple models [33, 37] have been applied.

Over the last few years, numerous prediction methods have been introduced. We distinguish between sequence-based and annotation-based methods. Sequence-based predictors make use of sequence-coded sorting signals [3, 9, 40, 62, 69, 138, 171], amino acid composition information [28, 34, 45, 77, 91, 103, 124, 131, 139, 146, 195], or even both type of information [72, 83, 88]. Annotation-based predictors use information about functional domains and motifs [32, 157], protein-protein interaction [107, 166], homologous proteins [71, 111], annotated Gene Ontology (GO) terms [80, 92, 108, 112], Swiss-Prot keywords [113, 122], or textual information from PubMed abstracts [14, 70]. Since proteins with sufficiently similar protein sequences are usually located in the same compartment [123], missing annotation information might also be transferred from close homologues. Annotation-based predictors often show higher accuracies than predictors based on sequence alone, however, they are less reliable for novel proteins without known close homologues. Hybrid prediction approaches take advantage of both types of information [8, 16, 33, 35, 37, 158]. Furthermore, there exist a range of highly specialized prediction methods that can be applied only to a selected type of protein [36, 140], expressed sequence tags [163], or predict only sub-locations of one particular location [39]. See recent reviews for more details on subcellular localization prediction [94, 145]. In Table 3.1, we give an overview of selected subcellular localization predictors.

Although there is evidence that more than one third of all eukaryotic proteins are transported to multiple compartments [197], multiple targeting of proteins has only rarely been considered by prediction methods. As one of the first groups, Scott *et al.* [157] introduced a method for multiple localization prediction based on about 500 multiple localized proteins. More recent predictors such as WoLF PSORT [88], Euk-mPloc [37], Euk-mPloc2 [35], iLoc-Euk [38], ngLoc [103], and KnowPred [111] use even up to 2,200 multiply targeted proteins as knowledge base for their predictions. Prediction methods that are specialized on the dual targeting into particular organelles

have been studied only sparsely [120]. Although there has been recent development on multiple localization prediction, we believe that there is still room for improvement.

Method	# locs	model	# features	annot.	homology	multi.
MultiLoc2	11	$>$ 10 SVMs	$>$ 1,000	yes	yes	no
WoLF PSORT	12	kNN	$\approx$ 30	yes	no	yes
Euk-mPloc	22	$>$ 300 kNNs	$>$ 1,000	no	yes	yes
KnowPred	10	PSI-BLAST	$>$ 1,000	no	yes	yes
BaCelLo	5	4 SVMs	280	no	no	no
LOCTREE	6	5 SVMs	$\approx$ 100	no	no	no
PSORT	4	rules	80	yes	no	no
TargetP	4	ANN	$>$ 600	no	no	no

Table 3.1: Overview of selected subcellular localization prediction methods. We show some basic properties of MultiLoc2 [8], WoLF PSORT [88], Euk-mPloc [37], KnowPred [111], BaCelLo [139], LOCTREE [124], PSORT [125], and TargetP [62]. For each subcellular localization predictor, we show the number of locations it predicts (# locs), the classification model used, the number of features used (# features), whether annotation information, such as domains or sorting signals, is used (annot.), whether features contain information of possible homologous proteins (homology), and whether multiple localization sites can be predicted (multi.). Note that for some predictors, the exact number of features depends on the version of the predictor.

The prediction performance of subcellular localization predictors has significantly improved over the years. Unfortunately, the machine learning models behind state-of-the-art predictors are often very complex, making it difficult to understand why a particular prediction was made. For example, MultiLoc2 consist of two layers, each containing an ensemble of SVMs in combination with an RBF kernel [8]. It returns a probability estimate for each location, but no additional information that could be helpful to understand why this outcome was obtained. Another popular state-of-the-art predictor is WoLF PSORT [88]. It uses a weighted kNN classification algorithm, which predicts the subcellular location by transferring the location of proteins with a similar feature encoding. Although this approach is not very complex, it is not obvious how a biological property influenced the prediction. Instead of giving a biologically meaningful explanation, predictions by WoLF PSORT are only supported by similarity information. The widely used Euk-mPloc uses a complex combination of kNNs

and is, hence, only able to return a plain list of locations [37]. Another drawback of current subcellular localization predictors is the absence of confidence estimates for individual predictions. Consequently, predictions cannot be verified with regard to their significance and reliability.

The "black box" character of current state-of-the-art prediction methods has a major impact on the credibility of *in silico* subcellular localization predictions. It is not evident how a prediction was obtained, nor which biological property influenced the algorithm towards this conclusion. Consequently, results are usually treated with great caution. In addition, it is often not possible to gain a deeper knowledge of the actual localization process. In particular when further experiments are planed, biologists could greatly benefit from knowledge like the position of sorting signals within the protein. Furthermore, a confidence estimation that helps to rate the reliability of individual predictions is vital if users rely on a small error rate. If future experiments are very expensive and time-consuming, biologists are interested in how much confidence they can put in a prediction.

In this chapter, we present YLoc, an interpretable method for predicting the subcellular localization of proteins. YLoc is based on the simple naïve Bayes classifier. It combines various feature types for its predictions ranging from simple amino acid composition to annotation information like PROSITE domains and GO terms from close homologues. Most importantly, it uses at most 30 of these features. The small number of features as well as the simple architecture guarantee interpretable predictions. YLoc is able to elucidate why a prediction was made and what attributes of the protein contributed most to this prediction. In addition, it returns confidence scores that rate predictions as reliable or not. YLoc is available in three versions. The low-resolution version, YLoc-LowRes, is specialized in distinguishing the localization of globular proteins and predicts up to five locations. The high-resolution version, YLoc-HighRes, covers 11 main eukaryotic subcellular locations. YLoc$^+$ is the most general predictor. It covers 11 main eukaryotic locations while integrating multiple localization sites. All three predictors are available for animal, fungal, and plant proteins.

We compared YLoc against other state-of-the-art protein subcellular localization predictors using two recently published independent datasets [8, 27]. The results confirm that YLoc, even though its architecture is very simple, performs comparable to current state-of-the-art predictors. To show that YLoc can benefit from confidence

information, we tested it with five different confidence estimation approaches. Since our newly introduced probabilistic confidence estimator performs best, we included it into YLoc. We found that confidence estimation results in a considerable enrichment of correct predictions. Hence, for instances predicted with high confidence, YLoc yields an even better prediction performance than state-of-the-art predictors. For proteins with multiple localizations, YLoc shows an outstanding accuracy compared to existing methods. In an example study, we show that YLoc prediction outputs can be easily interpreted making it possible to detect sorting signal relevant for protein localization. Moreover, we illustrate that YLoc can be applied to explain localization changes of proteins that are caused by mutations in the protein sequence.

3.2 Methods

3.2.1 Features

In the past, various types of feature encodings were studied in the context of subcellular localization. They range from sequence information like annotated sorting signals to knowledge inferred from homologous proteins. However, in many cases predictions methods employ only one or two types of feature encodings. In our study, we included numerous types of features and properties to benefit from their different characteristics and information.

First, we make use of sequence-derived features. These include amino acid composition, normalized amino acid composition, and pseudo amino acid composition [31]. In addition to counting simple amino acids, we use the compositions of certain amino acid types such as hydrophobic, positively charged, negatively charged, aromatic, large, and small. To encode for patterns in the protein sequence, we calculate sum and autocorrelation of properties like hydrophobicity, charge, and volume of the amino acids based on information from AAindex [101]. While the sum encodes for stretches of amino acids with similar properties, the autocorrelation measures the correlation of a signal with itself and can be used to identify periodic patterns. All features are calculated over the whole sequence length, as well as for subsequences of various lengths in the N-terminus (10 to 200), C-terminus (10 to 100), and middle part of the protein. In all cases, we omit the first residue to avoid a bias caused by methionine. In addition, various known

sorting signals such as mono NLS, bipartite NLS, NES, peroxisomal targeting signal, mTP, cTP, secretory pathways signal, and ER retention signal are considered.

Second, we make use of annotation-based features such as PROSITE patterns [93]. PROSITE patterns describe protein domains, families, as well as functional sites. A PROSITE pattern feature is assigned a value of one if the pattern is found in the protein sequence using PROSITE scan [47]. In addition, we create a feature for each location that is described by PROSITE patterns that are typical for this location. For example, DNA-binding domains are typical for nuclear proteins while certain receptors can only be found in the plasma membrane. We define a PROSITE pattern to be typical for a location if more than 80% of all proteins in the training dataset containing this pattern are present in this particular location. By manual inspection we found that a lower threshold results in false assignments while a larger threshold results in a very small number of typical patterns. The resulting feature is assigned a value of one if at least one typical PROSITE pattern of this location is present in the protein or zero otherwise.

Finally, we use GO terms [80] from close homologues from Swiss-Prot release 42.0. To find only proteins with well-conserved homologous regions, we require a homologous proteins to align with the query sequence with a very low E-value ($\leq 10^{-10}$) according to BLAST [1]. To ensure an even better match quality, we require a sequence identity of more than 30% for the aligned region. Using these alignment conditions, we are able to find proteins that very likely share a domain with the query protein. Since homologous proteins are often located within the same organelle, we can transfer the corresponding GO term information [123]. A GO term feature equals one if at least one homologous protein is annotated with that GO term. In addition, we grouped GO terms that are typical for a particular location. In this context, we define a GO term to be typical for a location if more than 95% of all proteins containing this GO term are located there. We found this threshold by manual inspection. We observed that using lower thresholds, GO terms are likely to be assigned as typical for a location even though they are not. This misclassification is likely due to the fact that GO terms naturally contain more noise since they were inferred from sequences which do not necessarily have to be orthologues, or even homologues. An additional feature indicates the location for which the most typical GO terms could be transferred.

The overall number of features considered for subsequent feature selection is about 30,000. Before selecting the features for our final YLoc predictor, we use the entropy-based supervised discretization of Fayyad and Irani [65] to discretize our features.

3.2.2 Feature Selection

The created features encode for a whole range of different characteristics that are important for subcellular protein localization. However, due to the large number of features, some of that information is likely to be redundant. Because of the limited number of learning examples available, learning with a small number of features often leads to a better generalization of machine learning algorithms (Occam's razor). Moreover, creating an understandable and manually readable prediction output based on all features is only possible if the number of features is rather small. On the other hand, the number of features should not be too small. If the selected features are not sufficient to cover all aspects of the localization process, the resulting classification model might show a reduced prediction performance. Hence, we aim at selecting as few features as possible without loosing the generalization ability of our model.

To find the set of the most important features, we started a large-scale feature selection using a correlation-based feature selection (CFS) approach [79]. This approach favors a feature set that shows high correlation with the class variable but low redundancy among the features in the set. In CFS, the quality of a feature subset I of size k is defined by

$$\frac{\sum_{i \in I} r_i}{\sqrt{k + \sum_{i,j \in I} r_{ij}}}, \tag{3.1}$$

where r_i is the correlation between feature i and the class variable, and r_{ij} is the correlation between two features i, j in the subset. Since we discretized our features, we cannot make use of Pearson's correlation coefficient. Instead, we use the information gain [141], defined by

$$\text{gain} = H(Y) - H(Y|X), \tag{3.2}$$

where $H(Y)$ equals the entropy of variable Y and $H(Y|X)$ is the entropy of variable Y after observing variable X. The information gain expresses how much additional information about a variable Y is provided by variable X [78]. Since we require a

symmetric measure, we have to normalize the information gain by the entropy of both features, giving us the symmetric uncertainty coefficient:

$$\frac{2 \times \text{gain}}{H(Y) + H(X)}. \tag{3.3}$$

The symmetric uncertainty coefficient is used to calculate the correlation of two discrete features r_{ij} or a discrete feature and the class variable r_i. Note that large feature subsets can be avoided by defining the subset quality to be zero if the feature set size k exceeds some threshold.

To select a feature subset, we use a backward best-first search strategy, which starts with the full feature set and greedily deletes the feature that results in the best subset quality according to CFS. Although a forward search is faster than our backward approach, we found a backward search to result in a feature set of higher quality (data not shown). The search continually caches the best 100 subsets and allows for 50 backtracking steps. CFS as well as the search algorithm are implemented in the Weka machine learning library [192].

Since we could not observe a significant improvement in a nested cross-validation of our method for more than 30 features (data not shown), we decided that 30 features are sufficient for our predictors. Due to the reduced number of locations in the low-resolution models, even 20 features are sufficient for YLoc-LowRes. The average running time of a feature selection on datasets with about 6,000 data points and 30,000 initial features was about two hours.

3.2.3 Naïve Bayes Classification

YLoc uses naïve Bayes, a probabilistic classification algorithm, introduced in Section 2.1.1. Since the naïve Bayes approach assumes features to be independent, it allows a straightforward decomposition of a prediction into the individual contributions of each feature.

Given a set of features $F = \{F_1, \ldots, F_k\}$, a set of locations $L = \{L_1, \ldots, L_m\}$, and a set of corresponding classes $C = \{C_{L_1}, \ldots, C_{L_m}\}$, naïve Bayes estimates the posterior probability by:

$$P(C_{L_j}|F) \propto P(C_{L_j}) \prod_{h=1}^{k} P(F_h|C_{L_j}). \tag{3.4}$$

The class priors and the feature probability distributions are estimated using the training data after discretization. The final probabilities are obtained by normalizing the posterior probabilities such that the sum of all posterior probabilities is one.

Since features are treated independently, we can easily assess the influence of a single feature F_h on the prediction. The probability of observing feature F_h ranges from $\min_j P(F_h|C_j)$ to $\max_j P(F_h|C_j)$ over the given classes C_j. Let $C_{max} = \arg\max_{C_j} P(C_j|F)$ be the predicted class. We define

$$\log \frac{P(F_h|C_{max})}{\min_j P(F_h|C_j)} \quad \text{and} \quad \log \frac{P(F_h|C_{max})}{\max_j P(F_h|C_j)} \tag{3.5}$$

to be the support and the opposition score, respectively. A large support score originates from a high probability for the observed feature value in the predicted class, compared to the class where this feature value is least likely. Thus, a large support score indicates that the observed feature value is very typical for the predicted class. In contrast, the opposition score is always a negative value or zero. If the observed feature value is more typical for some other class than the predicted one, the opposition score is negative. Hence, a prediction based on the feature alone would lead to a different decision than using all features. On the other hand, if a feature value is most likely to be observed the predicted class, the opposition score is zero, while the support score is positive. We merge both values in the discrimination score. If the support for C_{max} is stronger than the opposition, that is the sum of the scores is larger than zero, the discrimination score equals the support score and vice versa. We use the absolute value of the discrimination score to order the features according to their influence on the prediction.

To predict multiple localizations with YLoc$^+$, we transform our multi-label data into single-label data. For proteins labeled with multiple locations L_1 and L_2, we create a new class, $C_{L_1 \wedge L_2}$. When inferring predictions, the probability output of the naïve Bayes classifier is transformed as follows:

$$P(L_j|F) = \sum_{\{C_x | C_x \in C \wedge L_j \in \alpha(C_x)\}} P(C_x|F) \frac{1}{|\alpha(C_x)|}, \tag{3.6}$$

where $\alpha(C_x)$ is the set of labels of class C_x. This transformation is based on the assumption that proteins present in multiple locations are equally distributed between

the compartments. Obviously, this does not hold for all proteins with multiple localizations. However, given only qualitative data, this is the best assumption we can make. In order to report only relevant locations, YLoc employs a simple heuristic. After sorting the locations by probability, YLoc reports the locations with a probability better than chance, that is $P(L_j|F) > 1/|L|$, where L is the set of locations. To report only relevant locations with reasonable probability, YLoc stops reporting locations if a location is less than half as probable as the preceding location. Transforming the probabilities as above yields the advantage that label combinations not present in the training data can also be predicted.

3.2.4 Confidence Estimators

Providing users with an estimate of how reliable a prediction is can substantially improve the interpretability of YLoc's predictions. We tested YLoc with different confidence estimators, which rate the confidence in a prediction by a numerical value, called confidence score *cs*. Thus, predictions with a large confidence score are required to be more likely to be correct than predictions with a low confidence score. To obtain an interpretable score, we normalize scores to a range between zero and one. In the following, we call such a score *normalized confidence score ncs*. Confidence estimators can be distinguished between model-based confidence estimators and dataset-based confidence estimators. The former use distinct properties of the classification model to estimate the confidence in a prediction, whereas dataset-based estimators make use of the structure of the underlying training data. In the following, we present existing dataset-based confidence estimators used in this work and, further, introduce two novel confidence estimators.

Existing Dataset-based Estimation Approaches

Dataset-based confidence estimators do not rely on a particular model and, hence, are more universal. As a drawback, model-independent estimators cannot profit from the insights of a model and might be less effective for estimating confidences. In Chapter 4, we show how dataset-based confidence estimators can be applied with different kinds of regression models. Dataset-based confidence estimation approaches have been previously introduced as applicability domain (AD) estimators in the context of

QSAR [50, 60, 164]. They usually assume that a model can give more reliable predictions if a novel instances is located in a part of the input space that is also populated in the training set. Hence, if many similar instances exist in the training data, we assume that we can put more confidence in a particular prediction. In the following, we present three often applied AD domain estimators.

A popular estimator is based on the number of nearest neighbors [164]:

$$cs_{\text{NoNN}}(x) = \frac{1}{n_{max}(\alpha)} |\{(x_i, y_i) | (x_i, y_i) \in D, d(x_i, x) \leq \alpha\}|, \tag{3.7}$$

where α is a distance threshold and $n_{max}(\alpha)$ defines the maximum number of neighbors within a distance of α in the training dataset. If x has many neighbors in the input space, we assume a better generalization power of the model for this subspace resulting in a larger confidence score.

Another previously introduced AD estimator is based on the average Euclidean distance of a novel instance x to instances in the training dataset [60, 96]:

$$cs_{\text{AvgDist}}(x) = 1 - \frac{1}{d_{max}} \frac{\sum_{(x_i, y_i) \in D} d(x_i, x)}{|D|}, \tag{3.8}$$

where d is the Euclidean distance function and d_{max} is the maximum distance of instances within the training dataset. If a novel instance is so distant to the training set that we obtain a negative score, we assign a confidence score of zero.

Dimitrov *et al.* [50] introduced a confidence estimator based on the number of misclassified instances in the neighborhood of a novel instance. It assumes that a high accuracy in the neighborhood of a novel instance results in a highly reliable prediction. Hence, the accuracy confidence score cs_{Acc} is calculated as

$$cs_{\text{Acc}}(x) = \frac{1}{D'} |\{(x_i, y_i) | (x_i, y_i) \in D', \hat{y}_i = y_i\}|, \tag{3.9}$$

where D' is the set of the δ nearest neighbors of x and vector $\hat{y}$ contains the corresponding predicted labels. The latter can be obtained by performing a cross-validation on the training data. Scores are normalized to $[0, 1]$ by definition.

A Novel Dataset-based Estimation Approaches

We introduce a novel dataset-based confidence estimator based on the entropy of labels in the neighborhood of a novel instance. We assume that ambiguous regions in the

input space exhibit a high label entropy and are harder to predict, resulting in a lower prediction accuracy. The label entropy score cs_{LE} in the neighborhood of an instance is calculated as

$$cs_{\mathrm{LE}}(x) = 1 + \sum_{j}^{m} \left(p_{D'}(C_j) + \epsilon\right) \log_m \left(p_{D'}(C_j) + \epsilon\right), \tag{3.10}$$

where $p_{D'}(C_j)$ is the fraction of instances of class C_j in set D', which contains the δ nearest neighbors of x. By using a logarithm to base m, the number of classes, we receive normalized confidence scores between zero and one. To avoid zero probabilities, we add a small pseudo count of $\epsilon = 0.001$.

In Chapter 4, we generalize confidence estimation based on label entropy and neighborhood accuracy to regression problems, leading to confidence estimators CONFIVE and CONFINE.

A Novel Probabilistic Confidence Estimator

To account for the characteristics of the naïve Bayes classifier, we introduce a novel model-based confidence estimation approach. It uses the probability estimates of the underlying naïve Bayes classifier to measure the reliability of an individual prediction. The estimate is based on the fact that proteins can be predicted more reliably if the corresponding feature vector is typical for the predicted classes and less typical for any other class. Given the feature vector F of a novel protein, we calculate $P(F|\bigcup_{C_j \in C} C_j)$, the probability of observing F given our training dataset, by

$$P(F|\bigcup_{C_j \in C} C_j) = P(\bigcup_{C_j \in C} C_j) \prod_{h=1}^{k} \sum_{j=1}^{m} P(C_j) P(F_h|C_j), \tag{3.11}$$

where $P(\bigcup_{C_j \in C} C_j)$ equals one. On the other hand, we calculate $P(F|C_{max})$, the probability of F given the most probable class C_{max}. Since F should be more typical for the predicted class C_{max} than for the set of all proteins, $P(F|C_{max})$ should be greater than $P(F|\bigcup_{C_j \in C} C_j)$, the baseline probability of observing F. For our final confidence score, we calculate the fraction of both probabilities and additionally weight classes containing few training examples as less reliable by multiplying the class probability $P(C_{max})$. The final confidence score is calculated as follows:

$$cs_{\mathrm{NBconf}}(F) = \frac{P(C_{max})P(F|C_{max})}{P(C_{max})P(F|C_{max}) + P(F|\bigcup_{C_j \in C} C_j)}. \tag{3.12}$$

Since our probabilistic score lies between zero and one, we are not required to normalize it. A confidence score close to one indicates a reliable prediction, whereas a score close to zero indicates that we are less confident about the given prediction. Note that if we assume $P(F|\bigcup_{C_j \in C} C_j) = P(F)$, the presented confidence score would be a monotone transformation of $P(C_{max}|F)$, given by NBconf $= 1/(1 + \frac{1}{P(C_{max}|F)})$.

3.2.5 Creating Interpretable Output

To provide an interpretable output, YLoc creates a short reasoning in natural language that explains why a prediction was made. For this purpose, we first have to create a description for every YLoc feature and, secondly, have to assemble these descriptions into natural language.

A prediction is only interpretable for a user, if the individual features are interpretable too. A user should be able to relate a feature to a real world biological property. However, for some features, a biological explanation is not always obvious. For example, a high autocorrelation of hydrophobicity within the first 20 amino acids is not obviously an important characteristic in subcellular localization. However, we find that the sum of hydrophobicity within the first 20 amino acids, which encodes for long hydrophobic stretches in the N-terminus, is highly correlated with this feature. A long hydrophobic stretch in the N-terminus is in turn an important property of the secretory pathway signal. In similar cases, where a biological explanation is not obvious at first sight, we transfer the biological meaning from a strongly correlated feature. By doing so, we can manually describe all selected YLoc features in biological terms using natural language.

To be able to create a sentence that contains information about the actual feature value, we manually describe every discretization interval of the feature with an adjective. These descriptions are then linked to the actual feature descriptions. For example, a secretory pathway signal can be absent, weak, medium, or strong, depending on the underlying hydrophobicity. Obviously such descriptions are always a little arbitrary since individuals have different understandings of words like strong or weak. However, they are a good indicator of the underlying property. Such descriptions might be less detailed but are far easier to interpret by a user.

YLoc uses the manually provided description of the two most important features to create a short reason for the users. In this context, the two most important features

are the two features with the largest absolute discrimination score. In addition, the feature probability are used to underline the argumentation of YLoc. For example, it not only provides the user with the fact that the protein contains a strong secretory pathway signal, but also informs the user that 69% of the proteins from the secretory pathway share this property while almost no protein from the nucleus and cytoplasm show this attribute. Using this strategy YLoc can provide an interpretable reasoning that helps a user to understand the prediction output.

3.2.6 Datasets

3.2.6.1 BaCelLo Dataset

For training the YLoc-LowRes predictor, we used the BaCelLo training dataset [139]. The homology reduced dataset extracted from Swiss-Prot release 48.0 contains 2,597 animal, 1,198 fungal, and 491 plant proteins, resulting in three versions of YLoc-LowRes (see Table D.1 in the appendix for more details). Only globular proteins were considered in the annotation. Animal and fungal proteins originate from four locations: nucleus (nu), cytoplasm (cy), mitochondrion (mi), and the secretory pathway (SP). Plant proteins originate from five locations: nu, cy, mi, SP, and chloroplast (ch). The BaCelLo independent dataset [27] (IDS) contains proteins added to Swiss-Prot between release 49.0 and 54.0 with at most 30% sequence identity to proteins in the BaCelLo dataset. Moreover, proteins from the same location that align with an E-value lower than 10^{-3} according to BLAST [1] are clustered, resulting in 432 animal, 418 fungi, and 132 plant groups [27].

3.2.6.2 Höglund Dataset

For training YLoc-HighRes and YLoc^{+}, we used the Höglund training dataset [83]. The 5,959 eukaryotic proteins extracted from Swiss-Prot release 42.0 cover 11 locations: nu, cy, mi, ch, endoplasmic reticulum (er), Golgi apparatus (go), peroxisome (pe), plasma membrane (pm), extracellular space (ex), lysosome (ly), and vacuole (va) (see Table D.2 in the appendix for more details). For the Höglund dataset, Blum *et al.* [8] created an independent dataset (IDS) with proteins from Swiss-Prot release 55.3, which covers the locations er, go, pe, pm, ex, ly, and va. Proteins that share more than 30% sequence identity with proteins from the original Höglund dataset were excluded. In this study,

we only make use of the animal Höglund IDS, since it contains sufficient amount of proteins (198). By clustering proteins from the same location with more than 40% sequence identity, 158 groups of animal proteins were obtained.

3.2.6.3 DBMLoc Dataset

In addition to proteins from the Höglund dataset, YLoc$^+$ was trained using proteins from the DBMLoc database [197]. The DBMLoc database contains more than 10,000 proteins with multiple subcellular localization, which were experimentally determined or extracted from the literature. We extracted proteins which share less than 80% sequence similarity from DBMLoc. Most proteins in DBMLoc are present in two subcellular locations. Still, there is a small portion of proteins with three or more localizations. However, for training we selected only multiple locations with more than 100 representative proteins: cy and nu (cy_nu), ex and pm (ex_pm), cy and pm (cy_pm), cy and mi (cy_mi), nu and mit (nu_mi), er and ex (er_ex), and ex and nu (ex_nu). Due to the limited number of training examples for some localizations, we could not use a lower sequence similarity threshold. More details concerning the 3,054 proteins with multiple localization can be found in Table D.3 in the appendix.

3.2.7 Training and Evaluation

We implemented YLoc using Python, the machine learning library Weka [192], BLAST [1], and PROSITE scan [47]. Each YLoc predictor is available as an animal, fungal, and plant version.

To evaluate the prediction performance, we use the overall accuracy (ACC) and the F1-score averaged over all classes (F1). See Section 2.1.3 for a detailed introduction into both measures. Of the two measures, F1 is better suited than the ACC as an evaluation measure. Especially for unbalanced datasets, the ACC biases towards an overrepresented class. Thus, if all instances are predicted to belong to this class, the ACC is still rather high.

The ACC and F1 can be easily generalized using measures from multi-label classification [186]. Let $T = (x_i, y_i)$ denote a test dataset with y_i being the set of correct labels of an instance i. On the other hand, let $\hat{y}_i$ denote the set of predicted labels.

Then we can define the multi-label ACC as

$$\mathrm{ACC} = \sum_{\{i|i\in T\}} \frac{|y_i \cap \hat{y}_i|}{|y_i \cup \hat{y}_i|} \tag{3.13}$$

and the REC and PRE for label j as follows:

$$\mathrm{REC}_j = \sum_{\{i|i\in T \wedge j\in y_i\}} \frac{|y_i \cap \hat{y}_i|}{|y_i|} \tag{3.14}$$

$$\mathrm{PRE}_j = \sum_{\{i|i\in T \wedge j\in y_i\}} \frac{|y_i \cap \hat{y}_i|}{|\hat{y}_i|}. \tag{3.15}$$

Using multi-label measures, we can rate predictions as partially correct if only a portion of the correct labels were recovered or more labels than the correct ones were predicted.

To visualize the estimation quality of a confidence estimator, we plot a so-called confidence curve. It is based on the idea that we iteratively remove the prediction with the lowest normalized confidence score from our set of test predictions. Consequently, we would expect to observe better prediction qualities for the remaining set of high confidence predictions. Let $Q(i)$ be the quality value of the i predictions with the highest normalized confidence score, where Q can be any of the above quality measures, e.g. ACC, REC, PRE, or F1. Further, let $ncs(i)$ be the minimum normalized confidence score of the i predictions with the highest normalized confidence score. If we plot $Q(i)$ against $ncs(i)$, we expect $Q(i)$ to increase with increasing $ncs(i)$. On the other hand, if we plot $Q(i)$ against i, we would expect that $Q(i)$ is decreasing with increasing i (see Figure 3.1 for an example). An increase and decrease of the former and latter curve, respectively, implies an increasing prediction quality for predictions with a larger confidence score, which is a desirable behavior for a confidence estimator.

We measure the quality of confidence estimates by calculating the normalized area under the confidence curve (AUCC) as

$$\begin{aligned} AUCC_{\mathrm{inst},Q} &= \frac{\sum_{i=1}^{|T|} Q(i) - Q(|T|)}{Q(|T|)\cdot|T|}, \\ AUCC_{\mathrm{ncs},Q} &= \frac{\sum_{i=2}^{|T|} (Q(i) - Q(|T|))/(ncs(i) - ncs(i-1))}{Q(|T|)\cdot(ncs(1) - ncs(|T|))}, \end{aligned} \tag{3.16}$$

where T is the corresponding test dataset and $Q(|T|)$ is the baseline performance on the whole test set. The AUCC yields a positive value if correctly classified instances are

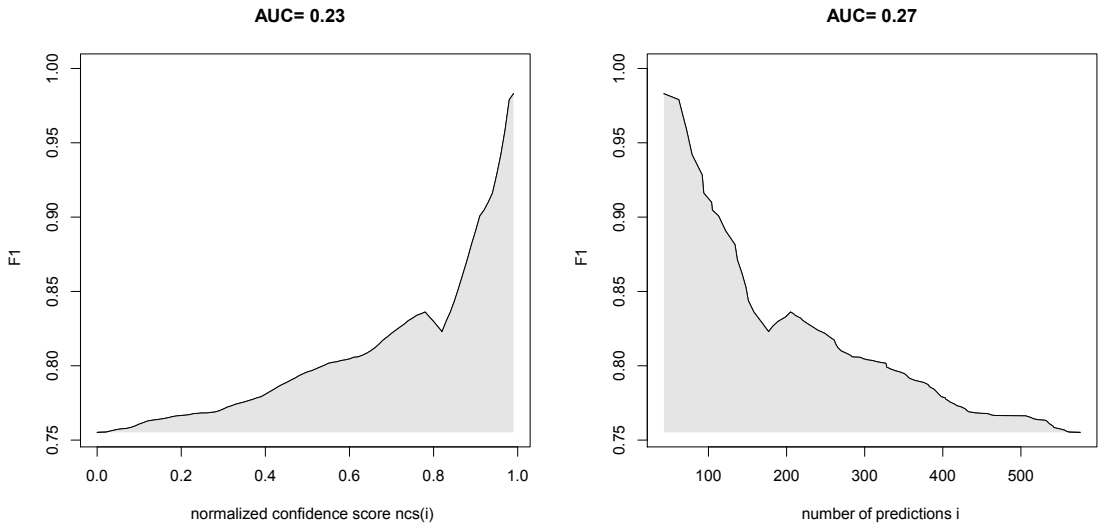

Figure 3.1: Example confidence curves. The left-hand confidence curve shows the F1 for each subset of predictions with a minimum normalized confidence score $ncs(i)$. The right-hand confidence curve shows the F1 for the i predictions with the highest confidence score. In this example, subsets of predictions with larger confidences score yield an increased accuracy. Both curves exhibit an AUCC of around 0.25 since the area under the curves covers approximately one fourth of the plot area.

enriched with high normalized confidence scores (see Figure 3.1). In contrast, an AUCC around zero implies that confidence scores were randomly assigned to predictions.

To estimate the optimal parameters of the nearest neighbor, accuracy, and label entropy confidence estimator, we performed a five-fold cross-validation on the training dataset and chose the parameters resulting in the highest average AUCC. The AUCC can be calculated for any prediction quality measure. However, calculating the AUCC based on F1 can create artefacts in the confidence curve due to very small classes. To reduce the influence of small classes, we exclude classes containing less than five instances from the calculation of the F1. While the F1 is a good measure for comparison, we found the AUCC to be more stable when used together with the ACC and, thus, optimized the average $\mathrm{AUCC}_{\mathrm{inst},ACC}$ in the above mentioned cross-validation.

3.3 Results and Discussion

3.3.1 Cross-Validation Evaluation

We evaluated all YLoc predictors in a five-fold nested cross-validation evaluation scheme on the training datasets. Since MultiLoc2 was trained on the same datasets, we compared both methods. Surprisingly, we did not observe a considerable difference between the predictors regarding prediction performance (see Table 3.2). All predictors show a very high ACC and F1. The major differences between the methods lie in their complexity and interpretability. MultiLoc2 is based on a complex SVM-ensemble classifier and uses more than 1,000 features, whereas YLoc is based on a very simple classification model. Due to its complexity, predictions made by MultiLoc2 are not interpretable. In contrast, using YLoc, it is obvious how biological features contributed to the outcome.

Version	MultiLoc2-LowRes	YLoc-LowRes
Animals	0.82 (0.84)	0.85 (0.87)
Fungi	0.79 (0.78)	0.77 (0.79)
Plants	0.79 (0.80)	0.78 (0.80)
Version	MultiLoc2-HighRes	YLoc-HighRes
Animals	0.84 (0.89)	0.85 (0.91)
Fungi	0.84 (0.89)	0.84 (0.91)
Plants	0.84 (0.89)	0.83 (0.90)

Table 3.2: Cross-validation performance comparison. Five-fold cross-validation performance of MultiLoc2-LowRes and MultiLoc2-HighRes compared to the five-fold nested cross-validation performance of YLoc-LowRes and YLoc-HighRes concerning the F1 and ACC (in brackets).

3.3.2 Benchmark Study on Two Independent Datasets

To show that YLoc is well-suited to predict the localization of novel proteins, we carried out a benchmark study using two recently published IDSs, the BaCelLo IDS [27] and the Höglund IDS [8]. We compared YLoc against six other state-of-the-art subcellular localization predictors, MultiLoc2 [8], BaCelLo [139], LOCTree [124], WoLF PSORT [88], Euk-mPloc [37], and KnowPred [111]. These predictors were chosen because they are quite recent and are available as online or as stand-alone version. In the

case of the BaCelLo IDS, we grouped predicted locations from the secretory pathway into the class SP to deal with predictors that distinguish between these locations. In contrast, for the Höglund IDS, we excluded predictors that cannot distinguish between the secretory pathway locations. To predict multiple locations with KnowPred, we defined a threshold of 30 for the multi-localized confidence score. As mentioned before, very similar proteins from the same location in the IDS are clustered. Instead of evaluating the performance based on one representative of each cluster, we re-weight instances such that the weight of all instances within one cluster sums to one. The results are summarized in Table 3.3.

Method	B Animals	B Fungi	B Plants	H Animals
YLoc-LowRes	0.75 (**0.79**)	**0.61** (0.56)	0.58 (0.71)	- (-)
YLoc-HighRes	0.69 (0.74)	0.51 (0.56)	0.54 (0.58)	0.34 (0.56)
YLoc$^+$	0.67 (0.58)	0.51 (0.48)	0.49 (0.53)	0.37 (0.53)
MultiLoc2-LowRes	**0.76** (0.73)	**0.61** (0.60)	**0.64** (**0.76**)	- (-)
MultiLoc2-HighRes	0.71 (0.68)	0.58 (0.53)	0.54 (0.62)	**0.41** (**0.57**)
BaCelLo	0.66 (0.64)	0.60 (0.57)	0.56 (0.69)	- (-)
LOCTree	0.58 (0.62)	0.43 (0.47)	0.58 (0.70)	- (-)
WoLF PSORT	0.67 (0.70)	0.51 (0.50)	0.46 (0.57)	0.18 (0.36)
Euk-mPloc	0.54 (0.61)	0.56 (0.60)	0.37 (0.46)	0.24 (0.27)
KnowPred	0.69 (0.75)	0.56 (**0.66**)	0.23 (0.29)	0.37 (0.49)

Table 3.3: Performance comparison using two independent datasets. Performance of the YLoc predictors and other state-of-the-art predictors using the BaCelLo (B) IDS and the Höglund (H) IDS concerning F1 and ACC (in brackets). The performance of YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred was measured using the generalized F1 and ACC. The highest-ranking method regarding each measure is highlighted in bold. Note that the WoLF PSORT results differ slightly from those obtained in Blum *et al.* [8] due to some changes in the underlying dataset. Also note that KnowPred does not predict chloroplasts.

We observed that YLoc-LowRes and MultiLoc2-LowRes yield the best overall performance on the BaCelLo IDS. This is due to the fact that both predictors are specialized on distinguishing globular proteins. Among the high-resolution predictors, MultiLoc2-HighRes and KnowPred perform best, followed by YLoc-HighRes. Although YLoc$^+$ was designed to predict multiple localizations, it performs comparably to Euk-

mPloc and WoLF PSORT. Clearly, the prediction performance depends on the origin of the proteins. In particular, the YLoc predictors are less accurate for fungal proteins, but yield good performance for animal and plant proteins. In contrast, Euk-mPloc performs well for fungal proteins but poorly for animal and plant proteins. Note that KnowPred does not predict chloroplasts and, thus, performs poorly on plant proteins. Most interestingly, the YLoc predictors perform comparable to the other predictors in the benchmark study, even though they have a very simple architecture and use at most 30 features. Similar results were observed for the animal Höglund IDS. MultiLoc2-HighRes performs best among the high-resolution predictors, followed by YLoc$^+$, YLoc-HighRes, and KnowPred. Euk-mPloc and WoLF PSORT, the other high-resolution predictors in this study, yield a poor F1 and ACC. In general, the performance of all predictors is comparably low for this dataset. This is due to the limited amount of available training data for the peroxisome and the secretory pathway locations. Since the number of protein sequences of the animal Höglund IDS is relatively small, the performance results should be seen as a trend. In addition, see Tables D.7, D.6, D.5, D.4, D.8, D.10,D.12, and D.14 in the appendix for more detailed results.

Using YLoc$^+$ has an advantage: Predictions can be borderline due to weak and noisy sorting signals. Hence, predicting all top-ranked locations leads to an increased recall. Moreover, it can help users to identify real multiple localization of proteins.

We also tested YLoc without transferring information from homologous proteins by excluding GO-term features from the feature selection. The resulting predictors show only slightly reduced prediction performance on the IDSs (see Tables D.9, D.11,D.13, and D.15). Additional versions of YLoc not using homology information can be helpful to analyze whether a prediction outcome would change if we were restricted to sequence information only.

3.3.3 Multiple-Localization Prediction

We compared YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred regarding their ability to predict multiple localization sites. The locations for all proteins in the DBMLoc dataset were predicted by WoLF PSORT, Euk-mPloc, and KnowPred by considering this dataset as an IDS. Since KnowPred returns only scores for each location but no location prediction, we predict all locations with a score above 30 if the multi-localized confidence score is larger than 30. For YLoc$^+$, we evaluated the predictions of the

DBMLoc proteins using the five-fold nested cross-validation results. We compared all predictors using single-label as well as multi-label measures.

Measures	YLoc$^+$	Euk-mPloc	WoLF PSORT	KnowPred
Single-label	**0.31** (0.35)	0.04 (0.05)	0.03 (0.05)	0.28 (**0.36**)
Multi-label	**0.68** (**0.64**)	0.44 (0.41)	0.52 (0.43)	0.66 (0.63)

Table 3.4: Performance comparison using the DBMLoc dataset. The performance was measured using F1 and ACC (in brackets). For YLoc$^+$ and WoLF PSORT, only the best performing version is shown. The highest-ranking method regarding each measure is highlighted in bold.

The results are shown in Table 3.4. YLoc$^+$ is superior to WoLF PSORT and Euk-mPloc in this study in terms of ACC as well as F1. While predicting at least one location correctly for many proteins, Euk-mPloc and WoLF PSORT are only able to predict 5% of the correct multiple locations. In contrast, YLoc$^+$ and KnowPred are able to recover more than one third of the multiple locations correctly. When excluding GO-term features from the feature selection, we observe only a slight performance decrease. For more details see Tables D.16 and D.17 in the appendix. In a similar study, we are able to show that the performance of all predictors remains almost unchanged if we use a cutoff of 40% in the homology reduction of the DBMLoc dataset, see Tables D.18 and D.19 in the appendix for performance details on the resulting DBMLoc40 dataset.

3.3.4 Evaluation of Confidence Estimates

To find a confidence estimator that is suitable to rate the confidence of individual YLoc predictions, we compared the estimation performance of the five presented estimators. For this purpose, we estimated the confidences of YLoc-LowRes predictions on the BaCelLo IDSs using the different presented confidence estimators. To show that the posterior probability is less suitable for confidence estimation, we additionally included it into the study. For all estimates, we plotted confidence curves and calculated the corresponding AUCCs. The obtained AUCC_{F1} values for the BaCelLo IDSs are shown in Table 3.5.

We observed that estimator NBconf performs superior to the four dataset-based estimators. Although estimators Acc and LE yield sometimes larger AUCCs, their es-

Measure	IDS	NBconf	Post	AvgDist	NoNN	Acc	LE
$AUCC_{inst,F1}$	Animals	0.27	0.13	-0.31	-0.25	0.27	0.18
	Fungi	0.19	0.09	-0.20	-0.17	-0.02	-0.09
	Plants	0.34	0.23	0.24	0.22	0.45	0.59
$AUCC_{ncs,F1}$	Animals	0.23	0.06	-0.07	-0.20	0.16	0.37
	Fungi	0.21	0.03	-0.06	-0.18	0.03	-0.02
	Plants	0.33	0.12	0.10	0.23	0.13	0.55

Table 3.5: $AUCC_{F1}$ of different confidence estimators for YLoc-LowRes predictions on the BaCelLo IDSs. We calculated $AUCC_{inst,F1}$ and $AUCC_{ncs,F1}$ based on confidence estimates for predictions of the BaCelLo IDSs using different confidence estimators. In addition, we show the AUCCs for the case of using the posterior probability as confidence estimate (Post).

timates are less robust. In particular on the BaCelLo fungi dataset, both estimators perform close to random. Estimators AvgDist and NoNN show the worst performance. On two of the three datasets they even yield a negative AUCC. In contrast, for confidence estimator NBconf, we observe an enrichment of correctly predicted instances for large normalized confidence scores on all BaCelLo IDSs. Moreover, NBconf is better suited for giving confidence estimates than the posteriori alone, see also Figure 3.2. Note that although an AUCC around 0.2 seems like a low value, it is already a very good result. As we can see from the confidence curves in Figure 3.1, confidence estimates yielding an AUCC around 0.2 show already a considerable enrichment of correct predictions. As a consequence, the prediction quality is increased by up to 20% given a normalized confidence scores of more than 0.5. The confidence curves and AUCCs based on the ACC look very similar, see Table D.20 in the appendix.

Our results suggest that our model-based confidence estimator NBconf performs superior to dataset-based estimators. This is no surprise since a model-based approach can use additional properties of the classification model for its estimation. As a consequence of our results, we integrated this confidence estimator into YLoc and use it to return confidence scores for every individual prediction.

To prove that YLoc highly benefits from confidence scores made by NBconf, we show the performance of YLoc on the animal BaCelLo IDS for different minimum normalized confidence scores in Table 3.6. The ACC and F1 of all predictors increase

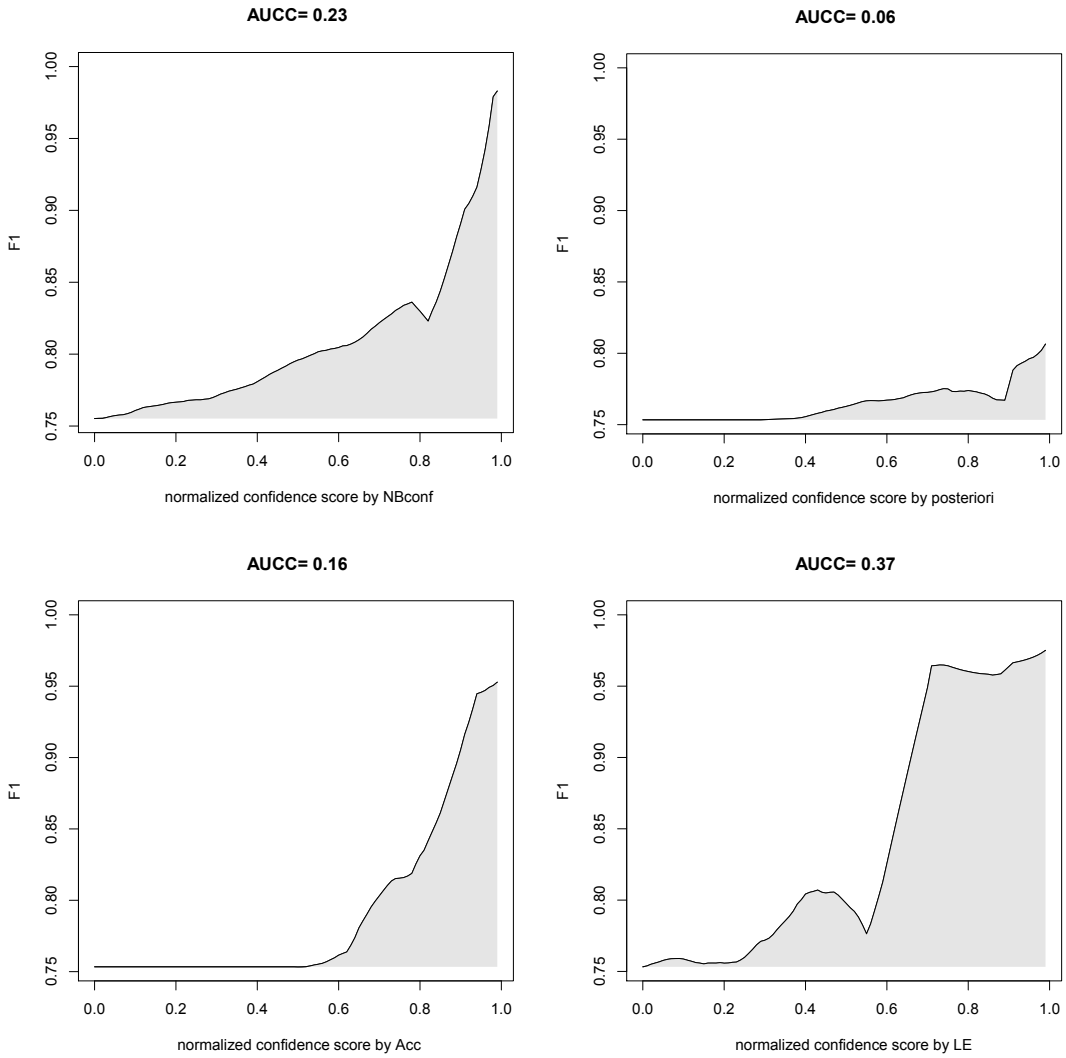

Figure 3.2: Confidence curves of different estimators on the BaCeLo animals IDS. We plotted score-based confidence curves resulting from estimates of NBconf, Acc, and LE as well as the posterior probability using predictions of the BaCelLo animals IDS made by YLoc-LowRes. Note that although the AUCC of estimator NBconf is smaller that the one of LE, the corresponding confidence curve is quite smooth.

with an increasing minimum normalized confidence score. The F1 and ACC of YLoc-HighRes increase by at least 4% given a minimum score of 0.2 and by at least 8% given a confidence threshold of 0.9. YLoc-LowRes and YLoc$^+$ show an even higher enrichment

for high confidence scores. For example, YLoc-LowRes achieves an F1 of 0.84 and an ACC of 0.91 for a minimum normalized confidence score of 0.8. Thus, YLoc-LowRes could correctly predict the location for 91% of the 189 proteins that have a normalized confidence score of at least 0.8. We got similar results for fungal and plant proteins (see Table D.21 in the appendix). Although only a certain portion of proteins can be predicted with high confidence, their predicted locations are much more likely to be correct.

Method	Measure	Minimum confidence score					
		0.00	0.20	0.40	0.60	0.80	0.90
YLoc-LowRes	F1	0.75	0.76	0.78	0.80	0.84	0.95
	ACC	0.79	0.79	0.81	0.86	0.91	0.93
	% of n	100	81	69	52	33	20
YLoc-HighRes	F1	0.69	0.74	0.76	0.76	0.77	0.77
	ACC	0.74	0.78	0.80	0.82	0.83	0.84
	% of n	100	88	82	74	68	61
YLoc$^+$	F1	0.67	0.69	0.72	0.77	0.76	0.81
	ACC	0.58	0.60	0.62	0.65	0.65	0.69
	% of n	100	86	73	56	38	25

Table 3.6: Performance of YLoc using the BaCelLo animal IDS for different minimum confidence levels. For each minimum normalized confidence score the prediction performance is given using F1 and ACC as well as the percentage of instances n that can be predicted with as least this score. The performance of YLoc$^+$ was measured using the generalized F1 and ACC.

3.3.5 The YLoc Web Server

The YLoc web server requires protein sequences in FASTA format as input. It allows users to predict the location of up to 20 proteins. For large-scale predictions, users can access YLoc via SOAP or HTTP using the Python-based client scripts provided on the YLoc web site. Users can choose between three YLoc predictors, YLoc-LowRes, YLoc-HighRes, and YLoc$^+$, and three protein origins (animals, fungi, and plants). In addition, they can switch off the use of GO term-based features. In this case, YLoc uses models in which the GO terms from close homologues are replaced by sequence-

based features. Consequently, these YLoc models rely less on the presence of close homologous proteins. Every prediction will be assigned a prediction ID which can be used to retrieve results later on. Alternatively, users can simply bookmark the waiting page or the result page to obtain results later. Currently, predictions are kept for two weeks. The location prediction of a single protein takes 10-20 s, depending on the protein length. Note that about 95% of the runtime origins from BLAST and PROSITE scan. The actual prediction using naïve Bayes is very fast and requires only a few milliseconds.

Prediction results are displayed at three different levels of details. The prediction summary presents the predicted location(s), the probability of those, and the normalized confidence score for every query protein. The probability of a location is simply how likely the protein is located in this compartment. In contrast, the confidence score is a measure of reliability. A low confidence score implies the possibility that the real probability can differ considerably from the predicted probability. However, a high confidence score signifies that the predicted probability is close to the real probability for being located in the predicted location. Consequently, higher confidence scores imply a higher reliability of the prediction for the individual sequence. In addition, an explanation in natural language clarifies why the prediction has been made. This explanation includes the two most likely reasons for this localization, for example: "The most important reason for making this prediction is the strong secretory pathway sorting signal" or "Moreover, it is a barely charged protein." This information can be very important since it might already give a hint of the underlying mechanism of this sorting process responsible for the localization.

The detailed prediction page provides more information on a particular protein prediction. For example, the probability distribution of locations is provided. It is important to know the runner-up locations, especially for low confidence predictions, since rather ambiguous predictions should be inspected manually. YLoc also provides the most similar protein from Swiss-Prot 42.0 and associated GO terms. More details of how protein attributes influence the prediction are given in a large attribute table (see Figure 3.3). The attributes are expressed in biological terms and ordered according to their absolute discrimination score, which corresponds to their influence on the prediction. A positive discrimination score implies that the attribute value is very typical for the predicted location but atypical for some other location. In contrast,

a negative discrimination score implies that the attribute value is more typical for some other location than the predicted one. A simple +/- encoding shows whether an attribute is typical for a location or not. By inspecting only the first lines of the table, it is sometimes already obvious which biological property led to the prediction outcome and is likely to be responsible for the real localization of the protein. In addition, it gives hints which parts of the protein should be considered for mutations that result in an altered subcellular localization.

Attribute		Discrimination Score	Cy	Mi	Nu	SP	Detailed Attribute Information
strong	secretory pathway sorting signal	5.72	--	--	--	++	Attribute Details...
barely	charged protein	2.89	+	--	-	++	Attribute Details...
no	mono NLS sorting signal	2.58	--	+	--	++	Attribute Details...
strong	putative mitochondrial or secretory pathway sorting signal	2.42	--	++	--	++	Attribute Details...
very	hydrophobic protein	2.32	--	-	--	++	Attribute Details...
very	hydrophobic N-terminus	2.06	--	--	--	++	Attribute Details...
absent	GO:0005576 (extracellular region)	-1.77	+	+	+	--	Attribute Details...
no	putative mitochondrial sorting signal	1.68	+	--	+	++	Attribute Details...
barely	negatively charged N-terminus	1.56	--	++	--	++	Attribute Details...
absent	GO term GO:0005739 (mitochondrion)	1.55	+	--	+	+	Attribute Details...

Figure 3.3: The attribute table of the YLoc web service. All attributes are listed in order of their influence on the prediction outcome and are expressed in biological terms. The +(+) or -(-) indicates whether an attribute value is (very) typical or (very) untypical for a location.

How a particular biological attribute is calculated can be found on a detailed attribute page (see Figure 3.4). For example, YLoc-LowRes (animal version) calculates the strength of the secretory pathway sorting signal using the "autocorrelation of every third hydrophobic amino acid within the first 20 amino acids in the N-terminus". Knowing how the attribute value is calculated is essential to understand which particular amino acids and properties encode for a possible sorting signal. Furthermore, the attribute is visualized. Embedded JavaScript code displays the distribution of proteins from the different locations regarding this feature. The provided protein distributions are very helpful for understanding how proteins from different locations behave with respect to a biological property or sorting signal.

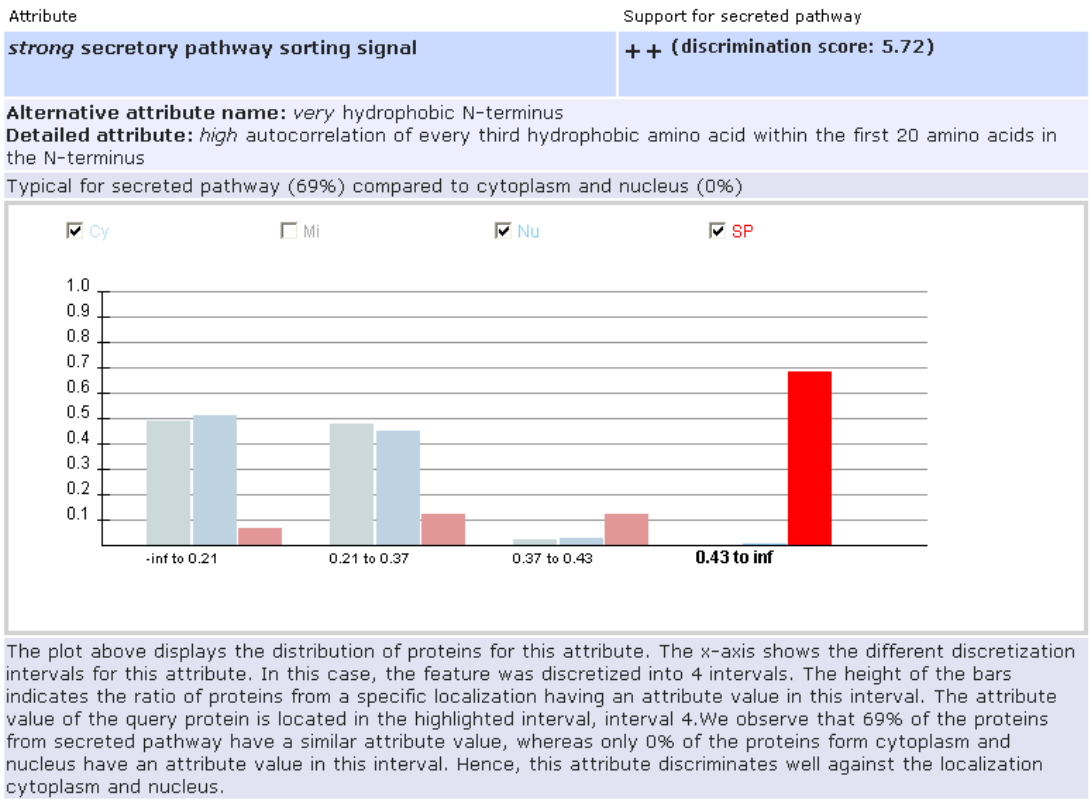

Figure 3.4: Detailed attribute page of the feature "secretory pathway sorting signal" in YLoc-LowRes (animal version). The distribution of proteins from the cy, mi, nu, and SP over the different attribute intervals is shown.

To show how YLoc elucidates a subcellular localization prediction, we provide an interpretable example prediction output. The example protein *Neurotoxin magi-12* (U13-HXTX) with Swiss-Prot AC Q75WG7, obtained from the animal BaCelLo IDS, was predicted to be located in the SP by YLoc-LowRes with a probability of 99.99% and a normalized confidence score of 0.99. Hence, users can be very confident that the prediction is correct. U13-HXTX is known to be secreted into the extracellular space. YLoc found that U13-HXTX contains a strong secretory pathway signal, which is known to mediate the transport into the SP. Moreover, YLoc identified this feature to be the most discriminating, since 69% of all proteins in the SP have a similar secretory pathway signal, whereas only 0%, 2%, and 1% of all proteins present in the cy, mi, and nu, respectively, have the same kind of feature. Figure 3.4 shows the distribution of proteins from different locations concerning this particular feature. In addition, YLoc

identified other features that highly influenced the prediction, such as the low charge of the protein and the lack of a mono NLS. Table 3.7 shows an example output of YLoc for the six most discriminating attributes. Given this output, it is easy to understand why this prediction was made and what features were responsible for it.

Sequence Feature	DS	Nu	Cy	Mi	SP
Strong secretory pathway sorting signal *(High hydrophobic autocorrelation within first 20 amino acids)*	5.72	0.01	0.00	0.02	0.69
Barely charged *(Low overall charge autocorrelation)*	2.89	0.10	0.16	0.02	0.28
No mono NLS sorting signal	2.89	0.04	0.12	0.02	0.26
Strong putative mitochondrial or secretory pathway sorting signal *(Large weighted sum of amino acids, typical for mi and SP)*	1.68	0.58	0.62	0.16	0.84
Very hydrophobic protein *(High pseudo amino acid count of hydrophobic amino acids [CITVWY])*	2.32	0.08	0.13	0.04	0.36
Very hydrophobic N-terminus *(High pseudo amino acid count of very hydrophobic residues within the first 90 amino acids)*	2.06	0.09	0.05	0.08	0.41

Table 3.7: YLoc output of an example prediction. The six most discriminating protein features are displayed in order of their absolute discrimination score (DS). The features are manually annotated with a biological property. A more detailed description of each feature is given in italics. For each location, the ratio of proteins having this particular feature is shown. A detailed description of the fourth features is given in Nakai and Kanehisa [125].

The YLoc web service is available at www.multiloc.org/YLoc. For large-scale predictions, YLoc can be accessed via SOAP. The corresponding WSDL can be downloaded from the YLoc web site. In addition, we provide a Python-based client script. Alternatively, YLoc can be accessed via an HTTP-based client that is also available for download.

3.3.6 Understanding and Predicting Localization Changes

A key step in understanding the localization process of proteins is to elucidate why proteins localize to different compartments when undergoing mutation. Furthermore,

knowledge obtained from YLoc's prediction output can be used for supervised protein engineering. Due to its ability to identify biological properties that might be responsible for the localization (e.g. sorting signals) YLoc can be valuable for experimental biologists. In the following, we show examples of localization engineering taken from the literature, where YLoc could have been helpful to understand the underlying localization processes.

Human fumerate hydratase (FH, SwissProt AC P07954) is primarily located in the mitochondrion. The three YLoc predictors (animal version) detect the correct location and identify an mTP. After truncating the leading 43 residues, FH lacks an mTP and shows a negatively charged N-terminus, which is unfavourable for mitochondrial localization. Consequently, YLoc predicts FH to be cytoplasmic. In fact, the truncated FH protein is a known cytoplasmic isoform of FH encoded by the same gene [182].

Takada *et al.* [178] showed that human glyoxylate aminotransferase 1 (AGT1), located in the peroxisome, is likely to have lost its mTP by point mutation. In fact, the mTP of AGT1 of rat, located in the mitochondrion, shares 74% sequence identity with the upstream region of human AGT1. If we correct the single point mutation, we extend human AGT1 by 22 residues. YLoc-HighRes (animal version) is then able to predict a localization shift from the peroxisome to the mitochondrion. In addition, it recognizes the appearance of an weak mTP. According to YLoc$^+$, the extended AGT1 is very likely localized in the mitochondrion.

In 1982, Carlson and Botstein [26] found two isoforms of glycosylated invertase in yeast, which is encoded by the SUC2 gene. The extracellular isoform is regulated by glucose repression, whereas the N-terminal truncated cytosolic isoform is constitutively expressed. YLoc-LowRes (fungal version) is able to predict the localization change of this truncation, although it still recognizes associated GO terms that indicate a secreted localization. In addition, the truncation of the signal peptide was recognized by YLoc. Four years later, Kaiser and Botstein [98] examined the signal peptide of the same protein by inducing multiple mutations in the signal peptide region ranging from short deletions up to long substitutions. Five of the the ten functional mutants lack extracellular invertase activity and show only cytoplasmic activity. Three of these cases could be validated by YLoc-LowRes. In one case, YLoc predicts a localization change, but not to the cytoplasm. In all five cases, YLoc confirms the loss of a signal peptide. In addition, YLoc reproduces that five mutants remain in the secretory pathway.

The GLR1 gene of yeast encodes two different isoforms of glutathione reductase: a longer, mitochondrial isoform and a shorter, cytoplasmic isoform [130]. The two different isoforms very likely arise from leaky ribosomal scanning. YLoc-LowRes (fungal version) predicts GLR1 as mitochondrial and identified an mTP within the first 20 amino acids. The truncated isoform is still predicted to be located in the mitochondrion but with a decreased probability. Moreover, YLoc identified the loss of the mTP. Both YLoc-HighRes and YLoc$^+$ reproduce the location shift and state a change in the mTP.

A detailed overview of predicted locations of all discussed isoforms and mutated proteins can be found in Table D.22 in the appendix.

3.4 Conclusion

Understanding protein subcellular localization is crucial for the functional annotation of proteins. In contrast to many prediction methods, predictions made by YLoc are interpretable. The YLoc web server explains why a prediction was made and shows which particular attributes contributed most and in which direction. Explaining why a subcellular localization prediction was made does clearly influences the trust in the results. A user might find a prediction reasonable but might also find attributes indicating a different localization that are more convincing to him. In addition, users can use the YLoc web server to identify properties of their proteins that are typical or atypical for a certain cell organelle. YLoc can thus be helpful to understand the localization of novel proteins that have not been annotated before.

We show that YLoc performs comparable to or even better than state-of-the-art subcellular localization predictors. Our benchmark results suggest that using complex computational models is less important than using a well defined set of highly discriminating features. When predicting proteins from multiple locations, YLoc yields often better prediction quality than current state-of-the-art predictors. Moreover, YLoc's flexible probability transformation allows predicting novel location combinations, which are not part of the training data.

We could show that confidence estimation is helpful to rate the reliability of predictions. In particular, a probabilistic confidence estimator based on naïve Bayes is well suited to detect erroneous predictions. When considering only proteins that can be predicted with a certain normalized confidence score, the prediction performance

increases considerably. We believe that a confidence estimate is of great interest since it increases the trust in prediction results.

YLoc provides textual explanations in natural language, which allows every user to understand and interpret its output. The provided reasoning takes away some of the "black box" character of the prediction. YLoc's interpretability lays the foundation for using YLoc for protein localization engineering. For several examples, we can demonstrate that YLoc predicts experimentally validated changes of localization sites and known sorting signals caused by mutations. Since we applied YLoc successfully to proteins with alternative isoforms that differ in localization, it seems promising to include alternative transcription and translation sites as features for YLoc$^+$.

We are convinced that both performance and interpretability of YLoc can be improved by integrating further biologically relevant features. Improvement will rely on traditional biology and computational biology proceeding hand in hand. Discovering novel protein sorting signals can improve the performance of YLoc, whereas an improved predictor can help biologists to elucidate the localization of novel proteins.

Chapter 4

Interpretable Regression With CONFINE and CONFIVE

MHC-I–peptide binding represents an important step of the adaptive immune response. By binding peptides of proteins within the cell, MHC-I molecules are able to present a "fingerprint" of a cell's proteome on its surface. This includes peptides from pathogens. A subsequent binding of a T cell to an MHC-I–peptide complex can trigger apoptosis of the infected cell and induces immunological memory against future infections of the respective pathogen.

The identification of MHC-I binding peptides is an important step in identifying immunogenic peptides, i.e. epitopes. Since binding data is only available for a small number of peptides [116], computational MHC-I–peptide binding prediction has become an attractive alternative. However, the "black box" character of current state-of-the-art prediction approaches is often a drawback when predicting potential T-cell epitopes. Predictions are often not interpretable and provide no confidence information for individual predictions, which is of essential interest in critical tasks like epitope-based vaccine design. In this chapter, we show how to overcome the drawbacks of state-of-the-art prediction methods by introducing two novel confidence estimators, CONFINE and CONFIVE. In analogy to experimental measurements, they associate each affinity prediction with a confidence interval, which allows users to estimate the potential error of an individual prediction. Parts of this chapter will be published in Briesemeister *et al.* [19].

4.1 Introduction

The first steps towards MHC-I–peptide binding prediction were taken in the 1980s by identifying binding motifs [160]. It was found that certain positions of binding peptides show a strong preference to a small set of amino acids [143]. These positions are called anchor positions [64]. Motifs became very popular and were widely used to distinguish between binders and non-binders. However, binding motifs are a very simplified representation of binding since they ignore the fact that the whole peptide contributes to the binding affinity to the MHC molecule. To overcome this shortcoming, matrix-based methods and other classification algorithms have been applied. Matrix-based approaches such as BIMAS [132], SYFPEITHI [144], and TEPITOPE [7] calculate a sequence profile of validated binders. The binding score of a peptide is then obtained by combining the profile scores of the amino acids at each position in the peptide. In addition, various classification algorithms such as ANNs [85], hidden Markov models (HMMs) [23, 117], and SVMs [44, 53, 54, 95, 187] have been applied. However, classification approaches can only help to categorize peptides and ignore the fact that peptides exhibit different binding affinities to different MHC molecules. In particular, for epitope-based vaccine design, we rely on quantitative binding predictions [184].

Over the last years, various prediction methods have been introduced to model this quantitative structure-activity relationship (QSAR) [57]. In particular, linear QSAR approaches that aim at predicting the binding affinity from the peptide sequence without using structural information have been the center of attention. The simplest quantitative prediction approaches are either based on score matrices [24, 114, 135, 136] or on simple linear regression algorithms [58, 74, 81]. They assume that the amino acids of the peptide contribute independently to the overall binding affinity. More advanced approaches that consider interactions of peptide side chains are usually based on nonlinear regression algorithms [115]. Most state-of-the-art methods like NetMHC-3.0 [116] are based on ANNs [25, 116, 127]. But also SVR has been used for binding affinity prediction [191]. In addition to linear QSAR methods, structure-based prediction approaches have been investigated. They range from 3D-QSAR [56, 155] over docking and threading [97, 148] to molecular dynamics simulations [63]. However, structural approaches usually require large computational resources. Moreover, their prediction

quality is not yet comparable with sequence-based approaches, also due to the limited amount of structural MHC-I–peptide complex data.

The *in silico* prediction methods introduced above are an attractive alternative to costly and time-consuming experiments. In the past, they could support the identification of MHC-I binding peptides [21, 151, 168]. Moreover, the knowledge of MHC-I binding affinities can also support the design process of epitope-based vaccines. Only peptides bound to MHC molecules can induce T-cell reactivity and are, hence, able to induce immunity. It was also found that T-cell reactivity is correlated with MHC-I binding affinity [161]. As a consequence, predicted MHC-I binding affinities are often used as an approximation of T-cell reactivities. These are an important prerequisite to discover potential epitopes for vaccines. Note that epitope-based vaccine design does not only require detection of potential epitopes, but also to select an optimal subset of epitopes for a given antigen and target population [48, 183, 185, 189]. Furthermore, the selected epitopes have to be assembled in a poly-peptide such that peptides are likely to be presented to MHC molecules. See Toussaint and Kohlbacher [184] for more details on epitope-based vaccines.

As an alternative to naturally processed and presented peptides, vaccines can also benefit from engineered peptides with an enhanced T-cell reactivity, called heteroclitic peptides. Sometimes it is distinguished between fixed anchor epitopes, which are modified at the anchor residues leading to an increased MHC-I binding affinity, and heteroclitic peptides, which exhibit an increased TCR binding affinity [159]. After all, both types of epitope analogs aim at inducing a stronger immune response than the original peptide. There have been rational approaches to design epitopes by considering structural information [55, 180]. In addition, computational prediction approaches have been used to find peptides with maximum affinity to a particular MHC molecule [59, 104]. Other *in silico* approaches aim at increasing the affinity of a wildtype epitope by introducing mutations [6, 90]. Interestingly, state-of-the-art predictors like NetMHC-3.0 are often not considered for epitope engineering. Due to the "black box" character of complex regression approaches, their explicit regression function remains unknown. Hence, it is unknown how strongly an amino acid affects the binding affinity. Replacing amino acids that are unfavourable for binding, as performed in previous studies [59, 90, 180], cannot be guided by predictions that are not interpretable.

Using computational predictions methods for epitope-based vaccine design is fraught with high risk. It is often assumed that *in silico* prediction methods perform equally well for all peptides. But this is not the case. In particular, if they are applied to peptides that are very dissimilar to the corresponding training peptides, predictors perform usually poor. Thus, when MHC-I–peptide binding predictors are used for discovering or engineering epitopes, they might provide unreliable binders. In the experimental sciences, the concept of a measurement and its associated error is a cornerstone in understanding the reliability of a data point. In contrast, statistical measures that capture the prediction error are not a direct replacement for the measurement error. In most cases, it is not even clear what the reliability of a prediction method means. For example, specifying the correlation coefficient for a training dataset is not sufficient to really give the user an idea of the error of an individual prediction. To overcome these problems, confidence estimation, which determines the reliability of individual predictions, is desirable. In cases where highly accurate predictions are required, e.g. for choosing candidates for expensive experiments, confidence intervals would be especially valuable. Despite their importance, confidence estimation for regression models has not been applied extensively in the context of computational biology.

In the area of QSAR, where regression methods are applied to predict the biological activity of small molecules, the concept of confidence estimation was introduced through so-called applicability domains (AD) [164]. The AD defines the input space on which the model is expected to give reliable predictions [50]. However, AD estimators were designed to detect possible extrapolation errors but not to measure the error of instances within the AD. Consequently, some estimators cannot express the confidence in a prediction in a quantitative manner. Although some estimators can provide quantitative scores, it is usually difficult to relate a score to an actual error. One way to overcome these problems are confidence estimators that express the reliability of an individual MHC-I–peptide binding prediction in a quantitative and intuitive manner.

In this chapter, we introduce a novel concept to confidence estimation. In analogy to experimental measurements, we associate each individual affinity prediction with an estimate of its error. We propose two novel confidence estimators, CONFINE and CONFIVE, which return confidence intervals with only a small computational overhead. These intervals contain the real affinity value with a certain probability, while

being very small for confident predictions and fairly broad if the prediction is likely to be erroneous. Hence, in contrast to other estimation approaches, their error estimates are very intuitive and easy to interpret. CONFINE and CONFIVE estimate the confidence of a prediction by inspecting local properties of the input space. CONFINE determines the error rate of the nearest neighbors of a test instance in the training data. CONFIVE examines the variance in the surrounding local environment and assumes that large variances result in higher error rates. Since both estimators are strictly model-independent, they can be applied with any linear and nonlinear regression algorithm.

After introducing our regression approach, we present related work on confidence estimation. We then introduce the methods underlying CONFINE and CONFIVE and discuss their applicability by analyzing the influence of noise, the number of features, and the dataset size on the quality of the estimated confidence intervals. We then compare our confidence estimators with other existing confidence and AD estimators on the well-studied IEDB benchmark datasets [137]. In addition, we apply our estimators to a set of 3D-QSAR datasets. Our results suggest that CONFINE and CONFIVE are able to interpolate the prediction error better than or comparable to other methods, given a sufficient amount of training data. We also show that confidence intervals are a very intuitive and informative way to express the reliability of individual predictions. To illustrate the universal character of CONFINE and CONFIVE, we apply them not only to linear regression but also to nonlinear SVR. In an example study, we show that considering confidence estimates does increase the probability of detecting suitable candidates for epitope-based vaccines. Furthermore, we show how confidence estimators can be used for automatic epitope engineering. By considering the reliability of predictions, our genetic algorithm approach returns only strong binders that originate from high-quality predictions. Our results confirm that the confidence estimators presented here can improve the user's confidence in MHC–peptide binding predictions and support epitope discovery and epitope engineering.

An open-source implementation of both methods is available in the R package `confReg` (http://cran.r-project.org/web/packages/confReg/index.html). Due to its design and object-oriented implementation using S4 classes, it can be easily extended and adapted to various regression tasks.

4.2 Methods

To estimate affinity values and the corresponding confidences, we first have to train a regression model for this particular task. To be able to provide interpretable binding affinity predictions in terms of IC_{50} values, we employ linear least squares regression, which is described in detail in Section 2.1.2. Before training the model, we perform a model selection step in which we try to find the optimal set of features. First, the quality of every feature is assessed by performing three five-fold cross-validations using only this feature. The features are then sorted in ascending order of their mean squared error (MSE). Starting with the feature with the lowest MSE, the features are iteratively added to the feature set. If the new feature set shows a higher average MSE in three five-fold cross-validations than the previous feature set, the newly added feature is removed. Since predictions are only interpretable in a manual fashion if the feature number is relatively low, we set the maximum number of features to 50. The resulting feature set is used to train our linear regression model.

Although we focus on predicting binding affinities of peptides to MHC-I molecules, we will use the term response instead of affinity in the following. This also enhances the fact that the described methods are universal and can be applied to any regression task.

4.2.1 Related Work on Confidence Estimation

There are various methods related to the estimation of individual confidences. It can be distinguished between methods that use certain properties of a regression model, e.g. the predictive variance of a Gaussian process, and methods that are independent from a particular regression model. In this work, we concentrate on the latter, since model-independent confidence estimators are more universal.

When the response of a novel instance x^* has been predicted using a trained regression model, confidence estimators try to determine the reliability of this particular prediction. A confidence estimator is a function $f : \mathbb{R}^k \rightarrow \mathbb{R}$, where the input is a test instance x^* and the output is a confidence score $cs(x^*)$. Note that confidence estimators and AD estimators do not try to predict the exact error of a prediction itself. Instead, they require predictions with a low error to have a small confidence score and predictions with a high error to have a large confidence score. Scores determined by

different estimators are not necessarily comparable, nor interpretable. Determining a threshold for the applicability domain of a model is, hence, often very vague. Instead of relying on non-interpretable scores that cannot be interpreted by a user, we propose an approach of translating confidence scores into interpretable confidence intervals, a more intuitive measure of confidence.

In the following, we will discuss related work before introducing our novel concepts for confidence estimation the following section.

Number of Nearest Neighbors

A traditional approach to estimate confidences utilizes the number of neighbors (NoNN) [164]. For this purpose, we define the local environment $E(x^*, d_E)$ as the set of instances from the training data with a maximum distance d_E to x^*. The optimal value of d_E can be found using a cross-validation scheme. The confidence value is calculated as follows:

$$cs_{\text{NoNN}}(x^*) = |E|. \tag{4.1}$$

Instead of using the number of neighbors, we can transform estimator NoNN into a density-based estimator using a Gaussian kernel [12]:

$$cs_{\text{NoNN}^*}(x^*) = \frac{1}{n}\sum_{i=1}^{n} e^{-0.5d(x_i,x^*)^2}. \tag{4.2}$$

Distance-based Estimators

Distance-based estimators, which are often used for AD estimation, try to distinguish between outliers and instances within the domain. The mistrust in a prediction grows with its distance to the training data. The following two estimators express the distance to the training dataset as the minimum distance or average distance:

$$cs_{\text{MinDist}}(x^*) = 1 - \min_i d(x_i, x^*) \text{ and} \tag{4.3}$$

$$cs_{\text{AvgDist}}(x^*) = 1 - \frac{1}{n}\sum_{i}^{n} d(x_i, x^*). \tag{4.4}$$

A slightly more involved estimator puts a bias on closer instances [164]:

$$cs_{\text{AvgBiasedDist}}(x^*) = 1 - \frac{\sum_i^n e^{-3d(x_i,x^*)} d(x_i, x^*)}{\sum_i^n e^{-3d(x_i,x^*)}}. \tag{4.5}$$

Since the training dataset itself might already contain some outliers that are rather hard to predict, one can exclude such outliers from the confidence estimation. This is done by considering only instances that can be predicted with a maximum prediction error of $\hat{\epsilon}_m$ [60]. Let MinDistOF, AvgDistOF, and AvgBiasedDistOF denote these "outlier-free" versions of the previously introduced distance-based estimators. An optimal threshold for the prediction error $\hat{\epsilon}_m$ can be estimated via cross-validation.

A generalization of the above distance-based AD estimators based on a one-class SVM [154] was introduced by Fechner *et al.* [66], here denoted as 1-SVM. It uses the decision value of a one-class SVM with an RBF kernel trained on the training dataset. The parameters of the SVM and the RBF kernel can be optimized via cross-validation. However, the optimization of all parameters via grid search comes with an increase in runtime.

Difference to Nearest Neighbor Prediction

A very intuitive approach of confidence estimation is based on the nearest neighbor prediction, which refers to the average response value of the nearest neighbors [12]. It assumes that the response value of a new instance x^* should be similar to the average response of its m nearest neighbors:

$$cs_{\text{DiffNN}}(x^*) = 1 - \left| \frac{\sum_{i=1}^m y_i}{m} - \hat{y}^* \right|. \tag{4.6}$$

An appropriate value for m can be obtained via cross-validation. We also tested a modified version of DiffNN where m is set to five, called Diff5NN in the following. Diff5NN requires no optimization but shows a slightly reduced performance.

Sensitivity Analysis of Local Variance and Local Bias

Bosnić and Kononenko [13] introduced confidence estimation based on the local sensitivity of a regression model. Sensitivity analysis determines how much the model is affected if we modify the training dataset. By introducing a local change into the learning data, we can explore the sensitivity of the regression model in this very local area

of the data. For this, we extend the training data by the predicted instance x^*. The response value of our new learning example is set to $\hat{y}^* + \delta(y_{max} - y_{min})$, where $\hat{y}^*$ is the predicted response value of x^*, δ is a sensitivity parameter, and y_{max} and y_{min} are the maximum and minimum response values of the training data, respectively. To measure the effects of this change, we predict the response value $\hat{y}^*_\delta$ of x^* using a regression model trained on the updated dataset. This approach is not strictly model-dependent since it does not rely on certain properties of a model but treats the model as a black box.

For predicting confidence values, several updated datasets with different sensitivity parameters $\delta \in \Delta = \{0.01, 0.1, 0.5, 1.0, 2.0\}$ are used. The local variance approach estimates how strong the predicted response value is changed by local changes in the training data:

$$cs_{\text{LocalVar}}(x^*) = 1 - \frac{1}{|\Delta|} \sum_{\delta \in \Delta} (\hat{y}^*_\delta - \hat{y}^*_{-\delta}). \tag{4.7}$$

The bias estimator measures how unstable the prediction is by expressing the amount of local bias:

$$cs_{\text{LocalBias}}(x^*) = 1 - \left| \frac{1}{2|\Delta|} \sum_{\delta \in \Delta} (\hat{y}^*_\delta - \hat{y}^*) + (\hat{y}^*_{-\delta} - \hat{y}^*) \right|. \tag{4.8}$$

We use the absolute value of the local bias since we are not interested in which direction the predictor is more unstable.

Local Cross-Validation

Local regression models, such as locally weighted regression, are often able to increase the prediction accuracy by adapting to local properties of the input space [2]. This idea has been adapted as a confidence estimator [12]. The estimator calculates the errors made by a locally trained model on the m nearest neighbors of the test instance. However, it does not consider errors made by one particular model. Instead, it tests whether a local part of the input space can in general be modeled with the given regression model. The errors of the local model are calculated by a leave-one-out cross-validation. The local environment $E(x^*, m)$ of x^* in training dataset D is defined as a set of the m nearest neighbors, the m instances $\{(x_1, y_1), \ldots, (x_m, y_m)\} \subseteq D$ with the

smallest Euclidean distance $d(x_i, x^*)$ to x^*. For every neighbor $(x_i, y_i) \in E$, a regression model is trained on $E \setminus (x_i, y_i)$. Then, the response $\hat{y}_i$ of x_i is predicted with this model and the absolute prediction error $\hat{\epsilon}_i = |\hat{y}_i - y_i|$ is calculated. By weighting the instances according to their distance to x^*, we receive the following confidence estimator:

$$cs_{\text{LocalCV}}(x^*) = 1 - \frac{\sum_{i=1}^{m} e^{-0.5d(x_i,x^*)^2} \hat{\epsilon}_i}{\sum_{i=1}^{m} e^{-0.5d(x_i,x^*)^2}}. \tag{4.9}$$

Obviously, estimation with LocalCV requires long runtimes, since the leave-one-out cross-validation has to be repeated for every single instances x^*. To reduce the runtime, we set m to $\min\{\frac{n}{20}, 50\}$.

Bagging

Another confidence estimation that treats the regression model as a black box is bootstrap aggregation of multiple predictions models, also known as bagging. It has been observed that bagged aggregates can increase the prediction performance [15], but it has been also used to estimate the reliability of predictions [60, 82]. Given our training dataset D, we create $m = 50$ new datasets D_i of the same size as D by uniformly sampling with replacement instances from D. Every dataset D_i is used to train a regression model and to predict our novel instance x^*, resulting in m predicted response values $\hat{y}_i^*$. Since we expect agreement among the predictors in case of a reliable prediction, the final confidence estimator is based on the variance of the predicted responses:

$$cs_{\text{bagging}}(x^*) = 1 - \frac{1}{m-1} \sum_{i=1}^{m} (\bar{y^*} - \hat{y}_i^*)^2, \tag{4.10}$$

where $\bar{y^*}$ denotes the mean of all predicted response values $\hat{y}_i^*$.

Predictive Variance

A classic approach of confidence estimation is the use of the predictive variance of Bayesian models. In particular, Gaussian processes have been successfully applied for AD estimation [156]. However, in contrast to the presented confidence estimators above, this way of estimation is strictly model-based and can only be applied if it is possible to estimate the models predictive variance. If the predictive variance is very small, we would assume the model to be very confident about this prediction. In contrast, if

the variance is rather high, the confidence is low. For a linear least squares regression model, the predictive variance is defined as $x^*(X^TX)^{-1}x^{*T}$, where X is the feature matrix of our linear regression.

We did not consider the predictive variance in this work for two reasons: First, confidence estimation using the predictive variance is only possible for certain models and, hence, not independent of the model as the other presented estimators. Second, although the confidence values estimated by LocalVar do not equal the predictive variances of the linear regression, it is safe to assume that LocalVar approximates the behavior of the predictive variance. In our experiments, we observed that LocalVar and the predictive variance represented the same order of the instances, regarding their error. Thus, both estimation approaches yield the same estimation quality.

4.2.2 Confidence Estimators CONFINE and CONFIVE

In the following, we introduce our two novel confidence estimators, CONFINE and CONFIVE. Both confidence estimators are model-independent and can be applied with any regression model. This has two advantages: First, both estimators do not impose a specific regression model. Second, the runtimes are independent of the model. Even if applied with computationally expensive models, CONFINE and CONFIVE estimate the confidence of an individual prediction with only a small runtime overhead.

CONFINE – Errors of Nearest Neighbors

Our first confidence estimator, CONFINE, is based on the squared error (SE) of the surrounding training instances. It has been adapted from an estimation approach of Dimitrov *et al.* [50], which has been introduced in the previous chapter as estimator Acc. If the SE of the m nearest neighbors is already very high, we do not expect the model to be very good on novel instances either. Thus, a large error in the local environment results in a low confidence score, whereas a low error results in a large score:

$$cs_{\text{CONFINE}}(x^*) = 1 - \frac{1}{m}\sum_{i=1}^{m}\hat{\epsilon}_i^2.$$

The prediction errors $\hat{\epsilon}_i$ can be obtained by predicting the response values of the training dataset using a model trained on the same data or by performing a cross-validation on the training data. The optimal value of m is obtained using five nested two-fold cross-validations on the training dataset by averaging the values of m resulting in the highest estimation quality of each fold.

CONFIVE – Variance of Nearest Neighbors

Our second confidence estimator, CONFIVE, is based on the variance of the response values of the m nearest neighbors of x^*. CONFIVE is a generalization of estimator LE from Chapter 3. It assumes that a large variance of the responses in a local region cannot necessarily be modeled with a regression approach. This is especially true if a linear model is applied. Thus, large variances result in a low confidence score, whereas small variances result in a large score:

$$cs_{\text{CONFIVE}}(x^*) = 1 - \frac{1}{m-1}\sum_{i=1}^{m}(\bar{y} - y_i)^2.$$

The optimal value of m is also obtained using five nested two-fold cross-validations.

Kernel-based Approaches of CONFINE and CONFIVE

As an alternative to the confidence estimators presented above, we propose two alternative estimators CONFINE* and CONFIVE*. Instead of relying on a fixed local environment, they use a Gaussian kernel to re-weight instances in the environment according to their distance to the test instance x^*. Hence, we put more weight on instances that are close to x^* and less weight on instances that are very distant to x^*:

$$cs_{\text{CONFINE}^*}(x^*) = 1 - \frac{\sum_{i=1}^{n} e^{-0.5d(x_i,x^*)^2}\hat{\epsilon}_i^2}{\sum_{i=1}^{n} e^{-0.5d(x_i,x^*)^2}} \tag{4.11}$$

$$cs_{\text{CONFIVE}^*}(x^*) = 1 - \frac{\sum_{i=1}^{n} e^{-0.5d(x_i,x^*)^2}(\bar{y} - y_i)^2}{\sum_{i=1}^{n} e^{-0.5d(x_i,x^*)^2}}. \tag{4.12}$$

To save runtime, both estimators are defined with a fixed kernel-width. Using this approach might result in a slightly reduced estimation quality. However, both estimators require no optimization and are computationally very cheap. Due to their reduced estimation quality, we show respective performance details only in the appendix.

4.2.3 Confidence Intervals

When the response of a novel instance x^* has been predicted using the trained regression model, we apply our confidence estimators for this particular prediction. Since obtained confidence scores $cs(x^*)$ determined by different estimators are not necessarily comparable nor interpretable, we calculate normalized confidence scores $ncs(x^*)$ as described below. We first predict the responses of the training data and then apply the confidence estimator for each prediction. The normalized confidence score $ncs(x^*)$ of a novel instance x^* is then calculated by determining the fraction of predictions from the training dataset with a smaller confidence value than x^*. Thus, an ncs of 0.8 implies that 80% of the instances in the training dataset have been predicted with a smaller confidence value. Using this approach, we obtain meaningful and interpretable scores which lie between zero and one.

Normalized confidence scores are useful indicators of the prediction error. We assume that the higher the score of a predicted instance, the more likely this instance was predicted with a small error. Still, it is not obvious how such a score relates to an actual error. For example, given an ncs of 0.9, it is not obvious how large the actual prediction error is.

Confidence intervals are a more intuitive concept than arbitrary scores. Instead of predicting only the response $\hat{y}$ and the corresponding normalized confidence score ncs, we predict an interval based on ncs which includes the correct response value y with a probability of 0.8. Since reliable predictions with a large ncs have, on average, a smaller SE, we expect them to have smaller confidence intervals. We can relate an ncs to confidence intervals (e.g., 80% confidence intervals) as follows.

The borders of confidence intervals are estimated on the training dataset. For this, we first predict the responses of the training instances using a model trained on the training dataset. Subsequently, the normalized confidence scores of all training instances are first estimated using a confidence estimator based on the training data and then sorted in ascending order $\{ncs_1, \ldots, ncs_n\}$. For every possible normalized confidence score nsc_i, we collect the errors of instances with an ncs of $\{ncs_{i-50}, \ldots, ncs_i, \ldots, ncs_{i+50}\}$ where possible. Otherwise, we use a reduced set of errors. Based on this set of errors E, we calculate the 0.1 quantile $q_{ncs_i}(0.1)$ and the 0.9 quantile $q_{ncs_i}(0.9)$ as interval borders for each ncs_i. By using empirical quantiles,

we do not assume a normal distribution and, hence, are independent of the underlying error distribution.

When predicting the response $\hat{y}^*$ and the confidence score ncs^* of a novel instance x^*, we calculate the 80% confidence interval as $[\hat{y}^* + q_{ncs^*}(0.1), \hat{y}^* + q_{ncs^*}(0.9)]$. In case of MHC-I–peptide binding prediction, we thus obtain a range of affinities instead of a plain affinity value. If the upper bound of the 80% confidence interval is below 500 nM, we can assume a probability of 90% that the peptide is going to be a real binder.

Note, in case of CONFINE, we could also simply use only the errors of the nearest neighbors of an instance to estimate intervals. However, we found this naive estimate to perform worse than the above described approach. One possible reason is the fact that such a naive approach considers only errors of instances in the nearest neighborhood. Whereas the more advanced confidence interval estimation applied to CONFINE considers the errors of instances with a similar error landscape regardless of their position.

4.2.4 Evaluation

The width of the predicted confidence interval is an indicator of the prediction error. Predictions with a large SE should yield a broad confidence interval, while predictions with a low SE are assumed to have a small confidence interval. Consequently, we can assess the quality of estimates by calculating the correlation ρ between the absolute prediction errors $|\hat{\epsilon}|$ and the corresponding confidence interval widths $ciw = q(0.9) - q(0.1)$. The resulting correlation is then normalized by the correlation obtained from a perfect confidence estimator. We define the *confidence–error correlation* (CEC) as

$$\text{CEC} = \frac{\rho(ciw, |\hat{\epsilon}|)}{\rho(\text{sort}(ciw), \text{sort}(|\hat{\epsilon}|))},$$

where *sort* is a sorting function. Since we wish to calculate an 80% confidence interval, we obviously also require about 80% of the test errors to lie within the confidence interval.

In the left-hand plot of Figure 4.1, we show absolute prediction errors of 200 example instances and the width of their corresponding confidence intervals estimated by CONFINE. We can observe that a considerable number of instances with a small error exhibit a small confidence interval, as we would have expected. The corresponding estimation quality in terms of the CEC equals 0.3. At a first glance, the resulting CEC does not seem all that impressive. It should be noted, however, that we do not expect

a perfect correlation between the error and the confidence interval width. It is only required that the error is smaller than the confidence interval. The right-hand plot of Figure 4.1 shows that a CEC of 0.3 already leads to a considerably reduced confidence interval for confident predictions. For larger confidence scores, we not only observe smaller absolute errors, but also smaller confidence intervals. While correlation is thus obviously not the perfect measure, we used it because of its rather intuitive nature.

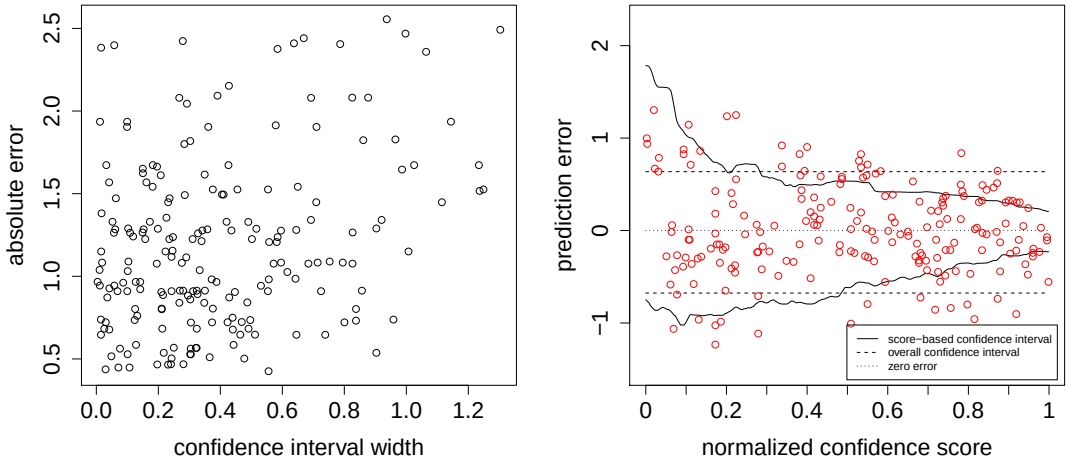

Figure 4.1: Example of estimating confidence intervals. In this example, we estimated the confidence intervals of 200 instances. The left-hand plot shows the confidence interval widths and the corresponding absolute errors. The corresponding CEC equals 0.3. Although the CEC is not very large, it is possible to see an increased number of small confidence intervals for predictions with a low error. In the right-hand plot, the estimated confidence interval borders are displayed. In addition, every prediction defined by its prediction error and its normalized confidence score is depicted by a red circle. On average, the absolute error is smaller for predictions with a high *ncs* and a small confidence interval.

In applications like epitope-based vaccine design, we are only interested in highly reliable predictions. To account for that, we also measure the *confidence-associated prediction improvement* (CAPI). Therefore, we calculate by what percentage the MSE is reduced if we consider only the top 20% predictions, i.e. the 20% predictions with the smallest confidence intervals.

4.2.5 Datasets

We benchmarked our methods on three different types of datasets: a synthetic dataset, an MHC–peptide binding dataset, and several 3D-QSAR datasets. The synthetic dataset was created using the Friedman function [68]. This test function has five relevant features where two are linear and three are nonlinear. All other features have no influence on the function value. We created datasets of different sizes $\{100, 500, 1000\}$ by sampling 10, 50, 100, or 500 features from $[0, 1]$ uniformly. The response values were calculated by applying the Friedman function to the first five features and introducing different levels of Gaussian noise $\mathcal{N}(\mu = 0, \sigma \in \{0.1, 0.5, 1.0, 2.0\})$ into the response value.

For our second study, we extracted peptides of length nine with known binding affinities to molecules from 12 different MHC class I alleles from the IEDB benchmark dataset [137]. We chose the 12 HLA alleles for which more than 1,000 examples are available: HLA-A*01:01, HLA-A*02:01, HLA-A*02:02, HLA-A*02:03, HLA-A*02:06, HLA-A*03:01, HLA-A*11:01, HLA-A*31:01, HLA-A*33:01, HLA-A*68:01, HLA-A*68:02, and HLA-B*07:02. We encoded the peptides using sparse binary encoding in which each peptide sequence is encoded by a 180-dimensional bit vector, where 9 times 20 bits represent each of the 20 amino acids at all nine positions [54]. For the response, we use the logarithm of the given IC_{50} values. The resulting datasets contain between 1,157 and 3,089 instances, each encoded by 180 features.

To show that CONFINE and CONFIVE can also be applied to other regression problems, we used eight well-studied benchmark datasets from 3D-QSAR [177], which consist of 66 to 397 chemical compounds. We calculated up to $1,872$ features using DragonX 1.4.0 [179]. Note that the QSAR datasets exhibit quite different properties with respect to the number of data points, the size of the input feature space, and the coverage of that input space. Due to the large number of features some instances can take values that might be unique within the whole dataset. In contrast, it is very unlikely that there is no peptide with the same amino acid at one particular position in a dataset of more than 1,000 peptides. More importantly, the QSAR datasets contain a lot less data points than the IEDB benchmark datasets.

4.3 Results and Discussion

Since some estimators perform very similar, we restricted our evaluation to CONFINE and CONFIVE, NoNN, AvgDist, LocalVar, LocalCV, and bagging. For details on the performance of the other predictors, we refer to the appendix.

4.3.1 Influence of Dataset Size, Features, and Noise

In an initial experiment, we analyzed how the introduced confidence estimators are influenced by the dataset size, the number of features, and noise in the data. The experiment was performed on the synthetic dataset, which gives us full control over these parameters. We performed five five-fold cross-validations on randomly generated artificial datasets, each with a different number of instances, features, and noise levels in the response variable, resulting in 48 combinations. The estimation quality of the confidence estimators in terms of the average CEC (avgCEC) are shown for different parameter combinations in Figure 4.2 and Table 4.1. See Tables D.23 and D.24 in the appendix for performance details of all estimators and details on qualities regarding the confidence-associated prediction improvement (CAPI).

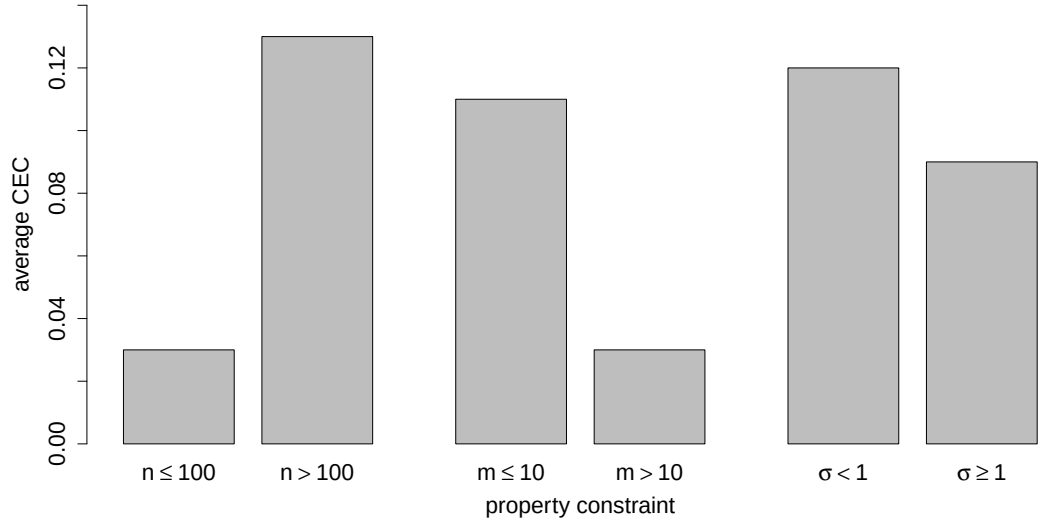

Figure 4.2: Influence of dataset size, features, and noise. The barplot shows the average CEC of the confidence estimators on artificial data with different property constraints: the dataset size n, number of selected features m, and the noise level σ.

We found that the dataset size has the strongest influence on the estimation performance. On very small datasets with only 100 instances, the estimators yield an avgCEC of 0.05. When considering datasets with more than 100 instances, the avgCEC of all estimators increases to 0.13. In addition, we observe a CAPI of 9% on small datasets and a CAPI of 21% if more than 100 training instances are given. Still, not all estimators are equally sensitive to the dataset size. While the avgCEC of estimator CONFIVE is only slightly influenced by the dataset size, the avgCEC of CONFINE increases by 0.17 when considering sufficiently large datasets. For large datasets, CONFINE shows a CAPI of 35%. Moreover, note that when the dataset size is increased from 100 to 1,000 instances, the standard deviation of the CECs decreased from 0.27 to 0.11.

We also observe that noisy features and noise in the responses have an influence on the quality of confidence estimates. Particularly when the initial number of features is high or the dataset size is low, noisy, non-predictive features were included in the feature set. When more than 10 features were selected, the avgCEC of all estimators decreases by 0.09. Similar results are obtained regarding the noise in the data. When random values with a low standard deviation ($\sigma < 1.0$) were added to the data, the avgCEC is up to 0.06 larger compared to avgCECs obtained on data with a higher noise level.

As expected, when we consider only datasets with 1,000 instances, ≤ 10 selected features, and a noise level of $\sigma = 0.1$, all estimators yield their best performance. In particular, CONFINE performs well, yielding an avgCEC of 0.30 and a CAPI of 0.48, i.e. the 20% of predictions that have the smallest confidence intervals exhibited a 48% lower MSE than an average prediction.

From our results, we can conclude that – not surprisingly – a larger amount of training data results in more robust estimates and higher confidence estimation quality. In addition, a good feature representation and a low noise level simplify confidence estimation. Clearly, these properties are not independent of each other. Distinguishing between informative and non-informative features is easier for large datasets, since the difference between noise and information becomes more evident. The same holds for datasets with a low level of noise, resulting in less noisy features. Since most confidence estimators discussed here inspect local properties of the input space, they rely on good feature representation. If noisy features are part of the feature set, instances in the local environment are not necessarily similar to the test instance and, thus, provide no

Table 4.1: Performance of confidence estimators on artificial data with different properties

	$n \leq 100$	$n > 100$	$m \leq 10$	$m > 10$	$\sigma < 1.0$	$\sigma \geq 1.0$	best
CONFINE	0.05	0.22	0.19	0.05	0.21	0.15	0.30
CONFIVE	-0.02	0.05	0.03	-0.01	0.04	0.02	0.07
AvgDist	0.02	0.12	0.10	0.03	0.11	0.08	0.16
Bagging	0.11	0.20	0.18	0.11	0.19	0.16	0.25
Diff5NN	0.01	0.17	0.14	0.02	0.14	0.11	0.29
LocalCV	0.01	0.05	0.04	0.02	0.04	0.03	0.05
LocalVar	0.00	0.12	0.10	0.00	0.09	0.08	0.16
NoNN	0.05	0.12	0.12	0.03	0.11	0.09	0.16

For every confidence estimator, we calculated the average CEC by considering datasets with a different number of instances n, a different number of selected features m, and a different noise level σ. In the last column, we show the average CEC for the best parameter combination ($n = 1{,}000$, $m \leq 10$, $\sigma = 0.1$).

reliable confidence information. Furthermore, given more instances in the dataset, we can define a local environment with a smaller diameter since the density of instances is higher. Consequently, the nearest neighbors are more similar to the test instance and contain more relevant confidence information.

4.3.2 Evaluation on IEDB Benchmark Datasets

To compare CONFINE and CONFIVE with existing confidence estimators, we performed five five-fold cross-validations on the IEDB benchmark datasets.

Our estimators CONFINE and CONFIVE, as well as DiffNN, with an avgCEC of around 0.25, perform better than other estimators, which yield an avgCEC around 0.12 (see Table 4.2). In addition, the best three confidence estimators show a CAPI of up to 39%, while the other estimators yield an average improvement of only 19%. In Figure 4.3, a boxplot shows the performance of the different confidence estimators. The boxplot confirms that CONFINE, CONFIVE, and DiffNN perform superior. Interestingly, on allele A*01:01, all confidence estimators yield a very high CEC (> 0.7). See Table D.25 in the appendix for performance details of all presented estimators.

Table 4.2: Performance of confidence estimators on biological datasets using linear regression

confidence	MHC			QSAR		
estimator	CEC	CAPI	runtime [ms]	CEC	CAPI	runtime [ms]
CONFINE	0.27	0.39	2	0.08	0.09	1
CONFIVE	0.24	0.35	2	0.09	0.13	1
AvgDist	0.11	0.18	2	-0.02	-0.10	1
Bagging	0.13	0.18	1	0.20	0.35	1
DiffNN	0.24	0.32	2	0.00	-0.14	1
LocalCV	0.16	0.27	214	0.08	0.10	353
LocalVar	0.10	0.17	482	-0.08	-0.22	430
NoNN	0.10	0.17	2	-0.03	-0.09	1

For every confidence estimator, the avgCEC, the confidence-associated prediction improvement (CAPI), and the time for an individual estimation in milliseconds on the MHC datasets and on the QSAR datasets is shown.

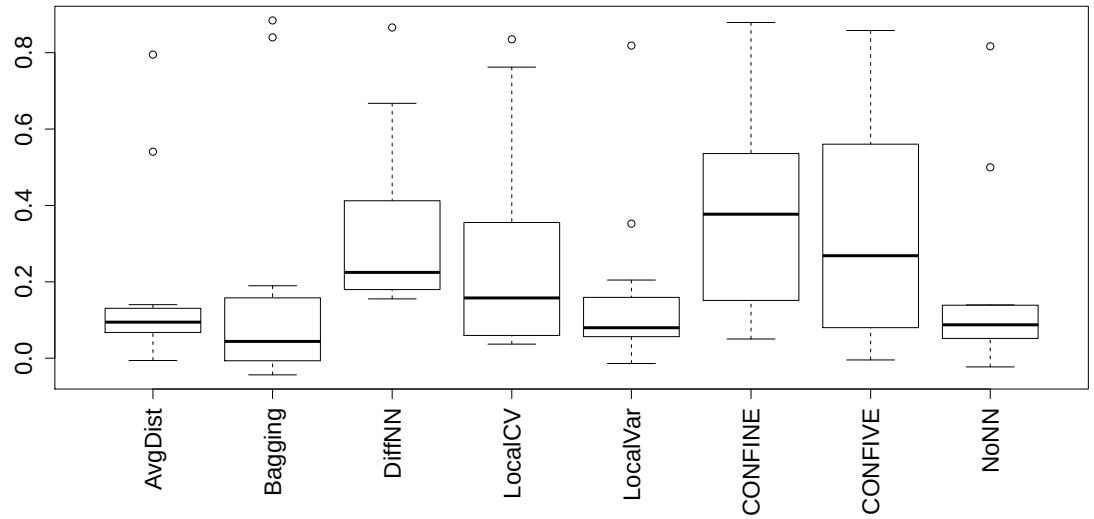

Figure 4.3: Boxplot of CECs of CONFINE, CONFIVE, and other estimators on the IEDB benchmark datasets.

In Figure 4.4, we plotted the avgCECs of CONFINE on the different IEDB benchmark datasets. Note that CONFIVE performs very similar to CONFINE. We observe

that the performance differs greatly from dataset to dataset. In particular, for allele A*02:02, A*02:03, and A*68:01, CONFINE performs poorly. In contrast, the avgCEC for allele A*01:01 and B*07:02 is even larger than 0.8. For most other dataset, CONFINE can provide good confidence estimates, yielding an avgCEC between 0.2 and 0.5.

Estimating confidences with CONFINE and CONFIVE is possible with only a minor computational overhead. Estimating the confidence intervals of one individual prediction requires about 2 ms on a 2GHz dual-core AMD Opteron with 4 GB of RAM using our R implementation. But also most other estimators need about 2 ms for an estimation. Only estimators LocalCV and LocalVar require more than 200 ms for an individual estimation. For each estimation, both estimators train multiple regression models, which results in a huge computational overhead. Bagging also uses multiple regression models for its estimations. However, these models are trained only once, making bagging faster than LocalCV and LocalVar. Note that confidence estimation with bagging can, however, require long runtime if it is applied with non-linear models (see Section 4.3.4).

Our results suggest that CONFINE and CONFIVE often perform better than most other confidence estimators while being comparable to DiffNN. They perform particularly well due to the large amount of training data contained in the IEDB benchmark datasets. Interestingly, CONFIVE performs well on the IEDB benchmark datasets, while yielding a poor performance on artificial data. The superior estimation quality of CONFINE and CONFIVE is of special interest for tasks that rely on high-quality predictions. If we would consider only peptides as epitopes candidates that could be predicted with a very small confidence interval, we would expect a considerably lower error rate for these predictions and, thus, more reliable epitope candidates. In addition, CONFINE and CONFIVE require only a small computational overhead.

4.3.3 Evaluation on 3D-QSAR Datasets

To show that our confidence estimators can also be applied to other regression problems than MHC-I–peptide binding prediction, we repeated the above study on the QSAR benchmark datasets.

The results differ slightly from the results obtained on the IEDB benchmark datasets (see Table 4.2). Bagging performs best, yielding an avgCEC of 0.20, while estimators

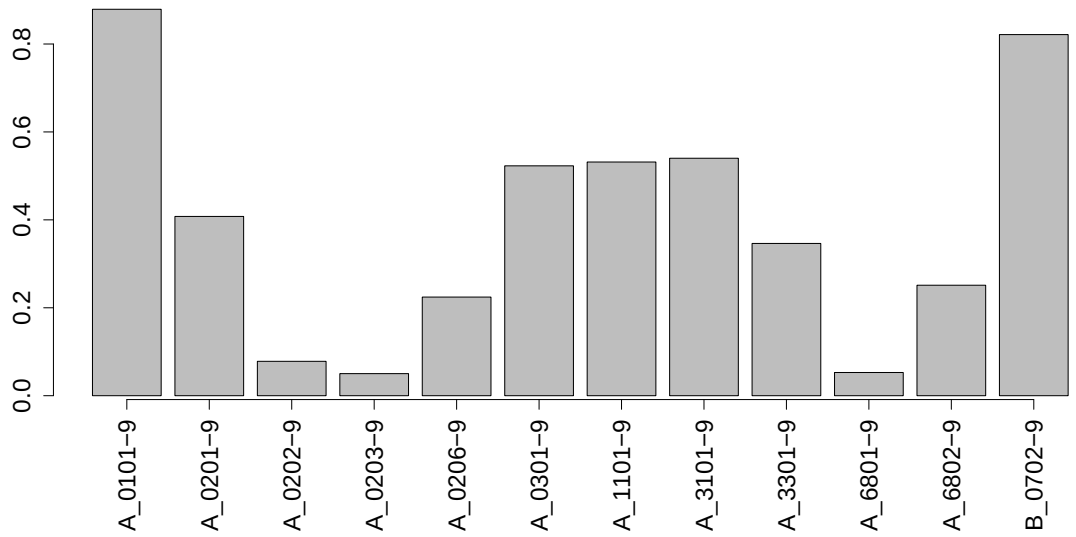

Figure 4.4: Barplot of the avgCEC of CONFINE on the different IEDB benchmark datasets.

CONFINE, CONFIVE, and LocalCV perform second-best, with avgCECs around 0.08. Since most estimators have been shown to be very sensitive to the dataset size, we also calculate the avgCEC considering only QSAR datasets with more than 100 learning examples. On large QSAR datasets, the avgCEC of most estimators is considerably improved. In the case of CONFINE and CONFIVE, the avgCEC improves to 0.13 and 0.15, respectively. A similar trend can be observed when considering prediction improvement. The estimation time of all estimators but LocalCV and LocalVar is only about 1 ms for an individual prediction. Again, see Table D.25 in the appendix for performance details of all estimators.

Our findings support the results obtained in Section 4.3.2. CONFINE and CONFIVE perform comparable to bagging and LocalCV on datasets of medium size with more than 100 training instances. Interestingly, commonly used AD estimators such as AvgDist, DiffNN, and NoNN often fail to give reasonable error estimates. On very small datasets our estimators exhibit also problems giving good confidence estimates. However, if sufficient training data is available, users can benefit from confidence estimates made by CONFINE and CONFIVE.

4.3.4 Confidence Estimation for Nonlinear Models

To show that CONFINE and CONFIVE can be also applied to nonlinear regression models, we repeated the evaluation on the MHC and QSAR datasets using SVR with an RBF kernel and the full set of features. Details on SVR can be found in Section 2.1.2. Since estimators LocalCV and LocalVar require too much runtime, we excluded them from this study. The parameters of the SVR and the estimators were optimized by performing cross-validations on the training dataset. Since optimizing SVRs requires more runtime, we restricted the evaluation to only one five-fold cross-validation. The results are shown in Figure 4.5 and Table 4.3.

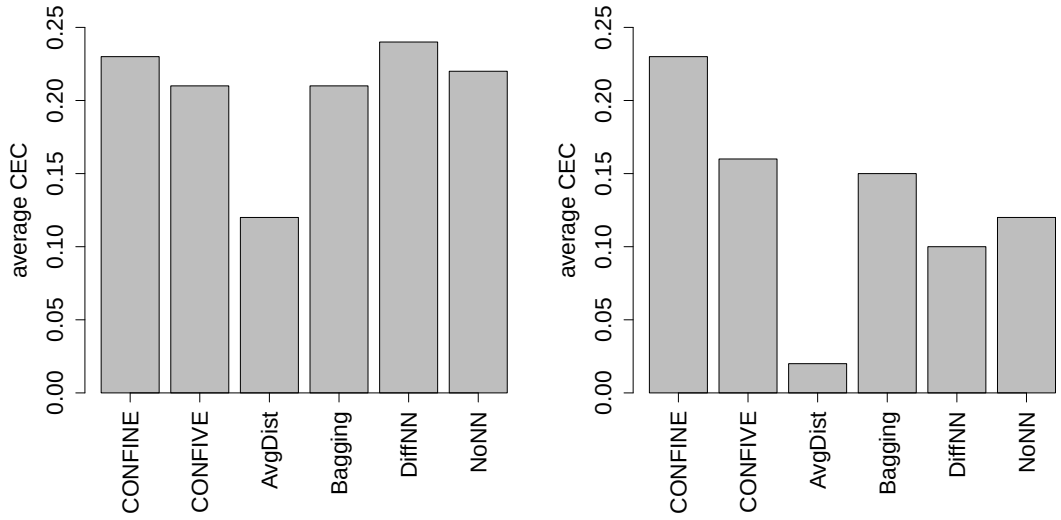

Figure 4.5: The average CEC of confidence estimators applied to SVR models on the IEDB benchmark datasets (left-hand barplot) and QSAR datasets (right-hand barplot). Due to the large runtimes of LocalCV and LocalVar, we excluded both estimators from this study.

Confidence estimators CONFINE, CONFIVE, and bagging perform best (see Figure 4.5). In particular, CONFINE yields the largest avgCEC and also a very large CAPI (see Table 4.3). The avgCEC on the MHC datasets is comparable to our previous results, whereas the avgCEC on the QSAR data is higher. In particular, the avgCEC of CONFINE and CONFIVE is considerable increased if being applied with nonlinear SVR. Again, we observe that CONFINE and CONFIVE perform better on

larger QSAR datasets.

Normalized confidence scores and confidence intervals can be predicted with only a small computational overhead using estimators CONFINE, CONFIVE, AvgDist, DiffNN, and NoNN. On the MHC and QSAR datasets they require between 9 to 44 ms for an individual prediction. The different estimation times between estimators and the differences compared to our previous results using linear regression originate from the different number of features. Confidence estimation based on bagging requires the largest runtime of up to 3 s for an individual prediction. The performance of all confidence estimators is shown in Table D.26 in the appendix.

Table 4.3: Performance of confidence estimators on biological datasets using SVR

confidence	MHC			QSAR		
estimator	CEC	CAPI	runtime [ms]	CEC	CAPI	runtime [ms]
CONFINE	0.23	0.41	9	0.23	0.32	9
CONFIVE	0.21	0.34	10	0.16	0.21	10
AvgDist	0.12	0.23	9	0.02	0.03	12
Bagging	0.21	0.50	374	0.15	0.17	3,064
DiffNN	0.24	0.35	9	0.10	0.20	10
NoNN	0.22	0.18	9	0.12	0.14	44

For every confidence estimator, the average CEC, the average confidence-associated prediction improvement (CAPI), and the time for an individual estimation in ms on the MHC datasets and on the QSAR datasets is shown.

Our findings support the assumption that CONFINE and CONFIVE show similar behavior when being applied in combination with nonlinear regression models. In particular, CONFINE shows again a very good and very robust performance, while being fast at the same time. Although confidence estimation based on bagging shows also good performance, bagging is less practical for real-world applications. If bagging is applied with a time-consuming regression model, runtimes can be quite high. In contrast, CONFINE and CONFIVE perform independent of the actual regression model, making them even more interesting for real-world applications like epitope-based vaccine design.

4.3.5 Evaluation of Confidence Intervals

To show that a score-based 80% confidence interval contains as many instances as an interval estimated independently from a confidence score, we compared it with a general 80% confidence interval. Therefore, we calculate the 0.1 quantile and the 0.9 quantile of the SEs of all training instances without considering the confidence scores. While the score-based confidence intervals are expected to be smaller for large ncs, the general interval is always of the same size. However, in both cases, we expect that about 80% of the test instances are within the confidence interval.

On the artificial dataset, we observe an almost equal fraction of 0.72 and 0.73 of the instances in the score-based interval and the general interval, respectively. If we consider only datasets with more than 100 instances, we find about 77% of the instances within both confidence intervals. Among the different confidence estimators, we could not find considerable differences. We obtain similar results for the IEDB benchmark datasets. A fraction of 0.74 and 0.77 of the instances are within the score-based interval and general interval, respectively. In contrast, only 54% and 55% of the instances from the QSAR datasets fall into the respective confidence intervals. However, when considering only QSAR datasets with more than 100 training examples, about 67% of the instances are within both confidence intervals.

Our results suggest that score-based confidence intervals contain nearly the same fraction of instances as general confidence intervals. In particular, on the large IEDB benchmark datasets, the fraction of instances within the confidence interval converges to 0.8. Furthermore, since the widths of score-based confidence intervals are correlated with the absolute prediction error, they are a very intuitive measure of confidence.

4.3.6 Predicting the Estimation Performance

Although confidence estimation can give valuable information in addition to plain response values, the quality of estimates differs from dataset to dataset (see Figure 4.4). To answer the question to what extend we can predict the quality of confidence estimates, we compared the CECs obtained from the training data ($\text{CEC}_{\text{train}}$) with the CECs obtained from the corresponding test data (CEC_{test}) for all estimators.

On the artificial dataset, we observe an average correlation coefficient ρ between $\text{CEC}_{\text{train}}$ and CEC_{test} of 0.16. When considering only datasets with more than 100

training examples, the average ρ increases to 0.38. The same trend can be observed in the biological datasets. For the considerably larger MHC datasets, we receive an average ρ between $\text{CEC}_{\text{train}}$ and CEC_{test} of 0.91, while no correlation is observed for the fairly small QSAR datasets. In particular, the training CECs of CONFINE and bagging show a comparably good correlation with their corresponding $\text{CECs}_{\text{test}}$ for all datasets. See Table D.27 in the appendix for a detailed overview.

If a sufficient amount of training data is available, the performance of confidence estimators correlates well with their performance on the training data.

4.3.7 Confidence Estimation in Epitope-Based Vaccine Design

Confidence estimation of individual regression predictions is of special interest in critical tasks which rely on high-quality predictions. MHC–peptide binding prediction, as an important step in the design process of epitope-based vaccines, is such a critical task. Only if we discover reliable epitope candidates, an immune response is triggered and the immune system can "remember" the corresponding antigen. Estimating the confidence in affinity predictions can help to distinguish between reliable epitope candidates and candidates that are likely to be false positives. In the following, we show that confidence estimation using CONFINE can help to improve epitope discovery and epitope engineering.

Parkhurst *et al.* [133] studied how mutations in known epitopes can improve MHC–peptide binding and T-cell response. They analyzed two epitopes of gp100, which is a melanocyte lineage-specific membrane glycoprotein expressed in melanoma. Peptides G9_{209} and G9_{280} show weak binding to HLA-A*02:01 and are able to induce an immune response [100]. The authors manually engineered both epitopes by designing 21 and 17 analogous peptides for G9_{209} and G9_{280}, respectively. Therefore, they introduced one or two mutations at positions 1, 2, 3, and 9. They experimentally determined the binding affinity of all analogs of G9_{209} and G9_{280} and found an improved binding affinity in 19 and 16 cases, respectively. In 2003, Bhasin and Raghava [6] performed an *in silico* study on the same data. Using a matrix-based classification approach, they predicted 16 and 11 peptides to bind stronger than G9_{209} and G9_{280}, respectively. The remaining eight epitopes could not be found by the method.

We repeated this study using the previously described linear regression approach. For training our model, we use available binding data for allele HLA-A*02:01 from the

IEDB benchmark dataset [137]. Peptides were sparsely encoded and the given IC_{50} values were log-transformed. In addition, we estimate confidence intervals for each prediction using CONFINE. In contrast to the matrix-based approach of Bhasin and Raghava [6], we predict that all of the 35 experimentally verified heteroclitic peptides yield a lower IC_{50} value than the wildtype peptide. In case of the $G9_{209}$ analogs, we predict two non-binders to bind to the MHC molecule (see Table 4.4). Interestingly, both non-binders yield the two largest confidence intervals among the predictions, which means that we can put only little confidence into the predicted affinities. If we were required to select an epitope candidate for a vaccine, we would not have chosen these peptides due to the low confidence scores. Furthermore, we can show that confidence estimates can also be beneficial to determine the quality of the predicted affinities among the binders. For the $G9_{209}$ and $G9_{280}$ analogs, our models show an MSE of 0.13 and 0.22, respectively. If we consider only the five predictions with the smallest confidence interval, the MSE decreases to 0.09 and 0.07, respectively. Hence, high-confidence predictions yield a 35% to 68% lower error than an average prediction. Considering only reliable binders as epitope candidates does decrease the chance of being misled by erroneous predictions. This can be beneficial for epitope discovery, which requires to select reliable epitope candidates. See Table D.29 for an overview of all 21 predictions of the $G9_{209}$ analogs.

The study of Parkhurst *et al.* [133] required a lot of experimental effort since for more than 30 peptides the MHC-I–peptide binding affinity had to be determined. To avoid these costly and time-consuming experiments, we suggest an alternative, computational approach based on a genetic algorithm to engineer epitopes. A similar approach that aims at finding the peptide with the strongest affinity to an MHC molecule has been recently published [104]. However, we are not interested in finding the best binder but in engineering a given epitope. Most importantly, our engineering approach aims at optimizing the predicted affinity and the confidence of the corresponding prediction at the same time. The peptide predicted with the highest binding affinity might be a low confidence prediction that exhibits a higher error than other predictions. To exclude low-confidence predictions, we aim at minimizing the upper bound of the predicted 80% confidence interval. The upper bound of this confidence interval corresponds to the maximum IC_{50} value the peptide is going to have with a probability of 90% ($IC_{50}^{90\%}$).

Table 4.4: Binding affinities and confidence intervals of $G9_{209}$ and selected analogs

peptide	exp IC_{50}	pred IC_{50}	lower bound IC_{50}	upper bound IC_{50}
ITDQVPFSV	172	156.2	50.9	759.9
WTDQVPFSV	716.7	158.7	28.3	1052.1
ITSQVPFSV	637.0	138.8	45.5	639.1
ILWQVPFSV	1.7	5.5	1.4	25.5
ILFQVPFSV	2.0	7.7	2.4	36.2
FLDQVPFSV	2.2	2.2	0.3	16.4
YLDQVPFSV	2.3	2.8	1.0	12.5

For each peptide, the experimentally determined IC_{50} value (exp IC_{50}), the predicted IC_{50} (pred IC_{50}), and the lower and upper bound of the 80% confidence interval are given (as IC_{50}). The first peptide is $G9_{209}$, followed by a selected set of analogs. The mutated positions are printed in bold. Note that we calculate the confidence interval width based on the log IC_{50} values.

In this way, we minimize the predicted IC_{50} value and maximize the corresponding confidence at the same time.

In addition, we have to consider two important constraints for our optimization:

1. The resulting peptide has to show a certain similarity to the wildtype peptide. Otherwise, the immunological memory might remember only the heteroclitic peptide but not the peptide of the pathogen. To avoid this problem, we restrict a candidate epitope to differ in at most two positions from the wildtype peptide.

2. Positions in the middle of the peptide are known to be important for TCR binding and, hence, should not be mutated. In case of allele HLA-A*02:01, it is known that residues 4 to 8 are less important for MHC-I–peptide binding but vital for the interaction of the MHC–peptide complex with the TCR [102]. Mutating one of these residues might result in a lower T-cell reactivity that could reduce the efficacy of the vaccine. Consequently, we fix these positions in the genetic algorithm.

For our optimization task, we use genetic algorithms implemented in the Python framework Pyevolve [134]. A genetic algorithm is a heuristic optimization technique

based on natural evolution. In every iteration, individuals of a population, peptides in our case, are mutated, combined, and selected according to their fitness. In our case, the fitness of a peptide is expressed as the $IC_{50}^{90\%}$. Peptides with a small $IC_{50}^{90\%}$ are more likely to be selected for the next iteration. Our approach uses a population consisting of 10 peptides and uses a mutation rate and crossover rate of 0.6 and 0.8, respectively. The peptides for the next generation are selected via rank selection. The optimization process terminates if the smallest $IC_{50}^{90\%}$ of a population remains unchanged for 50 iterations or if an upper bound of 200 iterations is reached. Note that due to the considerably small search space of 2,242 peptides, exhaustive enumeration would have been feasible for this particular optimization problem. Nevertheless, applying genetic algorithms results in a large speed-up. Our genetic algorithm approach requires less than 300 predictions to find the peptide with the smallest $IC_{50}^{90\%}$.

We applied our genetic algorithm approach to engineer epitopes $G9_{209}$ and $G9_{280}$ using the above described constraints. For epitopes $G9_{209}$ and $G9_{280}$, our approach returns peptides YLDQVPFSV and FLEPGPVTV with predicted IC_{50} values of 2.8 nM and 46.1 nM, respectively. Since there exists no experimental data that could prove the proposed binding affinity of the latter peptide, we concentrated our analysis on the engineered peptide of $G9_{209}$. Interestingly, Parkhurst *et al.* [133] found our epitope candidate to be the fourth strongest binder among their 21 analogous peptides with an IC_{50} value of 2.3 nM (see Table 4.4). We had a closer look at the three peptides exhibiting a stronger affinity than our epitope candidate. For two of the peptides, our linear regression approach predicts an IC_{50} larger than 2.3 nM. In accordance with the larger prediction error, the two corresponding confidence intervals are also larger than for our engineered peptide, showing the uncertainty of our prediction model. The third peptide with a stronger binding affinity than our epitope candidate is FLDQVPFSV with an IC_{50} value of 2.2 nM. Our regression approach confirms the peptide to be a stronger binder than the one obtained by the genetic algorithm. However, CONFINE seems to be unsure about this prediction, resulting in a fairly broad confidence interval and higher $IC_{50}^{90\%}$. After mining the literature, we found that Du *et al.* [61] assigned this peptide with an IC_{50} value of 17.7 nM. Hence, the correct affinity value might indeed be larger than 2.2 nM which makes it a less reliable candidate for an epitope. Although our epitope engineering strategy does not result in the peptide with the highest binding affinity, the obtained peptide is likely to be a reliable binder. By considering the

confidence of predictions, we are able exclude erroneous predictions and can select a suitable candidate for an epitope-based vaccine.

4.4 Conclusion

Epitope-based vaccines are a promising alternative to traditional vaccines. In the design process of epitope-based vaccines, biologists often rely on *in silico* MHC-I–peptide binding prediction as an approximation of T-cell reactivity. However, since computational prediction methods are prone to errors, epitope candidates suggested by prediction methods might be false positives. Due to the "black box" character of many state-of-the-art MHC-I–peptide binding prediction methods, it is usually not possible to detect a low quality prediction.

To overcome this "black box" character, we proposed two novel confidence estimators for regression, CONFINE and CONFIVE. They determine normalized confidence scores and confidence intervals that help biologists to rate the reliability of an individual affinity prediction. Both estimators are model-independent and can be applied with any regression model. In contrast to other methods, CONFINE and CONFIVE are computationally very efficient and can, thus, be added easily to existing predictors without a significant performance loss. By applying CONFINE and CONFIVE to an MHC-I–peptide binding prediction model, biologists do no longer rely on plain affinity values. Instead, they can benefit from the advantages of obtaining a range of affinities.

In an initial study on artificial data, we observe that CONFINE and CONFIVE, as well as other estimators, yield a better estimation performance on large datasets. A sufficient amount of training data helps to identify irrelevant features and increases the prospect of having adequate neighbors in the training dataset. We then compared CONFINE and CONFIVE with other existing confidence and AD estimators on a selected set of IEDB benchmark datasets. Our results suggest that CONFINE and CONFIVE give high-quality confidence estimates and perform better than existing estimators. The estimation quality is not equal on all tested alleles. However, users can predict the quality of estimates on novel data by considering the estimation quality on the training data. Similar results obtained using nonlinear support vector regression demonstrate that CONFINE and CONFIVE can be applied to nonlinear regression models as well. Only confidence estimation based on bagging performs comparably on

the tested datasets. However, depending on the regression method used, bagging can require a huge computational overhead. We also have seen that confidence intervals estimated by our two methods are comparable to fixed confidence intervals, while having the advantage of giving a very intuitive measure of confidence.

It is still a long way towards highly accurate confidence estimators that work equally well on all kind of data. A combination of multiple confidence estimators as well as an automated selection of the most appropriate estimator [11] could improve both the quality and the robustness of the estimation. Further, predicting not only the size of errors but also their sign will increase the amount of information gained from a confidence estimator. As an alternative, signed error estimates can be used to correct the prediction results and might increase the prediction performance of the regression model.

In an example study, we show that confidence estimates of CONFINE can be very beneficial for epitope discovery and epitope engineering. Considering only peptides from confident predictions results in epitope candidates that origin from less erroneous predictions. We also introduce a computational epitope engineering approach based on a genetic algorithm. Although, our approach does not reveal the strongest binder, the obtained peptide is a reliable binder of HLA-A*02:01 and, thus, a suitable epitope candidate. Our example study is a first step towards epitope engineering that benefits from confidence estimation. We believe that more effort in experimental validation will help to prove that the reliability of individual predictions is essential for designing epitope-based vaccines.

Estimating normalized confidence scores and confidence intervals is in general a step forward, moving away from plain affinity values and discrete applicability information. In particular, confidence intervals provide a very intuitive representation of reliability, which can be easily interpreted by biologists. Distinguishing between confident and almost random predictions will also help biologists to choose suitable candidates for experiments. We are convinced that confidence estimators can improve prediction methods with a “black box” character by making their predictions more transparent and usable for biologists.

Chapter 5

Conclusions and Perspectives

"In theory there is no difference between theory and practice. In practice there is."
– Yogi Berra also attributed to Chuck Reid, Jan L. A. van de Snepscheut, Manfred Eigen, et al.

Machine learning provides powerful tools for analyzing, predicting, and understanding biological processes. Nevertheless, computational models can only approximate the real world. In contrast to experiments in the laboratory, most theoretical models exhibit a "black box" character, making it difficult to understand why a particular outcome was obtained. Furthermore, it is often not evident how much confidence one can put into individual predictions. In this work, we focus on filling the gap between theoretical models and real-world experiments by making machine learning approaches more interpretable. In particular, we require our methods to answer two important questions: Why did the model predict this particular outcome? And, how reliable is this prediction? We consider two biological problems that can be modeled either by classification or regression methods and show how interpretability can be provided and how biologists can benefit from interpretable predictions.

In Chapter 3, we introduce YLoc, an interpretable classification approach for predicting the subcellular localization of proteins. YLoc uses a naive Bayes classifier and a small number of biological meaningful features to make predictions. In addition, it helps biologists to rate the reliability of individual predictions by returning a normalized confidence score. Due to discrimination scores that weight the influence of the different biological properties, YLoc is able to answer why a particular location has

been predicted. It creates a textual explanation that describes what feature had the strongest impact on the prediction. However, these explanations should always be inspected with caution. Since we transfer the biological meaning of correlated features, we might accidentally included a wrong interpretation. For the future, we suggest an alternative feature selection approach that also considers the interpretability of features. In this way, the selected feature set might already yield only highly interpretable features making a transfer of meanings obsolete.

Interpretable predictions made by YLoc can considerably support the work of biologists. We have shown that YLoc is able to detect properties of the protein sequence that are responsible for protein localization. This is of particular interest for experimental biologists since it can support the detection of sorting signal by guiding experiments in promising directions. Moreover, we have shown that YLoc can be used to predict localization changes caused by mutations or alternative isoforms. YLoc is even able to detect the site in the protein most likely to be responsible for the localization shift. However, YLoc has not been designed for this particular tasks. It uses biological properties to make predictions. Hence, minor changes in the protein sequence such as single point mutations might not be detected if the overall properties do not change significantly. It should be also considered that YLoc has been trained on a set of "natural" proteins. Applying it to proteins that exhibit properties that are not consistent with the training data might result in erroneous predictions. This issue shows the importance of confidence estimates for detecting unreliable predictions. In particular, if expensive and time-consuming experiments built on predictions, low confidence predictions that are likely to waste a lot of experimental effort should be excluded. In the future, it would be very interesting to create a prediction system tailored to automatic engineering of a protein's subcellular location. Such a system should include only sequence information to be able to recognize single point mutations. Most importantly, confidence scores should be considered in the engineering process. An objective function that considers the confidence in individual predictions can guarantee that only reliable candidates make their way into an experimental testing phase. We believe that YLoc as an interpretable classification approach is a key step towards understanding subcellular localization processes without conducting unnecessary experiments.

The second part of this thesis is dedicated to the interpretable prediction of peptide affinities to MHC-I molecules. In Chapter 4, we introduce two novel confidence

estimators, CONFINE and CONFIVE, that are able to rate the reliability of predicted MHC binding affinities. Biologists are provided with an interval of affinities instead of a plain affinity value. Hence, they are no longer compelled to assume that affinity prediction works equally well on all peptides but can use the size of the affinity interval as an indicator of reliability. This substantially improves the interpretability of current prediction methods. Confidence intervals are of special interest in critical tasks like epitope discovery for vaccines where only reliable predictions should be considered. We think that estimating confidence intervals is also vital for many other tasks where erroneous predictions can have fatal consequences. In the future, it would be interesting to analyze whether confidence estimation can be used to correct regression predictions. Instead of only warning for an error, a prediction could be automatically improved leading to lower error rates and a higher overall reliability of the model.

We have also shown that confidence estimation by CONFINE can be used for engineering the binding affinity of peptides to MHC class I, which is of special interest for critical tasks like epitope-based vaccine design. Although an engineered peptide can yield the highest predicted binding affinity, its real binding affinity might be far lower making it unfavorable for a vaccine. We propose a computational epitope engineering approach based on a genetic algorithm that optimizes the predicted binding affinity to MHC class I while maximizing the confidence of the prediction at the same time. Our approach provides reliable epitope candidates by ignoring low confidence predictions. In future work, it would be challenging to include T-cell reactivity in our epitope engineering approach. If we consider only MHC–peptide binding, we ignore the fact that an immune response is only possible if a TCR binds the MHC–peptide complex. Further, instead of using a search heuristic, one could think of using optimization approaches that guarantee finding the optimal epitope, such as integer linear programming. In any case, it would be necessary to invest more time in experimentally verifying the potential epitope candidates suggested by our engineering approach. Experimental biology could go hand in hand with confidence-based predictions to minimize the time and effort invested in experiments.

Beyond the two biological problems studied in this thesis, there exists a whole range of other applications where interpretable machine learning can substantially support the work of experimental biologists. We are convinced that interpretable predictions

and reliability information will encourage the trust of biologists in machine learning models and will contribute to interesting and important research in the future.

Bibliography

[1] Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *J Mol Biol*, **215**(3), 403–410. 50, 58, 59

[2] Atkeson, C., Moore, A., and Schaal, S. (1997). Locally weighted learning. *Artif Intell Rev*, **11**(1–5), 11–73. 19, 85

[3] Bannai, H., Tamada, Y., Maruyama, O., Nakai, K., and Miyano, S. (2002). Extensive feature detection of N-terminal protein sorting signals. *Bioinformatics*, **18**(2), 298–305. 46

[4] Ben-Hur, A., Ong, C., Sonnenburg, S., Schölkopf, B., and Rätsch, G. (2008). Support vector machines and kernels for computational biology. *PLoS Comput Biol*, **4**(10), e1000173. 15

[5] Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., and Bourne, P. (2000). The protein data bank. *Nucleic Acids Res*, **28**(1), 235–242. 42

[6] Bhasin, M. and Raghava, G. (2003). Prediction of promiscuous and high-affinity mutated MHC binders. *Hybrid Hybridomics*, **22**(4), 229–234. 43, 79, 102, 103

[7] Bian, H., Reidhaar-Olson, J., and Hammer, J. (2003). The use of bioinformatics for identifying class II-restricted T-cell epitopes. *Methods*, **29**(3), 299–309. 78

[8] Blum, T., Briesemeister, S., and Kohlbacher, O. (2009). MultiLoc2: integrating phylogeny and Gene Ontology terms improves subcellular protein localization prediction. *BMC Bioinformatics*, **10**, 274. 5, 46, 47, 48, 58, 62, 63

[9] Boden, M. and Hawkins, J. (2005). Prediction of subcellular localization using sequence-biased recurrent networks. *Bioinformatics*, **21**(10), 2279–2286. 5, 46

[10] Borbulevych, O., Do, P., and Baker, B. (2010). Structures of native and affinity-enhanced WT1 epitopes bound to HLA-A* 0201: Implications for WT1-based cancer therapeutics. *Mol Immunol*, **47**, 2519–2524. 42

[11] Bosnić, Z. and Kononenko, I. (2008a). Automatic selection of reliability estimates for individual regression predictions using meta-learning and internal cross-validation. *Knowl Eng Rev*, **25**(1), 27–47. 107

[12] Bosnić, Z. and Kononenko, I. (2008b). Comparison of approaches for estimating reliability of individual regression predictions. *Data Knowl Eng*, **67**(3), 504–516. 28, 83, 84, 85

[13] Bosnić, Z. and Kononenko, I. (2008c). Estimation of individual prediction reliability using the local sensitivity analysis. *Appl Intell*, **29**(3), 187–203. 28, 84

[14] Brady, S. and Shatkay, H. (2008). EpiLoc: a (working) text-based system for predicting protein subcellular location. In *Pacific Symposium on Biocomputing.*, pages 604–615. World Scientific. 46

[15] Breiman, L. (1996). Bagging predictors. *Mach Learn*, **24**(2), 123–140. 86

[16] Briesemeister, S., Blum, T., Brady, S., Lam, Y., Kohlbacher, O., and Shatkay, H. (2009). SherLoc2: a high-accuracy hybrid method for predicting protein subcellular localization. *J Proteome Res*, **8**(11), 5363–5366. 46

[17] Briesemeister, S., Rahnenführer, J., and Kohlbacher, O. (2010a). Going from where to why – interpretable prediction of protein subcellular localization. *Bioinformatics*, **26**(9), 1232–1238. 28, 45, 137

[18] Briesemeister, S., Rahnenführer, J., and Kohlbacher, O. (2010b). YLoc – an interpretable web server for predicting subcellular localization. *Nucleic Acids Res*, **38**(suppl 2), W497–W502. 45, 137

[19] Briesemeister, S., Rahnenführer, J., and Kohlbacher, O. (2012). No longer confidential: estimating the confidence of individual regression predictions. *PLoS One*, (to be submitted). 77, 137

[20] Brocard, C. and Hartig, A. (2006). Peroxisome targeting signal 1: Is it really a simple tripeptide? *Biochim Biophys Acta*, **1763**(12), 1565–1573. 35

[21] Brockman, A., Orlando, R., and Tarleton, R. (1999). A new liquid chromatography/tandem mass spectrometric approach for the identification of class I major histocompatibility complex associated peptides that eliminates the need for bioassays. *Rapid Commun Mass Spectrom*, **13**(11), 1024–1030. 79

[22] Brown, L. and Baker, A. (2008). Shuttles and cycles: transport of proteins into the peroxisome matrix (review). *Mol Membr Biol*, **25**(5), 363–375. 35

[23] Brusic, V., Petrovsky, N., Zhang, G., and Bajic, V. (2002). Prediction of promiscuous peptides that bind HLA class I molecules. *Immunol Cell Biol*, **80**(3), 280–285. 78

[24] Bui, H., Sidney, J., Peters, B., Sathiamurthy, M., Sinichi, A., Purton, K., Mothé, B., Chisari, F., Watkins, D., and Sette, A. (2005). Automated generation and evaluation of specific MHC binding predictive tools: ARB matrix applications. *Immunogenetics*, **57**(5), 304–314. 78

[25] Buus, S., Lauemøller, S., Worning, P., Kesmir, C., Frimurer, T., Corbet, S., Fomsgaard, A., Hilden, J., Holm, A., and Brunak, S. (2003). Sensitive quantitative predictions of peptide-MHC binding by a 'Query by Committee' artificial neural network approach. *Tissue Antigens*, **62**(5), 378–384. 78

[26] Carlson, M. and Botstein, D. (1982). Two differentially regulated mRNAs with different 5'ends encode secreted with intracellular forms of yeast invertase. *Cell*, **28**(1), 145–154. 73, 163

[27] Casadio, R., Martelli, P., and Pierleoni, A. (2008). The prediction of protein subcellular localization from sequence: a shortcut to functional genome annotation. *Brief Funct Genomic Proteomic*, **7**(1), 63–67. 48, 58, 62

[28] Cedano, J., Aloy, P., Perez-Pons, J., and Querol, E. (1997). Relation between amino acid composition and cellular location of proteins. *J Mol Biol*, **266**(3), 594–600. 46

[29] Chen, Y., Gao, Z., Kerris III, R., Wang, W., Binder, B., and Schaller, G. (2010). Ethylene receptors function as components of high-molecular-mass protein complexes in Arabidopsis. *PLoS One*, **5**(1), e8640. 36

[30] Chicz, R., Urban, R., Lane, W., Gorga, J., Stern, L., Vignali, D., and Strominger, J. (1992). Predominant naturally processed peptides bound to HLA-DR1 are derived from MHC-related molecules and are heterogeneous in size. *Nature*, **358**(6389), 764–768. 41

[31] Chou, K. (2001). Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins Struct Funct Genet*, **43**(3), 246–255. 49

[32] Chou, K. and Cai, Y. (2002). Using functional domain composition and support vector machines for prediction of protein subcellular location. *J Biol Chem*, **277**(48), 45765–45769. 46

[33] Chou, K. and Cai, Y. (2003a). A new hybrid approach to predict subcellular localization of proteins by incorporating Gene Ontology. *Biochem Biophys Res Commun*, **311**(3), 743–747. 46

[34] Chou, K. and Cai, Y. (2003b). Prediction and classification of protein subcellular location-sequence-order effect and pseudo amino acid composition. *J Cell Biochem*, **90**(6), 1250–1260. 46

[35] Chou, K. and Shen, H. (2010a). A new method for predicting the subcellular localization of eukaryotic proteins with both single and multiple sites: Euk-mPLoc 2.0. *PLoS One*, **5**(4), e9931. 46

[36] Chou, K. and Shen, H. (2010b). Plant-mPLoc: a top-down strategy to augment the power for predicting plant protein subcellular localization. *PLoS One*, **5**(6), e11335. 46

[37] Chou, K., Shen, H., *et al.* (2007). Euk-mPLoc: a fusion classifier for large-scale eukaryotic protein subcellular location prediction by incorporating multiple sites. *J Proteome Res*, **6**(5), 1728–1734. 5, 46, 47, 48, 62

[38] Chou, K., Wu, Z., and Xiao, X. (2011). iLoc-Euk: A multi-label classifier for predicting the subcellular localization of singleplex and multiplex eukaryotic proteins. *PLoS One*, **6**(3), e18258. 46

[39] Chou, W., Yin, Y., and Xu, Y. (2010). GolgiP: prediction of Golgi-resident proteins in plants. *Bioinformatics*, **26**(19), 2464–2465. 46

[40] Cokol, M., Nair, R., and Rost, B. (2000). Finding nuclear localization signals. *EMBO Reports*, **1**(5), 411–415. 46

[41] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach Learn*, **20**(3), 273–297. 11, 12, 14, 19, 20

[42] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans Inf Theory*, **13**(1), 21–27. 11

[43] Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. *J Mach Learn Res*, **2**, 265–292. 15

[44] Cui, J., Han, L., Lin, H., Zhang, H., Tang, Z., Zheng, C., Cao, Z., and Chen, Y. (2007). Prediction of MHC-binding peptides of flexible lengths from sequence-derived structural and physicochemical properties. *Mol Immunol*, **44**(5), 866–877. 78

[45] Cui, Q., Jiang, T., Liu, B., and Ma, S. (2004). Esub8: A novel tool to predict protein subcellular localizations in eukaryotic organisms. *BMC Bioinformatics*, **5**, 66. 5, 46

[46] Darwin, C. (1859). *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.* John Murry, London, UK, 1st ed. edition. 1

[47] de Castro, E., Sigrist, C., Gattiker, A., Bulliard, V., Langendijk-Genevaux, P., Gasteiger, E., Bairoch, A., and Hulo, N. (2006). ScanProsite: detection of PROSITE signature matches and prorule-associated functional and structural residues in proteins. *Nucleic Acids Res*, **34**(Web Server issue), W362–W365. 50, 59

[48] De Groot, A., Marcon, L., Bishop, E., Rivera, D., Kutzler, M., Weiner, D., and Martin, W. (2005). HIV vaccine development by computer assisted design: the GAIA vaccine. *Vaccine*, **23**(17-18), 2136–2148. 79

[49] Dellaire, G., Farrall, R., and Bickmore, W. (2003). The nuclear protein database (NPD): sub-nuclear localisation and functional annotation of the nuclear proteome. *Nucleic Acids Res*, **31**(1), 328–30. 33

[50] Dimitrov, S., Dimitrova, G., Pavlov, T., Dimitrova, N., Patlewicz, G., Niemela, J., and Mekenyan, O. (2005). A stepwise approach for defining the applicability domain of SAR and QSAR models. *J Chem Inf Model*, **45**(4), 839–849. 55, 80, 87

[51] Dingwall, C., Robbins, J., Dilworth, S., Roberts, B., and Richardson, W. (1988). The nucleoplasmin nuclear location sequence is larger and more complex than that of SV-40 large T antigen. *J Cell Biol*, **107**(3), 841–849. 34

[52] Domingos, P. (1999). The role of Occam's razor in knowledge discovery. *Data Min Knowl Disc*, **3**(4), 409–425. 20

[53] Dönnes, P. and Elofsson, A. (2002). Prediction of MHC class I binding peptides, using SVMHC. *BMC Bioinformatics*, **3**, 25. 78

[54] Dönnes, P. and Kohlbacher, O. (2006). SVMHC: a server for prediction of MHC-binding peptides. *Nucleic Acids Res*, **34**(Web Server issue), W194–W197. 41, 78, 92

[55] Douat-Casassus, C., Marchand-Geneste, N., Diez, E., Gervois, N., Jotereau, F., and Quideau, S. (2007). Synthetic anticancer vaccine candidates: rational design of antigenic peptide mimetics that activate tumor-specific T-cells. *J Med Chem*, **50**(7), 1598–1609. 79

[56] Doytchinova, I. and Flower, D. (2002). Physicochemical explanation of peptide binding to HLA-A* 0201 major histocompatibility complex: A three-dimensional quantitative structure-activity relationship study. *Proteins*, **48**(3), 505–518. 78

[57] Doytchinova, I. and Flower, D. (2008). QSAR and the prediction of T-cell epitopes. *Curr Proteomics*, **5**(2), 73–95. 78

[58] Doytchinova, I., Blythe, M., and Flower, D. (2002). Additive method for the prediction of protein-peptide binding affinity. application to the MHC class I molecule HLA-A* 0201. *J Proteome Res*, **1**(3), 263–272. 78

[59] Doytchinova, I., Walshe, V., Jones, N., Gloster, S., Borrow, P., and Flower, D. (2004). Coupling in silico and in vitro analysis of peptide-MHC binding: a bioinformatic approach enabling prediction of superbinding peptides and anchorless epitopes. *J Immunol*, **172**(12), 7495–7502. 79

[60] Dragos, H., Gilles, M., and Alexandre, V. (2009). Predicting the predictability: a unified approach to the applicability domain problem of QSAR models. *J Chem Inf Model*, **49**(7), 1762–1776. 55, 84, 86

[61] Du, Q., Wei, Y., Pang, Z., Chou, K., and Huang, R. (2007). Predicting the affinity of epitope-peptides with class I MHC molecule HLA-A* 0201: an application of amino acid-based peptide prediction. *Protein Eng Des Sel*, **20**(9), 417–423. 105

[62] Emanuelsson, O., Brunak, S., von Heijne, G., and Nielsen, H. (2007). Locating proteins in the cell using TargetP, SignalP and related tools. *Nat Protoc*, **2**(4), 953–971. 5, 46, 47

[63] Fagerberg, T., Cerottini, J., and Michielin, O. (2006). Structural prediction of peptides bound to MHC class I. *J Mol Biol*, **356**(2), 521–546. 78

[64] Falk, K., Rötzschke, O., and Rammensee, H. (1990). Cellular peptide composition governed by major histocompatibility complex class I molecules. *Nature*, **348**(6298), 248–251. 78

[65] Fayyad, U. M. and Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027. San Fransisco, Morgan Kaufman Publishers. 17, 51

[66] Fechner, N., Jahn, A., Hinselmann, G., and Zell, A. (2010). Estimation of the applicability domain of kernel-based machine learning models for virtual screening. *J Cheminform*, **2**(1), 2. 84

[67] Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Ann Hum Genet*, **7**(2), 179–188. 11

[68] Friedman, J. (1991). Multivariate adaptive regression splines. *Ann Stat*, **19**(1), 1–141. 92

[69] Fujiwara, Y. and Asogawa, M. (2001). Prediction of subcellular localizations using amino acid composition and order. *Genome Inform*, **12**, 103–112. 46

[70] Fyshe, A., Liu, Y., Szafron, D., Greiner, R., and Lu, P. (2008). Improving subcellular localization prediction using text classification and the Gene Ontology. *Bioinformatics*, **24**(21), 2512–2517. 46

[71] Garg, A. and Raghava, G. (2008). ESLpred2: improved method for predicting subcellular localization of eukaryotic proteins. *BMC Bioinformatics*, **9**, 503. 5, 46

[72] Garg, P., Sharma, V., Chaudhari, P., and Roy, N. (2009). SubCellProt: Predicting protein subcellular localization using machine learning approaches. *In Silico Biol*, **9**(1), 35–44. 46

[73] Gramatica, P. (2007). Principles of qsar models validation: internal and external. *QSAR Comb Sci*, **26**(5), 694–701. 28

[74] Guan, P., Doytchinova, I., Zygouri, C., and Flower, D. (2003). MHCPred: a server for quantitative prediction of peptide–MHC binding. *Nucleic Acids Res*, **31**(13), 3621–3624. 41, 78

[75] Guha, R. (2008). On the interpretation and interpretability of quantitative structure–activity relationship models. *J Comput Aided Mol Des*, **22**(12), 857–871. 26, 28

[76] Guha, R. and Jurs, P. (2005). Interpreting computational neural network QSAR models: a measure of descriptor importance. *J Chem Inf Model*, **45**(3), 800–806. 28

[77] Guo, J. and Lin, Y. (2006). TSSub: eukaryotic protein subcellular localization by extracting features from profiles. *Bioinformatics*, **22**(14), 1784–1785. 46

[78] Hall, M. (1999). *Correlation-based feature selection for machine learning*. Ph.D. thesis, The University of Waikato. 51

[79] Hall, M. (2000). Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 359–366. 51

[80] Harris, M., Clark, J., Ireland, A., Lomax, J., Ashburner, M., Foulger, R., Eilbeck, K., Lewis, S., Marshall, B., Mungall, C., *et al.* (2004). The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res*, **32**(Database issue), D258–D261. 46, 50

[81] Hertz, T. and Yanover, C. (2006). PepDist: a new framework for protein-peptide binding prediction based on learning peptide distance functions. *BMC Bioinformatics*, **7**(Suppl 1), S3. 78

[82] Heskes, T. (1997). Practical confidence and prediction intervals. In M. Mozer, M. Jordan, and T. Petsche, editors, *Adv Neural Inform Proc Sys, NIPS Vol. 9*, pages 176–182. MIT Press. 86

[83] Höglund, A., Dönnes, P., Blum, T., Adolph, H., and Kohlbacher, O. (2006). MultiLoc: prediction of protein subcellular localization using N-terminal targeting sequences, sequence motifs and amino acid composition. *Bioinformatics*, **22**(10), 1158–1165. 46, 58

[84] Holton, G. (1986). *The advancement of science, and its burdens: the Jefferson lecture and other essays*, chapter 1. Cambridge University Press. 1

[85] Honeyman, M., Brusic, V., Stone, N., and Harrison, L. (1998). Neural network-based prediction of candidate T-cell epitopes. *Nat Biotechnol*, **16**(10), 966–969. 78

[86] Hong, Z., Jin, H., Tzfira, T., and Li, J. (2008). Multiple mechanism–mediated retention of a defective brassinosteroid receptor in the endoplasmic reticulum of Arabidopsis. *Plant Cell*, **20**(12), 3418–3429. 36

[87] Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci USA*, **79**(8), 2554–2558. 11

[88] Horton, P., Park, K., Obayashi, T., Fujita, N., Harada, H., Adams-Collier, C., and Nakai, K. (2007). WoLF PSORT: protein localization predictor. *Nucleic Acids Res*, **35**(Web Server issue), W585–W587. 46, 47, 62

[89] Hotelling, H. (1957). The relations of the newer multivariate statistical methods to factor analysis. *Brit J Stat Psych*, **10**, 69–79. 19

[90] Houghton, C., Engelhorn, M., Liu, C., Song, D., Gregor, P., Livingston, P., Orlandi, F., Wolchok, J., McCracken, J., Houghton, A., *et al.* (2007). Immunological validation of the EpitOptimizer program for streamlined design of heteroclitic epitopes. *Vaccine*, **25**(29), 5330–5342. 43, 79

[91] Hua, S. and Sun, Z. (2001). Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, **17**(8), 721–728. 46

[92] Huang, W., Tung, C., Ho, S., Hwang, S., and Ho, S. (2008). ProLoc-GO: Utilizing informative Gene Ontology terms for sequence-based prediction of protein subcellular localization. *BMC Bioinformatics*, **9**, 80. 46

[93] Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., Cuche, B., de Castro, E., Lachaize, C., Langendijk-Genevaux, P., and Sigrist, C. (2008). The 20 years of PROSITE. *Nucleic Acids Res*, **36**(Database issue), D245–D249. 50

[94] Imai, K. and Nakai, K. (2010). Prediction of subcellular locations of proteins: Where to proceed? *Proteomics*, **10**(22), 3970–3983. 46

[95] Jacob, L. and Vert, J. (2008). Efficient peptide–MHC-I binding prediction for alleles with few known binders. *Bioinformatics*, **24**(3), 358. 41, 78

[96] Jaworska, J., Nikolova-Jeliazkova, N., and Aldenberg, T. (2005). QSAR applicability domain estimation by projection of the training set descriptor space: a review. *Altern Lab Anim*, **33**(5), 445–459. 28, 55

[97] Jojic, N., Reyes-Gomez, M., Heckerman, D., Kadie, C., and Schueler-Furman, O. (2006). Learning MHC I-peptide binding. *Bioinformatics*, **22**(14), e227–e235. 78

[98] Kaiser, C. and Botstein, D. (1986). Secretion-defective mutations in the signal sequence for Saccharomyces cerevisiae invertase. *Mol Cell Biol*, **6**(7), 2382–2391. 73, 163, 164, 165, 166

[99] Karniely, S. and Pines, O. (2005). Single translation–dual destination: mechanisms of dual protein targeting in eukaryotes. *EMBO Rep*, **6**(5), 420–425. 36

[100] Kawakami, Y., Eliyahu, S., Jennings, C., Sakaguchi, K., Kang, X., Southwood, S., Robbins, P., Sette, A., Appella, E., and Rosenberg, S. (1995). Recognition of multiple

epitopes in the human melanoma antigen gp100 by tumor-infiltrating T lymphocytes associated with in vivo tumor regression. *J Immunol*, **154**(8), 3961–3968. 102

[101] Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T., and Kanehisa, M. (2008). AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res*, **36**(suppl 1), D202–D205. 49

[102] Khan, A., Baker, B., Ghosh, P., Biddison, W., and Wiley, D. (2000). The structure and stability of an HLA-A* 0201/octameric tax peptide complex with an empty conserved peptide-N-terminal binding site. *J Immunol*, **164**(12), 6398–6405. 104

[103] King, B. and Guda, C. (2007). ngLOC: an n-gram-based Bayesian method for estimating the subcellular proteomes of eukaryotes. *Genome Biol*, **8**(5), R68. 46

[104] Knapp, B., Giczi, V., Ribarics, R., and Schreiner, W. (2011). PeptX: Using genetic algorithms to optimize peptides for MHC binding. *BMC Bioinformatics*, **12**, 241. 79, 103

[105] Kornfeld, S. (1987). Trafficking of lysosomal enzymes. *FASEB J*, **1**(6), 462–468. 36

[106] La Cour, T., Kiemer, L., Mølgaard, A., Gupta, R., Skriver, K., and Brunak, S. (2004). Analysis and prediction of leucine-rich nuclear export signals. *Protein Eng Des Sel*, **17**(6), 527–536. 34

[107] Lee, K., Chuang, H. Y., Beyer, A., Sung, M. K., Huh, W. K., Lee, B., and Ideker, T. (2009). Protein networks markedly improve prediction of subcellular localization in multiple eukaryotic species. *Nucleic Acids Res*, **36**(20), e136. 46

[108] Lei, Z. and Dai, Y. (2006). Assessing protein similarity with Gene Ontology and its use in subnuclear localization prediction. *BMC Bioinformatics*, **7**, 491. 46

[109] Lemeshow, S. and Hosmer, D. (1982). A review of goodness of fit statistics for use in the development of logistic regression models. *Am J Epidemiol*, **115**(1), 92–106. 11

[110] Levi, K. and Weiss, Y. (2004). Learning object detection from a small number of examples: The importance of good features. In S. Guler, A. Hauptmann, and

A. Henrich, editors, *Conference on Computer Vision and Pattern Recognition, CVPR Vol.2*, pages 53–60. IEEE Computer Society. 26

[111] Lin, H., Chen, C., Sung, T., Ho, S., and Hsu, W. (2009). Protein subcellular localization prediction of eukaryotes using a knowledge-based approach. *BMC Bioinformatics*, **10**, S8. 46, 47, 62

[112] Lu, Z. and Hunter, L. (2005). GO molecular function terms are predictive of subcellular localization. In *Pacific Symposium on Biocomputing*, pages 151–161. World Scientific. 46

[113] Lu, Z., Szafron, D., Greiner, R., Lu, P., Wishart, D., Poulin, B., Anvik, J., Macdonell, C., and Eisner, R. (2004). Predicting subcellular localization of proteins using machine-learned classifiers. *Bioinformatics*, **20**(4), 547–556. 46

[114] Lundegaard, C., Nielsen, M., Lamberth, K., Worning, P., Sylvester-Hvid, C., Buus, S., Brunak, S., and Lund, O. (2004). MHC class I epitope binding prediction trained on small data sets. In *Artif Immune Sys*, pages 217–225. Springer. 78

[115] Lundegaard, C., Lund, O., Keşmir, C., Brunak, S., and Nielsen, M. (2007). Modeling the adaptive immune system: predictions and simulations. *Bioinformatics*, **23**(24), 3265–3275. 78

[116] Lundegaard, C., Lamberth, K., Harndahl, M., Buus, S., Lund, O., and Nielsen, M. (2008). NetMHC-3.0: accurate web accessible predictions of human, mouse and monkey MHC class I affinities for peptides of length 8–11. *Nucleic Acids Res*, **36**(suppl 2), W509–W512. 7, 41, 77, 78

[117] Mamitsuka, H. (1998). Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models. *Protein Struct Funct Genet*, **33**(4), 460–474. 78

[118] Menachem, R., Tal, M., and Pines, O. (2011). A third of the yeast mitochondrial proteome is dual localized: a question of evolution. *Proteomics*, **11**(19), 446–455. 36

[119] Minsky, M. (1961). Steps toward artificial intelligence. In *Proc IRE*, volume 49, pages 8–30. IEEE. 11

[120] Mitschke, J., Fuss, J., Blum, T., Höglund, A., Reski, R., Kohlbacher, O., and A., R. S. (2009). Prediction of dual protein targeting to plant organelles. *New Phytol*, **183**(1), 224–236. 36, 47

[121] Moll, A., Hildebrandt, A., Lenhof, H., and Kohlbacher, O. (2006). BALLView: a tool for research and education in molecular modeling. *Bioinformatics*, **22**(3), 365–366. 42

[122] Nair, R. and Rost, B. (2002a). Inferring sub-cellular localization through automated lexical analysis. *Bioinformatics*, **18**(Suppl 1), S78–S86. 46

[123] Nair, R. and Rost, B. (2002b). Sequence conserved for subcellular localization. *Protein Sci*, **11**(12), 2836–2847. 5, 46, 50

[124] Nair, R. and Rost, B. (2005). Mimicking cellular sorting improves prediction of subcellular localization. *J Mol Biol*, **348**(1), 85–100. 5, 46, 47, 62

[125] Nakai, K. and Kanehisa, M. (1992). A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, **14**(4), 897–911. 45, 47, 72

[126] Neural, C., Williams, C., and Rasmussen, C. (1996). Gaussian processes for regression. In *Adv Neural Inform Proc Sys, NIPS vol. 8*, pages 514–520, Cambridge, MA, USA. The MIT Press. 19

[127] Nielsen, M., Lundegaard, C., Worning, P., Lauemøller, S., Lamberth, K., Buus, S., Brunak, S., and Lund, O. (2003). Reliable prediction of T-cell epitopes using neural networks with novel sequence representations. *Protein Sci*, **12**(5), 1007–1017. 78

[128] Nielsen, M., Lundegaard, C., Blicher, T., Lamberth, K., Harndahl, M., Justesen, S., Røder, G., Peters, B., Sette, A., Lund, O., *et al.* (2007). NetMHCpan, a method for quantitative predictions of peptide binding to any HLA-A and-B locus protein of known sequence. *PLoS One*, **2**(8), e796. 41

[129] Obozinski, G., Lanckriet, G., Grant, C., Jordan, M., and Noble, W. (2008). Consistent probabilistic outputs for protein function prediction. *Genome Biol*, **9**(Suppl 1), S6. 3, 28

[130] Outten, C. and Culotta, V. (2004). Alternative start sites in the Saccharomyces cerevisiae GLR1 gene are responsible for mitochondrial and cytosolic isoforms of glutathione reductase. *J Biol Chem*, **279**(9), 7785–7791. 74, 162, 163

[131] Park, K. and Kanehisa, M. (2003). Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics*, **19**(13), 1656–1663. 5, 46

[132] Parker, K., Bednarek, M., and Coligan, J. (1994). Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains. *J Immunol*, **152**(1), 163–175. 41, 78

[133] Parkhurst, M., Salgaller, M., Southwood, S., Robbins, P., Sette, A., Rosenberg, S., and Kawakami, Y. (1996). Improved induction of melanoma-reactive CTL with peptides from the melanoma antigen gp100 modified at HLA-A* 0201-binding residues. *J Immunol*, **157**(6), 2539–2548. 102, 103, 105

[134] Perone, C. (2009). Pyevolve: a Python open-source framework for genetic algorithms. *ACM SIGEVOlution*, **4**(1), 12–20. 104

[135] Peters, B. and Sette, A. (2005). Generating quantitative models describing the sequence specificity of biological processes with the stabilized matrix method. *BMC Bioinformatics*, **6**, 132. 78

[136] Peters, B., Tong, W., Sidney, J., Sette, A., and Weng, Z. (2003). Examining the independent binding assumption for binding of peptide epitopes to MHC-I molecules. *Bioinformatics*, **19**(14), 1765–1772. 7, 78

[137] Peters, B., Bui, H., Frankild, S., Nielson, M., Lundegaard, C., Kostem, E., Basch, D., Lamberth, K., Harndahl, M., Fleri, W., *et al.* (2006). A community resource benchmarking predictions of peptide binding to MHC-I molecules. *PLoS Comput Biol*, **2**(6), e65. 81, 92, 103

[138] Petsalaki, E., Bagos, P., Litou, Z., and Hamodrakas, S. (2006). PredSL: a tool for the N-terminal sequence-based prediction of protein subcellular localization. *Genomics Proteomics Bioinformatics*, **4**(1), 48–55. 46

[139] Pierleoni, A., Martelli, P., Fariselli, P., and Casadio, R. (2006). BaCelLo: a balanced subcellular localization predictor. *Bioinformatics*, **22**(14), e408–e416. 5, 46, 47, 58, 62

[140] Pierleoni, A., Martelli, P., and Casadio, R. (2011). MemLoci: predicting subcellular localization of membrane proteins in eukaryotes. *Bioinformatics*, **27**(9), 1224–1230. 46

[141] Quinlan, J. (1986). Induction of decision trees. *Mach Learn*, **1**(1), 81–106. 51

[142] Raiborg, C., Rusten, T., and Stenmark, H. (2003). Protein sorting into multivesicular endosomes. *Curr Opin Cell Biol*, **15**(4), 446–455. 36

[143] Rammensee, H., Friede, T., and Stevanović, S. (1995). MHC ligands and peptide motifs: first listing. *Immunogenetics*, **41**(4), 178–228. 41, 78

[144] Rammensee, H., Bachmann, J., Emmerich, N., Bachor, O., and Stevanović, S. (1999). SYFPEITHI: database for MHC ligands and peptide motifs. *Immunogenetics*, **50**(3), 213–219. 78

[145] Rastogi, S. and Rost, B. (2010). Bioinformatics predictions of localization and targeting. *Methods Mol Biol*, **619**, 285–305. 46

[146] Reinhardt, A. and Hubbard, T. (1998). Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Res*, **26**(9), 2230–2236. 46

[147] Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46. Morgan Kaufmann. 16

[148] Rognan, D., Lauemøller, S., Holm, A., Buus, S., and Tschinke, V. (1999). Predicting binding affinities of protein ligands from three-dimensional models: application to peptide binding to class I major histocompatibility proteins. *J Med Chem*, **42**(22), 4650–4658. 78

[149] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol Rev*, **65**(6), 386–408. 11

[150] Rothman, J. and Wieland, F. (1996). Protein sorting by transport vesicles. *Science*, **272**(5259), 227–234. 36

[151] Schirle, M., Keilholz, W., Weber, B., Gouttefangeas, C., Dumrese, T., Becker, H., Stevanović, S., and Rammensee, H. (2000). Identification of tumor-associated MHC class I ligands by a novel T cell-independent approach. *Eur J Immunol*, **30**(8), 2216–2225. 79

[152] Schnaitman, C., Erwin, V., and Greenawalt, J. (1967). The submitochondrial localization of monoamine oxidase. *J Cell Biol*, **32**(3), 719–735. 35

[153] Schölkopf, B., Bartlett, P., Smola, A., and Williamson, R. (1998). Support vector regression with automatic accuracy control. In *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN'98)*, pages 111–116. 22

[154] Schölkopf, B., Smola, A., Williamson, R., and Bartlett, P. (2000). New support vector algorithms. *Neural Comput*, **12**(5), 1207–1245. 84

[155] Schueler-Furman, O., Altuvia, Y., Sette, A., and Margalit, H. (2000). Structure-based prediction of binding peptides to MHC class I molecules: application to a broad range of MHC alleles. *Protein Science*, **9**(9), 1838–1846. 78

[156] Schwaighofer, A., Schroeter, T., Mika, S., Laub, J., Ter Laak, A., Sülzle, D., Ganzer, U., Heinrich, N., and Müller, K. (2007). Accurate solubility prediction with error bars for electrolytes: A machine learning approach. *J Chem Inf Model*, **47**(2), 407–424. 86

[157] Scott, M., Thomas, D., and Hallett, M. (2004). Predicting subcellular localization via protein motif co-occurrence. *Genome Res*, **14**(10 a), 1957–1966. 46

[158] Scott, M., Calafell, S., Thomas, D., and Hallett, M. (2005). Refining protein subcellular localization. *PLoS Comput Biol*, **1**(6), e66. 5, 46

[159] Sette, A. and Fikes, J. (2003). Epitope-based vaccines: an update on epitope identification, vaccine design and delivery. *Curr Opin Immunol*, **15**(4), 461–470. 43, 79

[160] Sette, A., Buus, S., Appella, E., Smith, J., Chesnut, R., Miles, C., Colon, S., and Grey, H. (1989). Prediction of major histocompatibility complex binding regions of protein antigens by sequence pattern analysis. *Proc Natl Acad Sci USA*, **86**(9), 3296–3300. 78

[161] Sette, A., Vitiello, A., Reherman, B., Fowler, P., Nayersina, R., Kast, W., Melief, C., Oseroff, C., Yuan, L., and Ruppert, J. (1994). The relationship between class I binding affinity and immunogenicity of potential cytotoxic T cell epitopes. *J Immunol*, **153**(12), 5586–5592. 79

[162] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press, New York, NY, USA. 15

[163] Shen, Y. and Burger, G. (2010). TESTLoc: protein subcellular localization prediction from EST data. *BMC Bioinformatics*, **11**(1), 563. 46

[164] Sheridan, R., Feuston, B., Maiorov, V., and Kearsley, S. (2004). Similarity to molecules in the training set is a good discriminator for prediction accuracy in QSAR. *J Chem Inf Comput Sci*, **44**(6), 1912–1928. 3, 55, 80, 83, 84

[165] Shi, S., Qiu, J., Sun, X., Huang, J., Huang, S., Suo, S., Liang, R., and Zhang, L. (2011). Identify submitochondria and subchloroplast locations with pseudo amino acid composition: Approach from the strategy of discrete wavelet transform feature extraction. *Biochim Biophys Acta*, **1813**, 424–430. 35

[166] Shin, C. J., S., W., Davis, M. J., and Ragan, M. A. (2009). Protein-protein interaction as a predictor of subcellular location. *BMC Syst Biol*, **3**, 28. 46

[167] Simon, P., Bonzon, M., Greppin, H., and Marme, D. (1984). Subchloroplastic localization of NAD kinase activity: evidence for a Ca2+, calmodulin-dependent activity at the envelope and for a Ca2+, calmodulin-independent activity in the stroma of pea chloroplasts. *FEBS Lett*, **167**(2), 332–338. 35

[168] Singh-Jasuja, H., Emmerich, N., and Rammensee, H. (2004). The Tübingen approach: identification, selection, and validation of tumor-associated HLA peptides for cancer therapy. *Cancer Immunol Immunother*, **53**(3), 187–195. 79

[169] Slot, J., Geuze, H., Gigengack, S., Lienhard, G., and James, D. (1991). Immunolocalization of the insulin regulatable glucose transporter in brown adipose tissue of the rat. *J Cell Biol*, **113**(1), 123–135. 36

[170] Small, I., Wintz, H., Akashi, K., and Mireau, H. (1998). Two birds with one stone: genes that encode products targeted to two or more compartments. *Plant Mol Biol*, **38**(1), 265–277. 36

[171] Small, I., Peeters, N., Legeai, F., and Lurin, C. (2004). Predotar: a tool for rapidly screening proteomes for N-terminal targeting sequences. *Proteomics*, **4**(6), 1581–1590. 46

[172] Smola, A. and Schölkopf, B. (2004). A tutorial on support vector regression. *Stat Comput*, **14**(3), 199–222. 22

[173] Sonnenburg, S., Zien, A., Philips, P., and Rätsch, G. (2008). POIMs: positional oligomer importance matricesunderstanding support vector machine-based signal detectors. *Bioinformatics*, **24**(13), i6. 28

[174] Specht, D. (1991). A general regression neural network. *IEEE Trans Neural Netw*, **2**(6), 568–576. 19

[175] Spowage, B., Bruce, C., and Hirst, J. (2009). Interpretable correlation descriptors for quantitative structure-activity relationship. *J Cheminform*, **1**, 22. 26

[176] Stanton, D. (2003). On the physical interpretation of QSAR models. *J Chem Inf Comput Sci*, **43**(5), 1423–1433. 27

[177] Sutherland, J., Lee, A., and Weaver, D. (2004). A comparison of methods for modeling quantitative structure-activity relationships. *J Med Chem*, **47**(22), 5541–5554. 92

[178] Takada, Y., Kaneko, N., Esumi, H., Purdue, P., and Danpure, C. (1990). Human peroxisomal L-alanine: glyoxylate aminotransferase. *Biochem J*, **268**(2), 517–520. 73, 162

[179] Talete srl (2007). DragonX 1.4 (molecular descriptor calculation software). 92

[180] Tangri, S., Ishioka, G., Huang, X., Sidney, J., Southwood, S., Fikes, J., and Sette, A. (2001). Structural features of peptide analogs of human histocompatibility leukocyte antigen class I epitopes that are more potent and immunogenic than wild-type peptide. *J Exp Med*, **194**(6), 833–846. 79

[181] Tikhonov, A. (1943). On the stability of inverse problems. In *Doklady Akademii Nauk SSSR*, volume 39, pages 195–198. 19, 20

[182] Tolley, E. and Craig, I. (1975). Presence of two forms of fumarase (fumarate hydratase EC 4.2.1.2) in mammalian cells: immunological characterization and genetic analysis in somatic cell hybrids. Confirmation of the assignment of a gene necessary for the enzyme expression to human chromosome 1. *Biochem Genet*, **13**(11), 867–883. 73, 161

[183] Toussaint, N. and Kohlbacher, O. (2009a). OptiTope–a web server for the selection of an optimal set of peptides for epitope-based vaccines. *Nucleic Acids Res*, **37**(suppl 2), W617–W622. 79

[184] Toussaint, N. and Kohlbacher, O. (2009b). Towards in silico design of epitope-based vaccines. *Expert Opin Drug Discov*, **4**(10), 1047–1060. 43, 78, 79

[185] Toussaint, N., Dönnes, P., and Kohlbacher, O. (2008). A mathematical framework for the selection of an optimal set of peptides for epitope-based vaccines. *PLoS Comput Biol*, **4**(12), e1000246. 43, 79

[186] Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *Int J Data Warehousing Min*, **3**(3), 1–13. 59

[187] Tung, C. and Ho, S. (2007). POPI: predicting immunogenicity of MHC class I binding peptides by mining informative physicochemical properties. *Bioinformatics*, **23**(8), 942–949. 41, 78

[188] Ustun, B., Melssen, W., and Buydens, L. (2007). Visualisation and interpretation of support vector regression models. *Anal Chim Acta*, **595**(1-2), 299–309. 28

[189] Vider-Shalit, T., Raffaeli, S., and Louzoun, Y. (2007). Virus-epitope vaccine design: informatic matching the HLA-I polymorphism to the virus genome. *Mol Immunol*, **44**(6), 1253–1261. 79

[190] von Heijne, G. (1990). The signal peptide. *J Membr Biol*, **115**(3), 195–201. 35

[191] Wan, J., Liu, W., Xu, Q., Ren, Y., Flower, D., and Li, T. (2006). SVRMHC prediction server for MHC-binding peptides. *BMC Bioinformatics*, **7**, 463. 78

[192] Whitten, I. and Frank, E. (2005). *Data mining: practical machine learning tools and techniques*. San Fransisco, Morgan Kaufman Publishers. 52, 59

[193] Wikimedia Commons (2006). File:biological_cell.svg. http://commons.wikimedia.org/wiki/File:Biological_cell.svg. [Online; accessed 20-June-2011], published under GNU Free Documentation License, Version 1.2. 30

[194] Wikimedia Commons (2008). File:scheme_chloroplast-es.svg. http://commons.wikimedia.org/wiki/File:Scheme_Chloroplast-es.svg. [Online; accessed 20-June-2011], published under GNU Free Documentation License, Version 1.2. 30

[195] Xie, D., Li, A., Wang, M., Fan, Z., and Feng, H. (2005). LOCSVMPSI: a web server for subcellular localization of eukaryotic proteins using SVM and profile of PSI-BLAST. *Nucleic Acids Res*, **33**(Web Server issue), W105–W110. 46

[196] Yang, Y. and Webb, G. I. (2002). A comparative study of discretization methods for naive-Bayes classifiers. In T. Yamaguchi, A. Hoffmann, H. Motoda, and P. Compton, editors, *Proc of the 2002 Pacific Rim Knowledge Acquisition Workshop (PKAW'02)*, pages 159–173, Tokyo. Japanese Society for Artificial Intelligence. 16

[197] Zhang, S., Xia, X., Shen, J., Zhou, Y., and Sun, Z. (2008). DBMLoc: a database of proteins with multiple subcellular localizations. *BMC Bioinformatics*, **9**, 127. 36, 46, 59

Appendix A

Abbreviations

ACC	Overall accuracy
AD	Applicability domain
AGT1	Human glyoxylate aminotransferase 1
ANN	Artificial neural network
AUCC	Area under the confidence curve
DNA	Deoxyribonucleic acid
ch	Chloroplast
CEC	Confidence–error correlation
CFS	Correlation-based feature selection
cTP	Chloroplast targeting peptide
cy	Cytoplasm
ER	Endoplasmic reticulum
ex	Extracellular space
F1	F1-score
FH	Human fumerate hydratase
FN	False negative
FP	False positive
GLR1	Glutathione reductase
GO	Gene Ontology
go	Golgi apparatus
HLA	Human leukocyte antigen
HMM	Hidden Markov model
IDS	Independent dataset
kNN	K-nearest neighbor
ly	Lysosome
MHC	Major histocompatibility complex
mi	Mitochondrion
mRNA	Messenger ribonucleic acid
mTP	Mitochondrial targeting peptide

MSE	Mean squared error
NES	Nuclear export signal
NLS	Nuclear localization signal
nu	Nucleus
pe	Peroxisome
pm	Plasma membrane
PRE	Precision
QSAR	Quantitative structure-activity relationship
R^2	Coefficient of determination
RBF	Radial basis function
REC	Recall
RNA	Ribonucleic acid
SE	Squared error
SOAP	Simple object access protocol
SSE	Sum of squared errors
SP	Secretory pathway
SUC2	Glycosylated invertase
SVM	Support vector machine
SVR	Support vector regression
Tc	T helper
TCR	T-cell receptor
Th	T cytotoxic
TN	True negative
TP	True positive
va	Vacuole

Appendix B

Publications

1. Briesemeister, S, Rahnenführer, J, and Kohlbacher, O. **No Longer Confidential: Estimating the Confidence of Individual Regression Predictions.** PLoS One, to be submitted.

2. Böcker, S, Briesemeister, S, and Klau, GW. **Exact Algorithms for Cluster Editing: Evaluation and Experiments.** Algorithmica, 2011, 60(2):316-334.

3. Briesemeister, S, Rahnenführer, J, and Kohlbacher, O. **YLoc an interpretable web server for predicting subcellular localization.** Nucl Acids Res, 2010, 38:W497-W502.

4. Briesemeister, S, Rahnenführer, J, and Kohlbacher, O. **Going from where to why - interpretable prediction of protein subcellular localization.** Bioinformatics, 2010, 26(9):1232-1238.

5. Kirchler, T, Briesemeister, S, Singer, M, Schütze, K, Keinath, M, Kohlbacher, O, Vicente-Carbajosa, J, Teige, M, Harter, K, and Chaban, C. **The role of phosphorylatable serine residues in the DNA-binding domain of Arabidopsis bZIP transcription factors.** Eur J Cell Biol, 2010,89(2-3):175-183.

6. Briesemeister, S, Blum, T, Brady, S, Lam, Y, Kohlbacher, O, and Shatkay, H. **SherLoc2: a high-accuracy hybrid method for predicting subcellular localization of proteins.** J Proteome Res, 2009, 8(11):53635366.

7. Blum, T, Briesemeister, S, and Kohlbacher, O. **MultiLoc2: integrating phylogeny and Gene Ontology terms improves subcellular protein localization prediction.** BMC Bioinformatics, 2009, 10:274.

8. Böcker, S, Briesemeister, S, Bui, QB, and Truss, A. **Going Weighted: Parameterized Algorithms for Cluster Editing.** Theor Comput Sci, 2009, 410(52):5467-5480.

9. Böcker, S, Briesemeister, S, and Klau, GW. **On Optimal Comparability Editing with Applications to Molecular Diagnostics.** BMC Bioinformatics, 2009, 10(Suppl 1):S61.

10. Böcker, S, Briesemeister, S, Bui, QB, and Truss, A. **Going Weighted: Parameterized Algorithms for Cluster Editing.** In: Proc of Conference on Combinatorial Optimization and Applications (COCOA), 2008, Volume 5165, pp. 1–12, Lect Notes Comput Sc, Springer.

11. Böcker, S, Briesemeister, S, and Klau, GW. **Exact Algorithms for Cluster Editing: Evaluation and Experiments.** In: Proc of Workshop on Experimental Algorithms (WEA), 2008, pp. 289-302, Springer.

12. Böcker, S, Briesemeister, S, Bui, QB, and Truss, A. **A fixed-parameter approach for Weighted Cluster Editing.** In: Proc of Asia-Pacific Bioinformatics Conference (APBC), 2008, Volume 5, pp. 211–220, Imperial College Press.

Appendix C

Contributions

All presented approaches and algorithms were discussed with my supervisor Prof. Dr. Oliver Kohlbacher as well as Prof. Dr. Jörg Rahnenführer, leading to new ideas and research directions.

Chapter 3 — YLoc – An Interpretable Classification Approach

The main methodology of YLoc was presented in the journal Bioinformatics [17] in 2010. Oliver Kohlbacher and myself designed YLoc and the evaluation study. The probabilistic confidence estimation approach has been developed by Jörg Rahnenführer and myself. The dataset-based confidence estimators were designed together with Johannes Junker as part of his master thesis, which I supervised. Johannes Junker also implemented the dataset-based confidence estimators. The YLoc web server was presented in the journal Nucleic Acids Research [18]. The YLoc software and the YLoc web server was designed and implemented by myself.

Chapter 4 — Interpretable Regression With CONFINE and CONFIVE

Major parts of this work are part of a manuscript in preperation [19]. Oliver Kohlbacher and myself designed the different confidence estimation strategies. All authors discussed and designed the evaluation study. The approach to estimate confidence intervals has been developed by Jörg Rahnenführer and myself. The R CRAN package `confReg` was designed and implemented by myself. Parts of the genetic algorithm approach were developed together with Leonie Martens as part of her bachelor thesis, which I supervised. The epitope engineering study has not been published elsewhere.

Appendix D

Tables

origin	class name	number of proteins
Animals	Cytoplasm	437
	Nucleus	1166
	Mitochondrion	188
	Secretory Pathway	804
Fungi	Cytoplasm	211
	Nucleus	711
	Mitochondrion	188
	Secretory Pathway	88
Plants	Cytoplasm	58
	Nucleus	121
	Mitochondrion	67
	Secretory Pathway	41
	Chloroplast	204

Table D.1: BaCelLo dataset: Class names and number of proteins of the three BaCelLo training datasets.

class name	number of proteins
Nucleus	837
Cytoplasm	1411
Mitochondrion	510
Chloroplast	449
Extracellular space	843
Plasma membrane	1238
Peroxisome	157
Endoplasmic Reticulum	198
Golgi apparatus	150
Lysosome	103
Vacuole	63

Table D.2: Höglund dataset: Class names and number of proteins of the Höglund training datasets. For training the animal, fungal, and plant versions of YLoc, the locations vacuole and chloroplast, lysosome and chloroplast, lysosome are not used, respectively.

class name	#	kind of proteins	example proteins
cy_nu	1882	various receptors and substrates	insulin receptor substrate, ionotropic glutamate receptor
ex_pm	334	proteins from signal pathways and immune pathways	autolysin , Fc receptor , kit ligand
cy_pm	252	structural proteins and messenger proteins	calmodulin, microtubule-associated protein tau
cy_mi	240	mostly enzymes	hemoglobin beta-1, homoaconitase
nu_mi	120	enzymes and DNA binding proteins	uracil-DNA glycosylase, estrogen receptor beta, DNA-glycosylase
er_ex	115	mostly enzymes	cholesterol 24-hydroxylase, calsequestrin
nu_ex	113	enzymes and viral proteins	puromycin-sensitive aminopeptidase, epstein-barr nuclear antigen

Table D.3: DBMLoc dataset: Class names and number of proteins (#) of the DBMLoc training datasets. Furthermore, the kind of proteins present in these multiple locations, including some example proteins, are shown. See Section 3.2.6 for the translation of the abbreviated class names.

Method	Location	REC	PRE	F1
YLoc-LowRes	mi	0.77	0.78	0.78
	SP	0.95	0.83	0.89
	cy	0.43	0.61	0.51
	nu	0.87	0.82	0.85
YLoc-LowRes*	mi	0.70	0.76	0.73
	SP	0.93	0.80	0.86
	cy	0.36	0.55	0.44
	nu	0.86	0.79	0.82
YLoc-HighRes	mi	0.71	0.77	0.74
	SP	0.91	0.77	0.84
	cy	0.32	0.50	0.39
	nu	0.86	0.78	0.82
YLoc-HighRes*	mi	0.73	0.74	0.70
	SP	0.88	0.74	0.81
	cy	0.45	0.43	0.44
	nu	0.75	0.82	0.78
YLoc$^+$	mi	0.68	0.66	0.67
	SP	0.86	0.69	0.77
	cy	0.89	0.46	0.61
	nu	0.85	0.52	0.65
YLoc$^+$*	mi	0.75	0.60	0.67
	SP	0.91	0.67	0.77
	cy	0.89	0.45	0.60
	nu	0.81	0.52	0.64

Table D.4: Detailed performance comparison on the animal BaCelLo IDS: Detailed performance of YLoc-LowRes, YLoc-HighRes, YLoc$^+$ on the animal BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). For predictors marked with *, GO-term features were excluded from the feature selection. The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	Location	REC	PRE	F1
YLoc-LowRes	mi	0.56	0.66	0.61
	SP	0.89	0.80	0.84
	cy	0.24	0.63	0.35
	nu	0.91	0.49	0.65
YLoc-LowRes*	mi	0.62	0.59	0.60
	SP	0.89	0.76	0.82
	cy	0.18	0.57	0.27
	nu	0.87	0.49	0.63
YLoc-HighRes	mi	0.42	0.83	0.56
	SP	1.00	0.22	0.36
	cy	0.46	0.66	0.54
	nu	0.71	0.51	0.60
YLoc-HighRes*	mi	0.45	0.83	0.58
	SP	1.00	0.20	0.33
	cy	0.51	0.64	0.57
	nu	0.65	0.53	0.58
YLoc$^+$	mi	0.44	0.49	0.46
	SP	0.67	0.29	0.40
	cy	0.89	0.46	0.61
	nu	0.82	0.45	0.58
YLoc$^+$*	mi	0.47	0.53	0.50
	SP	0.78	0.21	0.33
	cy	0.79	0.47	0.59
	nu	0.82	0.44	0.57

Table D.5: Detailed performance comparison on the fungi BaCelLo IDS: Detailed performance of YLoc-LowRes, YLoc-HighRes, YLoc$^+$ on the fungi BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). For predictors marked with *, GO-term features were excluded from the feature selection. The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	Location	REC	PRE	F1
YLoc-LowRes	mi	0.50	0.31	0.38
	ch	0.72	0.85	0.78
	SP	0.67	0.33	0.44
	cy	0.43	0.55	0.48
	nu	0.85	0.76	0.80
YLoc-LowRes*	mi	0.17	0.06	0.09
	ch	0.62	0.76	0.68
	SP	0.33	0.13	0.19
	cy	0.36	0.47	0.41
	nu	0.72	0.76	0.74
YLoc-HighRes	mi	0.83	0.35	0.49
	ch	0.42	0.95	0.59
	SP	1.00	0.29	0.46
	cy	0.57	0.28	0.38
	nu	0.78	0.77	0.77
YLoc-HighRes*	mi	0.67	0.26	0.38
	ch	0.46	0.90	0.61
	SP	0.83	0.22	0.35
	cy	0.53	0.34	0.42
	nu	0.80	0.79	0.80
YLoc$^+$	mi	0.83	0.25	0.39
	ch	0.40	0.91	0.56
	SP	0.83	0.24	0.38
	cy	0.82	0.31	0.45
	nu	0.88	0.53	0.66
YLoc$^+$*	mi	0.83	0.28	0.42
	ch	0.45	0.87	0.59
	SP	0.83	0.26	0.39
	cy	0.82	0.32	0.46
	nu	0.83	0.48	0.61

Table D.6: Detailed performance comparison on the plant BaCelLo IDS: Detailed performance of YLoc-LowRes, YLoc-HighRes, YLoc$^+$ on the plant BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). For predictors marked with *, GO-term features were excluded from the feature selection. The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	Location	REC	PRE	F1
YLoc-HighRes	er	0.10	0.45	0.16
	ex	0.81	0.86	0.84
	go	0.14	0.33	0.20
	ly	0.25	0.50	0.33
	pe	0.00	0.00	0.00
	pm	0.56	0.44	0.49
YLoc-HighRes*	er	0.10	0.29	0.15
	ex	0.87	0.87	0.87
	go	0.07	0.25	0.11
	ly	0.25	0.23	0.24
	pe	0.33	0.13	0.18
	pm	0.62	0.60	0.61
YLoc+	er	0.12	0.42	0.19
	ex	0.80	0.77	0.79
	go	0.07	0.21	0.11
	ly	0.25	0.12	0.16
	pe	0.67	0.35	0.46
	pm	0.74	0.43	0.54
YLoc+*	er	0.32	0.41	0.36
	ex	0.84	0.76	0.80
	go	0.07	0.21	0.17
	ly	0.00	0.00	0.00
	pe	0.67	0.44	0.53
	pm	0.68	0.46	0.55

Table D.7: Detailed performance comparison on the animal Höglund IDS: Detailed performance of YLoc-HighRes and YLoc$^+$ on the animal Höglund IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). For predictors marked with *, GO-term features were excluded from the feature selection. The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-LowRes	0.79	0.76	0.76	0.75
YLoc-HighRes	0.74	0.70	0.70	0.79
YLoc$^+$	0.58	0.82	0.58	0.67
MultiLoc2-LowRes	0.73	0.80	0.75	0.76
MultiLoc2-HighRes	0.68	0.75	0.71	0.71
BaCelLo	0.64	0.69	0.66	0.66
LOCTree	0.62	0.61	0.56	0.58
WoLF PSORT	0.70	0.70	0.64	0.67
Euk-mPloc	0.61	0.56	0.56	0.54
KnowPred	0.75	0.75	0.67	0.69

Table D.8: Performance comparison on the animal BaCelLo IDS: Performance of YLoc-LowRes, YLoc-HighRes, YLoc$^+$, MultiLoc2-LowRes, MultiLoc2-HighRes, BaCelLo, LOCTree, WoLF PSORT, Euk-mPloc, and KnowPred on the animal BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$, WoLF PSORT, Euk-mPloc, KnowPred were measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-LowRes*	0.76	0.71	0.73	0.71
YLoc-HighRes*	0.71	0.70	0.68	0.69
YLoc$^+$*	0.58	0.84	0.56	0.67

Table D.9: Performance of YLoc without GO terms on the animal BaCelLo IDS: Performance of YLoc-LowRes*, YLoc-HighRes*, and YLoc$^+$* on the animal BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-LowRes	0.56	0.65	0.66	0.61
YLoc-HighRes	0.56	0.65	0.55	0.51
YLoc$^+$	0.48	0.70	0.42	0.51
MultiLoc2-LowRes	0.60	0.66	0.59	0.61
MultiLoc2-HighRes	0.53	0.59	0.59	0.58
BaCelLo	0.57	0.71	0.56	0.60
LOCTree	0.47	0.55	0.43	0.43
WoLF PSORT	0.50	0.62	0.54	0.51
Euk-mPloc	0.60	0.67	0.53	0.56
KnowPred	0.66	0.69	0.53	0.56

Table D.10: Performance comparison on the fungi BaCelLo IDS: Performance of YLoc-LowRes, YLoc-HighRes, YLoc$^+$, MultiLoc2-LowRes, MultiLoc2-HighRes, BaCelLo, LOCTree, WoLF PSORT, Euk-mPloc, and KnowPred on the fungi BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred were measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-LowRes*	0.53	0.64	0.60	0.58
YLoc-HighRes*	0.56	0.65	0.55	0.52
YLoc$^+$*	0.48	0.71	0.41	0.50

Table D.11: Performance of YLoc without GO terms on the fungi BaCelLo IDS: Performance of YLoc-LowRes*, YLoc-HighRes*, and YLoc$^+$* on the fungi BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-LowRes	0.71	0.63	0.56	0.58
YLoc-HighRes	0.58	0.72	0.53	0.54
YLoc$^+$	0.53	0.75	0.45	0.49
MultiLoc2-LowRes	0.76	0.72	0.61	0.64
MultiLoc2-HighRes	0.62	0.65	0.52	0.54
BaCelLo	0.69	0.61	0.71	0.56
LOCTree	0.70	0.65	0.58	0.58
WoLF PSORT	0.57	0.46	0.48	0.46
Euk-mPloc	0.46	0.54	0.35	0.37
KnowPred	0.29	0.52	0.15	0.23

Table D.12: Performance comparison on the plant BaCelLo IDS: Performance of YLoc-LowRes, YLoc-HighRes, YLoc$^+$, MultiLoc2-LowRes, MultiLoc2-HighRes, BaCelLo, LOCTree, WoLF PSORT, Euk-mPloc, and KnowPred on the plants BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred were measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-LowRes*	0.58	0.44	0.44	0.42
YLoc-HighRes*	0.58	0.66	0.50	0.51
YLoc$^+$*	0.53	0.75	0.44	0.50

Table D.13: Performance of YLoc without GO terms on the plant BaCelLo IDS: Performance of YLoc-LowRes*, YLoc-HighRes*, and YLoc$^+$* on the plants BaCelLo IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-HighRes	0.56	0.31	0.43	0.33
YLoc$^+$	0.53	0.44	0.38	0.37
MultiLoc2-HighRes	0.57	0.38	0.46	0.41
WoLF PSORT	0.36	0.15	0.28	0.18
Euk-mPloc	0.27	0.23	0.29	0.24
KnowPred	0.49	0.34	0.50	0.37

Table D.14: Performance comparison on the animal Höglund IDS: Performance of YLoc-HighRes, YLoc$^+$, MultiLoc2-HighRes, WoLF PSORT, Euk-mPloc, and KnowPred on the animals Höglund IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred were measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc-HighRes*	0.60	0.37	0.39	0.36
YLoc$^+$*	0.56	0.44	0.42	0.40

Table D.15: Performance of YLoc without GO terms on the animal Höglund IDS: Performance of YLoc-HighRes*, and YLoc$^+$* on the animal Höglund IDS regarding overall accuracy (ACC), average recall (REC), average precision (PRE), and average F1-score (F1). The performance of YLoc$^+$ was measured using the generalized measures for multi-label classification.

Method	ACC	REC	PRE	F1
YLoc$^+$ Animals	0.63	0.58	0.85	0.69
YLoc$^+$ Animals	0.34	0.23	0.46	0.29
YLoc$^+$ Fungi	0.65	0.58	0.85	0.68
YLoc$^+$ Fungi	0.35	0.23	0.45	0.29
YLoc$^+$ Plants	0.64	0.58	0.84	0.68
YLoc$^+$ Plants	0.35	0.24	0.47	0.31
WoLF PSORT Animals	0.43	0.38	0.81	0.52
WoLF PSORT Animals	0.05	0.02	0.25	0.03
WoLF PSORT Fungi	0.42	0.36	0.81	0.49
WoLF PSORT Fungi	0.04	0.01	0.14	0.02
WoLF PSORT Plants	0.33	0.24	0.79	0.36
WoLF PSORT Plants	0.00	0.00	0.08	0.00
Euk-mPloc	0.41	0.32	0.76	0.44
Euk-mPloc	0.05	0.02	0.49	0.04
KnowPred	0.63	0.54	0.88	0.66
KnowPred	0.36	0.19	0.67	0.28

Table D.16: Performance comparison on the DBMLoc dataset: The prediction performance of the different versions of YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred on the DBMLoc dataset is shown. The prediction performance was measured using multi-label measures and singles label measures (in italic).

Method	ACC	REC	PRE	F1
YLoc$^+$* Animals	0.61	0.55	0.79	0.65
*YLoc$^+$ * Animals*	0.31	0.19	0.36	0.24
YLoc$^+$* Fungi	0.62	0.56	0.79	0.65
*YLoc$^+$ * Fungi*	0.33	0.20	0.36	0.24
YLoc$^+$* Plants	0.60	0.54	0.78	0.63
*YLoc$^+$ * Plants*	0.30	0.18	0.32	0.22

Table D.17: Performance of YLoc without GO terms on the DBMLoc dataset: The prediction performance of the different versions of YLoc$^+$ without the inclusion of GO terms on the DBMLoc dataset is shown. The prediction performance was measured using multi-label measures and singles label measures (in italic).

Method	ACC	REC	PRE	F1
YLoc40$^+$ Animals	0.65	0.58	0.84	0.68
YLoc40$^+$ Animals	0.38	0.23	0.51	0.31
YLoc40$^+$ Fungi	0.66	0.57	0.85	0.68
YLoc40$^+$ Fungi	0.37	0.22	0.42	0.28
YLoc40$^+$ Plants	0.65	0.56	0.84	0.67
YLoc40$^+$ Plants	0.35	0.20	0.43	0.26
WoLF PSORT Animals	0.43	0.38	0.81	0.52
WoLF PSORT Animals	0.05	0.02	0.25	0.03
WoLF PSORT Fungi	0.42	0.35	0.81	0.49
WoLF PSORT Fungi	0.04	0.01	0.14	0.02
WoLF PSORT Plants	0.33	0.24	0.79	0.36
WoLF PSORT Plants	0.00	0.00	0.08	0.00
Euk-mPloc	0.41	0.32	0.76	0.44
Euk-mPloc	0.05	0.02	0.49	0.04
KnowPred	0.64	0.54	0.88	0.67
KnowPred	0.36	0.19	0.67	0.28

Table D.18: Performance comparison on the DBMLoc40 dataset: The prediction performance of the different versions of YLoc$^+$, WoLF PSORT, Euk-mPloc, and KnowPred on the DBMLoc40 dataset is shown. The prediction performance was measured using multi-label measures and singles label measures (in italic).

Method	ACC	REC	PRE	F1
YLoc40$^+$* Animals	0.63	0.56	0.77	0.64
YLoc40$^+$ Animals*	0.35	0.19	0.38	0.24
YLoc40$^+$* Fungi	0.62	0.55	0.77	0.64
YLoc40$^+$ Fungi*	0.34	0.18	0.32	0.22
YLoc40$^+$* Plants	0.61	0.54	0.77	0.63
YLoc40$^+$ Plants*	0.33	0.17	0.32	0.22

Table D.19: Performance of YLoc without GO terms on the DBMLoc40 dataset: The prediction performance of the different versions of YLoc$^+$ without the inclusion of GO terms on the DBMLoc40 dataset is shown. The prediction performance was measured using multi-label measures and singles label measures (in italic).

Measure	IDS	NBconf	Post	AvgDist	NoNN	Acc	LE
$AUCC_{inst}$	Animals	0.35	0.29	-0.42	-0.17	0.42	0.14
	Fungi	0.25	0.17	-0.06	-0.02	0.22	0.17
	Plants	0.16	0.20	0.12	0.09	0.29	0.37
$AUCC_{score}$	B Animals	0.31	0.17	-0.10	-0.13	0.19	0.37
	Fungi	0.25	0.07	-0.02	-0.04	0.12	0.20
	Plants	0.18	0.11	0.05	0.09	0.07	0.29

Table D.20: $AUCC_{ACC}$ of different confidence estimators for YLoc-LowRes predictions on the BaCelLo IDSs. We calculated $AUCC_{inst,ACC}$ and $AUCC_{score,ACC}$ based on confidence estimates for predictions of the BaCelLo IDSs using different confidence estimators. In addition, we show the AUCCs for the case of using the posterior probability as estimate (Post).

Predictor	Version	Measure	Minimum confidence score					
			0.00	0.20	0.40	0.60	0.80	0.90
YLoc-LowRes	Animals	F1	0.75	0.76	0.78	0.80	0.84	0.95
		ACC	0.79	0.79	0.81	0.86	0.91	0.93
		% of n	100	81	69	52	33	20
	Fungi	F1	0.61	0.64	0.68	0.69	0.71	0.94
		ACC	0.56	0.58	0.62	0.68	0.73	0.86
		% of n	100	92	70	40	17	5
	Plants	F1	0.58	0.64	0.74	0.75	0.74	0.71
		ACC	0.71	0.75	0.76	0.78	0.78	0.75
		% of n	100	87	56	39	27	17
YLoc-HighRes	Animals	F1	0.69	0.74	0.76	0.76	0.77	0.77
		ACC	0.74	0.78	0.80	0.82	0.83	0.84
		% of n	100	88	82	74	68	61
	Fungi	F1	0.51	0.56	0.59	0.61	0.59	0.63
		ACC	0.56	0.58	0.59	0.62	0.61	0.65
		% of n	10	88	80	73	62	53
	Plants	F1	0.54	0.61	0.64	0.63	0.67	0.67
		ACC	0.58	0.65	0.68	0.67	0.71	0.71
		% of n	399	79	74	66	54	47
YLoc$^+$	Animals	F1	0.67	0.69	0.72	0.77	0.76	0.81
		ACC	0.58	0.60	0.62	0.65	0.65	0.69
		% of n	100	86	73	56	38	25
	Fungi	F1	0.51	0.56	0.58	0.58	0.64	0.68
		ACC	0.48	0.49	0.51	0.53	0.55	0.58
		% of n	100	89	79	64	40	22
	Plants	F1	0.49	0.60	0.66	0.69	0.77	0.84
		ACC	0.53	0.62	0.64	0.66	0.78	0.79
		% of n	100	82	69	58	44	34

Table D.21: Performance of YLoc for different minimum confidence levels using estimator NBconf: Performance of the YLoc predictors on the BaCelLo IDSs concerning F1 and ACC for different minimum normalized confidence scores of NBconf. Moreover, the percentage of instances n that can be predicted with the given minimum confidence level is given. The performance of YLoc$^+$ was measured using the generalized F1 and ACC. Values obtained from a very small set of proteins with less than 50 proteins are grayed out.

Protein	Origin	Swiss-Prot AC	Isoform	Change	Predicted
FH	Animal	P07954	delta43	mi→cy	mit→cy (YLoc-LowRes)
AGT1	Animal	P21549	extended	pe→mi	pe→mi (YLoc-HighRes)
					pe/mi/cy→mi (YLoc$^+$)
GLR1	Fungi	P41921	delta16	mi→cy	mi (YLoc-LowRes)
					mi→cy (YLow-HighRes)
					mi→cy/mit (YLoc$^+$)
SUC2	Fungi	P00724	delta20	ex→cy	ex→cy (YLoc-LowRes)
SUC2	Fungi	P00724	Mut113	ex→cy	ex→nu (YLoc-LowRes)
SUC2	Fungi	P00724	Mut210	ex	SP (YLoc-LowRes)
SUC2	Fungi	P00724	Mut236	ex	SP (YLoc-LowRes)
SUC2	Fungi	P00724	Mut211	ex	SP (YLoc-LowRes)
SUC2	Fungi	P00724	Mut437	ex→cy	SP (YLoc-LowRes)
SUC2	Fungi	P00724	Mut438	ex→cy	ex→cy (YLoc-LowRes)
SUC2	Fungi	P00724	Mut331	ex	SP (YLoc-LowRes)
SUC2	Fungi	P00724	Mut301	ex	SP (YLoc-LowRes)
SUC2	Fungi	P00724	Mut506	ex→cy	ex→cy (YLoc-LowRes)
SUC2	Fungi	P00724	Mut321	ex→cy	ex→cy (YLoc-LowRes)

Table D.22: Example of predicted localization changes: Isoforms and mutated proteins used in the localization change experiment from Section 3.3.6 are presented. In addition, the expected and predicted localization change is compared. See Appendix E for the used protein sequence data.

Table D.23: CEC of confidence estimators on artificial data with different properties

	$n \leq 100$	$n > 100$	$m \leq 10$	$m > 10$	$\sigma < 1.0$	$\sigma \geq 1.0$	best
CONFINE	0.05	0.22	0.19	0.05	0.21	0.15	0.30
CONFINE*	0.07	0.23	0.20	0.06	0.21	0.16	0.28
CONFIVE	-0.02	0.05	0.03	-0.01	0.04	0.02	0.07
CONFIVE*	-0.03	0.01	0.00	-0.02	0.01	-0.01	0.02
1-SVM	0.00	0.12	0.11	-0.02	0.10	0.08	0.17
AvgBiasedDist	0.01	0.03	0.03	0.01	0.04	0.02	0.05
AvgBiasedDistOF	0.02	0.08	0.07	0.01	0.09	0.05	0.13
AvgDist	0.02	0.12	0.10	0.03	0.11	0.08	0.16
AvgDistOF	0.05	0.11	0.11	0.03	0.11	0.09	0.14
Bagging	0.11	0.20	0.18	0.11	0.19	0.16	0.25
Diff5NN	0.01	0.17	0.14	0.02	0.14	0.11	0.29
DiffNN	-0.01	0.14	0.12	-0.04	0.11	0.08	0.27
LocalBias	0.01	0.02	0.01	0.02	0.03	0.01	0.03
LocalCV	0.01	0.05	0.04	0.02	0.04	0.03	0.05
LocalVar	0.00	0.12	0.10	0.00	0.09	0.08	0.16
MinDist	-0.01	0.04	0.03	0.00	0.03	0.02	0.04
MinDistOF	0.01	0.08	0.07	0.01	0.09	0.04	0.13
NoNN*	0.01	0.12	0.11	0.00	0.12	0.07	0.16
NoNN	0.05	0.12	0.12	0.03	0.11	0.09	0.16
PredVar	0.00	0.12	0.10	0.00	0.09	0.08	0.16

For every confidence estimator, we calculated the average CEC by considering datasets with a different number of instances n, a different number of selected features m, and a different noise level σ. In the last column, we show the average CEC for the best parameter combination ($n = 1,000$, $m \leq 10$, $\sigma = 0.1$).

Table D.24: Confidence associated prediction improvement of confidence estimators on artificial data with different properties

	$n \leq 100$	$n > 100$	$m \leq 10$	$m > 10$	$\sigma < 1.0$	$\sigma \geq 1.0$	best
CONFINE	0.11	0.35	0.31	0.14	0.37	0.24	0.48
CONFINE*	0.13	0.38	0.32	0.17	0.37	0.27	0.48
CONFIVE	0.02	0.09	0.08	0.02	0.12	0.05	0.12
CONFIVE*	0.03	0.01	0.03	-0.02	0.08	0.00	0.04
1-SVM	0.08	0.22	0.21	0.02	0.21	0.16	0.25
AvgBiasedDist	0.13	0.06	0.09	0.06	0.11	0.07	0.07
AvgBiasedDistOF	0.07	0.16	0.14	0.06	0.21	0.10	0.21
AvgDist	0.07	0.22	0.19	0.08	0.21	0.16	0.25
AvgDistOF	0.12	0.22	0.21	0.07	0.22	0.17	0.25
Bagging	0.23	0.26	0.25	0.23	0.32	0.23	0.29
Diff5NN	0.04	0.22	0.18	0.06	0.20	0.15	0.35
DiffNN	0.04	0.20	0.18	-0.01	0.17	0.14	0.34
LocalBias	0.04	0.05	0.06	-0.01	0.09	0.03	0.05
LocalCV	0.06	0.12	0.11	0.07	0.12	0.10	0.11
LocalVar	0.06	0.21	0.19	0.01	0.17	0.15	0.25
MinDist	0.11	0.08	0.09	0.08	0.09	0.09	0.05
MinDistOF	0.05	0.15	0.13	0.04	0.21	0.08	0.23
NoNN*	0.10	0.22	0.21	0.06	0.21	0.17	0.24
NoNN	0.16	0.22	0.22	0.10	0.24	0.18	0.26
PredVar	0.06	0.21	0.19	0.01	0.17	0.15	0.25

For every confidence estimator, we calculated the confidence associated prediction improvement (CAPI) by considering datasets with a different number of instances n, a different number of selected features m, and a different noise level σ. In the last column, we show the average CAPI for the best parameter combination ($n = 1,000$, $m \leq 10$, $\sigma = 0.1$).

Table D.25: Performance of confidence estimators on biological datasets using linear regression

confidence	MHC			QSAR		
estimator	CEC	CAPI	runtime [ms]	CEC	CAPI	runtime [ms]
CONFINE	0.27	0.39	2	0.08	0.09	1
CONFINE*	0.23	0.33	2	0.07	-0.08	1
CONFIVE	0.24	0.35	2	0.09	0.13	1
CONFIVE*	0.17	0.22	2	0.08	0.11	1
1-SVM	0.02	-0.02	1	-0.08	-0.23	1
AvgBiasedDist	0.03	0.00	3	-0.04	-0.17	1
AvgBiasedDistOF	0.00	0.01	3	0.02	0.00	1
AvgDist	0.11	0.18	2	-0.02	-0.10	1
AvgDistOF	0.02	0.00	2	-0.03	-0.02	1
Bagging	0.13	0.18	1	0.20	0.35	1
Diff5NN	0.16	0.17	2	0.05	0.13	1
DiffNN	0.24	0.32	2	0.00	-0.14	1
LocalBias	0.08	0.11	481	0.01	-0.05	429
LocalCV	0.16	0.27	214	0.08	0.10	353
LocalVar	0.10	0.17	482	-0.08	-0.22	430
MinDist	0.04	0.05	2	-0.04	-0.24	1
MinDistOF	0.01	0.02	2	0.03	0.05	1
NoNN*	0.07	0.09	2	-0.02	-0.06	1
NoNN	0.10	0.17	2	-0.03	-0.09	1
PredVar	0.10	0.17	1	-0.08	-0.22	1

For every confidence estimator, the avgCEC, the confidence associated prediction improvement (CAPI), and the time for an individual estimation in miliseconds on the MHC datasets and on the QSAR datasets is shown.

Table D.26: Performance of confidence estimators on biological datasets using support vector regression

confidence	MHC			QSAR		
estimator	CEC	CAPI	runtime [ms]	CEC	CAPI	runtime [ms]
CONFINE	0.23	0.41	9	0.23	0.32	9
CONFINE*	0.11	0.18	9	0.15	0.09	9
CONFIVE	0.21	0.34	10	0.16	0.21	10
CONFIVE*	0.08	0.28	9	0.11	0.07	10
AvgDist	0.12	0.23	9	0.02	0.03	12
Bagging	0.21	0.50	374	0.15	0.17	364
DiffNN	0.24	0.35	9	0.10	0.20	10
NoNN	0.22	0.18	9	0.12	0.14	44

For every confidence estimator, the average CEC, the average confidence associated prediction improvement (CAPI), and the time for an individual estimation in milliseconds on the MHC datasets and on the QSAR datasets is shown.

Table D.27: Correlation of estimated performance and real performance using linear regression

	artificial data	artificial data $n > 100$	MHC data	QSAR data
CONFINE	0.13	0.43	0.88	0.06
CONFINE*	-0.38	-0.35	-0.53	0.15
CONFIVE	0.06	0.25	0.96	0.32
CONFIVE*	0.01	-0.05	0.92	0.12
1-SVM	0.21	0.33	0.52	-0.02
AvgBiasedDist	0.02	0.17	0.56	-0.15
AvgBiasedDistOF	0.19	0.25	-0.05	0.02
AvgDist	0.27	0.36	0.90	-0.21
AvgDistOF	0.31	0.42	0.10	0.17
Bagging	0.03	0.39	0.95	0.13
Diff5NN	0.46	0.65	0.90	0.12
DiffNN	0.51	0.74	0.95	-0.12
LocalBias	-0.02	0.08	0.88	0.17
LocalCV	0.04	0.13	0.90	0.10
LocalVar	0.24	0.35	0.88	-0.04
NoNN*	0.26	0.37	0.79	-0.15
NoNN	0.00	0.41	0.89	-0.12
PredVar	0.24	0.35	0.88	-0.04

For every confidence estimator, we calculated the correlation ρ between the CEC on test data (CEC_{test}) and the CEC on the training data using an estimator trained on the same data ($\text{CEC}_{\text{train}}$). In case of MinDist, CEC_{test} is obtained by averaging the CECs of a cross-validation on the training data.

Table D.28: Correlation of estimated performance and real performance using support vector regression

	artificial data	artificial data $n > 100$	MHC data	QSAR data
CONFINE	0.22	0.44	0.84	0.12
CONFINE*	-0.33	-0.59	-0.56	0.04
CONFIVE	-0.11	0.21	0.94	0.39
CONFIVE*	-0.13	-0.30	0.60	-0.17
AvgDist	0.44	0.74	0.80	-0.10
Bagging	-0.33	-0.54	-0.47	0.05
DiffNN	0.03	0.27	0.90	0.17
NoNN	0.49	0.59	0.32	0.34

For the given confidence estimator, we calculated the correlation ρ between the CEC on test data (CEC_{test}) and the CEC on the training data using an estimator trained on the same data ($\text{CEC}_{\text{train}}$).

Table D.29: Binding affinities and confidence intervals of $G9_{209}$ and analogs

peptide	Exp IC_{50}	Pred IC_{50}	lower bound IC_{50}	upper bound IC_{50}
ITDQVPFSV	172	156.2	50.9	759.9
ILDQVPFSV	3.3	12.5	3.2	60.9
IMDQVPFSV	19.1	12.7	3.3	60.1
ILDQVPFSV	40.0	12.5	3.2	60.9
FTDQVPFSV	61.4	28.0	7.1	128.3
WTDQVPFSV	716.7	158.7	28.3	1052.1
YTDQVPFSV	86.0	35.5	11.8	146.6
ITWQVPFSV	34.4	68.1	17.7	327.0
ITFQVPFSV	66.2	96.0	26.6	450.6
ITYQVPFSV	33.1	95.9	32.2	426.0
ITAQVPFSV	95.6	135.2	44.6	572.2
ITMQVPFSV	40.0	41.0	13.7	183.6
ITSQVPFSV	637.0	138.8	45.5	639.1
ILWQVPFSV	1.7	5.5	1.4	25.5
ILFQVPFSV	2.0	7.7	2.4	36.2
ILYQVPFSV	4.9	7.7	2.6	35.7
ILAQVPFSV	11.5	10.8	3.6	45.9
ILMQVPFSV	7.5	3.3	1.1	16.4
ILSQVPFSV	20.0	11.1	3.5	49.8
WLDQVPFSV	11.5	12.7	1.6	94.3
FLDQVPFSV	2.2	2.2	0.3	16.4
YLDQVPFSV	2.3	2.8	1.0	12.5

For each peptide, the experimentally determined IC_{50} value (Exp IC_50), the predicted IC_{50} (Pred IC_{50}), and the lower and upper bound of the 80% confidence interval are given (as IC_{50}). The first peptide is $G9_{209}$, followed by a set of analogs. The mutated positions are printed in bold. Note that we calculate the confidence interval based on the log IC_{50} values.

Appendix E

Sequence Data

In the following, we list the proteins used in the localization change experiments from Section 3.3.6 including their Swiss-Prot AC, name, possible isoform, and the experimentally observed location and, if possible, references to the original paper.

Q75WG7 U13-HTXT (secreted)

MKLSALVFVASVMLVAASPVKDVEEPVETHLAADLKTIEELAKYEEAAVQKRSCIVGSKN
IGETCVASCQCCGATVRCIGEGTKGICNNYQTNNILGQILLYAKDTVVNTAGLLVCAQDL
SEYE

P07954 FH (mitochondrion) [182]

MYRALRLLARSRPLVRAPAAALASAPGLGGAAVPSFWPPNAARMASQNSFRIEYDTFGEL
KVPNDKYYGAQTVRSTMNFKIGGVTERMPTPVIKAFGILKRAAAEVNQDYGLDPKIANAI
MKAADEVAEGKLNDHFPLVVWQTGSGTQTNMNVNEVISNRAIEMLGGELGSKIPVHPNDH
VNKSQSSNDTFPTAMHIAAAIEVHEVLLPGLQKLHDALDAKSKEFAQIIKIGRTHTQDAV
PLTLGQEFSGYVQQVKYAMTRIKAAMPRIYELAAGGTAVGTGLNTRIGFAEKVAAKVAAL
TGLPFVTAPNKFEALAAHDALVELSGAMNTTACSLMKIANDIRFLGSGPRSGLGELILPE
NEPGSSIMPGKVNPTQCEAMTMVAAQVMGNHVAVTVGGSNGHFELNVFKPMMIKNVLHSA
RLLGDASVSFTENCVVGIQANTERINKLMNESLMLVTALNPHIGYDKAAKIAKTAHKNGS
TLKETAIELGYLTAEQFDEWVKPKDMLGPK

P07954 FH delta 43 (cytoplasm) [182]

MASQNSFRIEYDTFGEL
KVPNDKYYGAQTVRSTMNFKIGGVTERMPTPVIKAFGILKRAAAEVNQDYGLDPKIANAI

MKAADEVAEGKLNDHFPLVVWQTGSGTQTNMNVNEVISNRAIEMLGGELGSKIPVHPNDH
VNKSQSSNDTFPTAMHIAAAIEVHEVLLPGLQKLHDALDAKSKEFAQIIKIGRTHTQDAV
PLTLGQEFSGYVQQVKYAMTRIKAAMPRIYELAAGGTAVGTGLNTRIGFAEKVAAKVAAL
TGLPFVTAPNKFEALAAHDALVELSGAMNTTACSLMKIANDIRFLGSGPRSGLGELILPE
NEPGSSIMPGKVNPTQCEAMTMVAAQVMGNHVAVTVGGSNGHFELNVFKPMMIKNVLHSA
RLLGDASVSFTENCVVGIQANTERINKLMNESLMLVTALNPHIGYDKAAKIAKTAHKNGS
TLKETAIELGYLTAEQFDEWVKPKDMLGPK

P21549 AGT1 (peroxisome) [178]

MASHKLLVTPPKALLKPLSIPNQLLLGPGPSNLPPRIMAAGGLQMIGSMSKDMYQIMDEI
KEGIQYVFQTRNPLTLVISGSGHCALEAALVNVLEPGDSFLVGANGIWGQRAVDIGERIG
ARVHPMTKDPGGHYTLQEVEEGLAQHKPVLLFLTHGESSTGVLQPLDGFGELCHRYKCLL
LVDSVASLGGTPLYMDRQGIDILYSGSQKALNAPPGTSLISFSDKAKKKMYSRKTKPFSF
YLDIKWLANFWGCDDQPRMYHHTIPVISLYSLRESLALIAEQGLENSWRQHREAAAYLHG
RLQALGLQLFVKDPALRLPTVTTVAVPAGYDWRDIVSYVIDHFDIEIMGGLGPSTGKVLR
IGLLGCNATRENVDRVTEALRAALQHCPKKKL

P21549 AGT1 extendedisoform (mitochondrion) [178]

MFQALAKASAAPGSRAAGWVRTMASHKLLVTPPKALLKPLSIPNQLLLGPGPSNLPPRIM
AAGGLQMIGSMSKDMYQIMDEI
KEGIQYVFQTRNPLTLVISGSGHCALEAALVNVLEPGDSFLVGANGIWGQRAVDIGERIG
ARVHPMTKDPGGHYTLQEVEEGLAQHKPVLLFLTHGESSTGVLQPLDGFGELCHRYKCLL
LVDSVASLGGTPLYMDRQGIDILYSGSQKALNAPPGTSLISFSDKAKKKMYSRKTKPFSF
YLDIKWLANFWGCDDQPRMYHHTIPVISLYSLRESLALIAEQGLENSWRQHREAAAYLHG
RLQALGLQLFVKDPALRLPTVTTVAVPAGYDWRDIVSYVIDHFDIEIMGGLGPSTGKVLR
IGLLGCNATRENVDRVTEALRAALQHCPKKKL

P41921 GLR1 (mitochondrion) [130]

MLSATKQTFRSLQIRTMSTNTKHYDYLVIGGGSGGVASARRAASYGAKTLLVEAKALGGT
CVNVGCVPKKVMWYASDLATRVSHANEYGLYQNLPLDKEHLTFNWPEFKQKRDAYVHRLN
GIYQKNLEKEKVDVVFGWARFNKDGNVEVQKRDNTTEVYSANHILVATGGKAIFPENIPG
FELGTDSDGFFRLEEQPKKVVVVGAGYIGIELAGVFHGLGSETHLVIRGETVLRKFDECI
QNTITDHYVKEGINVHKLSKIVKVEKNVETDKLKIHMNDSKSIDDVDELIWTIGRKSHLG
MGSENVGIKLNSHDQIIADEYQNTNVPNIYSLGDVVGKVELTPVAIAAGRKLSNRLFGPE
KFRNDKLDYENVPSVIFSHPEAGSIGISEKEAIEKYGKENIKVYNSKFTAMYYAMLSEKS
PTRYKIVCAGPNEKVVGLHIVGDSSAEILQGFGVAIKMGATKADFDNCVAIHPTSAEELV
TMR

P41921 GLR1 delta16 (cytoplasm) [130]

MSTNTKHYDYLVIGGGSGGVASARRAASYGAKTLLVEAKALGGT
CVNVGCVPKKVMWYASDLATRVSHANEYGLYQNLPLDKEHLTFNWPEFKQKRDAYVHRLN
GIYQKNLEKEKVDVVFGWARFNKDGNVEVQKRDNTTEVYSANHILVATGGKAIFPENIPG
FELGTDSDGFFRLEEQPKKVVVVGAGYIGIELAGVFHGLGSETHLVIRGETVLRKFDECI
QNTITDHYVKEGINVHKLSKIVKVEKNVETDKLKIHMNDSKSIDDVDELIWTIGRKSHLG
MGSENVGIKLNSHDQIIADEYQNTNVPNIYSLGDVVGKVELTPVAIAAGRKLSNRLFGPE
KFRNDKLDYENVPSVIFSHPEAGSIGISEKEAIEKYGKENIKVYNSKFTAMYYAMLSEKS
PTRYKIVCAGPNEKVVGLHIVGDSSAEILQGFGVAIKMGATKADFDNCVAIHPTSAEELV
TMR

P00724 SUC2 (extracellular space) [26]

MLLQAFLFLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 delta20 (cytoplasma) [26]

MTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut113 (cytoplasm) [98]

MLLQASFPFPFGWFCDQNICINDKRN

P00724 SUC2 Mut210 (extracellular space) [98]

MLPLFLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut236 (extracellular space) [98]

MLLRLFLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut211 (extracellular space) [98]

MLLRLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut437 (cytoplasm) [98]

MLLAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT

IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut438 (cytoplasm) [98]

MLLSMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut331 (extracellular space) [98]

MLLRSTLFLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut301 (extracellular space) [98]

MLLRSTFLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG

SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut506 (cytoplasm) [98]

MLLVDRPLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK

P00724 SUC2 Mut321 (cytoplasm) [98]

MLLVDRSTGRPLLAGFAAKISASMTNETSDRPLVHFTPNKGWMNDPNGLWYDEKDAKWHLYFQ
YNPNDTVWGTPLFWGHATSDDLTNWEDQPIAIAPKRNDSGAFSGSMVVDYNNTSGFFNDT
IDPRQRCVAIWTYNTPESEEQYISYSLDGGYTFTEYQKNPVLAANSTQFRDPKVFWYEPS
QKWIMTAAKSQDYKIEIYSSDDLKSWKLESAFANEGFLGYQYECPGLIEVPTEQDPSKSY
WVMFISINPGAPAGGSFNQYFVGSFNGTHFEAFDNQSRVVDFGKDYYALQTFFNTDPTYG
SALGIAWASNWEYSAFVPTNPWRSSMSLVRKFSLNTEYQANPETELINLKAEPILNISNA
GPWSRFATNTTLTKANSYNVDLSNSTGTLEFELVYAVNTTQTISKSVFADLSLWFKGLED
PEEYLRMGFEVSASSFFLDRGNSKVKFVKENPYFTNRMSVNNQPFKSENDLSYYKVYGLL
DQNILELYFNDGDVVSTNTYFMTTGNALGSVNMTTGVDNLFYIDKFQVREVK