Technische Universität Berlin
Fachbereich Robotics and Biology Laboratory

Masterarbeit

Im Studiengang Informatik

Predicting protein contacts by combining information from sequence and physicochemistry

eingereicht von	Kolja Stahl Matrikelnummer: 325372
eingereicht am	5.2.2016
Gutachter	Prof. Dr. Oliver Brock Prof. Dr. Klaus-Robert Müller
Betreuer	Prof. Dr. Oliver Brock Dr. Michael Bohlke-Schneider

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den	Unterschrift

Abstract

Different kinds of information are used in contact prediction, e.g., structure-based, co-evolutionary or sequence-based information. They have inherent specific strengths and weaknesses. The hypothesis is that different types of information capture different aspects of the data and combining them can alleviate some of the weaknesses. The usefulness of combining different information has been demonstrated by multiple groups. We will extend this to include all of the aforementioned information.

In this thesis we develop a sequence-based learner that is later combined with physico-chemical information. The neural network uses a feature set that has been evolved over the years. A main insight is, that a popular feature in the field, the local amino acid composition, has been rendered redundant. Our hypothesis is that is caused by the introduction of co-evolutionary information that extract similar information. Removing the local amino acid composition results in a drastically reduced dimensionality (by almost 75%) that allows us to train more complex networks and increase the size of the training set considerably.

We use stacking to combine the models and supply additional indicator variable to help the learner to identify when a source of information is most likely to be effective.

In our experiments, we outperform the current state-of-the-art in contact prediction MetaPSICOV on the CASP11 data set by 11% on $1.5L$ for long-range contacts.

Zusammenfassung

Es gibt verschiedene Arten von Informationen die in der Kontaktvorhersage benutzt werden, wie z.B. physicochemische, co-evolutionäre oder sequenz-basierte Informationen. Jede Informationsart hat ihre eigenen Stärken und Schwächen. Die Hypothese ist, dass unterschiedliche Arten von Informationen, unterschiedliche Aspekte der Daten erfassen und durch eine Kombination die jeweiligen Schwächen abgemildert werden können. Das es sinnvoll ist, verschiedene Arten von Informationen zu kombinieren, haben einige Gruppe bereits demonstriert. Wir werden die Kombination auf die oben genannten Informationen ausweiten.

In dieser Arbeit präsentieren wir zunächst einen sequenz-basierten Lerner der später mit physicochemischen Informationen kombiniert wird. Dem neuronalen Netzwerk liegt ein Feature Set zugrunde, das seit Jahren fortentwickelt wird. Eine Haupterkenntnis dieser Arbeit ist, dass ein sehr beliebtes Feature in der Kontaktvorhersage, die Zusammensetzung der Aminosäuren, redundant ist. Unsere Hypothese ist, dass das durch die vor Kurzem hinzugefügten co-evolutionären Informationen kommt, die das gleiche bezwecken. Die Dimensionalität des Featuresets wird durch das Entfernen des Features um fast 75% reduziert, was uns ermöglicht das Trainingsset zu vergrößern und komplexere Netzwerke zu trainieren.

Wir benutzen “stacking” um die Modelle zu kombinieren. Dem Lerner werden Indikatorvariablen zur Verfügung gestellt, die jeweils dabei helfen sollen die Informationsquelle zu identifizieren, die wahrscheinlich am effektivsten ist.

In unseren Experimenten auf den CASP11 Daten hat das neue Modell eine “mean precision” die ca. 11% höher ist, auf long-range Kontakten für $1.5L$, als die vom aktuellen State of the Art MetaPSICOV.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Proteins are the building blocks of life. They transport oxygen, provide structure for cells, fight intruders (antibodies), help digestion and the building of new molecules (enzymes), and perform a myriad of other functions [1, 2]. The function of a protein is determined by its structure. Knowing the structure is essential in drug design and biotechnology. The 3D structure can be determined in the laboratory by nuclear magnetic resonance spectroscopy (NMR), with electron microscopy, or X-ray crystallography. Unfortunately, the process is very time consuming and cost intensive. A comparably cheap approach is computation.

This thesis focuses on *contact prediction*, an intermediate step to the *protein structure prediction (PSP)* problem.

The protein structure prediction problem is one of the most important problems in Bioinformatics. The goal is, given a sequence of amino acids to predict the 3D structure of the protein. The huge search space makes this problem very difficult. It is generally unfeasible to do an exhaustive search. Therefore, it is necessary to devise strategies for targeted subsampling. Exploiting additional information can guide the sampling process.

Contact prediction tries to solve a smaller and easier problem first. Instead of predicting the 3D structure, we will try to predict inter-residue contacts. The problem is now, given two residues to predict if they are in contact. We define residues to be in contact if they are within 8 Ångström of each other in the 3D structure of the protein (see left-hand side of figure 1.1). The resulting contact map (see right-hand side of figure 1.1) can then be used to reconstruct the 3D structure of a protein [3–5].

There are mainly three kinds of information available in contact prediction to help identifying contacts.

Structure-based information uses predictions from either template structures (similar structures picked from a database) or from search space samples (decoys). Predictions from template structures are very accurate for good template matches, but it requires that

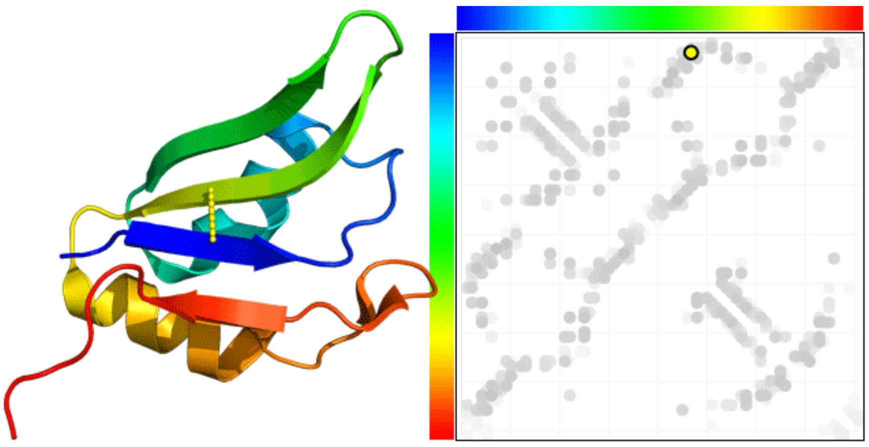

Fig. 1.1 Cartoon representation of the 3D structure (left) and the resulting contact map (right). The yellow dotted line (center) between two β-sheets (green-ish and blue) in the 3D structure denotes a contact, it is marked as a yellow circle in the contact map. Image source: [6]

similar structures are available. Which, especially in ab initio prediction, is seldom the case. Predictions from decoys work very well, even in *ab initio* prediction. Contacts can be taken directly from the decoys. The sampling process is guided by an energy function that takes physicochemical information as a basis, thus encoding the information directly into the decoy. The quality of physicochemical information is very much dependent on the quality of the decoys. The decoy quality generally suffers for more complex proteins. This is due to the bigger search space and the lower probability of sampling complex conformations.

The second type are co-evolutionary information. Residues that mutate in pairs are indicative of contacts. Since the mutation of a residue can lead to a destabilization of the structure, the other residue mutates as well to maintain stability. Multiple sequence alignments are used to identify co-mutating patterns. Co-evolutionary methods can work very well on their own, but they are highly dependent on sufficiently large multiple sequence alignments. This may become a limiting factor in ab initio prediction.

Finally, the third type of information is sequence-based. The features are derived directly from the sequence of amino acids. They contain for instance the amino acid composition, or secondary structure prediction and employ machine learning to identify patterns indicative of contacts. The biggest advantage of sequence-based information is that they work well in light of few additional information. Most successful predictors in CASP had a sequence-based component [7–9]. The Critical Assessment of protein Structure Prediction (CASP) [10] is a bi-annual set of blind studies to assess the current performance and progress in contact prediction.

The hypothesis of this thesis is that the information presented here capture different aspects of the data and the different profiles (strengths and weaknesses) can be exploited to further improve performance. The idea is to develop a model that identifies when a source of information is most likely to be effective, mostly with the help of indicator variables. Skwark et al. [8], Kosciolek [9] showed that combining sequence-based and co-evolutionary information work well. We will extend this to also include physicochemical information.

It is not clear what the best way to combine the different types of information is, the relationship is unknown and non-linear. For this purpose, we will use machine learning. The general goal in machine learning is to approximate an unknown function. In case of supervised learning, this is done by training the model on data with known truth (here: contact/non-contact at sequence positions i,j). We have a lot of data available, thus it is possible to treat the combination process as a learning problem.

For the physicochemical information we will use EPC-map, a contact predictor developed by Schneider et al. [11] that primarily leverages physicochemical information. The sequence-based component will be based on the feature set used by [9]. Both approaches also leverage co-evolutionary methods.

1.1 Contributions

The contributions of this thesis are as follows:

- a critical analysis of the sequence-based feature set used in [9] that led to a much reduced feature set (by approx. 75%)
- a sequence-based learner
- a new, state-of-the-art contact predictor combining physicochemical, co-evolutionary and sequence-based information

1.2 Thesis Structure

The thesis structure is as follows. In chapter 2 we review the types of information available, as well as look at prominent representatives. Chapter 3 includes background information and lays the ground work for chapter 4, where we will develop the sequence-based component. The final model is presented in chapter 5. In chapter 6 is the final conclusion and possible future directions.

Chapter 2

Related Work

2.1 Introduction

We want to combine multiple sources of information to boost performance. The different types of information have specific strengths and weaknesses. A main incentive is to reduce or remove weaknesses by including models that do not exhibit the same shortcomings. This section reviews the different sources of information that are used in contact prediction, with a particular focus on their strengths and weaknesses.

This section starts with physicochemical information.

2.2 Structure-based Information

Physicochemical Information

Physicochemical information is a variant of structure-based information. The search space is sampled and the resulting decoys are used to extract information. Decoys can for instance be generated by the standard ab initio protocol of Rosetta [12], where the decoy generation is guided by an energy function that encodes *physicochemical information* into the resulting decoy. This includes for instance the packing density, or the distance between hydrogen bonds [13]. Physicochemical information can work very well for ab initio prediction, because they don't require additional information. But the decoy quality generally degrades for more complex proteins or high *contact order* proteins.

The (relative) contact order is defined as the "[..]average sequence separation of residues that form contacts in the three-dimensional structure divided by the length of the protein" [14, p. 1937]. A high contact order usually implies a higher number of long-range contacts. Rosetta predictions are biased towards low contact order predictions [14]. According to

Wu et al. [15] folding simulations become the limiting factor for proteins exceeding 120-150 residues. In Rosetta, high contact order conformations are undersampled for proteins exceeding 80 residues [14]. This is mainly due to the very large search space. The available computing power becomes the limiting factor in the search.

After obtaining the decoys, potential contacts can be identified and ranked by looking at the number of occurrences in the decoys [16]. Zhu et al. [17] refine this approach by doing an energy-dependent weighting of the decoys. EPC-map [11] uses an intermediate graph structure instead, based on the identified contacts and its neighbors in the decoys to extract additional features that are fed into a SVM. An example of local graph-based feature is the diameter that conveys some information about the packing or compactness of the protein.

[16, 11] were both successful in recent CASP experiments.

Template-based Information

Template-based methods are structure-based as well. They use template structures instead of decoys. Templates are obtained by comparing the sequence or the sequence profile to a database of known structures [18–20]. To obtain predictions, the contacts are taken directly from the template. This can yield high accuracy predictions for good template matches, but assumes that there exist similar structures.

LOMETS [21] is a meta approach that pools results of different template-methods. The individual threading algorithms differ in the databases and scoring functions they use to find good matches [22, 23]. The final prediction of the meta predictor is based on a consensus score [24] reached by looking at 30 models of the top predictions [21].

2.3 Evolutionary Information

Residues that mutate in pairs are indicative of contacts. Since the mutation of a residue can lead to a destabilization of the structure, the other residue mutates as well to maintain stability.

Evolutionary methods look for co-evolving patterns in multiple sequence alignments (MSA). Evolutionary information have been used to improve secondary structure predictions [25] and to identify functional sites [26]. They can work very well on their own for contact prediction and have been included in most of the recent contact predictors. Combining multiple different sources of evolutionary information may further increase the performance [25, 27]. PConsC and MetaPSICOV combine up to 3 different methods.

Because evolutionary methods rely solely on the multiple sequence alignments, the performance is dependent on the quality of the MSA. There has to be a big enough sample size and sufficient diversity present. EVFold needs 5L sequences [26]. PSICOV [28] warns the user if fewer than 1L sequences are available. L refers to the length of the sequence. This can become a limiting factor in ab initio prediction, where usually only few sequences are available.

Furthermore, similar looking sequences can fold into different structures [29, 26]. This has a direct implication not only for evolutionary methods, but for all methods relying on MSA and may require a filtering step to avoid learning wrong patterns. Although co-evolutionary information have been successfully exploited in a range of different prediction tasks, they can cause false positives in contact prediction [26].

There exist a variety of different evolutionary methods. The main difference is how they handle and remove background noise. The primary problems are *phylogenetic bias* and *indirect coupling* [30]. Phylogenetic bias occurs if the method assumes i.i.d. samples, but the input sequences are close to one another in the phylogenetic tree. Dunn et al. [31] introduced the *average product correction (APC)* to mitigate the effect of phylogenetic bias in the computation of mutual information by doing a normalization. In indirect coupling, if AB and BC are in contact, it might lead to a stronger signal that AC are in contact as well, which may introduce false positives [30, 28]. PSICOV [28] tries to remove the effect of indirect coupling, expanding on the product corrected MI by Dunn et al. PSICOV works on the covariance matrix, another approach is pseudo-likelihood maximization (PLM) [32, 33]. PLM-based methods are expected to yield higher precision [32].

2.4 Sequence-based Information

Sequence-based machine learning approaches derive most of the features directly from the sequence of amino acids. Those features include for instance the amino acid composition, secondary structure prediction, or solvent accessibility [27, 8, 7]. Machine learning tries to identify patterns that are indicative of contacts.

Sequence-based methods have proven to be robust in light of few additional information. The quality of the features is mostly independent of the complexity of the protein and the lack of external dependencies is an advantage in ab initio prediction. Most successful entries in recent CASP experiments had a significant sequence-based component [27, 8, 7].

The approaches vary primarily in their use of machine learning algorithms and composition of the feature set [34, 27, 7, 35, 8].

2.5 Combining Multiple Sources of Information

Recent approaches in contact prediction showed improved performance by combining different sources of information.

MetaPSICOV [27] is the current state-of-the-art in contact prediction. Both, MetaPSICOV and PconsC2 [8] combine predictions from sequence-based and co-evolutionary methods.

EPC-map [11] used physicochemical and co-evolutionary information, BCL::Contact [36] combined sequence-based with physicochemical information.

MetaPSICOV

Given that MetaPSICOV is the current benchmark and our main comparison point, it makes sense to look at MetaPSICOV more closely.

MetaPSICOV consists of two stages. The CASP11 results are based on stage 2.

Stage 1 is an ensemble of 6 neural networks. The final prediction is the average over all predictions in the ensemble. The neural networks are shallow, single hidden layer networks with sigmoid activation and 55 hidden units. The networks are trained on different distance cut offs for the contacts.

They use a 672-feature set that we are going to adopt as a starting point and will more thoroughly introduce in section 4, see also [27].

Stage 2 is another neural network with the same architecture as described above. They use a slight alteration of the column features and sequence separation of stage 1. In addition, an excerpt of the contact map created by the stage 1 predictor is used as an input feature, which corresponds to a 11-by-11 window centered at i, j. A total of 731 features are used.

Stage 2 yielded higher accuracies than stage 1 in most cases. Although the accuracies were higher, the structure quality was worse. Stage 2 is "[..]a more accurate contact predictor, but at the expense of biasing the distribution of contacts to regions of the protein where adjacent contacts are made (beta-sheets)" [27, p. 7].

Summary

We will focus on EPC-map and MetaPSICOV, two of the currently best approaches combining multiple sources of information. The following table summarizes the information they leverage.

The new model we develop in chapter 5 will feature physicochemical, evolutionary and sequence-based information. Indicator features will be used to differentiate when a model is most likely to be effective and includes, e.g., the length of the sequence, number of

Algorithm	physicochemistry	evolutionary	sequence	indicator
EPC-map	✓	✓		
MetaPSICOV		✓	✓	✓
New model	✓	✓	✓	✓

Table 2.1 Overview: Leveraged information per algorithm

sequences in the alignment, presence of medium or long-range contact or secondary structure predictions.

Chapter 3

Background

The purpose of this section is to give a general introduction into the methods we will be using for the sequence-based learner.

3.1 Machine learning

"Machine Learning is the study of computer algorithms that improve automatically through experience" [37]. It has become ubiquitous in recent years, in some areas rivaling or even surpassing human-level expertise [38, 39]. Example applications include spam detection [40], face detection [41], object recognition [42], speech recognition and translation [43].

We will focus on supervised learning. The task is, given data X to predict the target variable y. The function $f : X \rightarrow y$ is generally unknown and needs to be estimated based on training examples.

The primary issue in machine learning is the bias-variance tradeoff. The difficulty is to find the balance between *overfitting* and *underfitting*. Overfitting usually occurs when a model is overly complex and starts to fit the idiosyncrasies of the data (noise). The result is worse performance on unseen data (overall worse generalization, has high variance). The opposite is underfitting. Due to the lower complexity, the model is unable to capture the underlying patterns of the data.

Machine learning algorithms require features. Our features will be based on the information reviewed in chapter 2. The feature engineering part is a crucial step in the learning process.

In the end, the learner should be able to discriminate contacts (1) from non-contacts (0).

We will now introduce the machine learning algorithms we use as part of the model selection in chapter 4.

3.2 Classification and Regression Trees (CART)

Classification and regression trees [44] are binary *decision trees*. A decision tree is a cascade of simple if-then-else constructs. The CART-algorithm uses feature thresholding (e.g., age ≤ 5 and age > 5) to recursively partition the data. The goal is to create subsets of the data that correspond to the same target variable. In each step, a feature is selected that generates the best split, according to some criteria (e.g., information gain that measures the reduction of entropy). This can be used to rank features, i.e., features higher up in the tree are more important (feature importance). The path from root to leaf is called a *decision rule*. Decision rules can be arbitrarily complex, which makes decision trees prone to overfitting. A major advantage of decision trees is that they are *white box* models, meaning, it is possible to inspect how and why a particular solution resulted. In addition, they are fairly low maintenance. The use of feature thresholding makes decision trees invariant to monotone transformations, thus reducing the need for preprocessing.

We will use two decision tree-based methods: *Random Forests* and *XGBoost*. Random Forests were specifically developed to counter the overfitting problem of decision trees. XGBoost is a newer approach that uses boosting instead of bagging. The main difference is in the training process.

Random Forest Classifier (RFC)

Random Forests were developed by Leo Breiman [45]. The idea is to create a forest of uncorrelated decision trees. Random Forests combine bagging and random feature selection. The randomness is injected at two stages into the training process. First, each tree is grown on a random subset of the data. The random samples are picked with replacement. This is the bootstrapping part of bagging (short for bootstrap aggregating). Second, at each split only a random subset of the features is used for the decision. Both measures try to avoid highly correlated trees and overfitting. The final prediction is the result of a majority vote over all trees, the aggregating. This mainly reduces the variance and improves the predictive power. Given the random nature of the tree building, there are a lot of trees that aren't particular good. By averaging over the predictions, the hope is that those cancel out. The tree ensembling is embarrassingly parallel.

The Random Forest Classifier has been recently used in PConsC2 [8].

XGBoost

XGBoost (or XGB, short for extreme gradient boosting) [46] has been successfully used in recent Kaggle competitions, usually as an integral part of the winning ensemble [47–49]. It implements a variety of *Gradient Boosting* algorithms, including Generalized Linear Model (GLM) and Gradient Boosted Decision Tree (GBDT). The focus is on scalability.

XGBoost differs from Random Forests mainly in the way it creates the tree ensemble. Trees do not have to be trained on a subset of the data or a subset of the features. The ensemble is build sequentially. In each round, k-trees are used to classify examples into k classes. New trees focus on previously misclassified examples to improve the discriminative power of the ensemble. Boosting increases the risk of overfitting, to prevent this, XGBoost employs early stopping. XGBoost can use any loss function that specifies a gradient.

3.3 Support Vector Machines (SVM)

The support vector machine (SVM) has been used with a lot of success in recent years [11, 7]. The goal is to find a hyperplane that best separates the two classes (see figure 3.1, filled and not filled circles). There are many possible hyperplanes. The choice is based on the training data. Intuitively, the most robust hyperplane is the hyperplane that puts the most space between samples of any class, resulting in a small buffer area. The optimization problem is to find the hyperplane with the biggest margin. It can be solved using quadratic programming and yields a unique solution.

The hyperplane is defined by its support vectors (circles on the dashed line in 3.1). We consider the *soft margin* SVM (see equation (3.1)), that introduces a *slack variable* ξ_i to improve the generalization ability by allowing some mislabeled samples.

$$\begin{gathered} \min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t. } y_i(w \cdot x_i - b) \geq 1 - \xi_i,\ \xi_i \geq 0 \end{gathered} \tag{3.1}$$

The leniency is controlled by the hyperparameter C. Big values for C lead to very slim margins that increase the susceptibility to overfitting, not allowing many mislabeled examples. On the other hand, very small values for C can lead to underfitting.

In the scenario depicted in the image, the two classes are linearly separable. For non-linear cases, we employ the *kernel trick*. A kernel function represents a dot-product. The idea is to project the data into a usually much higher dimensional space, where the data is hopefully linearly separable (see figure 3.2). The linear boundary corresponds to a non-linear

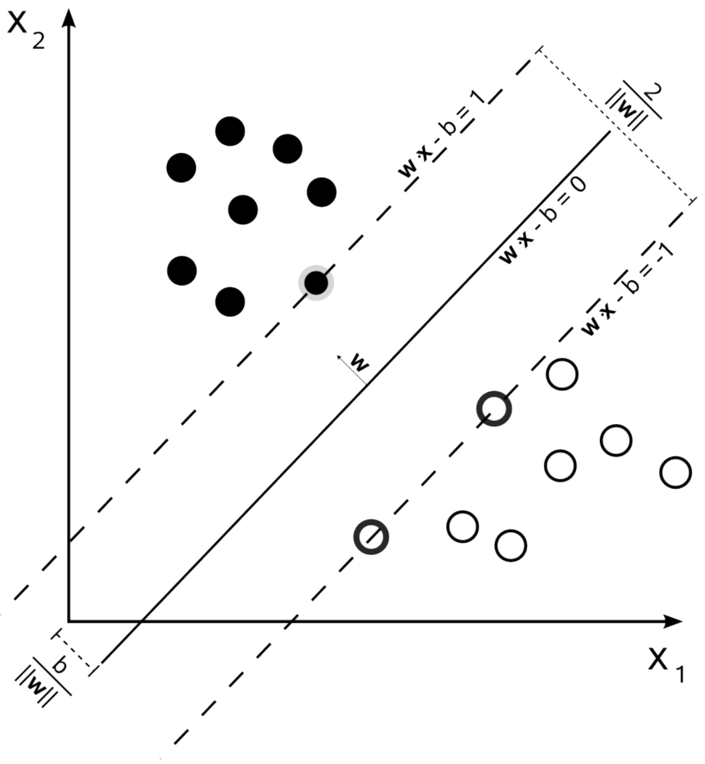

Fig. 3.1 Maximum margin hyperplane separating the two classes (filled, not filled circles), circles on dashed line represent support vectors that define the margin. Image source: [50]

boundary in the original data space. The most commonly used kernel is the radial basis function (RBF).

The hyperparameters depend on the kernel we choose. For the RBF kernel, in addition to C, there is the radius γ of the RBF.

A downside of the SVM is scaling. It can be assumed that the complexity is between $O(n^2)$ and $O(n^3)$ [52, p. 10] for n samples. A major component of the cost is the number of support vectors needed, which has a direct impact on testing time.

3.4 Neural Networks (NN)

Faster hardware and refined training strategies [53–55] helped the resurgence of neural networks.

The most exposure comes from *Deep Learning*. Deep learning is the term for neural networks with many hidden layers (usually with a minimum of 3 to 5+). They have cemented themselves as the state-of-the-art for many tasks in computer vision, improving on methods that incorporated years of manual feature engineering. Recently, recurrent neural networks

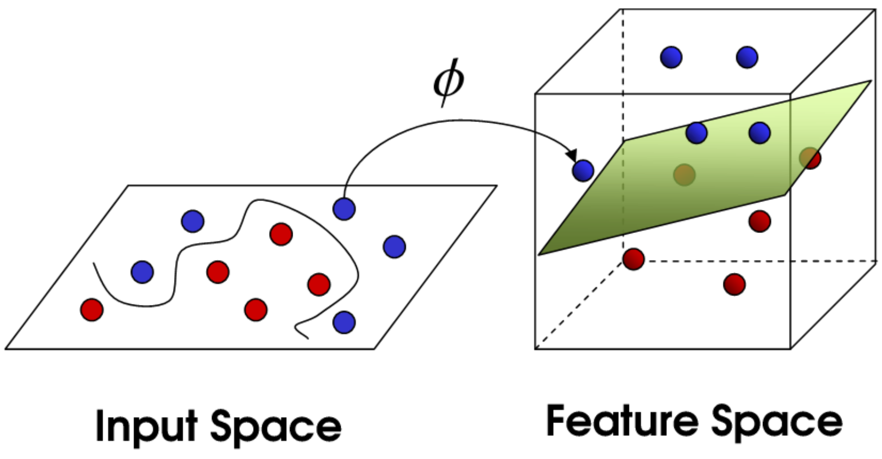

Fig. 3.2 Not linearly separable data (left) is mapped into higher dimensional feature space (right) via a kernel function ϕ, where data is now linearly separable. Image source: [51]

(RNN), a neural network architecture allowing loops and working with inputs of arbitrary length have been successfully applied to speech recognition [56].

The core idea of neural networks is inspired by biology and attempts to mimic the behavior of neurons in the brain. Neurons are connected through synapses and exchange signals. If the incoming signals exceed a given threshold, a neuron fires a signal, possibly igniting a cascade of further signals. The resulting neural stimulus patterns are associated with responses. We have billions of neurons in the brain that build a gigantic network. Learning happens by creating new connections and adapting already existing connections in the network.

In a neural network the artificial neurons receive a weighted combination of the inputs and produce an output. In the simplest case, the neural network learns a linear combination of the input data associated with the desired output. The interactions can be made more complex by introducing non-linear activation functions and by increasing the number of neurons in the hidden layers, as well as the number of hidden layers.

Neural networks are trained by backpropagation and usually optimized by stochastic gradient descent. The learning process is divided into the forward and the backward pass. In the forward pass, the input is threaded through the network producing an output. The output is then compared to the ground truth using a loss function and the resulting error is back propagated through the network in the backward pass. In the process, the weights of each layer are slightly changed in such a way that the error decreases. This is repeated multiple times. For binary classification the commonly used loss function is the log-loss or cross-entropy error. The cross entropy measures the similarity between two distributions p,q.

Assume p is the distribution of the true labels (based on training data), q is the distribution of the predicted labels. Ideally, $p = q$, to approach this, we try to minimize the cross-entropy (see equation 3.2).

$$H(p,q) = -\sum_i p_i \log q_i \tag{3.2}$$

A major advantage of neural networks is that they scale linearly with the number of samples. A downside is the plethora of hyperparameters and the somewhat non-schematic training procedure. It is often necessary to rely on intuition and rules of thumbs to properly tune the hyperparameters. Some of the hyperparameters include the overall architecture, that is the number of hidden layers and hidden units, the activation functions, the weight initialization, learning rate and the choice of momentum. Activation functions may have themselves additional parameters. A grid-search approach is practically unfeasible given the huge number of parameters.

For most of the hyperparameters we will follow current recommendations. The current recommendation for activation functions is to not use sigmoid activation.

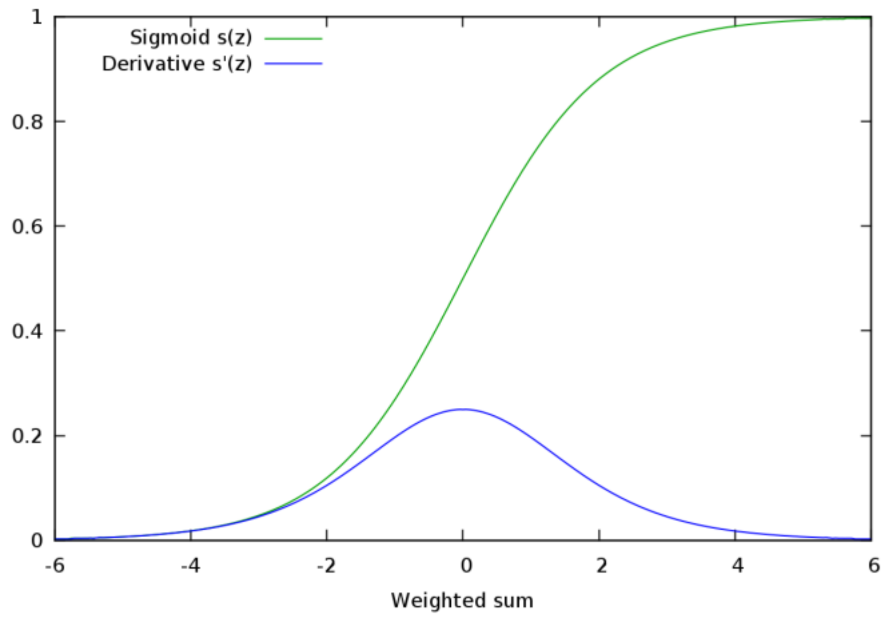

Fig. 3.3 Sigmoid function (green) and its derivative (blue). Image source: [57]

The problem with sigmoid activations is that they saturate, that is, the gradient is close to zero at either ends of the tail of the sigmoid function (see figure 3.3, where s(z) is close to either 0 or 1). Neural networks learn by back propagating the error through the network. The weights are updated based on the gradient. If a gradient is close to zero or actually zero, the neuron is essentially "dead" and nothing flows through it. In the same vein, the small magnitude of the derivative (maximum at 0.25) can cause another problem in deeper networks, called the vanishing gradient problem [58, 59]. The learning process can be slowed down or coming to a halt, because we have products of gradients and updates may get exponentially smaller in lower layers [60]. The slower learning or dead neurons in lower layers can result in subsets of the input that aren't propagated/used. New activation

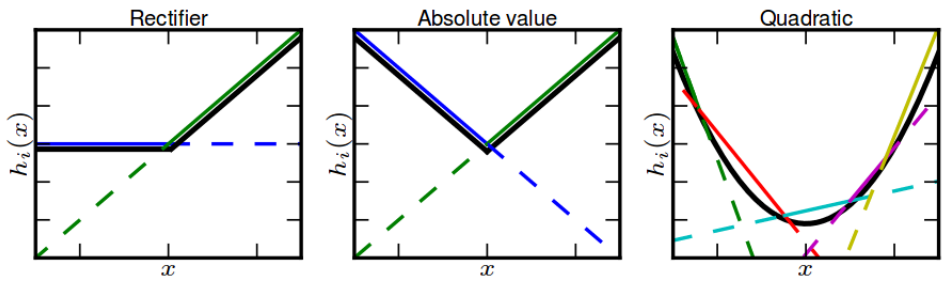

Fig. 3.4 Implementation of the rectified linear unit (ReLU) (left plot), the absolute value rectifier (middle plot) and an approximation to a quadratic activation function (right plot) by the Maxout function. Linear pieces are depicted as colored lines. Image source: [61]

functions have been developed that avoid the aforementioned problems. The rectified linear unit (ReLU) is such a non-saturating function, computing: $f(x) = max(0,x)$. The gradient is either 0 for negative values or 1 for positive values. Unfortunately, neurons can still "die" [62]. Maxout units learn a piecewise linear approximation of an arbitrary convex function [61], see also figure 3.4. They are generalizations of rectified linear units and do not suffer from dying neurons [62]. The downside is that they need to learn additional parameters.

Maxout units have been specifically developed to work well with *Dropout*. Neural networks have a disposition to overfit. Instead of learning the underlying patterns of the data that can be generalized to new data, the neural network starts to fit noise or memorize instead. Dropout randomly "drops" units and their connections during training. It forces the network to learn a more robust representation and approximately corresponds to training and averaging many smaller networks. Unfortunately, this approximation is only accurate for linear layers [63]. Because Maxout learns an activation function by combining linear pieces, it is linear almost everywhere except at the intersections of the linear pieces. Dropout is therefore more accurate in Maxout networks compared to networks that use non-linear functions [61].

Our experiments showed that Maxout units worked well and Dropout was essential for better performance. The experimental evaluation is part of the next section.

3.5 Ensembling and Stacking

Combining different models can improve the predictive power. The models should be as diverse as possible to capture different aspects of the data. Diversity can be created through the use of different machine learning algorithms or by training on different subsets of the data. The two main approaches are *ensembling* and *stacking*. The starting point is a pool of predictors.

In ensembles, the final prediction is reached by a vote or a simple average. For example, Random Forest uses a majority vote with equal voting rights. More often than not, the final prediction is a weighted average. The weights can be determined by regression. The approach is simple but can work rather well.

Stacking treats the combination process as another learning problem. The predictions become input features of a new learner. The advantage is that additional information can be used to aid the learning process. For example, by identifying when a particular model is most effective. The downside is that stacking is prone to overfitting and requires extra care to avoid information leakage. Therefore, the predictions are usually taken from the validation sets of the cross-validation.

Many machine learning algorithms, like Random Forest and XGBoost already incorporate some form of ensembling. Boosting and bagging is used to improve the performance of many weak learners by reducing the variance. In boosting the ensemble is constructed iteratively. Training samples are reweighted and subsequent models focus on previously misclassified data.

Chapter 4

Sequence-based information

4.1 Introduction

The goal of this thesis is to improve the accuracy in contact prediction by combining different sources of information. Chapter 2 reviews the types of information available in contact prediction. The different information have their specific strengths and weaknesses. The idea is that by combining them, we can alleviate some of the weaknesses. We will use sequence-based, co-evolutionary, and physicochemical information. Physicochemical information is a kind of structure-based information. For the physicochemical information we will use EPC-map, a contact predictor developed by Schneider et al. EPC-map has lower performance on complex proteins, due to the employed sampling method. The sequence-based component we develop in this chapter will put a special focus on more complex proteins. Both approaches also leverage co-evolutionary information.

We will start this section by giving a broad overview of the algorithm we are going to develop. This includes a glimpse at the data and the feature set. Following this is the implementation part that focuses on the model selection and hyperparameter tuning. We will conclude this section with the results of the model selection and a look at the implications.

4.2 Experiment Setup

We are going to develop a sequence-based classifier. It's not clear which machine learning algorithm will work best, that's why we will test different algorithms, namely support vector machine, neural network, and XGBoost.

Machine learning algorithms need features. Feature engineering is a crucial step in the development process. As a starting point, we take the feature set used by MetaPSICOV

[27]. The core of the features has been evolved over many years [34, 7]. Different groups have made adjustments. We want to critically analyze the feature set, especially in regard to combination approaches. It might also be important for scaling, because the focus on complex proteins leads to a lot of data.

Features

We will take the feature set of MetaPSICOV [27] as a basis. Jones et al. extended the feature set of Cheng and Baldi [7] with different co-evolutionary methods. They showed that PSICOV [28], CCMpred [32] and mfDCA [64] augment each other to some extent and ensembling them improves the predictive power. Some other aspects are a little bit different as well, e.g., replacing the binary indicators for the secondary structure (SS) predictions with the probability emitted by PSIPRED. The adaptation has been made for SOLVPRED as well. Instead of 3 different contact potentials, Jones et al. use the mean contact potential averaged over [65] and [66]. The residue type feature present in SVMcon has been dropped. In addition to the mutual information, there is now the average product corrected mutual information introduced by Dunn et al. [31]. The mutual information is computed over the MSA profiles for the contacting sequence positions i,j and measures the inter-dependence.

The features are divided into local and global features to limit the overall number of features and make them more accessible to machine learning algorithms.

The *local features* are computed on a window of the sequence of amino acids. Two windows of length 9 are centered at i and j. One window of length 5 is located at the midway point $(i+j)/2$. Most of the features consist of the amino acid composition or amino acid profile of the residue column (483 out of 672) in the multiple sequence alignment. The sequences of the multiple sequence alignment (MSA) are weighted. The sequences are compared pairwise and the weight is incremented if at least 35% of the positions are matching. The amino acid profile consists of the relative frequency of the 20 amino acids and a 21st position for the gap character. For each of the columns $(2 \cdot 9 + 5)$ there is also a secondary structure prediction (predicted probability of the structure being either a helix, coil, or strand, so all in all 3 features) and the solvent accessibility (exposed or buried). Also included is an indicator for missing data (window is overreaching). We decided to drop this feature, because it's already encoded in the column-based features with all zeros. This removes 23 features. Finally, there is the Shannon entropy of the i-th and j-th alignment column.

All of the column-based features are applied on the global (whole sequence) level as well, either as a relative frequency or average. In addition, there is the sequence separation $|i-j|$

Table 4.1 Overview of the features for the sequence-based learner.

Group	Features	Inputs
Column Features	amino acid composition, secondary structure prediction, solvent accessibility, alignment entropy	598
Co-evolutionary Features	GaussDCA, PSICOV, CCMpred, mfDCA, GREMLIN, mutual information, mutual information (APC), mean contact potential	9
Global Sequence Features	sequence positions, column features repeated on global level, sequence separation, log sequence length, log number of sequences in the alignment, log number of effective sequences	46

discretized and binned and the log of sequence length, number of sequences in the alignment and number of effective sequences determined in the weighting step.

We make initially the following changes: In addition to dropping the missing window indicator, we include two additional co-evolutionary methods with GaussDCA [67] and GREMLIN [68] and the sequence positions i,j.

The feature set is fairly high dimensional with 653 dimensions and the data gets big very quick, especially with the focus on longer and more complex proteins. For example, a protein with 500 residues contributes 124,750 samples to the data set. MetaPSICOV used very shallow networks and alternations of online and offline training to cope with the complexity. We had scaling issues as well.

The ongoing problems triggered a reevaluation of the feature set.

Feature Importance and Refined Feature Set

During construction, tree-based models do a feature importance ranking. The feature importance can be used as a starting point to evaluate the feature set. Although interesting, the feature importance sometimes lack meaningfulness. Correlation can inflate or deflate the importance of a feature.

The feature importance of Random Forest and XGBoost has a significant difference: If two features are strongly correlated, XGBoost keeps only one of the features. Random Forest may use both interchangeably on different occasions, thus deflating the importance of both features. The way XGBoost handles it seems more sensible and thus we will stick to the feature importance of XGBoost.

As mentioned in section 3.2, XGBoost splits the data set recursively. In each split, the feature that best separates the two classes is chosen. Features used in earlier splits are deemed more important. The specific feature importance measure we use is called *mean decrease of impurity.*

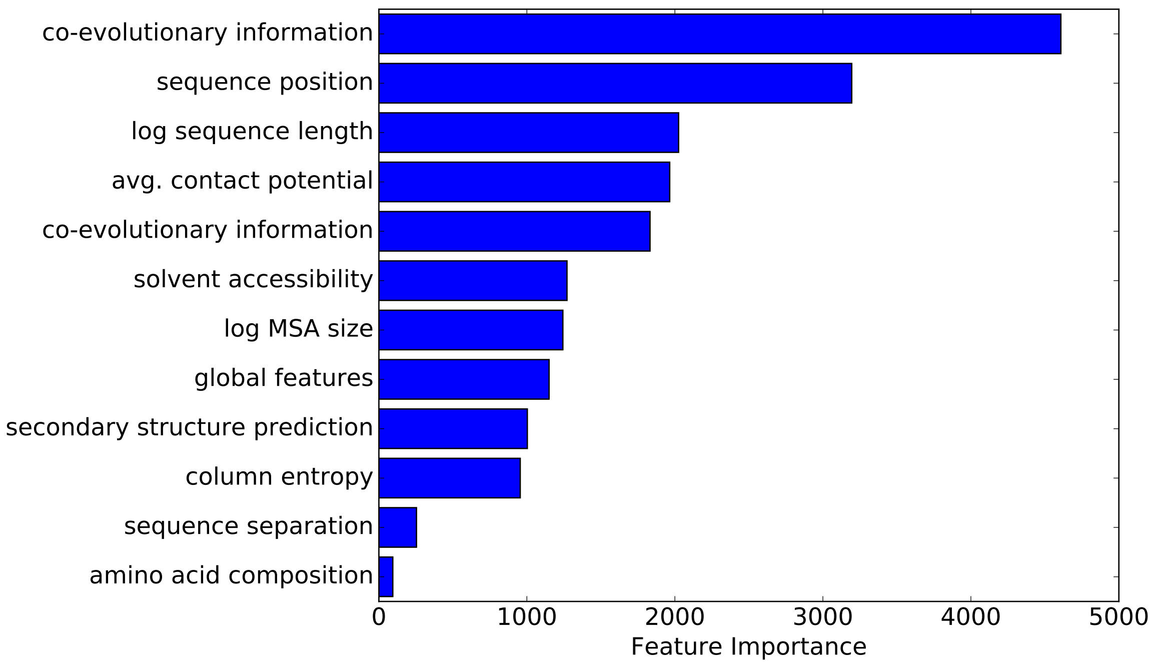

Fig. 4.1 Shows an excerpt of the feature importance ranking emitted by XGBoost. The higher the value is, the more important is the feature.

Figure 4.1 shows an excerpt of the feature importance ranking as emitted by XGBoost. Some features are aggregated (see, e.g., sequence position instead of i,j). The values depicted here are the average. In general, the higher the value, the more important the feature. For a complete overview consult table B.1 in the appendix.

We won't go too much into detail. The most interesting aspect is the difference in importance. It's clearly visible that, overall, co-evolutionary information are most important. This is not surprising, they have shown to yield good performance on their own, when enough alignments are available. It's also apparent that combining multiple different methods is helpful, as shown by [27]. Even factoring in correlation, the individual methods still show a strong signal. The newly added GaussDCA scores the highest amongst the evolutionary methods.

Also highly scoring are features that allow for a rough assessment of the quality of the data and the complexity of the protein with the log sequence length, log MSA size and the number of effective sequences.

Solvent accessibility and secondary structure prediction are very important, especially the window located at the midway point between i,j, that covers most of the medium-range contacts (see also table B.2 in the appendix) and contacts with smaller sequence separation. In experiments, dropping the mid window had almost no effect on the performance on long-range contacts, but a big impact on medium-range contacts.

Most noticeably is that the amino acid composition is ranked last with great distance (ignoring sequence separation). This disparity becomes even more striking, if we account for the dimensionality of the features. The amino acid composition makes up approximately 74% of the features.

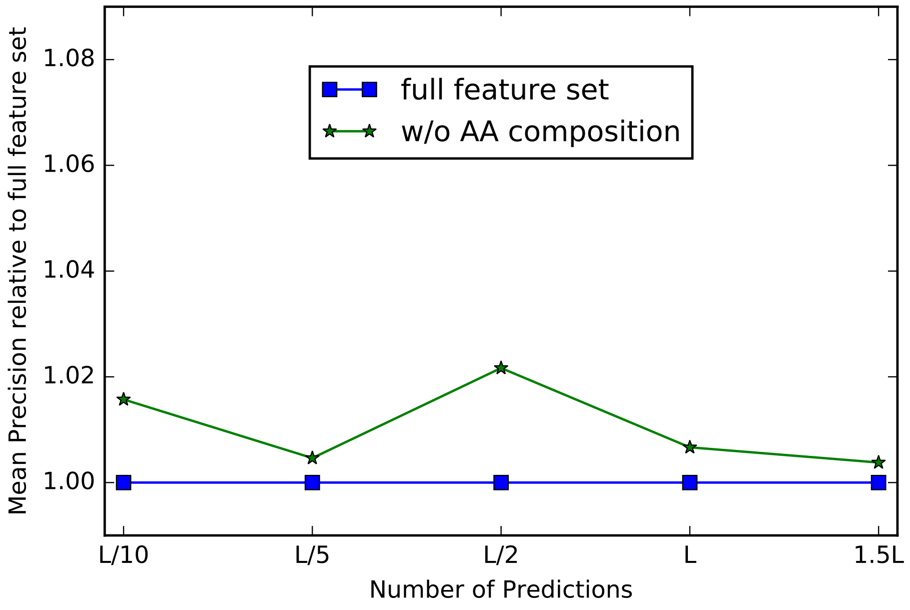

Fig. 4.2 Comparison of the neural network performance on long-range contacts with (square marker, blue line) and without (star marker, green line) the amino acid composition. The performance is shown relative to the full feature set.

The next logical step was to remove the amino acid composition. The impact on one of our models (here: neural network) is shown in figure 4.2. The performance is depicted relative to the full feature set (square marker, blue line). In contact prediction, the performance is evaluated on subsets of the data (see also more thorough description in appendix A) relative to the length of the sequence (L). The rationale is that only a high quality subset is necessary for reconstruction. The precision looks at the best, e.g., L predictions, that is the predictions with highest confidence and measures how many predictions were indeed contacts in the native structure. We will use the mean precision throughout the thesis, which is the average precision for a given cut off over all proteins in the data set.

Removing the amino acid composition leads to a slight increase in performance. The increase can be explained with the much reduced dimensionality and the overall easier optimization problem, also the curse of dimensionality might be a factor. The results are similar for medium-range contacts and our other models (XGBoost etc.). The main takeaway is that we don't sacrifice performance by dropping the local amino acid composition.

The much reduced dimensionality allows us to increase the training data for some of our models. Especially the neural network profits from increased data. We can now also increase the model complexity.

Hypothesis: Amino Acid Composition Redundant

Amino acid compositions or evolutionary profiles were added to identify evolutionary patterns. The idea is that if two residues are in contact and one of the residues mutates, the other residue mutates as well in order to maintain or restore stability of the structure (see also section 2.3). The Co-evolutionary information that have been recently added to the feature set fulfill this exact task. They are highly specialized and also account for different kinds of biases. Our hypothesis is that co-evolutionary information make the amino acid composition redundant.

Unfortunately, it seems to be a bit more complicated than that. We conducted some additional experiments where we removed the co-evolutionary information and then compared the performance with and without amino acid composition. On all occasions, removing the amino acid composition improved the performance. Further evidence that the curse of dimensionality may play a role.

Interestingly, a lot of papers mention that the amino acid composition or evolutionary profiles were essential for the performance [8, 69, 34]. They have in common that they used a broader definition of evolutionary profiles that, e.g., included the number of sequences in the alignment or the information per position in the position-specific weight matrix used for the sequence profile information that can be interpreted as the column entropy. All features that according to the feature importance ranking are more important than the amino acid composition itself.

We will use the refined feature set in the upcoming model selection.

4.3 Implementation

In this section we will do the model selection. We will briefly look at each classifier (Random Forest, XGBoost, SVM, Neural Network) and their respective hyperparameters. Classifier performance can vary based on data/ task, thus it makes sense to look at different approaches.

Data

We have initially a lot of data with roughly 140GB. With the reduced dimensionality, this shrinks to merely 35GB. For training and validation we use the EPC-map_train [11] training set in combination with a subset of the MetaPSICOV [27] training and test set. EPC-map_train consists of 742 proteins with $50-150$ residues. The MetaPSICOV training set is originally made up of 624 proteins and contains longer sequences and more complex proteins. The MetaPSICOV test set contains initially 434 proteins with 50 to 400 residues. The proteins have only 50 to 500 homologous structures, which makes them a lot harder. All three sets cap the sequence identity at 25%.

We decided to drop proteins with chain breaks from the MetaPSICOV sets out of convenience. Furthermore, we removed proteins with an HHsearch [70] E-value of $< 10^{-3}$ matching a protein in either EPC-map_train or EPC-map_test. This included some duplicate proteins, leaving a final set of 1543 sequences. The filtering is done to avoid overfitting.

63 proteins from the MetaPSICOV set are kept as a hold out set for the stacked learner (see chapter 5). The other 1479 proteins are used in training. For the final model evaluation of the sequence-based learner we use another hold out set with RBO_test [11], consisting of 132 proteins.

To avoid overfitting, the parameter selection for a particular classifier is done via a 5-fold cross validation on the training set. The initial split and the split for the folds are along the proteins, rather than the resulting feature vectors to prevent overfitting. Feature vectors of a protein might share distinctive characteristics/features.

A special attribute of the data is the high imbalance. Contacts make up only 3.5% of the data. Some machine learning algorithms struggle with imbalanced data and need extra attention. How we are going to cope with the imbalance depends on the algorithm used.

Software used

We used the following libraries: scikit-learn [71], Keras [72], XGBoost [46] and libSVM [73]. The code is primarily in Python, performance critical code is written in Fortran.

4.3.1 Support Vector Machine (SVM)

The data is highly imbalanced. Not addressing the imbalance can lead to a classifier that simply classifies everything as the majority class. This results in a high accuracy, but is of no practical use. Complicating is the runtime complexity of the SVM, which is between $O(n^2)$ and $O(n^3)$ (see also section 3.3), that makes it necessary to discard data.

Possible solutions to the class imbalance problem have been discussed in [74]. The most common approach is sampling. The main idea is to balance out the data by either dropping instances of the majority class (*undersampling*) or sampling with replacement from the minority class (*oversampling*). Oversampling can lead to overfitting by putting more emphasize on the redundant samples. Undersampling leads to a loss of information.

Regarding the size of the data, initial experiments showed that 100-120k training samples worked fine in training attempts. Taking roughly 25+ hours to a couple of days depending on the goodness of fit of the chosen parameters. 100k samples translate to only $10-15$ proteins, depending on the number of residues. This isn't enough to get any generalization ability.

We need to deal with the imbalance and reduce the data set size at the same time. The only viable option is undersampling. We will fix the number of contacts per protein to 200 (same as EPC-map) and reduce the overall number of proteins to 685. A 5-fold CV leaves approx. 105k training samples.

To further reduce the data set size and focus only on the interesting samples, we will exclusively train on medium- and long-range contacts. Spot tests showed that using a broader spectrum of samples, starting with a sequence separation of 6, what is commonly used, performed slightly worse. The effect isn't as pronounced, because the mass of the samples already lies within the interesting spectrum.

The data is scaled such that the mean is zero and the standard deviation is 1.

Hyperparameter tuning

We will use the RBF kernel. Polynomial and linear kernel didn't perform well. The interesting hyperparameters are C and the radius of the RBF γ. Undersampling leaves us with another hyperparameter to tune, the ratio of minority to majority examples (or split). The 3 hyperparameters have been determined by a grid search.

We will first look at the impact different ratios have on the mean precision (see also section A in the appendix). The notation is as follows. For 1to3 we randomly pick 1 minority example for every 3 majority examples. Figure 4.3 depicts the performance relative to 1to3 on medium-range contacts. The parameters are the same for all runs. We use a RBF kernel

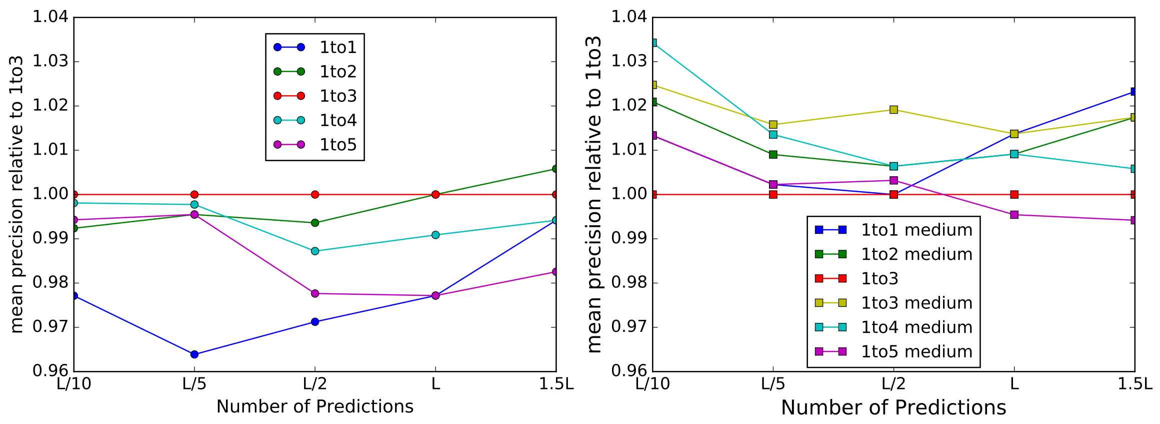

Fig. 4.3 Performance comparison on medium-range contacts using different ratios in undersampling. SVMs in left plot are trained on medium- and long-range contacts, SVMs in right plot solely on medium-range contacts. A split is based on a fixed number of 200 contacts per protein. 1to3 means for every randomly picked sample of the minority class, we pick 3 samples of the majority class. As a baseline, we took the mean precision achieved by the SVM trained on the 1to3 split. It's the average over a 5-fold cross validation. The hyperparameters of the SVMs were $\gamma = 0.001$, C = 10 with a RBF kernel.

with $\gamma = 0.001$ and $C = 10$. A grid search showed that $\gamma = 0.001$ and $C = 10$ yielded overall the best result. The mean precision is the average over a 5-fold cross validation.

The split 1to3 strikes the best balance between contacts and non-contacts (see left plot of figure 4.3). It achieves the highest mean precision for all cut-offs except $1.5L$, where 1to2 is better by approximately half a percent. Also noticeable is the negative effect of fewer minority samples on the performance from $L/5$ to $L/2$ relative to 1to3.

We still need to discard data, despite only training on medium- and long-range contacts. In the right plot (see figure 4.3) we further restrict the focus to only medium-range contacts. The baseline is again the 1to3 split (red line). All other results are obtained from SVMs only trained on medium-range contacts. 1to3, now only trained on medium-range contacts, has still the most consistent performance. The performance increased for all predictors but 1to5 (on L and 1.5L).

Unfortunately, we didn't see the same effect for training solely on long-range contacts. The performance was almost equal to training on both medium- and long-range contacts. Predicting long-range contacts with SVMs only fitted on medium-range contacts yield performances that are $10-15\%$ below SVMs fitted on medium- and long-range contacts.

Coincidentally, the parameters, including the split ratio, are the same used by EPC-map.

Results

The results of the cross validation show that the overall best performance is achieved by training medium and long contacts differently. The best performance on medium contacts was achieved by only training on medium contacts and using a RBF kernel with $\gamma = 0.001$ and $C = 10$ on a split that features 50 randomly picked positive samples and 150 randomly picked negative samples from each protein of the training set. The best performance for long-range contacts yields the SVM that is jointly trained on medium- and long-range contacts with $\gamma = 0.01$ and $C = 5$ and a 1to2 split.

4.3.2 Neural Network (NN)

Neural networks have been used in a couple of recent contact predictors [75, 76] and most notably the current state-of-the-art MetaPSICOV [27]. A significant advantage of the neural network, especially in comparison to the SVM, is the scaling behavior. Neural networks scale linearly with the number of data points. A downside is the more complex training. Finding the right architecture and good hyperparameters can be very time consuming.

Overall, the training approach is less systematic, compared to the grid search we employed for XGBoost and the SVM and more based on intuition, rules of thumbs, best practices and educated guesses. A grid search approach is practically unfeasible, because of the sheer number of hyperparameters. There are some general efforts to try to automate and optimize the hyperparameter search with *spearmint* [77] and *hyperopt* [78], that make for instance use of Bayesian optimization to quickly generate promising settings. The integration with neural networks is, however, lacking or very immature. [79] propose a gradient-based approach that is tightly integrated into the training process of the neural network, but isn't yet usable in conjunction with current neural network libraries. We will have to resort to spot checking.

One of the first choices we have to make is to pick a library. At the moment there are lot of evolving neural network libraries and backends. We settled on Keras [72]. Keras is a Python library that supports most of the cutting edge approaches and is build with modularity in mind, which makes it easy to extend. We will use the Theano [80] backend.

Prior to learning we have to prepare the data. Preprocessing is essential to neural networks and its learning ability. Features should be of similar scale, to avoid too big or too small steps in the gradient update. We use standardization (mean of zero, variance of 1).

Hyperparameter tuning

There are a lot of different hyperparameters to consider, including the overall architecture of the network, the choice of activation function, weight initialization, learning rate, and the batch size. We will present in this section an excerpt of the process that lead to the currently best model.

The main decision is the architecture. It's necessary to determine the number of hidden layers, as well as the number of neurons/units per hidden layer. We picked initially 50 hidden units, inspired by the 55 that MetaPSICOV uses and experimented then with multiples of 50 close to the actual number of features. We tried different combinations and finally settled on four hidden layers with $400-200-100-50$ units. Increasing the number of hidden layers to 4 improved the performance considerably and led to faster convergence. Deepening the network further didn't improve the results. A representative excerpt of the results from one of the cross-validation folds is depicted in figure 4.4, e.g., "400-50" refers to a two hidden layer network with 400 and 50 hidden units. The difference in performance on $1.5L$ long-range contacts between the shallow network with 50 hidden units and the neural network with 4 hidden layers is 5.5%. The difference is only marginal with 0.5% between the 3 and the 4 hidden layer network. A single epoch takes 190 seconds for the single layer network, 1400 seconds for the 3 hidden layer network and 1650 seconds for the 4 layer network.

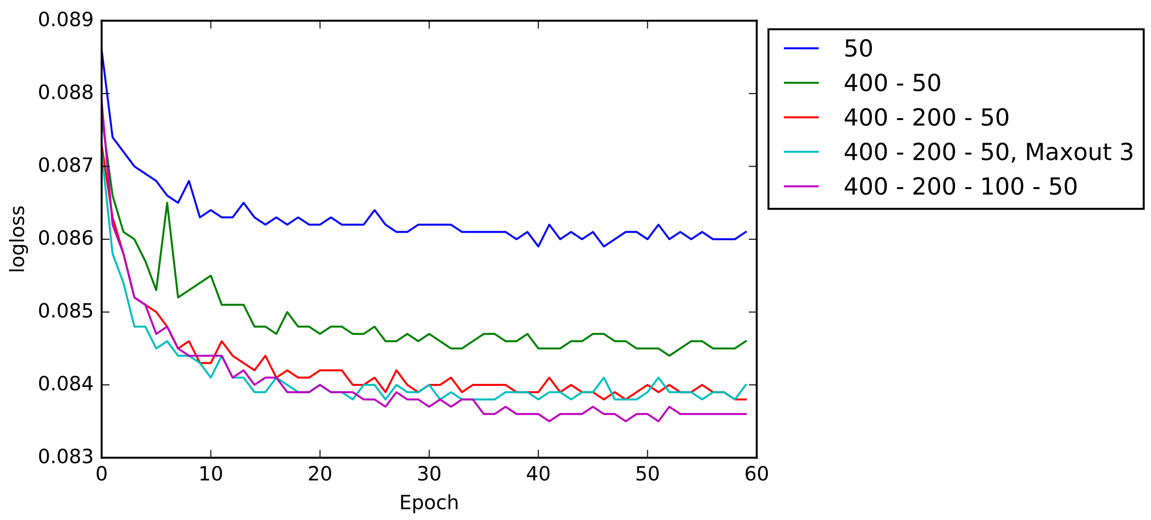

Fig. 4.4 logloss of different networks on a validation set

For the activation function we picked Maxout units after preliminary experiments. The advantages have been discussed in section 3.4. Maxout is a generalization of the widely used rectified linear unit (ReLU), but doesn't exhibit the same problems with dying units. It was clear early on that we needed strong regularization. Maxout has been developed to work well with Dropout, which made it a natural choice. The Maxout activation learns a piecewise linear function. We will stick to two linear pieces (similar to ReLU). Increasing the number of linear pieces reintroduced overfitting and led to worse generalization.

Knowing the activation function, it's easier to settle on the weight initialization. The weight initialization is crucial, especially for saturating units or units that can die. A bad initialization can cause the vanishing or the exploding gradient problem. The goal of the initialization is to spark learning (diversity) and aid learning (gradients should have similar variance). Progress in the weight initialization made it easier to train deeper networks and diminished the need for pretraining [38]. The current recommendation is to use the weight initialization introduced by He et al. [38, 81]. The weights are scaled by the respective layer dimensions.

Having set most of the parameters of the network, we are left with the parameters of stochastic gradient descent, the optimization algorithm we use and the choice of regularization. Stochastic gradient descent (SGD) has two parameters, the learning rate and (if present) momentum. The learning rate defines the size of the steps we are taking. If the learning rate is set to high, we can overshoot the local or global minimum. A learning rate of 0.01 worked best. Momentum helps to accelerate SGD [82] by reusing a fraction of the previous weight update. It was shown that momentum can have a negative effect on Maxout units [83], 0.5 worked best in experiments. SGD is computed over batches of training samples to speed-up training. We used 100 samples for a single batch and didn't conduct further experiments. A

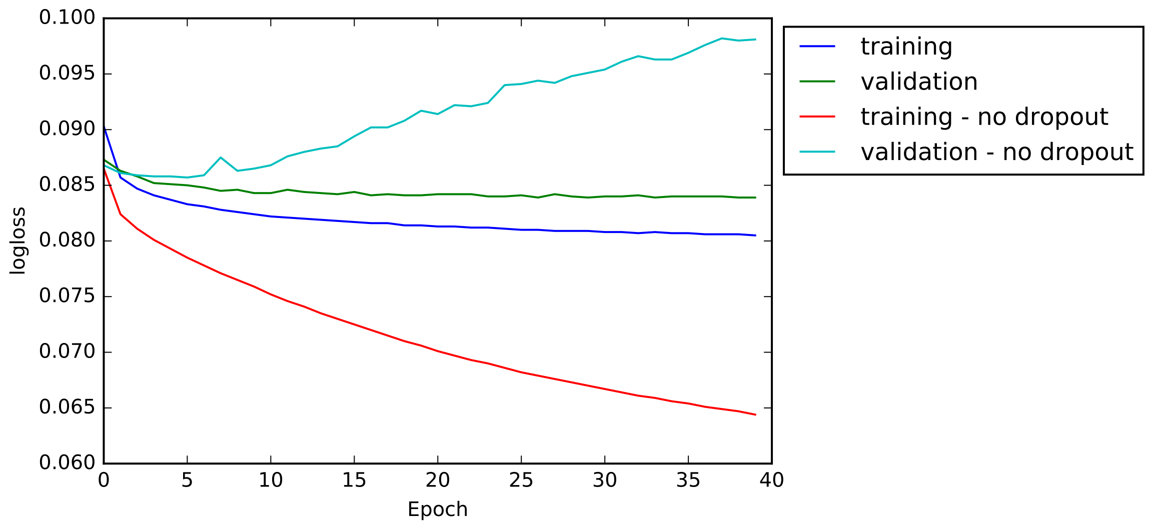

Fig. 4.5 logloss with and without dropout

common recommendation for neural networks is to increase the size of the network until it overfits and then to regularize strongly. We use dropout for regularization. The best result were obtained by dropping half of the units at random $(p = 0.5)$. Dropout allowed us to train the network longer, with no apparent overfitting and led to overall better generalization. The performance with and without dropout is depicted in figure 4.5. We can see that the validation error quickly increases without dropout and the network apparently starts to overfit within the first 10 epochs. The overfitting is further controlled by using early-stopping on the validation set and a 5-fold cross-validation.

All of the data fits into memory. A single epoch takes roughly 1650 seconds for 18mio. samples. Most networks reached the best result after around 40 to 50 epochs.

Essential to the performance was the reduced feature set and dropout. Increasing the data had a regularizing effect and overall improved the results significantly.

4.3.3 XGBoost

There aren't as many hyperparameters for Random Forest and XGBoost. A coarse grid search is sufficient.

Initial results of Random Forest were nowhere near XGBoost. We also had memory issues and decided to drop the Random Forest in later experiments and focus solely on XGBoost.

The most important parameters of XGBoost are the depth of the trees and the learning rate (eta). Increasing the maximum depth increases the chance of overfitting. The number of trees are determined by the number of rounds. We use early stopping on the validation set to determine the number of rounds. XGBoost can impose an asymmetric error to artificially balance out the data. It made no difference. The best result didn't balance the data and used eta = 0.1 and a maximum depth of 6.

XGBoost performed slightly better with the reduced feature set, with an increase in AUC on the validation, while simultaneously decreasing the AUC on the training set. Thus, improving generalization. The increase in data didn't have the same effect as with the neural network. But it was already possible to train on more data due to the sparse binary file format XGBoost uses.

4.4 Results and Discussion

We set out to do two things in this section. First, critically analyze the feature set. This hadn't been done before, also one aspect was the continuing difficulty with scaling. Second, test different machine learning algorithms, because it wasn't clear which one is the best choice.

In this section we will present the result of the model selection and the feature set analysis.

Reduced Feature Set

We used XGBoost as one of the machine learning algorithms in the model selection. XGBoost provides a feature importance ranking, which we used to assess the feature set.

Strikingly, the local amino acid composition, that makes up roughly 74% of the feature set was deemed unimportant, relative to the other features. To confirm this, we compared the performance with and without the amino acid composition. The result was, that the neural network without the amino acid composition performed slightly better. The performance improvement can be attributed to the easier optimization problem. The important aspect is, that we do not sacrifice performance by removing the amino acid composition.

Our hypothesis is, that this is the result of the newly added co-evolutionary features, that also focus on identifying co-evolutionary patterns and do a better job, in turn making the amino acid composition redundant.

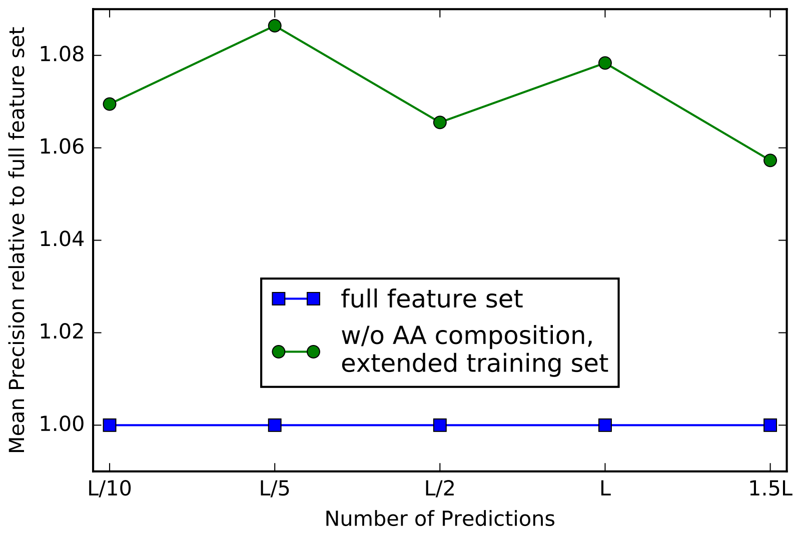

Fig. 4.6 Comparison of the neural network performance on long-range contacts with (square marker, blue line) and without (star marker, green line) the amino acid composition. It was possible to increase the size of the training data for the network with the reduce feature set. The performance is shown relative to the full feature set.

The much reduced feature set allowed us to significantly increase the size of the training data. The neural network profited the most from this increase (see figure 4.6), improving the mean precision by at least 6% on RBO_test.

A second observation is that the newly added features are amongst the highest ranked features. GaussDCA outperforms all other co-evolutionary methods, GREMLIN outperforms 2 of the 3 co-evolutionary methods that were in the original feature set. The sequence positions were the 3rd and 5th most important features.

Next, we will look at the result of the model selection.

Model Selection

The reduced feature set had a big influence on the model selection as well. Beforehand, the performance of XGBoost and the neural network was roughly on par. It also led to faster training of the SVM with fewer support vectors. The result of the model selection is depicted in figure 4.7. The neural network outperforms the other two algorithms. The algorithms are fairly close together on medium-range contacts (see left plot, in figure 4.7), the difference is more distinct on long-range contacts (see right plot, in figure 4.7) with almost 7% between the neural network and XGBoost on $1.5L$. The difference in performance between the SVM trained on only medium-range contacts and the fully trained SVM is at most 2.5% on $L/5$.

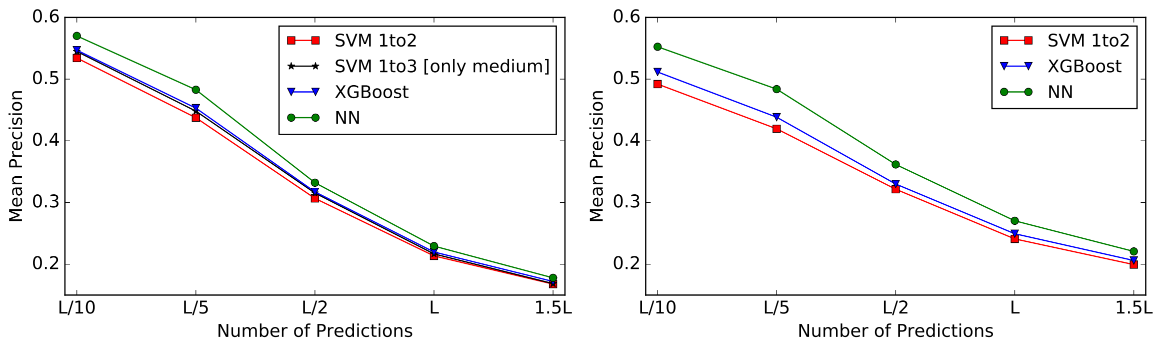

Fig. 4.7 Result of the model selection. Left plot: performance on medium-range contacts. Right plot: performance on long-range contacts.

4.5 Conclusion

We did a critical analysis of the feature set initially introduced by [7] and now used in the state-of-the-art contact predictor MetaPSICOV. We could show that a big component of the feature set can be removed without sacrificing performance. Our hypothesis is that the recently added co-evolutionary information make the amino acid composition redundant.

The new feature set is reduced in size by 74%. This allows us to train on more data, which improves the performance of the neural network significantly, that emerges as the best model in our experiments.

We will make a comparison to current contact predictors at the end of the next section, where we combine the newly developed sequence-based learner with EPC-map.

Chapter 5

Combining sequence-based and physicochemical information

5.1 Introduction

We want to combine multiple sources of information to further boost performance in contact prediction. Chapter 2 reviews the types of information available in contact prediction, along with current approaches that leverage different kinds of information.

The utilized information have specific strengths and weaknesses. The idea is that by combining them, we can alleviate some of the weaknesses. The usefulness has been successfully demonstrated by, e.g., [27, 8, 11, 36]. For example [27] combine sequence-based with co-evolutionary information, [11] leverages both physicochemical and co-evolutionary information. We are going to extend this to physicochemical, co-evolutionary, and sequence-based information.

The hypothesis is that the different kinds of information capture different aspects of the data. But it's generally unclear, what the best way to combine them is. We will treat the combination process as a learning problem. For this purpose, we will use machine learning and more precisely stacking. In stacking, the output of predictors become input features of a new learner. We will supply the model with additional, mostly indicator features, to help the model to identify when a source of information is most likely to be effective. A downside of stacking is the susceptibility to overfitting.

The chapter is structured as follows. We will start with a brief overview of the algorithm and followed by some implementation details. The section is concluded by an experimental evaluation.

5.2 Overview of the Algorithm

There are different approaches to combining models in machine learning. We discussed ensembling and stacking in section 3.5. Stacking allows us to use additional information to aid the learning process. For example, EPC-map should by construction perform very well on medium-range contacts and low complex proteins. Features like the length of the sequence or the sequence separation will help to identify such proteins and contacts.

We reuse the refined feature set introduced in section 4. The feature set contains a lot of global features, that can help the learner to assess the overall complexity of the protein, as well as the quality of the predictions. For example, a small number of alignments is generally detrimental to the performance of co-evolutionary methods. In addition to the sequence separation, there is now also an indicator for medium or long-range contact. By construction EPC-map doesn't have predictions for all possible contact pairs (i,j). The absence of a prediction is encoded as 0. In total there are 174 features.

5.3 Implementation

We will now briefly look at the implementation.

Data

We can only use a subset of the training data described in section 4.3. All RBO_train proteins have to be excluded to avoid information leakage. EPC-map has been trained on RBO_train and it would be necessary to change parts of the model to obtain untainted predictions. This removes 742 proteins from the training set, leaving 737. For the validation set, we kept 63 proteins, that have never been used in training.

Also, we have to take extra care to avoid information leakage. We need the predictions from the neural network and EPC-map as input features. The predictions have to be untainted. For this reason, we use the predictions on the validation sets used in the 5-fold cross-validation.

There is another cutback. EPC-map produces by default only medium- and long-range contacts. Further, depending on the decoys, there aren't necessarily predictions for all possible contacts. This isn't a problem per se, although having more samples can have a regularizing effect. To focus on the proper combination, we decided to only train on medium- and long-range contacts.

For evaluation we will use the following data sets:

CASP11 21 free modeling (fm, or template-free) targets with 110-470 residues, 9 targets are excluded, because we don't have the official predictions or PDBs

D329 [84]: 329 proteins with 55-458 residues

SVMcon_test [7]: 48 proteins with 46-198 residues, we removed protein 1a1hA, data was corrupt

RBO_test [11]: 132 proteins with 55-150 residues

PSICOV [28]: 150 proteins with 52-266 residues, we removed 7 proteins that overlapped with training data

Stacked Learner

The meta classifier is again a neural network. The best performance yielded a network with 400, 200, and 50 hidden units. The parameters are the same as described in section 4.3.2. We use Maxout units and Dropout. The network quickly converges to a good solution (10-15 epochs). Increasing the set should further improve the performance. It might be worthwhile to also produce predictions for smaller sequence separations with EPC-map. Including smaller ranges improved the performance of the sequence-based learner.

Next is the experimental evaluation.

5.4 Results and Discussion

In this chapter, we used stacking to combine physicochemical, co-evolutionary and sequence-based information. The learning process is aided by including indicator variables that allow, e.g., to assess the complexity of the protein, or the quality of the co-evolutionary predictions by including the number of alignments.

In this section we will present the final results. The evaluation is done on multiple data sets that are currently used in contact prediction to assess performance. The most important data set is CASP. CASP is a bi-annual set of blind studies. The targets are unknown at the time of the experiments. We will use the current CASP11 data set and concentrate on the hard free modeling targets.

The focus of our evaluation will be on long-range contacts. They are the most interesting contacts, because they put the most constrain on the structure. Also, they are generally the hardest contacts to predict. Long-range contacts have a sequence separation of at least 24.

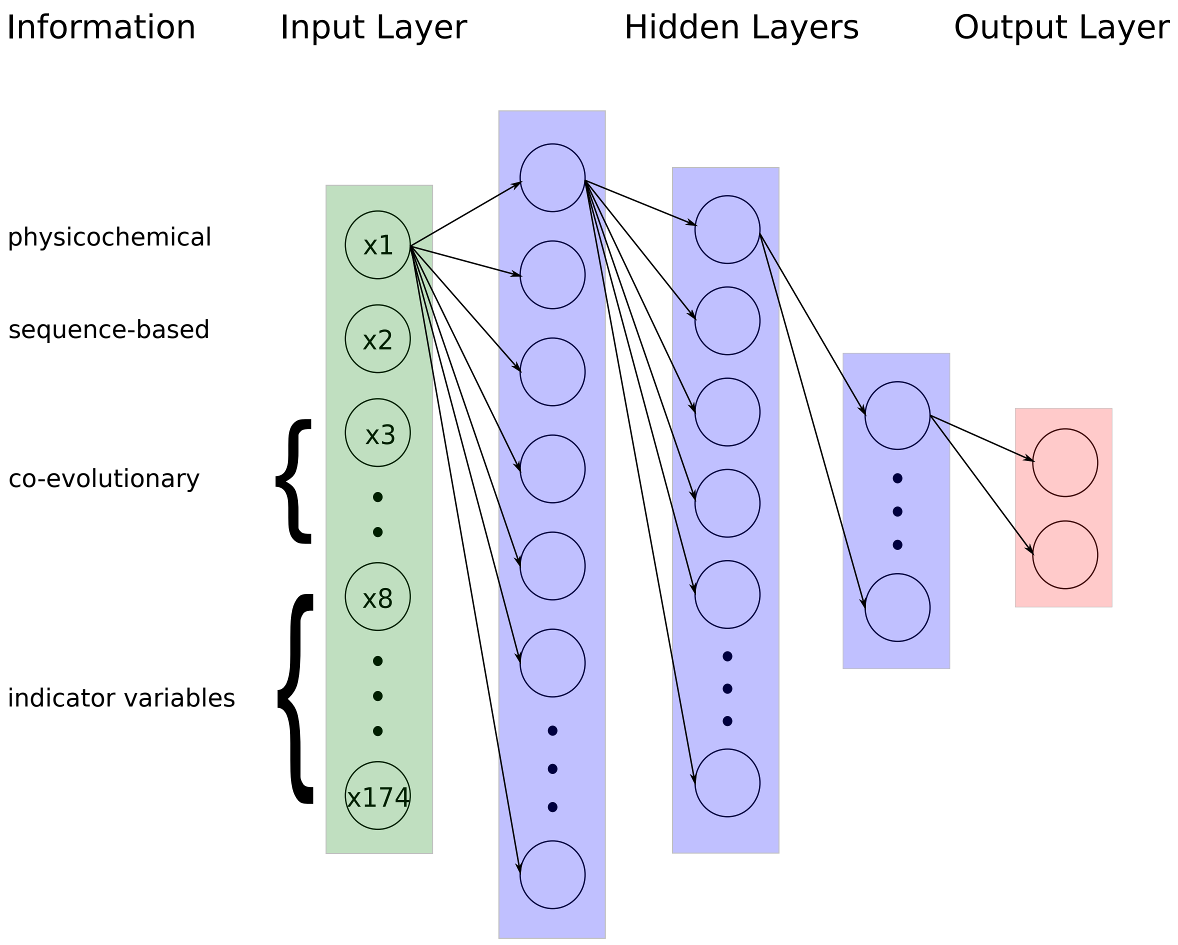

Fig. 5.1 Cartoon representation of the neural network. The network is fully connected, some connections are left for visibility reasons.

The performance is evaluated in contact prediction on a subset of the predictions relative to the length of the sequence. The rationale is that only a subset of the contacts is necessary for reconstruction. We will consider $L/10, L/5, L/2, L$, and $1.5L$, where L is the length of the sequence. The most important performance metric in contact prediction is precision (see also A in the appendix). We take, e.g., the L best predictions (sorted by confidence) and see how many of the L predictions were indeed contacts. For the comparison, we will use the mean precision. The precision is averaged over all proteins for a given cut-off. The results, together with the standard error are also in the appendix.

Our main comparison point is MetaPSICOV, the current state-of-the-art in contact prediction. It's interesting to look at both stage 1 and stage 2 of MetaPSICOV, because although the accuracy of stage 2 is higher, the structure quality of stage 1 is usually better.

All MetaPSICOV predictions are done with the latest version. To ensure fairness, the data (alignments, co-evolutionary predictions and such) are the same for all algorithms.

Weighting vs Stacking

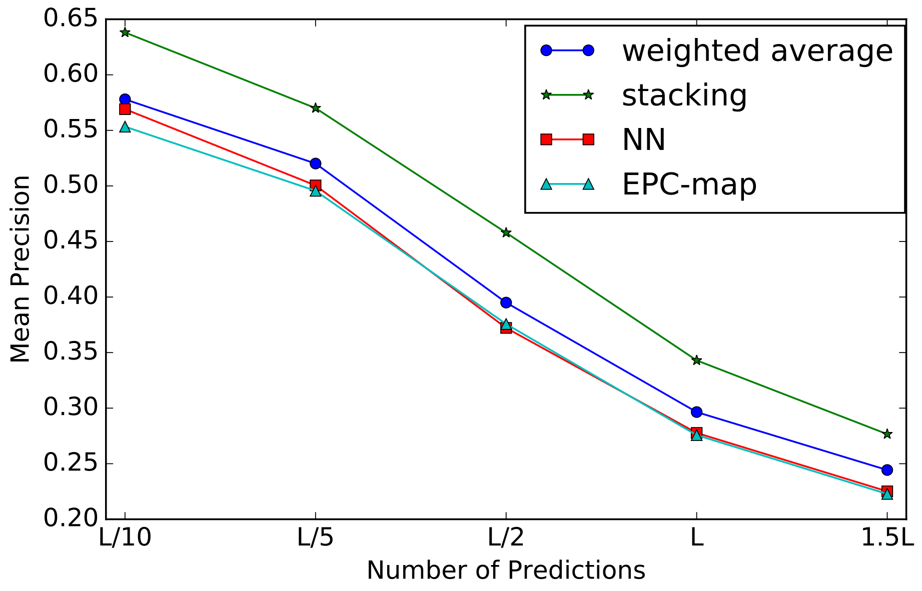

Fig. 5.2 Comparison of the performance on long-range contacts for stacking and taking the weighted average on RBO Test.

We decided to use stacking for the model combination. The idea is to learn, when a specific model is most likely effective. Another common approach in ensemble learning is to use a weighted average of the model predictions. In figure 5.2 is a comparison between weighting and stacking on RBO_Test for long-range contacts. The weights have been determined by ridge regression, included were the predictions from EPC-map, the neural network and all five of the co-evolutionary methods. We can see that taking the weighted average (blue line, circle marker) improves the result in comparison to the individual methods (NN, red line, square marker and EPC-map, cyan line, triangle marker), but falls short compared to the stacking approach. Stacking improves the results by a big margin, with almost 12% over the weighted average and over 18% compared to the neural network. This reinforces the initial assumption, that we are able to identify when a model is most likely effective and that the neural network is able to pick up on this non-linear relationship.

We will now look at the performance on the current CASP data set.

CASP

CASP is the current benchmark in structure prediction. The targets are evaluated on a per domain-basis. Not all PDBs are available, we restrict the evaluation to 21 targets.

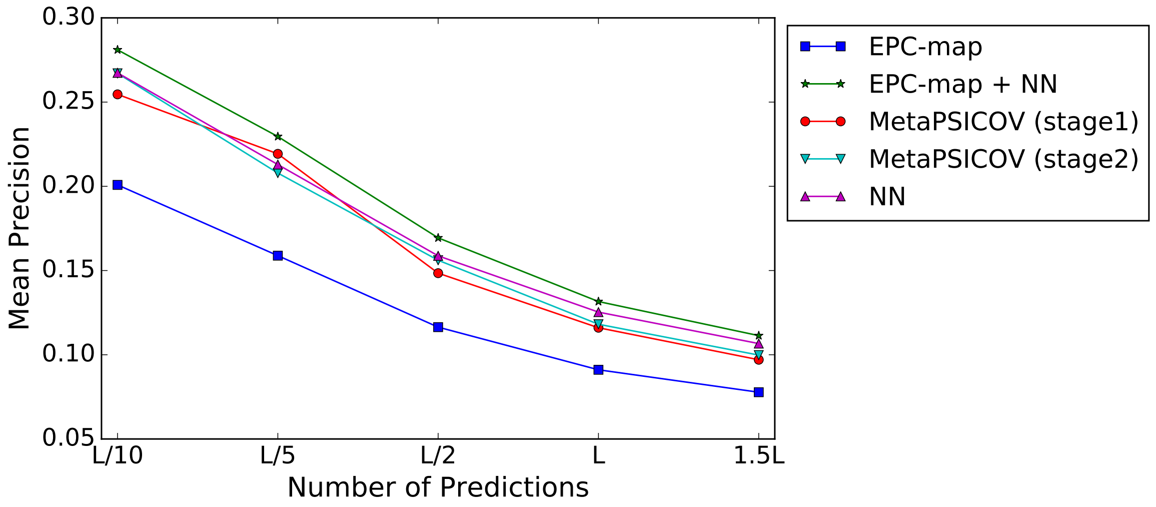

Fig. 5.3 Performance comparison on CASP11 on long-range contacts.

The results are depicted in figure 5.3. The stacked learner (EPC-map + NN, green line, star marker) outperforms all other algorithms. Starting with $L/2$, the newly developed sequence-based learner outperforms MetaPSICOV (red and cyan line, circle and lower triangle marker) as well. Although there is a big gap in performance between the neural network (magenta line, upper triangle marker) and EPC-map (blue line, square marker), combining the predictions improves the results, especially on $L/10$ through $L/2$.

One particular focus was to improve the performance on proteins with a complex topology. The CASP11 free modeling (template-free) targets are exactly such proteins. We improve the mean precision over MetaPSICOV by 11% on $1.5L$ for long-range contacts and by almost 30% compared to EPC-map. The improvement over the neural network is smaller with almost 4% on $1.5L$ and approx. 7.5% on $L/5$. The standard error is the same for all methods except for EPC-map on $L/5$, which is slightly higher, and the neural network on $L/10$, which is slightly smaller (see table in appendix C.1).

A particular difficulty of the hard targets is that they have comparably few alignments. The success of Jones et al. in CASP11 had two components: MetaPSICOV and a custom MSA pipeline [9]. The performance of machine learning algorithms is very much dependent on good data and many features make use of the multiple sequence alignments. The custom MSA pipeline they employed increased the number of alignments considerably (see table 5.1), which had a big impact on the performance (see figure 5.4).

The approach was born out of necessity. The database most commonly used is Uniprot [85]. The sequences are clustered to find similar structures. Due to the computational power required, the database is updated infrequently. At the time of CASP11, the database was at

database	min	max	median	mean	std. deviation
uniprot20_2013_03	1	766	49	110.57	172.6
uniprot20_2015_06	1	572	68	146.48	176.46
custom MSA pipeline	1	6044	179	676.38	1398.76

Table 5.1 Overview: Number of alignments per database

least a year old. To account for newer structures and get more alignments, they constructed a custom database with jackHMMer [86] to use in conjunction with Uniprot, if there were fewer than 2000 alignments available.

In the mean time there has been a new update (as of June 2015). In regard to the upcoming CASP12, it would be interesting to see how the different data sets fare. Unfortunately, we don't have predictions for the new database at this point. We will limit the comparison to the old database (that we used throughout training for all our algorithms) and the alignments of the custom database. Tomasz Kosciolek [9] was kind enough to provide us with the alignments they used in CASP11. In addition, we have the official CASP11 predictions for stage 2.

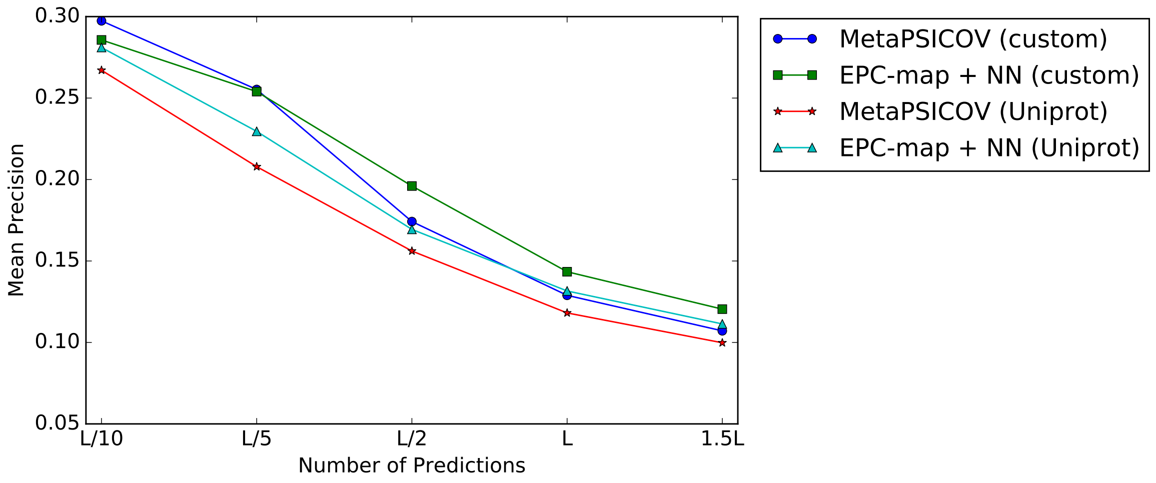

Fig. 5.4 Performance of MetaPSICOV on CASP11 on long-range contacts with and without custom MSA pipeline.

In figure 5.4 we can see the results with and without the custom MSA pipeline. The algorithms that use the custom pipeline (blue and green line, circle and square marker) outperform their counterparts without the additional alignments considerably. Also, MetaPSICOV stage2 seems to profit more from the increased MSA size, at least on $L/10$ and $L/5$. Interesting is that the stacked learner is beating MetaPSICOV on L and $1.5L$ regardless of the additional alignments. The marginal increase in the number of alignments from uniprot20_2013_03 to uniprot20_2015_06 compared to the custom MSA pipeline (still median size approx. 2.5 times higher) suggests that it is still a crucial factor and should always be done.

Additional Data Sets

The results are shown in figure 5.5, the mean precision together with the standard error is also tabled in the appendix C.

We can see that the stacked learner (green, star marker), combining EPC-map and the sequence-based neural network, comes out ahead on all data sets and for all cut-offs except $L/10$ on SVMcon_test. The gap between the stacked learner and MetaPSICOV (red and cyan, circle and lower triangle marker) is the higher, the smaller the gap is between EPC-map (blue, square marker) and the neural network (NN, magenta, upper triangle). Overall, there is a big improvement in performance compared to the individual models.

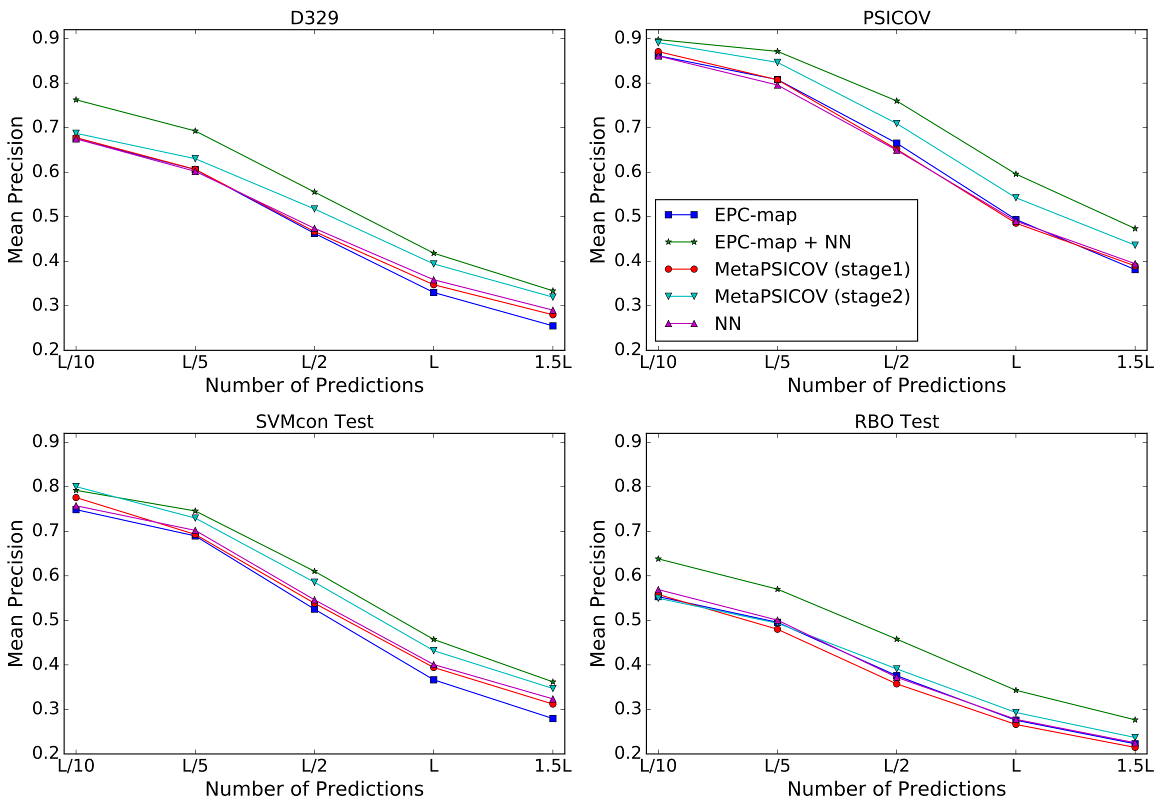

Fig. 5.5 Performance comparison on long-range contacts for different data sets.

5.5 Conclusion

We looked at the performance of different classifiers with and without the custom MSA pipeline that Jones et al. used in CASP11. The custom MSA pipeline resulted in much better performance, due to the larger number of alignments. Even with the updated database, the advantage is still present.

But the main focus was on the stacking. We built a new model that combines physicochemical, co-evolutionary and sequence-based information. The new, stacked learner outperforms the current state-of-the-art contact predictor MetaPSICOV on all data sets we looked at. The performance was the higher, the smaller the gap in performance between EPC-map and the sequence-based learner was. But even with a bigger gap (see, e.g., CASP11 results), we were still able increase the performance considerably.

Chapter 6

Conclusion

Our goal was to improve the performance in contact prediction by combining different kinds of information, namely physicochemical, co-evolutionary, and sequence-based. The underlying assumption is that they capture different aspects of the data and combining them can alleviate some of the weaknesses the individual types of information exhibit.

For this purpose, we built a sequence-based learner (see chapter 4). Because it wasn't clear, which machine learning algorithm performs best on the given task, we tested several algorithms, including support vector machine, neural network, and XGBoost. The neural network emerged as the best model in the experimental evaluation (see p. 28ff).

As a basis for the sequence-based learner we took a feature set has been evolved over the years. The feature set has been critically analyzed in chapter 4 (see p. 18ff). One aspect of the analysis was the high dimensionality that, combined with the big data set size, led to continuing scaling issues. To assess the individual features, we used a feature importance metric that is emitted by XGBoost, a decision tree-based machine learning algorithm. The result was that the least important feature (see p. 21ff), the local amino acid composition, that makes up 74% of the size of the feature set can be removed without sacrificing performance. Moreover, the performance increased, likely due to the easier optimization problem (see p. 21). Our hypothesis is that the amino acid composition is made redundant by the co-evolutionary methods that have been recently added to the feature set.

A second result was that the newly added features, that is, the sequence positions and two additional co-evolutionary methods with GaussDCA and GREMLIN ranked amongst the most important features (see p. 20). GaussDCA outperformed all co-evolutionary methods present in the original feature set.

The much reduced feature set, and in turn the smaller data set size allowed us to significantly increase the training data and to train more complex networks. The neural network profited the most from this increase (see p. 31).

With the sequence-based learner in hand, we set out to combine the neural network with EPC-map in chapter 5. It was unclear, what the best way to combine the different kinds of information is. We decided to use stacking and treat the combination process as a learning problem (see p.33ff). In stacking, predictions of learners become input features for a new learner. We supplied the stacked learner with additional indicator features to aid the learning process (see p. 33). The final learner is a 3-hidden layer neural network (see also p. 35).

In the final evaluation we focused on the hardest data set CASP11, where the new learner outperformed the current state-of-the-art in our experiments. An interesting aspect was the impact the custom MSA pipeline by Jones et al. had on the performance. Experiments suggest that the custom pipeline approach is also in the future preferable to simply using Uniprot (see p. 38).

We also looked at additional data sets (see p. 39ff). We were able to show that on RBO_test stacking was superior to just using a weighted average of the models (see p. 36), reinforcing our initial hypothesis. Further, the stacked learner outperformed the current state-of-the-art contact predictor MetaPSICOV on all additional data sets. The performance of the stacked learner was the higher, the smaller the gap was between EPC-map and the neural network. But even with a bigger gap, the new learner outperformed the individual methods significantly.

In conclusion, we developed a new contact predictor that combines physicochemical, co-evolutionary, and sequence-based information. The stacked learner outperformed the current state-of-the-art in our experiments.

Future Directions

We have already some ideas for improvements. First, it's still possible to almost double the size of the current training data, given our memory restrictions.

Second, an interesting direction is shown by the stage 2 approach Jones et al. (amongst other) took, where another network is trained on an excerpt of the contact map in order to correct and filter wrong predictions. It significantly improved the performance, but led to overall worse structures, due to redundancy. It would be interesting to see, if more complex models can help to lessen this effect. For this purpose, graphical models, that allow a cyclic structure, would be interesting as well. In which case adjustments to the contact map are immediately reflected and can be responded to.

A third idea concerns the features, and more precisely, the feature representations. Currently, we use a fixed window. Adding another window at the midway point between the contacts improved the performance on medium-range contacts. That is because the windows cover most of the medium-range contacts. It had almost no effect on long-range contacts. Distributed representations or the use of recurrent neural networks to summarize the sequence between sequence positions may allow for a more dynamic and fluid representation.

The fourth idea concerns EPC-map. The performance of EPC-map is very much dependent on the quality of the decoys. We can use predictions from, e.g., the sequence-based learner to improve the quality said decoys.

Finally, a custom MSA pipeline should be established, mainly in view of CASP12.

References

[1] National Library of Medicine. What are proteins and what do they do? http://ghr.nlm.nih.gov/handbook/howgeneswork/protein, 2015. Accessed: 2015-12-27.

[2] Protein functions. http://www.nature.com/scitable/topicpage/protein-function-14123348, 2014. Accessed: 2015-12-27.

[3] Badri Adhikari, Debswapna Bhattacharya, Renzhi Cao, and Jianlin Cheng. Confold: Residue-residue contact-guided ab initio protein folding. *Proteins: Structure, Function, and Bioinformatics*, 2015.

[4] Debora S Marks, Lucy J Colwell, Robert Sheridan, Thomas A Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3d structure computed from evolutionary sequence variation. *PloS one*, 6(12):e28766, 2011.

[5] Mirco Michel, Sikander Hayat, Marcin J Skwark, Chris Sander, Debora S Marks, and Arne Elofsson. Pconsfold: improved contact predictions improve protein models. *Bioinformatics*, 30(17):i482–i488, 2014.

[6] Contact map. http://gremlin.bakerlab.org/img/contact_map.gif. Accessed: 2015-11-05.

[7] Jianlin Cheng and Pierre Baldi. Improved residue contact prediction using support vector machines and a large feature set. *BMC bioinformatics*, 8(1):113, 2007.

[8] Marcin J Skwark, Daniele Raimondi, Mirco Michel, and Arne Elofsson. Improved contact predictions using the recognition of protein like contact patterns. 2014.

[9] Tomasz Kosciolek. Metapsicov: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. http://www.bioinfo.org.cn/~casp/papers/CONSIP2_presentation.pdf, 2014. Accessed: 2016-2-1.

[10] Critical Assessment of protein Structure Prediction (casp). http://www.predictioncenter.org/, 2007. Accessed: 2016-01-31.

[11] Michael Schneider and Oliver Brock. Combining physicochemical and evolutionary information for protein contact prediction. *PloS one*, 9(10):e108438, 2014.

[12] Carol A Rohl, Charlie EM Strauss, Kira MS Misura, and David Baker. Protein structure prediction using rosetta. *Methods in enzymology*, 383:66–93, 2004.

[13] Steven A Combs, Samuel L DeLuca, Stephanie H DeLuca, Gordon H Lemmon, David P Nannemann, Elizabeth D Nguyen, Jordan R Willis, Jonathan H Sheehan, and Jens Meiler. Small-molecule ligand docking into comparative models with rosetta. *Nature protocols*, 8(7):1277–1298, 2013.

[14] Richard Bonneau, Ingo Ruczinski, Jerry Tsai, and David Baker. Contact order and ab initio protein structure prediction. *Protein Science*, 11(8):1937–1944, 2002.

[15] Sitao Wu and Yang Zhang. A comprehensive assessment of sequence-based and template-based methods for protein contact prediction. *Bioinformatics*, 24(7):924–931, 2008.

[16] Jesse Eickholt, Zheng Wang, and Jianlin Cheng. A conformation ensemble approach to protein residue-residue contact. *BMC structural biology*, 11(1):38, 2011.

[17] Jiang Zhu, Qianqian Zhu, Yunyu Shi, and Haiyan Liu. How well can we predict native contacts in proteins based on decoy structures and their energies? *Proteins: Structure, Function, and Bioinformatics*, 52(4):598–608, 2003.

[18] Andrea Hildebrand, Michael Remmert, Andreas Biegert, and Johannes Söding. Fast and accurate automatic structure prediction with hhpred. *Proteins: Structure, Function, and Bioinformatics*, 77(S9):128–132, 2009.

[19] Jian Peng and Jinbo Xu. Raptorx: exploiting structure information for protein alignment by statistical inference. *Proteins: Structure, Function, and Bioinformatics*, 79(S10): 161–171, 2011.

[20] Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.

[21] Sitao Wu and Yang Zhang. Lomets: a local meta-threading-server for protein structure prediction. *Nucleic acids research*, 35(10):3375–3382, 2007.

[22] Ying Xu and Dong Xu. Protein threading using prospect: design and evaluation. *Proteins: Structure, Function, and Bioinformatics*, 40(3):343–354, 2000.

[23] Jiye Shi, Tom L Blundell, and Kenji Mizuguchi. Fugue: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *Journal of molecular biology*, 310(1):243–257, 2001.

[24] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4): 702–710, 2004.

[25] Burkhard Rost and Chris Sander. Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins: Structure, Function, and Bioinformatics*, 19(1):55–72, 1994.

[26] Debora S Marks, Thomas A Hopf, and Chris Sander. Protein structure prediction from sequence variation. *Nature biotechnology*, 30(11):1072–1080, 2012.

[27] David T Jones, Tanya Singh, Tomasz Kosciolek, and Stuart Tetchner. Metapsicov: Combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, page btu791, 2014.

[28] David T Jones, Daniel WA Buchan, Domenico Cozzetto, and Massimiliano Pontil. Psicov: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[29] S. Rackovsky. Nonlinearities in protein space limit the utility of informatics in protein biophysics. *Proteins: Structure, Function, and Bioinformatics*, 83(11):1923–1928, 2015. ISSN 1097-0134. doi: 10.1002/prot.24916. URL http://dx.doi.org/10.1002/prot.24916.

[30] Alan S Lapedes, Bertrand G Giraud, LonChang Liu, and Gary D Stormo. Correlated mutations in models of protein sequences: phylogenetic and structural effects. *Lecture Notes-Monograph Series*, pages 236–256, 1999.

[31] Stanley D Dunn, Lindi M Wahl, and Gregory B Gloor. Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, 24(3):333–340, 2008.

[32] Stefan Seemayer, Markus Gruber, and Johannes Söding. Ccmpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

[33] Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, and Erik Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87(1):012707, 2013.

[34] Marco Punta and Burkhard Rost. Profcon: novel prediction of long-range contacts. *Bioinformatics*, 21(13):2960–2968, 2005.

[35] Jesse Eickholt and Jianlin Cheng. Predicting protein residue–residue contacts using deep networks and boosting. *Bioinformatics*, 28(23):3066–3072, 2012.

[36] Mert Karakaş, Nils Woetzel, and Jens Meiler. Bcl:: Contact–low confidence fold recognition hits boost protein contact prediction and de novo structure determination. *Journal of Computational Biology*, 17(2):153–168, 2010.

[37] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45, 1997.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[39] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. *arXiv preprint arXiv:1404.3840*, 2014.

[40] Pranam Kolari, Akshay Java, Tim Finin, Tim Oates, and Anupam Joshi. Detecting spam blogs: A machine learning approach. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1351. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[41] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[42] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[43] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

[44] Leo Breiman, Jerome Friedman, Richard Olshen, Charles Stone, D Steinberg, and P Colla. Cart: Classification and regression trees. *Wadsworth: Belmont, CA*, 156, 1983.

[45] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[46] XGBoost github page. https://github.com/dmlc/xgboost, 2015. Accessed: 2015-09-13.

[47] Kaggle otto competition. https://www.kaggle.com/c/otto-group-product-classification-challenge. Accessed: 2015-08-25.

[48] Higgs boson competition. https://www.kaggle.com/c/higgs-boson/. Accessed: 2015-08-25.

[49] Liberty mutual competition. https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction/. Accessed: 2015-09-02.

[50] SVM hyperplane. https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:Svm_max_sep_hyperplane_with_margin.png. Accessed: 2015-09-10.

[51] Application of kernel function. http://www.nectarineimp.com/wp-content/uploads/2013/08/machine-learning-svm.jpg. Accessed: 2016-01-31.

[52] Léon Bottou and Chih-Jen Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.

[53] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[54] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[55] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[56] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[57] sigmoid derivative. http://whiteboard.ping.se/uploads/MachineLearning/sigmoid700.png. Accessed: 2015-11-30.

[58] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[59] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[60] Michael Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/chap5.html, 2015. Accessed: 2015-11-30.

[61] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[62] Andrej karpathy. CS231n convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-1/, 2015. Accessed: 2015-11-30.

[63] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[64] László Kaján, Thomas A Hopf, Debora S Marks, Burkhard Rost, et al. Freecontact: fast and free software for protein contact prediction from residue co-evolution. *BMC bioinformatics*, 15(1):85, 2014.

[65] Sanzo Miyazawa and Robert L Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18(3):534–552, 1985.

[66] Marcos R Betancourt and D Thirumalai. Pair potentials for protein folding: choice of reference states and sensitivity of predicted native states to variations in the interaction schemes. *Protein Science*, 8(02):361–369, 1999.

[67] Carlo Baldassi, Marco Zamparo, Christoph Feinauer, Andrea Procaccini, Riccardo Zecchina, Martin Weigt, and Andrea Pagnani. Fast and accurate multivariate gaussian modeling of protein families: predicting residue contacts and protein-interaction partners. *PloS one*, 9(3):e92721, 2014.

[68] Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence-and structure-rich era. *Proceedings of the National Academy of Sciences*, 110(39):15674–15679, 2013.

[69] Jesse Eickholt and Jianlin Cheng. A study and benchmark of dncon: a method for protein residue-residue contact prediction using deep networks. *BMC bioinformatics*, 14(Suppl 14):S12, 2013.

[70] Johannes Söding, Andreas Biegert, and Andrei N Lupas. The hhpred interactive server for protein homology detection and structure prediction. *Nucleic acids research*, 33 (suppl 2):W244–W248, 2005.

[71] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[72] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[73] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[74] Xinjian Guo, Yilong Yin, Cailing Dong, Gongping Yang, and Guangtong Zhou. On the class imbalance problem. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 4, pages 192–201. IEEE, 2008.

[75] Allison N Tegge, Zheng Wang, Jesse Eickholt, and Jianlin Cheng. Nncon: improved protein contact map prediction using 2d-recursive neural networks. *Nucleic acids research*, 37(suppl 2):W515–W518, 2009.

[76] Pietro Di Lena, Ken Nagata, and Pierre Baldi. Deep architectures for protein contact map prediction. *Bioinformatics*, 28(19):2449–2457, 2012.

[77] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[78] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of The 30th International Conference on Machine Learning*, pages 115–123, 2013.

[79] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Gradient-based hyperparameter optimization through reversible learning. *arXiv preprint arXiv:1502.03492*, 2015.

[80] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[81] Andrej karpathy. CS231n convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-2/, 2015. Accessed: 2015-12-03.

[82] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

[83] Anderson de Andrade. Best practices for convolutional neural networks applied to object recognition in images.

[84] Yunqi Li, Yaping Fang, and Jianwen Fang. Predicting residue–residue contacts using random forest models. *Bioinformatics*, 27(24):3379–3384, 2011.

[85] UniProt Consortium et al. Uniprot: a hub for protein information. *Nucleic acids research*, page gku989, 2014.

[86] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden markov model speed heuristic and iterative hmm search procedure. *BMC bioinformatics*, 11(1):1, 2010.

Appendix A

L-metrics

The de facto standard metric in contact prediction is the *positive predictive value* (PPV, see definition A.1) or *precision*. The performance is measured for different cut offs relative to the length of the protein.

$$\text{PPV} = \frac{\text{number of true positives}}{\text{number of true positives + number of false positives}} \tag{A.1}$$

The rationale behind this is, that only a high quality subset of the contacts are needed to rebuild the 3D structure of a protein. Thus, it make sense to prioritize optimizing for high confidence predictions of the subsets of interest. The most interesting cut offs are $1.5L$ and L, as they come close to the number of contacts necessary for reconstruction of the 3D structure [cite!!].

For each protein we predict the probability that a residue-pair is indeed a contact. The probabilities are sorted in descending order and we look at i.e. our top $1.5L$ predictions. It is assumed that there exist $1.5L$ contacts and the PPV measures how many we were able to find or appropriately rank. This might skew the results, if for instance, we have less than $1.5L$ contacts. Another metric called *coverage* which is akin to what is commonly called *recall* measures how many of the existing contacts we were able to identify. That is, it is possible to have a PPV or precision lower than 1 with a coverage of 1, meaning all contacts have been found.

Contacts are divided into ranges, most notably medium and long range. The performance is usually measured independently. The sequence separation of contact building residues in medium range contacts are between 12 and 23. Contact pairs with a sequence separation greater 23 are referred to as long range contacts. Medium range contacts are usually found within secondary structures. They form early on in the folding process and stabilize the structure. Long contacts are of most value for contact prediction, because they put the most constrain on the structure and its complexity. We don't consider short range contacts ($|i-j| < 12$) as they aren't adding much information and are usually fairly easy to predict.

CASP only considers $L/5$ in the performance evaluation.

Appendix B

Feature Importance

Table B.1 Feature Importance as emitted by XGBoost, the higher the better. Average over a 5-fold cross validation.

Feature	Feature Importance
GaussDCA	4607.6
DCA	4010.4
j	4008.8
GREMLIN	2667
i	2382.4
log sequence length	2025.4
Contact Potential	1965.2
PSICOV	1832.4
SOLVPRED	1270.97
log MSA size	1242.8
global SS prediction	1227.8
no. of effective sequences	1121.4
CCMpred	1104.4
global AA composition	1102.4
global solvent exposure	1028.6
PSIPRED	1002.84
local alignment entropy	954.76
global avg. entropy	724.8
normalised mutual information	679.4
mutual information	576.2
sequence separation	253.69
local AA composition	94.61

Table B.2 Feature Importance of local features broken down for i,j and mid window

Feature	Feature Importance
local AA composition window i	**145.6**
local AA composition window j	47.06
local AA composition window mid	88.42
SOLVPRED window i	1086.96
SOLVPRED window j	1299.24
SOLVPRED window mid	**1551.28**
PSIPRED window i	790.11
PSIPRED window j	934.38
PSIPRED window mid	**1508.97**
Alignment Entropy window i	707.18
Alignment Entropy window j	1000.6
Alignment Entropy window mid	**1317.92**

Appendix C

Results

Table C.1 Results for long-range contacts on CASP11

Algorithm	Mean Precision (Standard Error)				
	L/10	L/5	L/2	L	1.5L
NN	0.267 (.03)	0.213 (.03)	0.159 (.02)	0.125 (.01)	0.107 (.01)
EPC-map	0.201 (.04)	0.159 (.04)	0.116 (.02)	0.091 (.01)	0.078 (.01)
EPC-map + NN	**0.281 (.04)**	**0.230 (.03)**	**0.169 (.02)**	**0.132 (.01)**	**0.111 (.01)**
Stage1	0.255 (.04)	0.219 (.03)	0.148 (.02)	0.116 (.01)	0.097 (.01)
Stage2	0.267 (.04)	0.208 (.03)	0.156 (.02)	0.118 (.01)	0.100 (.01)

Table C.2 Results for long-range contacts on RBO Test

Algorithm	Mean Precision (Standard Error)				
	L/10	L/5	L/2	L	1.5L
NN	0.569 (.02)	0.500 (.02)	0.372 (.01)	0.278 (.01)	0.225 (.01)
EPC-map	0.553 (.02)	0.496 (.02)	0.376 (.01)	0.276 (.01)	0.223 (.01)
EPC-map + NN	**0.638 (.02)**	**0.570 (.02)**	**0.458 (.02)**	**0.343 (.01)**	**0.277 (.01)**
Stage1	0.558 (.02)	0.480 (.02)	0.357 (.01)	0.266 (.01)	0.215 (.01)
Stage2	0.550 (.02)	0.494 (.02)	0.391 (.02)	0.293 (.01)	0.237 (.01)

Table C.3 Results for long-range contacts on PSICOV

Algorithm	Mean Precision (Standard Error)				
	L/10	L/5	L/2	L	1.5L
NN	0.861 (.01)	0.796 (.01)	0.649 (.01)	0.490 (.01)	0.395 (.01)
EPC-map	0.862 (.01)	0.808 (.01)	0.665 (.01)	0.494 (.01)	0.381 (.01)
EPC-map + NN	**0.898 (.01)**	**0.871 (.01)**	**0.760 (.01)**	**0.596 (.01)**	**0.473 (.01)**
Stage1	0.871 (.01)	0.808 (.01)	0.652 (.01)	0.485 (.01)	0.390 (.01)
Stage2	0.891 (.01)	0.847 (.01)	0.709 (.01)	0.543 (.01)	0.436 (.01)

Table C.4 Results for long-range contacts on SVMcon Test

Algorithm	Mean Precision (Standard Error)				
	L/10	L/5	L/2	L	1.5L
NN	0.757 (.03)	0.702 (.03)	0.546 (.03)	0.401 (.02)	0.324 (.02)
EPC-map	0.749 (.03)	0.690 (.03)	0.525 (.03)	0.367 (.02)	0.280 (.02)
EPC-map + NN	0.792 (.03)	**0.746 (.03)**	**0.610 (.03)**	**0.457 (.02)**	**0.362 (.02)**
Stage1	0.776 (.03)	0.693 (.03)	0.538 (.03)	0.394 (.02)	0.312 (.02)
Stage2	**0.800 (.04)**	0.730 (.04)	0.586 (.03)	0.432 (.03)	0.347 (.02)

Table C.5 Results for long-range contacts on D329

Algorithm	Mean Precision (Standard Error)				
	L/10	L/5	L/2	L	1.5L
NN	0.675 (.01)	0.602 (.01)	0.474 (.01)	0.359 (.00)	0.290 (.00)
EPC-map	0.675 (.01)	0.606 (.01)	0.463 (.01)	0.330 (.01)	0.255 (.00)
EPC-map + NN	**0.763 (.01)**	**0.693 (.01)**	**0.556 (.01)**	**0.418 (.01)**	**0.334 (.00)**
Stage1	0.678 (.01)	0.606 (.01)	0.467 (.01)	0.347 (.00)	0.280 (.00)
Stage2	0.687 (.01)	0.630 (.01)	0.517 (.01)	0.394 (.01)	0.319 (.00)