IDENTIFICATION AND PREDICTION OF INTRINSICALLY DISORDERED REGIONS IN PROTEINS

by

Mauricio Oberti

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Bioinformatics and Computational Biology

Committee:

_____	Dr. Iosif Vaisman, Dissertation Director
_____	Dr. Dmitri Klimov, Committee Member
_____	Dr. Monique van Hoek, Committee Member
_____	Dr. Iosif Vaisman, Director, School of Systems Biology
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
_____	Dr. Peggy Agouris, Dean, College of Science

Date: _____ Spring Semester 2019
George Mason University
Fairfax, VA

Identification and Prediction of Intrinsically Disordered Regions in Proteins

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Mauricio Oberti
Master of Science
Johns Hopkins University, 2008

Director: Iosif Vaisman, Professor
School of Systems Biology

Spring Semester 2019
George Mason University
Fairfax, VA

DEDICATION

This dissertation is dedicated to my family, Marie, Louisa and Raoul, who have given me all their love, support, and patience.

To my father, Dr. Hugo Oberti, and mother, Maria Isabel Blanco, for being a constant source of inspiration.

ACKNOWLEDGEMENTS

I am the most grateful to Dr. Iosif Vaisman for agreeing to be my advisor. Under his guidance, I have learned and acquired the skills needed to become a better scientist.

I would like to especially thank Dr. Dmitri Klimov and Dr. Monique van Hoek for agreeing to be on the dissertation committee, and for all advice, corrections, and comments made on my dissertation and presentations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

IDENTIFICATION AND PREDICTION OF INTRINSICALLY DISORDERED REGIONS IN PROTEINS

Mauricio Oberti, Ph.D.

George Mason University, 2019

Dissertation Director: Dr. Iosif Vaisman

It has been the dominant paradigm in structural biology that a well-defined structure determines protein function. Intrinsically disordered proteins (IDPs), which lack a stable three-dimensional structure under normal physiological conditions, are a challenge to the structure-to-function paradigm. Disorder exists in up to half of the amino acids in eukaryotic proteins, and disordered regions are involved in numerous biological functions, as a result of their flexibility. Since amino acid sequence is known to determine protein structure, sequence information can be used to identify disordered regions. Protein disorder is involved in the development of many diseases, and identifying disordered regions can help us understand how to use them as potential drug targets. The identified regions can also be used to better understand the pathways of protein folding and provide insights into protein function.

In this study, we developed two machine-learning based algorithms to distinguish between disordered and ordered residues within a sequence-based on n-gram frequencies content and reduced amino acid alphabets.

Our results show that using n-gram frequencies is an accurate, computationally inexpensive and fast method to predict disordered regions, based on raw protein sequence data. Furthermore, we show that an algorithm using a combination of Convolutional Neural Networks architecture and reduced amino acid alphabets encoding achieves state-of-the-art prediction results on the CASP datasets. Both prediction algorithms can subsequently aid in the development of next-generation treatments for a variety of biomedical applications.

1. BACKGROUND AND SIGNIFICANCE

It has been the dominant paradigm in structural biology that a well-defined structure determines protein function. Intrinsically disordered proteins (IDPs), which lack a stable three-dimensional structure under normal physiological conditions, are a challenge to the structure-to-function paradigm [1]. Disorder exists in up to half of the amino acids in eukaryotic proteins [2], and disordered regions are involved in numerous biological functions, as a result of their flexibility. Since amino acid sequence is known to determine protein structure, sequence information can be used to identify disordered regions. Protein disorder is involved in the development of many diseases and identifying disordered regions can help us understand how to use them as potential drug targets. The identified regions can also be used to better understand the pathways of protein folding and provide insights into protein function.

In this study, we developed two machine-learning based algorithms to distinguish between disordered and ordered residues within a sequence-based on n-gram frequencies content and reduced amino acid alphabets.

Our results show that using n-gram frequencies is an accurate, computationally inexpensive and fast method to predict disordered regions, based on raw protein sequence data. Furthermore, we show that an algorithm using a combination of Convolutional Neural Networks architecture and reduced amino acid alphabets encoding achieves state-

of-the-art prediction results on the CASP datasets. Both prediction algorithms can subsequently aid in the development of next-generation treatments for a variety of biomedical applications.

n-grams are a commonly used technique in computational linguistics, probability, text categorization, and biology. In this study, an *n*-gram has been denoted as a contiguous string of *n* amino acid residues in a protein sequence [3]. The primary structure, or the amino acid sequence of a protein, determines the protein's three-dimensional structure. This implies that disorder, or lack of stable structure, can also be encoded in the sequence. A sequence can be decomposed into a list of overlapping *n*-grams. *n*-gram patterns have been previously used to show evolutionary relationships between protein sequences and to predict protein secondary structure. A key advantage to using *n*-gram frequencies is that they are a computationally inexpensive way of analyzing patterns in protein sequences.

Current predictors of protein disorder use multiple sequence alignments, secondary structure analyses, PSI-BLAST sequence profiles, or distinctive residue compositions [4]. These predictors require analyzing and comparing entire sequences, taking relatively longer time compared to *n*-grams, which decompose sequences into smaller pieces, each of which can be readily analyzed quantitatively. Computational techniques used for predictions are generally referred to as "black-box" models such as Neural Networks and Support Vector Machines, and the features that these models utilize are generally not fully understood. To help with feature selection, *n*-gram

frequency data can also be used to train decision trees, which can provide more insight into how the training data is used to create the decision-making process [3].

Current disorder prediction methods

Over 60 protein disorder prediction servers are currently available, although not all publicly available [5]. The methods they are based on can be classified in one of the following categories: (1) Ab initio or sequence-based, (2) clustering, (3) template based and (4) meta or consensus. The methods that can't easily be assigned to one of the above categories fall into the hybrid classification. Below is a brief description of the basis of each of the type of method.

Ab initio methods: They rely almost exclusively on amino acid sequence information to make a prediction. Features extracted from the primary sequence, alignment profiles or scoring matrices are used as input for statistical models to make predictions of disorder regions. This class of methods was widely used in the CASP8 and CASP9 experiments and a few examples of it are DISOPRED, ESpritz [6] and PreDisorder [7].

Clustering methods: This approach generates tertiary structure models from the primary sequence. It then superimposes the different models onto each other under the assumption that positions in ordered regions will be conserved across the models [8] and residues in disordered regions are likely to vary. Since this approach doesn't rely on a training set, it is less likely to be biased by the training data.

Template based methods: Template methods, similar to clustering, predict disordered regions of proteins by aligning the input sequence to homologous proteins with known

structure. Homologous proteins are found by database search or by fold recognition methods. This method is frequently used in combination with other prediction approaches and they generally fall into the hybrid category.

Meta methods: They combine the output of several disordered predictors into a single average, which tends to have a moderate increase in accuracy. This is one of the most popular methods and is used by metaPrDOS [9] and GSmetaDisorder among others.

Hybrid methods: These are the methods combining two or more of the previous approaches in order to improve prediction accuracy. PrDOS [10] is one common example, combining ab initio and template based homologous alignment to output a prediction.

2. METHODS AND MATERIALS

Databases and programs

DisProt is a manually curated database collection of intrinsically unstructured proteins. Latest release (6.02) of DisProt contains 694 unique proteins. DisProt has been used to understand the properties of intrinsically unstructured proteins and as diverse training set for numerous methods.

PDB (Protein Data Bank) is the worldwide repository of information about the 3D structures of large biological molecules, including proteins and nucleic acids. The structures are typically obtained by X-ray crystallography and NMR spectroscopy and submitted by biologists and biochemists from around the world.

CASP (Critical Assessment of protein Structure Prediction) is a worldwide experiment for protein structure prediction taking place every two years since 1994. CASP provides research groups with an opportunity to objectively test their structure prediction methods and delivers an independent assessment of the state-of-the-art in protein structure modeling to the research community and software users. CASP had a specific category for protein disorder prediction but it was terminated due to lack of suitable targets for the 11^{th} edition.

CAMEO (Continuous Automated Model Evaluation) is a protein structure model evaluation resource that is hinged on the PDB pre-release cycle. Each protein is classified into three categories based on the difficulty of the structure: hard, medium and easy. The

resource provides evaluations for Protein Structure and Contact Prediction on proteins yet to be released to the public on a weekly basis.

CPTAC Program the Clinical Proteomic Tumor Analysis Consortium (CPTAC) analyzes cancer biospecimens by mass spectrometry, characterizing and quantifying their constituent proteins, or proteome. Mass spectrometry enables highly specific identification of proteins and proteoforms, accurate relative quantitation of protein abundance in contrasting biospecimens, and the localization of post-translational protein modifications, such as phosphorylation, on a protein's sequence.

Machine learning

Machine learning is a subfield of computer science at the center of data mining, focusing on the development and design of algorithms capable of automatically recognize meaningful patterns and make predictions on data [11]. The quality of the patterns and information inferred from the data is highly dependent on the information content and nature of the data itself. Moreover, specific predictions made for a subset does not always imply it will be true in the larger data from which the sample was taken.

This section is a summary of machine learning methods commonly used in the training of disordered region predictors and some of their fundamental principles. They fall into the broad category of supervised learning, as a particular instance called classification algorithms. They map a given input vector or instance in one of the available categories or classes. This assignment is done based on the model learning from training instances being labeled with their actual classes (test cases).

Support Vector Machine

Support vector machine (SVMs) are a class of supervised learning algorithm and are an extension of simple linear models [12]. SVMs use linear models to implement nonlinear class boundaries.

Given a set of N training instances $(x_i, y_i)_{i=1}^{N}$, where x_i is an observation vector and y_i is a class label vector [-1,1], the goal of SVM is to learn a linear decision boundary or hyperplane that maximally separates instances of each class [-1,1]. The function describing the decision boundary is:

$$f(x) = w^T x - b$$

Where **x** is the vector describing the attributes of a new instance which class is to be predicted, **w** is the normal vector to the hyperplane and $\frac{b}{||w||}$ is the distance to the origin.

In the instance space, a hyperplane and supporting hyperplanes for a group of instances **x** are defined by:

$$w^T x - b = 0$$

$$w^T x - b = 1$$

$$w^T x - b = -1$$

The distance margin between the supporting hyperplanes is equal to $\frac{2}{||w||}$ and instances on the margin are called supported vectors.

Some data may not be completely separated by a linear hyperplane, SVMs should account for a degree of misclassification. Taking into account the slack the hyperplane equation for a set of points x can be written as

$$y_i(w^T x - b) \geq 1 - \xi_i \, , 1 \leq i \leq n$$

Using Lagrange multipliers and quadratic programming techniques to solve the minimization problem, the maximum margin hyperplane can be written as a function of the support vectors

$$x = \; b + \Sigma \alpha_i y_i a(i) \cdot a$$

Where $\mathbf{y_i}$ is the class value for **a(i)**, **b** and $\boldsymbol{\alpha_i}$ are numeric parameters and **a** represents a new instance to be evaluated.

The dot product is also called kernel function and allows mapping the original instance space to higher dimensional space (kernel trick). Selecting an appropriate kernel function allows finding non-linear decision boundaries to better fit the data.

Naïve Bayes

A naïve Bayes classifier (NBC) is based on Bayes' theorem which assumes that each feature in a dataset is independent of one another. This assumption of feature independence is the naivety of the classifier, but the NBC has been shown to often outperform several sophisticated classification algorithms [13]. One of the advantages of the NBC is that only a handful of training data is needed to estimate the parameters necessary for classification. Because of the feature independence assumption, only the variances of the features for each class are calculated.

Let $\boldsymbol{x} = [x_1, x_2, \ldots, x_k]^T$ be the feature vector. To label $\boldsymbol{x}$ in one of the M classes, $C_1, C_2, \ldots, C_M$, the posterior probability of a class, C_i, given the feature vector, $\boldsymbol{x}$, is: $P(C_i|\boldsymbol{x})$. The NBC assigns $\boldsymbol{x}$ to a class $\check{C}$ with the highest posterior probability among M classes

$$\check{\mathrm{C}} = argmax_i^M P(C_i|x)$$

The posterior probability of each class P(Ci|x) is calculated by

$$P(C_i|x) = \frac{P(x|C_i).P(C_i)}{P(x)}$$

Since the NBC has the assumption that all features are independent of one another, $P(\boldsymbol{x}|C_i)$ can be calculated by

$$P(x|C_i) = \prod_{j=1}^{K} P(x_j)|C_i$$

where k is the total number of features. Both $P(Ci)$, the prior probability of each class, and $P(x)$ can be ignored due to uniformity and constant assumption, respectively.

Thus, the scoring function can be rewritten as

$$\check{\mathrm{C}} = argmax_i \prod_{j=1}^{K} P(x_j)|C_i$$

Random Forest

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual tree [14].

Each tree gives a classification, and it is said that the tree “votes" for that class. The forest chooses the classification having the most votes, over all the trees in the forest. Each tree is created as follows [14]:

- If the number of cases in the training set is *N*, sample *N* cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
- If there are *M* input variables, a number $m<<M$ is specified such that at each node, *m* variables are selected at random out of the *M* and the best split on these *m* is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

The original paper on random forests [15] shows that the forest error rate depends on two things [14]:

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.
- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Reducing *m* reduces both the correlation and the strength. When increased, it increases both. Somewhere in between is an optimal range of *m*, which is usually quite wide. This is the only adjustable parameter to which random forests is somewhat sensitive.

Artificial Neural Networks

Neuron models

The building blocks of a neural network and part of every network architecture are the neurons. Neurons are often referred to as units and are mathematically equivalent to activation functions. Two types of neuron models are used in current state-of-the-art implementations of deep convolutional neural networks [16]: the rectified linear unit (ReLU) and the softmax unit. We also review two other activation functions (binary threshold neuron and hyperbolic tangent neuron) to put the benefits of ReLU and softmax into context.

Binary Threshold Neuron

$$y = \begin{cases} 1 \; if \; M \leq b + \sum_{i=1}^{k} x_i \cdot w_i \, , where \; M \; is \; a \; threshold \; parameter \\ 0 \; otherwise \end{cases}$$

This activation function behaves very similarly to a biological neuron model. The output *y* takes a binary decision, either to activate or not. If the input parameters met the threshold for activation, a signal is sent as output. Despite being closer to a biological neuron than other activation functions, it is not differentiable. This makes it impossible to use local greedy optimization learning algorithms such as gradient descent, which computes the derivatives of the activation function in order to reduce the error.

Logistic Sigmoid Neuron

$$y = \frac{1}{1 + \exp(-z)}, where\ z = \sum_{i=1}^{k} x_i \cdot w_i$$

Like the binary threshold neuron, the output domain of this neuron is bounded by 0 and 1. The main advantages of using this activation function are that it is fully differentiable and it is non-linear, which helps to increase performance [17]. The main disadvantage of this model is that it is computationally expensive to compute.

Rectified Linear Neuron (ReLU)

$$y = \max\{0, b + \sum_{i=1}^{k} x_i \cdot w_i\}$$

The rectified linear neuron is not fully differentiable or bounded above. It only has two values for slopes, so its derivative with respect to x_i can only be 0 or w_i. Despite its simpler appearance when compared to the other activation models, it is very efficient to compute in terms of value and partial derivatives. This simplicity enables much larger network implementations [18] and has been demonstrated to enable better training of deeper networks [19]. ReLU introduces a non-linearity with its angular and is currently the most popular activation function used in deep neural networks [20].

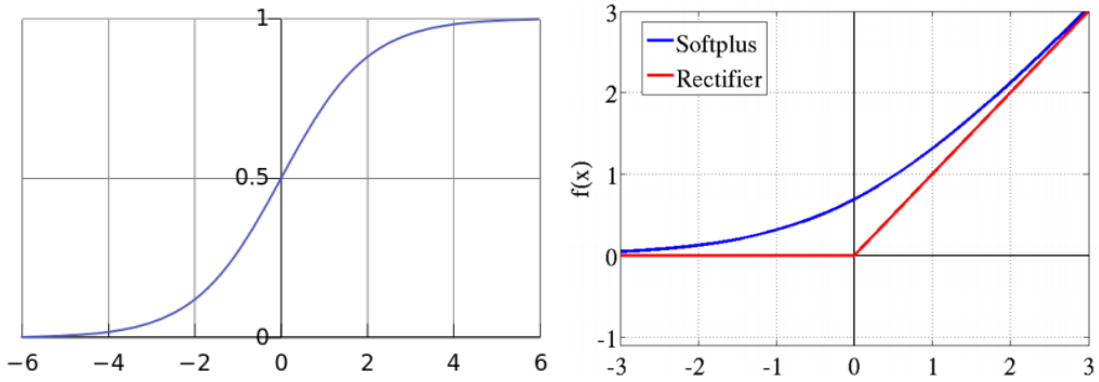

Figure 2.1 Logistic sigmoid (left) and rectified linear (right) activation functions

Softmax Neuron

$$y_j = \frac{\exp(z_j)}{\sum_{i=1}^{k} \exp(z_i)}, where\ z_j = \sum_{i=1}^{k} x_i \cdot w_i + b$$

The equation of a SoftMax neuron needs to be understood in the context of a layer of k such neurons within a neural network. The notation y_j corresponds to the output of the j^{th} SoftMax neuron, and $w_{i,j}$ corresponds to the weight of x_i as in input for the j^{th} SoftMax neuron. SoftMax function takes as input the vector of all the SoftMax neurons $z_1, z_2, ..., z_k$, and normalizes it into a probability distribution consisting of K probabilities [16]. After applying SoftMax, each component will be in the interval (0,1), and the components will add up to 1 so that they can be interpreted as probabilities. This makes the SoftMax layer ideal for classification; neuron j can be made to represent the probability that the input is an instance of class j. One important point about the SoftMax function is that its derivative is quick to compute, and it is given by $\frac{dy}{dz} = \frac{y}{1-y}$.

Feed-forward Architecture

A feed-forward neural network is a form of artificial neural network where connections between nodes form a directed acyclic graph. A node represents an activation function f, an edge is the composition of two activation functions $f \circ g$, and an edge weight is a parameter of f. This defines an input and output layer and enables the representation of a mathematical function [16]. The feed-forward architecture means that data travels in one direction across the network, from the input nodes, through the hidden nodes and the output nodes.

Shallow Feed-Forward Neural Networks The most straightforward feed-forward neural network, called the perceptron, consists of a single layer of output nodes. The inputs are fed directly to the outputs via a series of weights. The first neural networks introduced in the 1960s [17], were of this type and used a binary threshold activation function. This architecture severely reduces the function space, and the single layer perceptrons are only capable of learning linearly separable patterns. This was generalized and proved [21] and lead to a move away from artificial neural networks for machine learning by the academic community during the late 1970s. However, a single-layer neural network can compute other activation functions instead of a step function. When the logistic sigmoid function is used as activation, the single-layer network is identical to the logistic regression model, widely used in statistical modeling.

Deep Feed-Forward Neural Networks This class of network is also called Multilayer Perceptron (MLP) and consists of multiple layers of computational units, interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons

of the next layer and can be represented by a directed acyclic graph made up of more than two layers. The network units usually apply the sigmoid function as the activation function. The hidden layers, which roles are not set from the start, are learned throughout the training. When successfully trained, each neuron becomes a feature detector [17].

It can be mathematically proven that MLPs are universal approximators [22]. The universal approximation theorem for neural networks [23] states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds for a wide range of activation functions and solves the initial limitation faced by perceptrons.

Backpropagation

In the context of multi-layer feed-forward neural networks and supervised training, backpropagation is an algorithm used to train a neural network. The objective is to learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to output [24]. Gradient descent relies on the partial derivatives of the error (or cost) function with respect to each parameter of the network. The backpropagation algorithm is an implementation of gradient descent which efficiently computes these values.

Compute Error-Weight Partial Derivatives

Let t be the target output and let $y = (y_1, y_2, ..., y_P)$ be the actual value of the output layer on a training case. The error is given by

$$E = C(t - y)$$

Where C is the chosen cost (or loss) function. The error-weight partial derivatives are calculated using the chain rule twice and given by:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial net} \cdot \frac{\partial net}{\partial w_{i,j}}$$

The derivative $\frac{\partial f}{\partial x}$ is numerically obtained by perturbing x and taking the change in $f(x)$. The advantage of this formula is that instead of individually perturbing each weight w_{ij}, only the unit outputs y_i are perturbed. In a neural network with k fully connected layers and n units per layer, this amounts to $\Theta(k \cdot n)$ unit perturbations instead of $\Theta(k \cdot n^2)$ weight perturbations [16]. This means that backpropagation scales linearly with the number of neurons making it very efficient to calculate even on large networks.

Weight Values Update with Gradient Descent

The learning rule is given by:

$$w_{i,t+1} = w_{i,t+1} + \tau \cdot \frac{\partial E}{\partial w_{i,t}}$$

Being τ the learning rate

Weight values move in the direction that will reduce the error faster, which is the direction of steepest descent on the error surface (given by the partial derivative). Gradient descent converges ($w_{i,t+1}$ equals $w_{i,t+1}$) when the partial derivative reaches zero. This corresponds to a local minimum on the error surface. Figure 2.2 shows two training sessions where the only difference is the initialization of the two weights, and the minima attained in each case are different [16]. Gradient descent with backpropagation does not guarantee to find the global minimum of the error function, but only a local minimum. This is caused by the non-convexity of the error functions in neural networks.

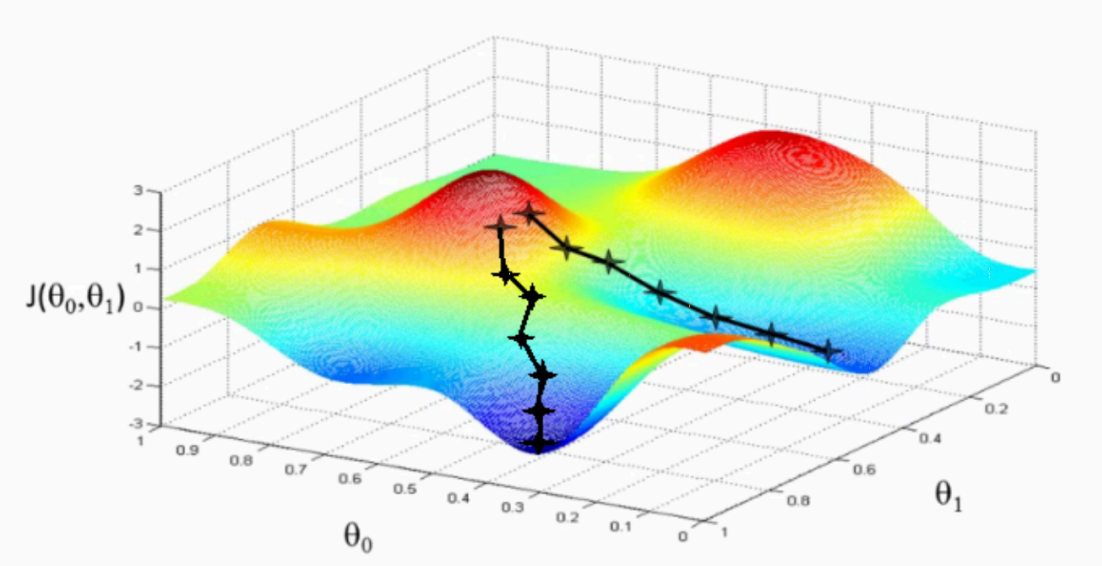

Figure 2.2 Error surface illustrating local minima issue

Stochastic Gradient Descent

Stochastic gradient descent (SGD), is an incremental implementation of the gradient descent algorithm. To perform the weight updates, the partial derivatives are obtained by averaging the weights over a randomly selected subset of the training set,

instead of the full set. These randomly selected groups are referred to as mini batches and the algorithm implementation as mini batch training [17].

Overfit

Highly expressive and parameterizable models such as deep feed-forward neural networks are prone to overfit. This means that the model may work perfectly over the training set but will fail to generalize and perform poorly on unseen data. A good illustration of the overfitting problem is shown in Figure 2.3 [16]. This regression example tries to fit a polynomial curve of different orders (0, 1, 3, 9) to a set of points sampled uniformly with noise from a curve.

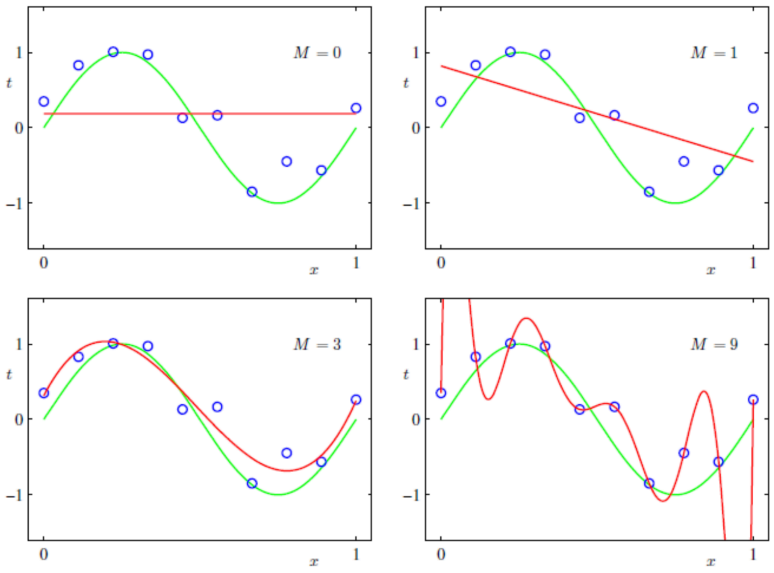

Figure 2.3 Overfit as polynomial order increases

Cross Validation and Early stopping

Cross validation is a method for estimating the generalization accuracy of a supervised learning algorithm. It consists in separating the labeled dataset into a training set, a validation set, and a test set. The partial derivatives are computed from the error over the training set, but the function that is retained at the end of training is the one that minimizes the error over the validation set [16]. The test set is used to measure the performance (by a chosen metric). The separation between the test set and the validation set is to obtain a stochastically impartial measure of performance.

Early stopping is a form of cross validation that aims to prevent overfitting by allowing the training algorithm to continue while the validation error decreases. Training stops as soon as the error in the validation set increases, even if the training error is still decreasing.

Dropout

Dropout is a regularization technique aimed to reduce overfitting in deep neural networks by preventing complex co-adaptations on the training data [16]. Dropout randomly turns off a fixed proportion $k \in (0, 1)$ of neurons at every training iteration, but uses the entire network (with weights scaled down by k) at test time. This significantly reduces overfitting and gives major improvements over other regularization methods [25]. At test time, the average of all models is used, and it can be seen as a powerful ensemble method.

This technique has been shown to improve performance on the MNIST image classification benchmark task [26], though this increase is only significant when compared to models where neither data augmentation, convolutional layers or unsupervised pre-training are used.

Deep Convolutional Neural Networks

Convolutional neural networks (CNNs) are deep feed-forward neural networks containing at least one convolutional layer. As a general rule, deep neural networks are difficult and hard to train, but convolutional neural networks (using ReLU activation functions) are an exception to this [27]. They were inspired by the visual system's structure, obtaining and maintaining state-of-the-art performances on several computer vision tasks [20]. A convolutional layer takes advantage of the spatial structure of the inputs which makes them different from a traditional fully connected layer. It imposes specific operations on the data before and after the data is fed to the activation functions [16]. Figure 2.4 shows the standard structure of a CNN, consisting of alternating convolutional layers and pooling layers (often each pooling layer is placed after a convolutional layer) [27]. The last layers are a small number of fully-connected layers, and the final layer is a SoftMax classifier which makes the final class prediction.

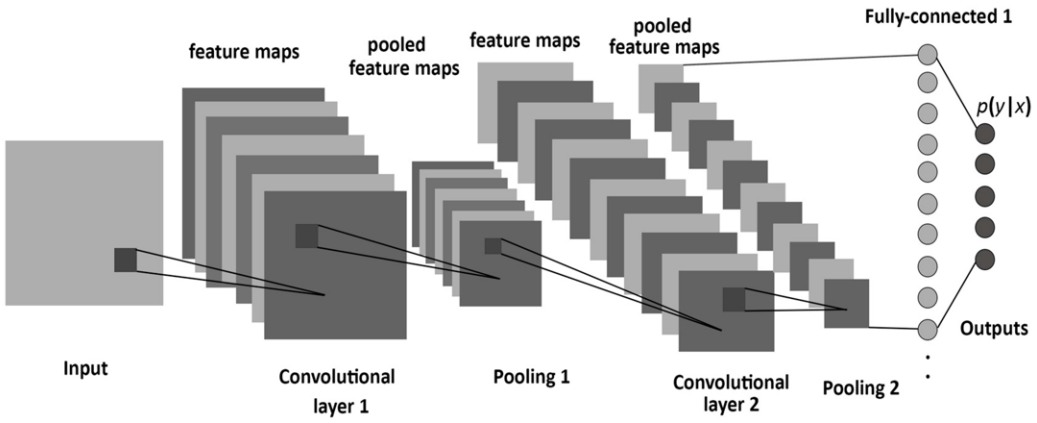

Figure 2.4 Two convolutional layer neural network

Sliding kernel

The convolutional layer is comprised of a set of learnable kernels or filters tasked to extract local features from the input. Each kernel is used to calculate a feature map. The units of the feature maps can only connect to a small region of the input, called the receptive field [27]. A new feature map is typically generated by sliding a filter over the input. The sliding kernel is defined as a $k \times k$ matrix W that is applied to $k \times k$ windows X of the object by performing the matrix dot product $\sum_{i=0}^{k-1} \sum_{j=1}^{k} x_{ij} \cdot w_{ij}$.

This is equivalent to the vector product component of the generic neural network inputs combination $z = b + \sum_{i=0}^{n-1} x_i \cdot w_i$. The pixel matrix representation of the entire image is flattened into the vector x, the weights of the kernel are flattened into a portion of the weight vector w, and all weights corresponding to a pixel that is not part of the window are set to zero [16].

All units share the same filters among each feature map, reducing the number of parameters and increasing the ability to detect the same feature, regardless of its location in the vector input [28].

Class Imbalance

Class imbalance is defined as the situation where the sample distribution of classes is significantly non-uniform, some classes have a significantly higher number of examples in the training set than other classes [16] [29]. The protein disorder prediction problem is an example where the training set present these characteristics, being the disordered class under-represented when compared to the ordered class. It is established that class imbalance can have a significant detrimental effect on the training of traditional classifiers and that it also affects deep convolutional neural networks [29]. Class imbalance can be addressed at the data or classifier level. Undersampling and oversampling are two data level methods which have been shown to perform better in the context of CNNs [29]. Oversampling, on its most basic implementation, replicates randomly selected samples from minority classes until achieving a balanced data set. Undersampling achieves the same results by randomly removing members of the majority class.

Initial problem exploration

In order to explore the different machine learning methods, parameters and how they affect the prediction capabilities of our models, we developed a highly parameterizable model builder trained on a smaller dataset (DisProt). This simplified version, allowed us to explore different strategies for feature extraction and generation, sliding window construction, reduced amino acid alphabets, minimum data set size and machine learning methods. The findings and software artifacts from this exploratory work were used in the implementation of the final algorithms.

Data Retrieval and datasets generation

For our exploratory models, disordered and ordered regions were obtained from the Database of Protein Disorder (DisProt) version 6.02. DisProt dataset is a set of proteins containing experimentally determined IDRs using a variety of indirect biochemical methods. A total of 694 sequences were extracted from the online FASTA file, each of which at least 27 residues long. A total of 347,037 residues were analyzed (265,599 ordered and 81,438 disordered) in order to construct the initial complete set. A balance dataset was created by randomly drawing the same number of ordered and disordered residues from the complete set. Using this technique, a maximum set size of 162,876 residues is possible but for practical reasons, we limited the maximum size to 80,000. Datasets from the Critical Assessment of protein Structure Prediction experiments (CASP8, CASP9, and CASP10) were used for initial benchmarking and parameter selection (containing 122, 117 and 94 sequences respectively). The balanced

dataset was randomly divided into disProt_TRAIN and disProt_TEST, each set containing 70% and 30% of the residues respectively.

Figure 2.5 and Figure 2.6 show the distribution of the length of the disordered regions in the source datasets. The four datasets look very similar in terms of the distribution of IDRs lengths, with the exception of DisProt data, which contains longer IDRs.

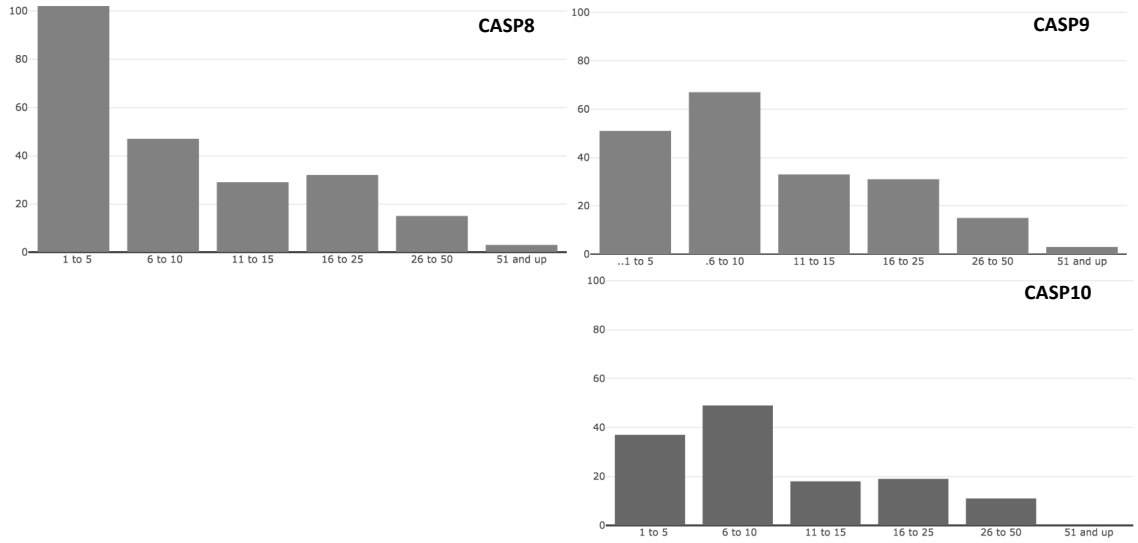

Figure 2.5 Distribution of length of disordered regions in the CASP8, CASP9 and CASP10 datasets

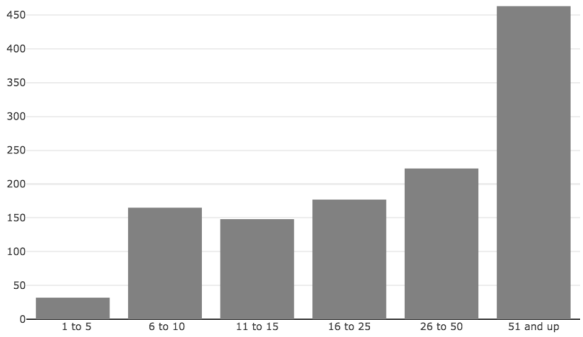

Figure 2.6 Distribution of length of disordered regions in DISPROT datasets.

n-gram frequencies

n-gram frequencies are used as the primary target feature on all of our exploratory models. We calculated frequencies for each of the 347,037 residues in the DisProt database using a sliding window approach. The frequency of a certain *n*-gram in a given window is the number of times the *n*-gram appears in the window out of the total number of *n*-grams in the window. *n*-gram frequencies were calculated as follows:

$$f_{\text{n-gram}} = \frac{i}{m}$$

$$m = w - n + 1$$

i represents the number of times the particular *n*-gram occurs in the window, *m* is the total number of *n*-grams in the window, and *w* is the window size.

n-gram frequencies were then normalized in order to prevent the frequency of a feature from skewing the decision process. The following shows how the 3-gram “BBB” was normalized, as an example:

$$q_{BBB} = \frac{f_{BBB}}{f_B * f_B * f_B}$$

As is shown, an *n*-gram frequency was normalized by dividing the frequency by the product of the frequencies of each of its constituent (reduced) residues. The residue frequencies were calculated from all the sequences in the initial training data set. Each residue possessed a 3^n-dimensional feature vector with the normalized frequencies of all possible *n*-grams. The disordered and ordered residues were labeled into two different classes for machine learning.

Alphabet Reduction

The 20-letter amino acid alphabet had to be reduced to a 3-letter alphabet in order to simplify and quicken the machine learning process, as the model would have to analyze fewer n-grams during training and testing. Another problem with using the original 20-letter alphabet is the total number of features (n-grams) would exceed the number of sequences. By reducing the number of letters in the alphabet to three, and thus decreasing the number of possible n-grams, the number of sequences would be almost 15 times the total number of features, so it would be highly unlikely for the model to overfit.

We explore five reduced amino acid alphabets taken from outside sources (Table 2.1). Residues can be clustered based on various properties, including chemical and genetic properties. A critical feature of these alphabets is that they cluster residues in ways that prevent the loss of key biochemical information.

Table 2.1 The five initial reduced alphabets and their sources. Each letter contains a cluster of amino acid residues (one-letter abbreviations). The residue clusters were denoted by the letters "B", "J", and "U" in this study.

Reduced Alphabet	Letter 1 (B)	Letter 2 (J)	Letter 3 (U)	Reference
1	CFILMVWY	AGHPRT	DEKNQS	[30]
2	AFGILMPV	DEKR	CHNQSTWY	[31]
3	CFILMVWY	AGPST	DEHKNQR	[32]
4	DHIMNVY	EFKLQ	ACGPRSTW	[33]
5	ACGILMPSTV	EKRDNQH	FYW	[34]

Feature extraction and process Automation

A collection of Java/Groovy scripts were written for data parsing and extraction, normalization, alphabet reduction, *n*-gram frequency calculations and generation of receiver operating characteristic (ROC) curves. The scripts were streamlined in such a way that automated the entire process from parsing the sequence data in the source databases to creating comma-separated values (CSV) files with the feature vectors for each sequence. Weka libraries [35] were used to calculate the ROC Area for each one of the sets.

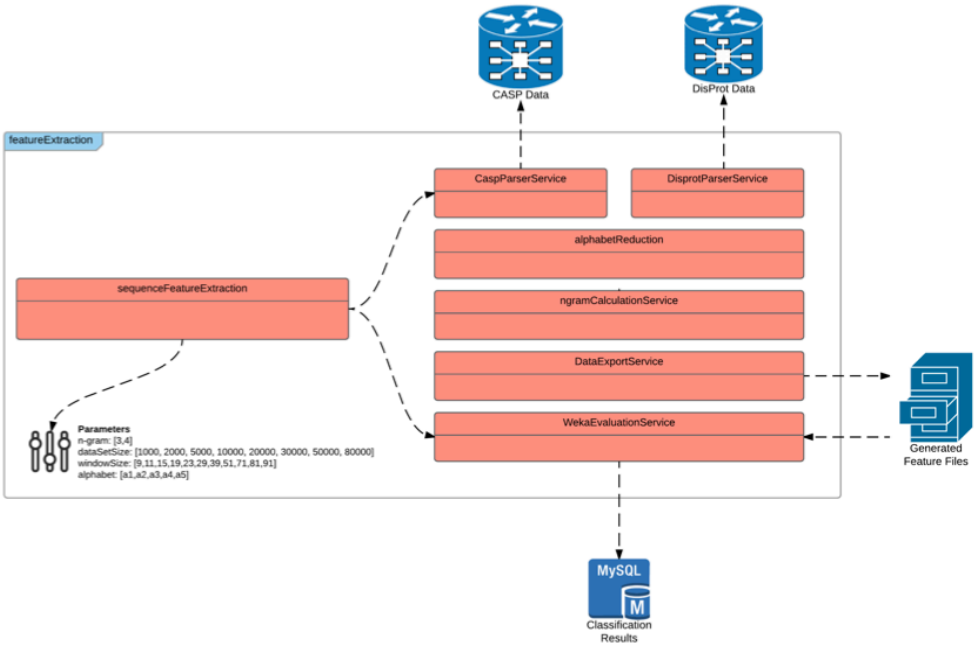

Figure 2.7 Data process and script diagram

Parameter selection and evaluation

In order to construct an optimal model, sets of different parameters and values were initially selected based on empirical data and literature [36][37].

Table 2.2 Tested parameters and values

Parameter	Tested Values
Training Dataset Size	1000, 2000, 5000, 10000, 20000, 30000, 50000, 80000
Sliding Window Size	9,11,15,19,23,29,39,51,71,81,91
Alphabet Reduction Mapping	1,2,3,4,5
n-gram	3,4
Prediction algorithm	Logistic Regression, Random Forest

A total of 1760 models were created using DisProt database as the base dataset. Each model was used to test its predictive value against each CASP dataset and performance metrics were stored in a local database. The main metric used for comparison was an average of the AUC for the three CASP datasets (AUC_AVG). Datasets with less than 10,000 observations showed a lot of variabilities and were filtered out from further analysis. Figure 2.8 shows an optimal dataset size between 50,000 and 80,000 observations.

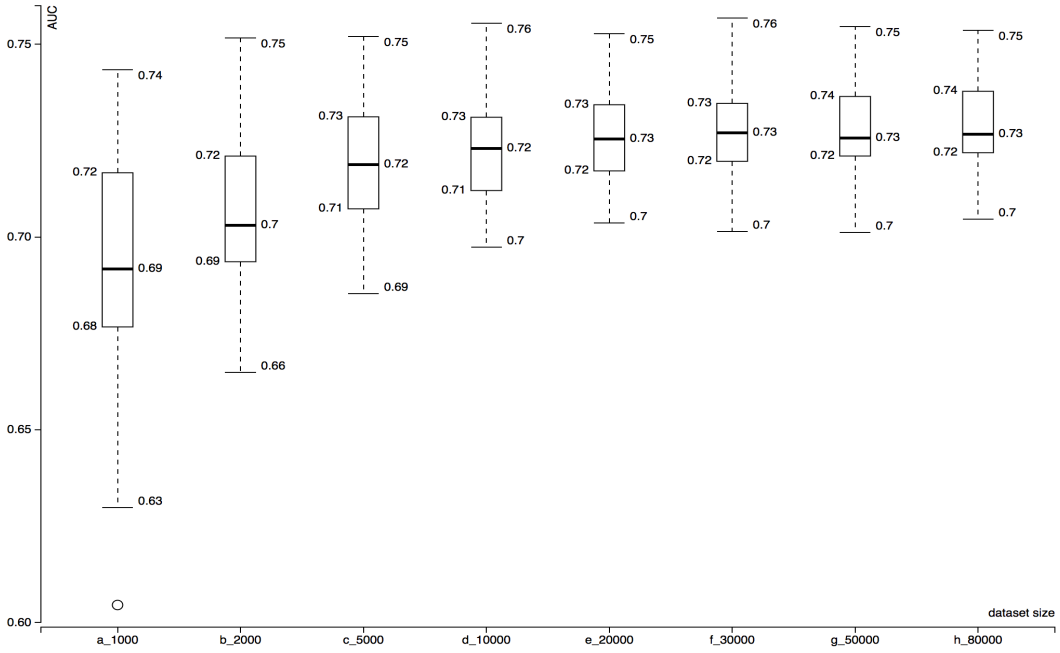

Figure 2.8 Dataset size performance, learning curve

Machine learning method and n-gram selection accounted for most of the improvement in performance. Figure 2.9 shows Logistic Regression and 3-grams having a better performance across tested models.

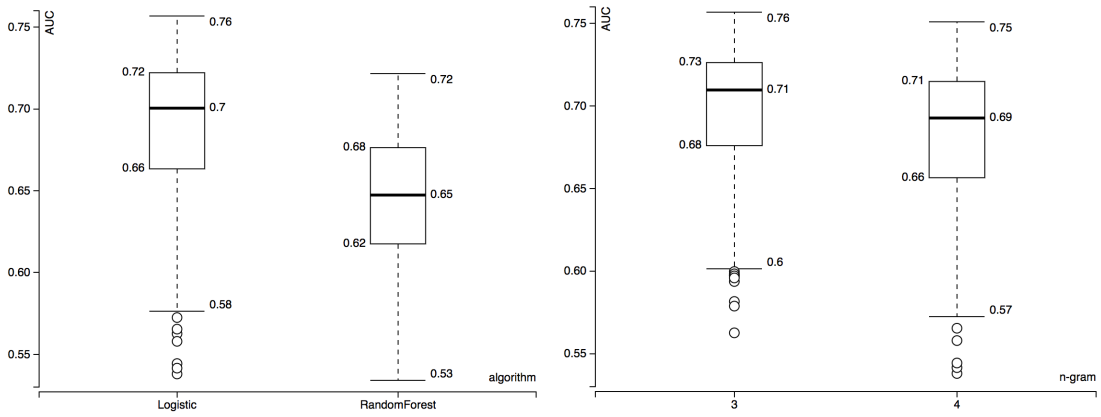

Figure 2.9 Algorithm and n-gram performance (dataset size greater than 10,000)

There is evidence that window sizes of at least 19 residues are necessary to capture secondary structure features [36]; this may not be necessarily true for disorder prediction and n-gram. We tested several window sizes ranging 9-91 residues long; Figure 2.10 shows a clear improvement in performance when larger windows are used (greater than 71 residues).

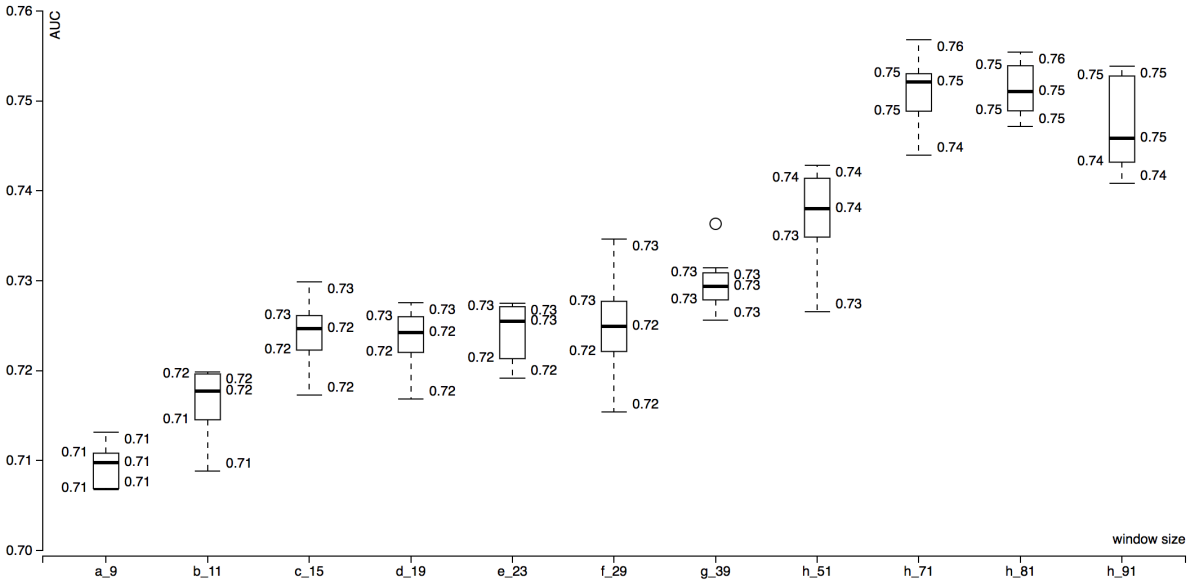

Figure 2.10 Sliding window size performance (3-gram, Logistic Regression)

Finally, there seems to be significant improvement by using mapping Alphabet 1 and 3, as shown in Figure 2.11.

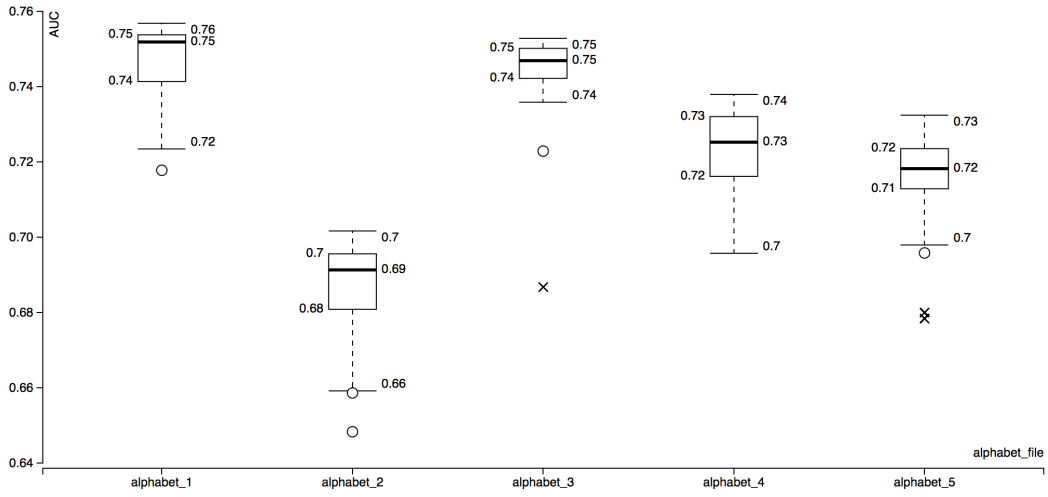

Figure 2.11 Mapping alphabet performance (3-gram, Logistic Regression, Window size greater 71)

Prediction Algorithms Development

Based on the findings from our exploration phase, we developed two slightly different algorithms to predict disordered protein regions. For the full algorithm development, we used a more extensive but less accurate training and test set based on PDB sequences. This extended set allowed us to experiment with more sophisticated machine learning methods that required a large number of instances during training. In particular, we expand our tests to include Random Forest and Convolutional Neural Network architectures. We also extended our reduced amino acid alphabet set to include an additional alphabet based on information content [38]. Experimentation with various window sizes leads us to select a somewhat bigger window (W=101) in both cases.

Regarding feature generation, we adopted two different approaches depending on the machine learning classification method implemented. For Random Forest based method (Chapter 4) we use 3-grams frequencies calculated over reduced amino acid alphabet sequences. Our Convolutional Neural Network models are trained directly over the 3-letter reduced amino acid sequences without any further feature extraction (Chapter 5).

3. SPECIFIC AIMS

Problem statement

The objective of this dissertation is to develop a novel protein disorder prediction method based on n-grams frequencies and machine learning principles. The underlying hypotheses behind this work are:

- N-gram frequencies produce a signal that we will be able to detect through machine learning methods and used to predict order/disordered regions accurately.
- Reducing the amino acid space from a 20-letter alphabet to a 3-letter alphabet will still make the signal detectable, despite the reduction in information content.

To assess the efficiency of this method, it will be benchmarked against existing tools using CASP targets and a selected number of PDB structures.

The specific aims that will be addressed are:

1. Algorithm development and implementation
    a. Develop an n-gram based approach to classify individual residues in protein sequences into one of the following classes (disorder, order).
    b. Calculate a position-dependent disorder score for each individual residue in the analyzed sequence.

c. Benchmark the performance of the method against existing models using proteins from the CASP experiments, as well as a large subset of proteins extracted from PDB database.

2. Explore advanced machine learning methods to improve on the accuracy and prediction capabilities of our original n-gram based algorithm.

3. Develop a parameterized online resource for the prediction of disordered residues of a protein chain from its amino acid sequence-based on our best performant method.

4. IDENTIFICATION AND PREDICTION OF INTRINSICALLY DISORDERED REGIONS IN PROTEINS USING N-GRAMS

In this chapter, we present the paper submitted to the 8th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM BCB) and published as part of the Conference proceedings [39].

Abstract

Intrinsically disordered proteins (IDPs) play an important role in many biological processes and are closely related to human diseases [40]. They also have the potential to serve as targets for drug discovery, especially in disordered binding regions [41], [42]. Accurate prediction of IDPs is challenging, most methods rely on sequence profiles to improve accuracy making them computationally expensive.

This paper describes a method based on n-gram frequencies using reduced amino acid alphabets, which tries to overcome this challenge by utilizing only sequence information.

Our results show that the described IDP prediction approach performs at the same level as some of the other state-of-the-art ab initio methods. However, the simplicity of n-grams allows constructing decision trees which can provide important insights into common patterns and properties associated with disordered regions.

Introduction

It has been the dominant paradigm in structural biology that a well-defined structure determines protein function. Intrinsically disordered proteins (IDPs), which lack a stable three-dimensional structure under normal physiological conditions, are a challenge to the structure-to-function paradigm [43]. Disorder exists in up to half of the amino acids in eukaryotic proteins, and disordered regions are involved in numerous biological functions [4], as a result of their flexibility. Since amino acid sequence is known to determine protein structure, sequence information can be used to identify disordered regions. Protein disorder is involved in the development of many diseases, and identifying disordered regions can help us understand how to use them as potential drug targets [41], [44], [45]. The identified regions can also be used to better understand the pathways of protein folding and provide insights into protein function.

In this study, machine-learning classification techniques will be applied to distinguish between disordered and ordered residues within a sequence-based on n-gram frequencies content. Initial results show that using n-gram frequencies is an accurate and computationally inexpensive method to predict disordered regions, based on raw sequence data. This method of prediction can subsequently aid in the development of next-generation treatments for a variety of biomedical applications.

N-grams are a commonly used technique in computational linguistics, probability, text categorization, and biology [46]. In this study, an n-gram has been denoted as a contiguous string of n amino acid residues in a protein sequence. The primary structure, or the amino acid sequence, of a protein determines the protein's three-dimensional

structure. This implies that disorder, or lack of stable structure, can also be encoded in the sequence. A sequence can be decomposed into a list of overlapping n-grams. N-gram patterns have been previously used to show evolutionary relationships between protein sequences and to predict protein secondary structure [46]. We also successfully used n-gram analysis to predict functional properties of proteins, including drug resistance [47].

Current predictors of protein disorder use multiple sequence alignments, secondary structure analyses, PSI-BLAST sequence profiles, or distinctive residue compositions [4]. These predictors require analyzing and comparing entire sequences, taking relatively longer compared to n-grams, which decompose sequences into smaller pieces, each of which can be readily analyzed quantitatively. Computational techniques used for predictions are generally "black-box" models such as Neural Networks and Support Vector Machines. In addition, the features that these models use are not fully understood. To help with feature selection, n-gram frequency data can also be used to train decision trees, which can provide more insight into how the training data is actually used to create the decision-making process.

Experimental and Computational Methods

Current prediction methods

Over 60 protein disorder prediction servers are currently available, although not all publicly. The methods can be classified in one of the following categories [5]: (i) Ab initio or sequence-based, (ii) clustering, (iii) template based and (iv) meta or consensus. The methods that can't easily be assigned to one of the above categories fall into the

hybrid classification. Below is a brief description of the basis of each of the type of method.

Ab initio methods. They rely almost exclusively on amino acid sequence information to make a prediction. Features extracted from the primary sequence, alignment profiles or scoring matrices are used as input for statistical models to make predictions of disorder regions. This class of methods was widely used in the CASP8 [48] and CASP9 [49] experiments. A few examples of it are Disopred [50], ESpritz [6] and PreDisorder [7].

Clustering methods. This approach generates tertiary structure models from the primary sequence. It then superimposes the different models onto each other under the assumption that positions in ordered regions will be conserved across the models [8]. Residues in disordered regions are likely to vary. Since this approach doesn't rely on a training set, it is less likely to be biased by the training data.

Template based methods. Similar to clustering, it predicts disordered regions of proteins by aligning the input sequence to homologous proteins with known structure. Homologous proteins are found by database search or by fold recognition methods. This method is frequently used in combination with other prediction approaches and they generally fall into the hybrid category.

Meta methods. They combine the output of several disordered predictors into a single average, which tends to have a moderate increase in accuracy. This is one of the most popular methods and is used by metaPrDOS [9] and GSmetaDisorder [51] among others.

Hybrid methods. These are the methods combining two or more of the previous approaches in order to improve prediction accuracy. PrDOS [10] is one common example, combining ab initio and template based homologous alignment to output a prediction.

Alphabet Reduction Evaluation and Selection

The 20-letter amino acid alphabet was reduced to a 3-letter alphabet in order to simplify and quicken the machine learning process, as the model would have to analyze fewer n-grams during training and testing. One problem with using the original 20-letter alphabet is that the total number of features (n-grams) would exceed the number of sequences in the training set. By reducing the number of letters in the alphabet to three, and thus decreasing the number of possible n-grams, the number of sequences would be almost 250 times the total number of features. Therefore, it would be highly unlikely for the model to overfit. The reduction of the amino acid space from a 20-letter alphabet to a 3-letter alphabet carries a reduction in the information content. One key hypothesis in our method is that this reduction won't affect the n-gram signal or the method predictive capability.

The reduced alphabets were taken from outside sources (Table 4.1). Residues can be clustered based on various properties, including chemical and genetic properties. Reduced alphabets cluster residues in ways that prevent the loss of key biochemical information.

Table 4.1 The six reduced alphabets and their sources. Each letter contains a cluster of amino acid residues (one-letter abbreviations). The residue clusters were denoted by the letters "B", "J", and "U" in this study

Alphabet/ref	Letter 1 (B)	Letter 2 (J)	Letter 3 (U)
a1 [30]	CFILMVWY	AGHPRT	DEKNQS
a2 [31]	AFGILMPV	DEKR	CHNQSTWY
a3 [32]	CFILMVWY	AGPST	DEHKNQR
a4 [33]	DHIMNVY	EFKLQ	ACGPRSTW
a5 [34]	ACGILMPSTV	EKRDNQH	FYW
a6 [38]	CILMVFWY	AGHST	DEKNPQR

Disorder prediction by n-gram frequencies

Feature Generation. For a given protein sequence of length L and a reduce alphabet of A letters, we extract L feature vectors of size T where T is the total numbers of possible n-grams ($T=A^n$). In order to calculate the n-gram feature vector for a particular position, we use a centered sliding window of size W.

Each position of the N-size feature vector contains the frequency of a particular n-gram for the current residue centered window. N-gram frequencies are then normalized to prevent the frequency of a feature from skewing the decision process. The following shows how the 3-gram BJU was normalized, as an example:

$$q_{BJU} = \frac{f_{BJU}}{f_B . f_J . f_U} \quad (1)$$

Where q is the normalized frequency and f is the frequency of the 3-gram of the reduce residue within the protein sequence. Finally, each feature vector is assigned a label (0=order, 1=disorder) based on the residue annotation and is used as input to a classification algorithm. Random Forest [52] was chosen based on its known performance and ability to provide insight into the mechanism of classification. It follows

a decision tree model, which creates different nodes on a tree based on the provided attributes in the training set [52], [53]. The algorithm constructs an ensemble of decision trees, which reduces overfitting, a common issue when only a single decision tree is used.

Parameter Selection. In order to select the reduced alphabet, window size and n-gram that maximize the model predictive value, we construct a simplified classification model for each possible combination of a select group of values. We test each model performance using a dataset containing a combination of CASP8, CASP9 and CASP10 targets [48], [49], [54]. The AUC for each model is calculated and used as main metric to compare performance. Table 4.2 shows a summary of the parameters tested and Figure 4.1 shows the results for a selected group of parameters. It was found that 3-grams, alphabets a1, a3 and a6 and W >71 have better overall performance and increase model predictive value. The final implementation of the method uses 3-gram, alphabet 6 and W=101. The window size was derived after experimentation within the final model.

Control Experiments. In order to ensure that the observed AUC values were different from those in a random feature space, controls were created by shuffling the residue labels. Labels were shuffled such that half of the disordered residues and half of the ordered residues were labeled into the incorrect class. The shuffled datasets were used to train the same machine learning algorithm.

Table 4.2 Model construction parameters

Parameter	Tested values
Sliding Window Size (W)	9,11,15,19,23,29,39,51,71,81
n-gram (n)	3,4
Reduce Alphabet (A)	a1,a2,a3,a4,a5,a6
Classification algorithm	Naïve Bayes

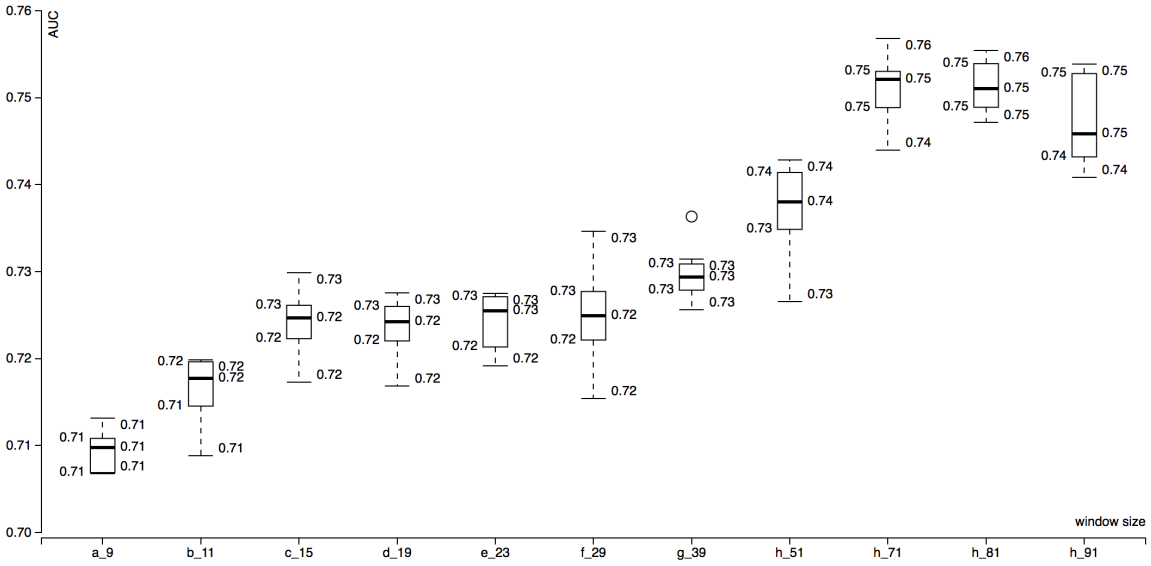

Figure 4.1 Sliding window size models performance (3-gram, all alphabets)

Results and Discussion

Datasets

Publicly available datasets are used to train and evaluate our method. High resolution X-ray crystal structures from the Protein Data Bank (PDB) [55] are used to construct the training and test data sets while CASP10 [54] and CAMEO [56] (http://www.cameo3d.org) are used for further validation. We use Pisces protein sequence

culling server (http://dunbrack.fccc.edu) [57] to extract sequences from PDB to filter for high resolution and reduce redundancy. Parameters selected for culling are: (i) proteins sharing >25% sequence identity (ii) maximum resolution of 1.8 Angstroms (iii) R-value >0.30. In total, 7,119 proteins are retrieved from PDB with an average length of 349 residues. We randomly divide the initial set into two distinct subsets: PDB70 containing 70% of the sequences for training and PDB30 with the remaining 30% for testing.

There is no common agreement on how to define disorder residues from PDB files [58]. For the purpose of this work, we consider a residue to be in a disorder position if it appears in the sequence records but its coordinates are missing from the electron density map. This is not a perfect definition since there are other reasons why a residue can have missing coordinates (i.e.: crystallization artifacts). However, it allows us to use a large number of proteins from PDB without further experimental validation.

CASP10 is the latest dataset available from the series experiments, which released specific targets for protein disorder prediction. The 94 available targets are used for validation and as an independent benchmark set. To prevent any redundancy between training and validation sets, we use BLASTClust [59] to filter for sequence identity between the PDB70 (training set) and CASP10 (validation set). Finally, to further assess our method, we tested against CAMEO 6 Months targets released from August 26, 2016 to February 18th, 2017 (504 targets, categorized in 3 groups).

Program benchmarking

To benchmark our method, we selected previous CASP participating programs, in particular, a sequence-only method (Espritz) and a sequence profile based method (Disopred3). Espritz is an ensemble of sequence-only and multiple sequence alignments disorder prediction methods. The sequence-only method has three different versions, depending on the initial set used for training (X-ray, NMR, Disprot). We used X-ray and NMR versions. The X-ray trained version is the one that performs best among the three. Disopred3 runs a PSI-BLAST search for each of the residues in a 15-residue window. The profile is then used as input to a neural network classifier which outputs a probability estimate of the residue being disordered.

Metrics and Evaluation Criteria

Disorder data is characterized by high class imbalance, disordered residues account for less than 5% of the data in the PDB set (training and test). Since disordered residues are relatively rare compared to ordered ones, they are harder to predict. Performance metrics should account for this imbalance and reward correct prediction of disordered residues higher than the correct prediction of ordered ones [49]. We selected a subset of the metrics commonly used for the assessment of disorder data [37], [60], [61] that take into account the nature of the imbalanced data: (i) specificity (ii) sensitivity (iii) balanced accuracy (iv) Matthews correlation coefficient and (v) AUC.

Binary metrics

$$\text{Specifity} = \frac{TN}{TN + FP} \tag{2}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{Acc} = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} \tag{4}$$

$$\text{MCC} = \frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

True positives (TP) and true negatives (TN) are the numbers of correctly predicted disordered and ordered residues. False positives (FP) and false negatives (FN) are the numbers of incorrectly predicted disordered and ordered residues.

Statistical metrics. The ROC curve is a plot that compares the true positive rate against the false positive rate under various threshold values for a binary classifier.

$$\text{AUC} = \text{Area under the ROC curve plotting 1-specificify and sensitivity} \tag{6}$$

To confirm the robustness of our classifier we perform 10-fold cross validation [35], [62] across our PDB set. Complete data set is divided into ten equally-sized groups. One group is used as a validation set to test the model, while the other groups are used to train it. This process is executed ten times, with a different group used for testing each time. Results for cross validation are shown in Figure 4.5.

Method Performance

We compare the performance of our n-gram method against the benchmark programs using CASP10 (Figure 4.2), PDB30 (Figure 4.3) and CAMEO hard targets (Figure 4.4) sets. Our method performs at the same level than Espritz NMR (sequence-only) for PDB30 and CAMEO sets and outperforms it for the CASP10 set. The comparison with Espritz X-ray in the PDB30 set is not relevant since there is a significant overlap between sequences in our test set (PDB30) and Espritz X-ray training set. This overlap explains in part the relatively high performance. For the CASP10 dataset, we included results from Disopred3 to have a comparison point with a sequence profile method. Both methods were downloaded and ran locally in a Linux server using the default parameters.

The performance of the method was evaluated on disordered regions of various lengths for the CASP10 dataset (Table 4.3). The percentage of residues correctly predicted to be disordered is reported in

Table 4.4. As only disordered residues are considered for each category, the percentage of correctly predicted as disordered corresponds to the recall.

Finally, to evaluate the speed at which our method performs predictions we created a script which takes as input a FASTA file of proteins to predict, performs predictions and saves the results to a file. Average execution time needed to perform predictions in a standard Linux server (4 CPUs/4GB memory) was 5.86s for the CASP10 dataset.

Table 4.3 Performance of methods predictor against CASP10 targets

Method	Acc.	Sens.	Spec.	MCC	AUC
Espritz (Xray)	0.72	0.54	0.89	0.30	0.81
Espritz (NMR)	0.68	0.56	0.79	0.20	0.73
ngram	0.72	0.61	0.83	0.26	0.80
Disopred3	0.64	0.32	0.97	0.32	0.85

Table 4.4 Prediction accuracy by region length in CASP10

Method	<10AA	10-30AA	>30AA
Espritz (Xray)	50.5	58.8	48.3
ngram	48.8	46.3	54.7
Disopred3	25.1	33.1	45.2

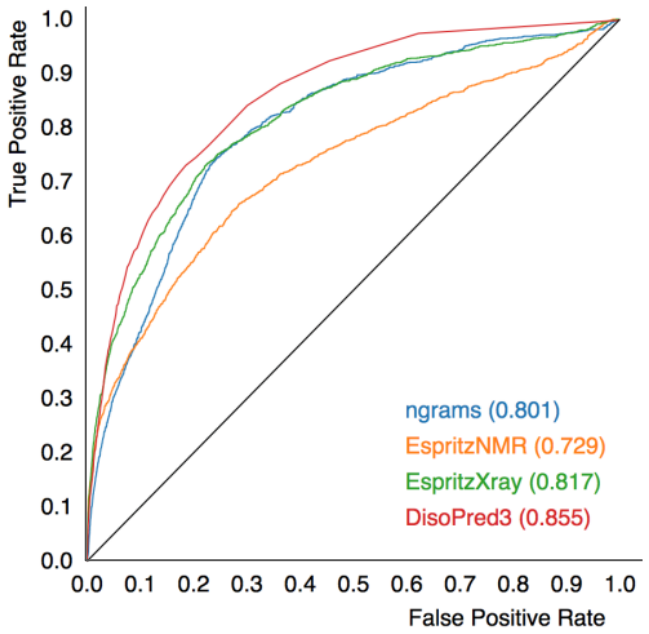

Figure 4.2 CASP10 targets performance

Table 4.5 Performance of methods predictor against PDB30 targets

Method	Acc.	Sens.	Spec.	MCC	AUC
Espritz (Xray)	0.83	0.78	0.87	0.39	0.90
Espritz (NMR)	0.78	0.75	0.80	0.27	0.86
ngrams	0.79	0.73	0.86	0.33	0.85

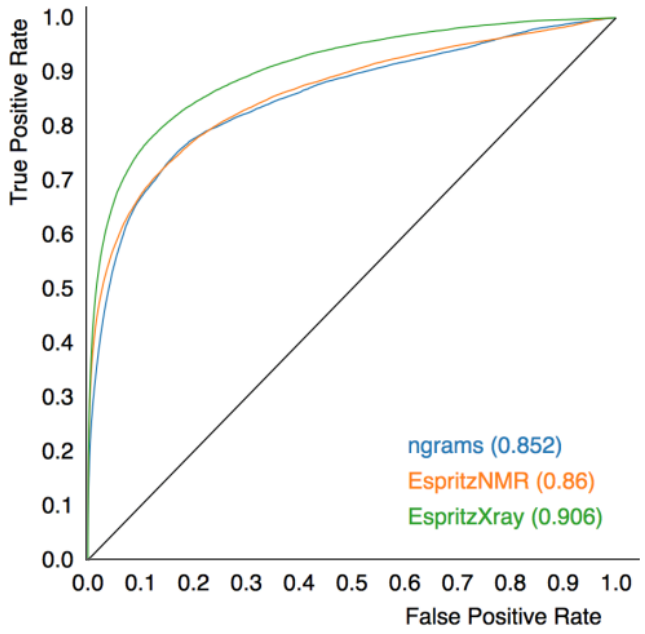

Figure 4.3 PDB30 targets performance

Table 4.6 Performance of methods predictor against CAMEO targets

Method	Acc.	Sens.	Spec.	MCC	AUC
Espritz (Xray)	0.74	0.74	0.74	0.27	0.80
ngrams	0.73	0.56	0.89	0.34	0.79

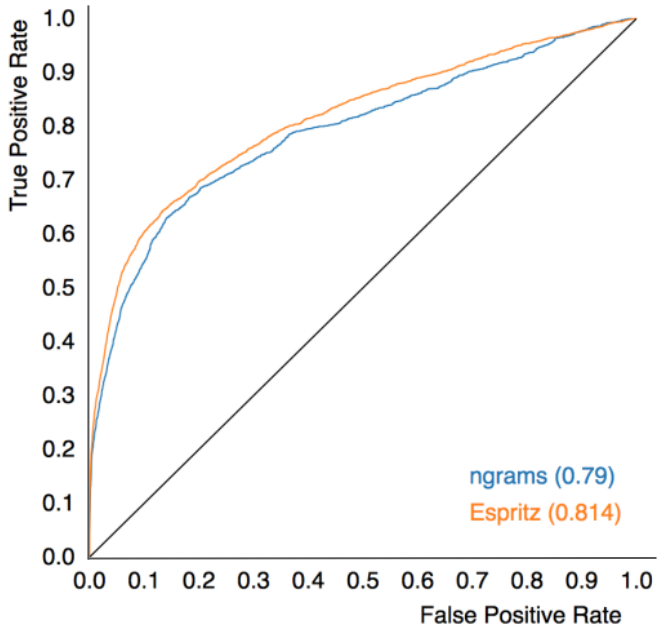

Figure 4.4 CAMEO targets (hard) performance

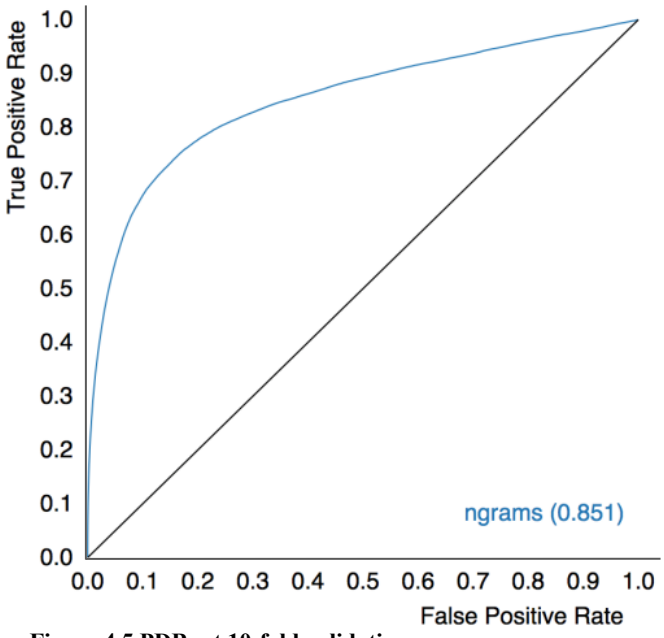

Figure 4.5 PDB set 10-fold validation

Decision Tree Analysis

Figure 4.6 provides an example visual representation of the type of trees generated by the random forest algorithm. Trees tended to use n-gram features with polar and charged residues (letter 'U' in the reduced alphabet). Since these n-grams tended to be close to the root of the tree, these features were particularly important in classification. This may imply that the presence of these residues is a key factor in how an n-gram based model differentiates between disorder and order. In fact, disordered sequences are known to contain higher proportions of polar and charged residues than ordered sequences [50]. This suggests a possible link between the importance of such residues in the decision trees and this previously known observation.

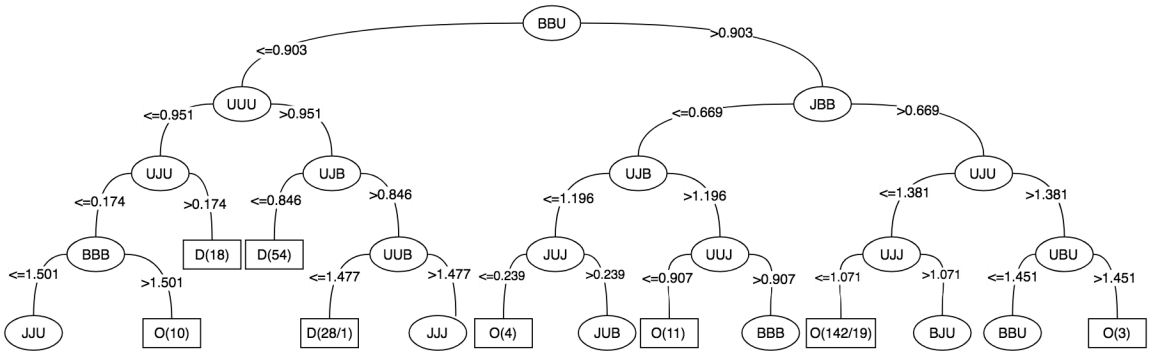

Figure 4.6 Visual representation of 3-gram based C4.5 decision tree using reduced alphabet 1. Each circular node represents a 3-gram, and the edges show how the 3-gram frequencies were used in the decision-making process

Conclusions

This paper presents a new n-gram based classification method for protein disorder prediction. We demonstrated that a combination of an alphabet reduction strategy with n-gram frequency patterns leads to an approach which can successfully compete with more elaborated and computationally expensive algorithms.

Overall and despite its simplicity, our method performs at similar levels as other sequence-based algorithms across all tested data sets. It has the additional advantage of providing insight into the mechanism of classification on the supplied data set.

Analysis of the decision trees showed that many n-grams were necessary for the decision-making process. However, the presence of polar and charged residues in the n-grams contributes to how the model differentiates between disorder and order state.

5. CNNALPHA: PROTEIN DISORDER REGIONS PREDICTION BY REDUCED AMINO ACID ALPHABETS AND CONVOLUTIONAL NEURAL NETWORKS

In this chapter, we present the work submitted to "PROTEINS: Structure, Function, and Bioinformatics" journal, currently under revision.

Abstract

Intrinsically disordered regions (IDR) play an important role in key biological processes and are closely related to human diseases [40]. They have the potential to serve as targets for drug discovery, especially in disordered binding regions [41], [42]. Accurate prediction of IDRs is challenging because their genome wide occurrence and a low ratio of disordered residues make them difficult targets for traditional classification techniques. Existing computational methods mostly rely on sequence profiles to improve accuracy which is time consuming and computationally expensive. This article describes an ab initio sequence-only prediction method -- which tries to overcome the challenge of accurate prediction posed by IDRs -- based on reduced amino acid alphabets and convolutional neural networks (CNNs). We experiment with six different 3-letter reduced alphabets. We argue that the dimensional reduction in the input alphabet facilitates the detection of complex patterns within the sequence by the convolutional step.

Experimental results show that our proposed IDR predictor performs at the same level or outperforms other state-of-the-art methods in the same class, achieving accuracy levels of 0.76 and AUC of 0.85 on the publicly available Critical Assessment of protein

Structure Prediction dataset (CASP10). Therefore, our method is suitable for proteome-wide disorder prediction yielding similar or better accuracy than existing approaches at a faster speed.

Introduction

Intrinsically disordered proteins (IDP) or intrinsically disordered regions (IDR) are segments within a protein chain lacking a stable three-dimensional structure under normal physiological conditions. They have been known to scientists for over 50 years and since then, linked to key biological processes including regulation of transcription, signal transduction, cell cycle control, post-translational modifications, ligand binding, protein interaction, and alternative splicing [63] [64]. Disorder regions exist in up to half of the amino acids in eukaryotic proteins [43] and at least 6% of all the residues in SwissProt are believed to be within a disordered region [65].

Experimental structure resolution of IDP/IDRs is complex, lengthy and expensive. DisProt database [66], a community resource annotating protein sequences for intrinsically disordered regions, currently contains just over 800 proteins. A large number of computational prediction methods have been developed ([4],[8]) because of this inherent complexity. Existing methods can be classified in one of the following categories [5]: (i) Ab initio or sequence-based. They rely almost exclusively on amino acid sequence information to make a prediction. Features extracted from the primary sequence, alignment profiles or scoring matrices are used as input for statistical models which then make predictions of disorder regions. Generally, methods that do not rely on

complex external sources of information fall into this category and are referenced as sequence-only. (ii) Clustering. This approach generates tertiary structure models from the primary sequence. It then superimposes the different models onto each other with the assumption that positions in ordered regions will be conserved across models. (iii) Template based. Similar to clustering, template-based method predicts disordered regions of proteins by aligning the input sequence to homologous proteins with a known structure. Homologous proteins are found by doing a database search or by fold recognition methods. (iv) Meta or consensus. They combine the output of several disordered predictors into a single average, which tends to have a moderate increase in accuracy. Evolutionary information contained in sequence profiles helps ab initio methods to improve prediction accuracy. However, generating sequence profiles is time consuming and methods relying on them for predictions may not be suitable for large proteome-wide analysis.

This article presents a sequence-only ab initio method for predicting protein disorder based on reduced amino acid alphabets and convolutional neural networks (cnnAlpha). Our method relies solely on the amino acid sequence for determining disorder positions and is aimed to proteome-wide applications where speed and low false positive rate are prioritized over maximum accuracy [67].

Among the main challenges with sequence-based prediction methods are (a) the highly class imbalance nature of the datasets and (b) the difficulty in accurately capturing the interdependency of adjacent residues in determining the transitions between disorder and order states. If not addressed, a class imbalance can severely bias predictions toward

the majority class (order state). To solve the imbalance problem, we choose an undersampling technique where we randomly remove examples from the majority class until we have a balanced dataset. Undersampling has been proven to be highly successful yielding a positive performance within the context of convolutional networks and extreme ratio imbalance datasets [29]. In order to capture local sequence context, we use a sliding window approach which feeds into a convolutional neural network that is tasked with learning rich higher-order sequence features.

Convolutional neural networks proved to be very efficient and well performing in the field of computer vision, excelling in tasks such as object detection and image classification [20]. The adaptation of convolutional neural networks architectures for biological problems has been successful in the context of DNA-protein binding prediction [68] and DNA function modeling [69]. Reducing the amino acid alphabet from 20 to 3 letters enables a seamless adaptation of convolutional neural networks for protein models. Instead of analyzing 2-D images with three color channels (R,G,B), fixed length protein sequence windows are mapped to 1-D input vectors with three channels. This translation allows mapping the protein disorder prediction problem to the 2-class image classification problem in the computer vision domain.

Methods and Materials

Disorder definition and feature extraction

There is no universal agreement on how to define disorder residues from PDB files [58]. In the context of this work, we consider a residue to be in a disorder position if

it appears in the sequence records, but its coordinates are missing from the electron density map. We annotated our PDB training and CAMEO validation sets using this definition. The annotation provided by the CASP experiments was created using a similar definition.

This is not a perfect definition since there are other reasons why a residue can have missing coordinates (i.e., crystallization artifacts). However, it allows us to use a large number of proteins from PDB without further experimental validation.

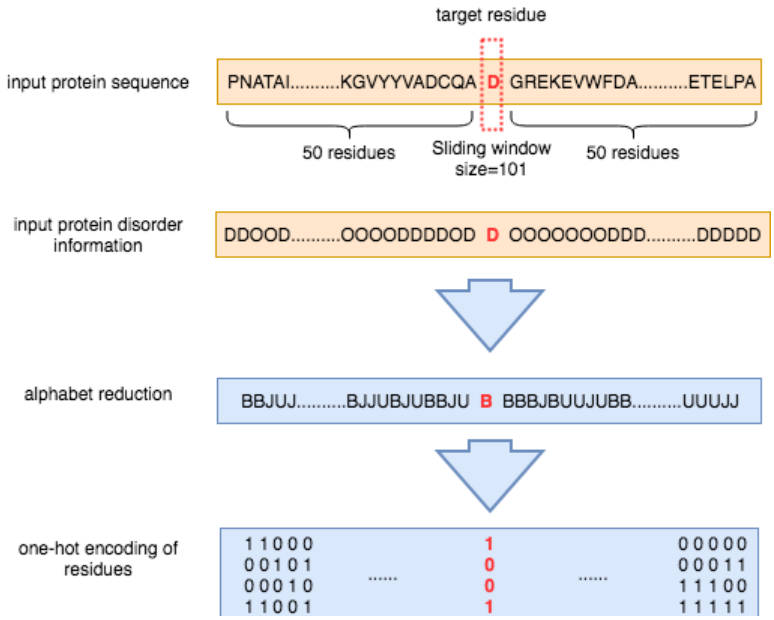

Figure 5.1 Sequence encoding, window generation and feature extraction steps using sliding window approach

The primary sequences from our training set had to be translated to numerical features to be fed into the convolutional network. For that purpose, we implemented a 101-residue length sliding window centered on the target residue. The window length

was set after experimenting with different sizes, finding that larger windows were more consistent in capturing disorder information. For each window, residues are represented by letters from the reduced amino acid alphabet and encoded using a one-bit hot encoding scheme. This generates a 3-D input feature matrix per target residue of size [3 x 101]. This process is illustrated in Figure 5.1

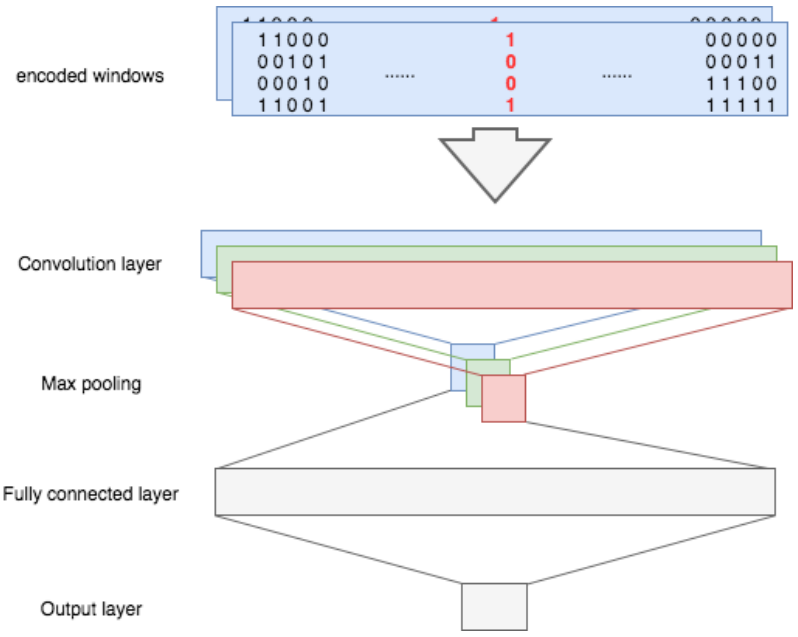

Figure 5.2 Basic 1-layer CNN architecture shared among all models

Reduced alphabets

Reduced alphabets cluster residues in ways that prevent the loss of key biochemical information. The 20-letter amino acid alphabet was reduced to a 3-letter alphabet in order to simplify and quicken the network learning process, reducing the

number of possible encodings and size of the input feature vectors. The reduced alphabets were selected from the literature (Table 5.1), where each was designed with a specific structural protein task in mind. In each alphabet, residues are clustered based on various properties, including chemical and genetic properties.

We found that alphabets 1,2 and 6 performed better in our specific classification task. Alphabet 1 achieves the reduction by mismatch minimization between the reduced interaction matrix and the Miyazawa and Jernigan (MJ) matrix. Alphabet 2 identifies the reduced alphabet which simplified sequence performs best in the context of protein fold recognition using global sequence alignments with the parent sequence. Alphabet 6 implements an automated reduction protocol using information theory metrics tailored to the prediction of solvent accessibility.

Table 5.1 The six reduced alphabets and their sources. Each letter contains a cluster of amino acid residues (one-letter abbreviations). The residue clusters were denoted by the letters "B", "J", and "U"

Alphabet/ref	Letter 1 (B)	Letter 2 (J)	Letter 3 (U)
a1 [30]	CFILMVWY	AGHPRT	DEKNQS
a2 [31]	AFGILMPV	DEKR	CHNQSTWY
a3 [32]	CFILMVWY	AGPST	DEHKNQR
a4 [33]	DHIMNVY	EFKLQ	ACGPRSTW
a5 [34]	ACGILMPSTV	EKRDNQH	FYW
a6 [38]	CILMVFWY	AGHST	DEKNPQR

Convolutional neural network architectures

The convolutional neural network architectures used in our models are variations of Figure 5.2. The input is a 3 x L matrix where L is the length of the sequence window (101 residues). Each symbol of the 3-letter reduced alphabet is mapped to one of the three one-hot vectors (B=[0,0,1], J=[0,1,0], U=[1,0,0]).

The first layer of our network is a convolutional layer, step size 1 and window size of 32. The output of each neuron on a convolutional layer is the convolution of the kernel matrix. The second layer is a max-pooling layer, one for each convolutional layer. Each of these max-pooling layers only outputs the maximum value (global or local) of its respective convolutional layer outputs. The third layer is a fully connected layer of size 256 where each of its neurons is connected to all of the neurons in the max-pooling layer. We use a dropout layer [25] after the fully connected layer to avoid overfitting. The final output layer consists of two neurons corresponding to the two classification results. These two neurons are fully connected to the previous layer. Table 5.2 highlights the differences between each of the tested models.

Table 5.2 Description of the CNN architectures tested

Method	Architecture description
64-ker-local	1-convolutional layer, 64 kernels, local max pooling
128-ker-local	1-convolutional layer, 128 kernels, local max pooling
64-ker-global	1-convolutional layer, 64 kernels, global max pooling
128-ker-global	1-convolutional layer, 128 kernels, global max pooling
2-conv-local	2-convolutional layers, [64,32] kernels, local max pooling

Network training details

We train our models using stochastic gradient descent (SGD) with mini batches of size 128. SGD works by utilizing chain ruling to take the partial derivative of the loss function with respect to each weight vector in the network and use the derivative to update the weights. We use a version of SGD with support for momentum and learning rate decay with default parameters and a learning rate set to 1e-3. All models are trained using the same setup and configuration the only difference being the seeds for initializing weights. We use early stopping, based on the validation set in order to pick the optimal set of weights. We train all our neural network models on AWS using G3 instances (NVIDIA Tesla M60 GPU) using python Keras libraries [70] running on top of TensorFlow library to assure model portability.

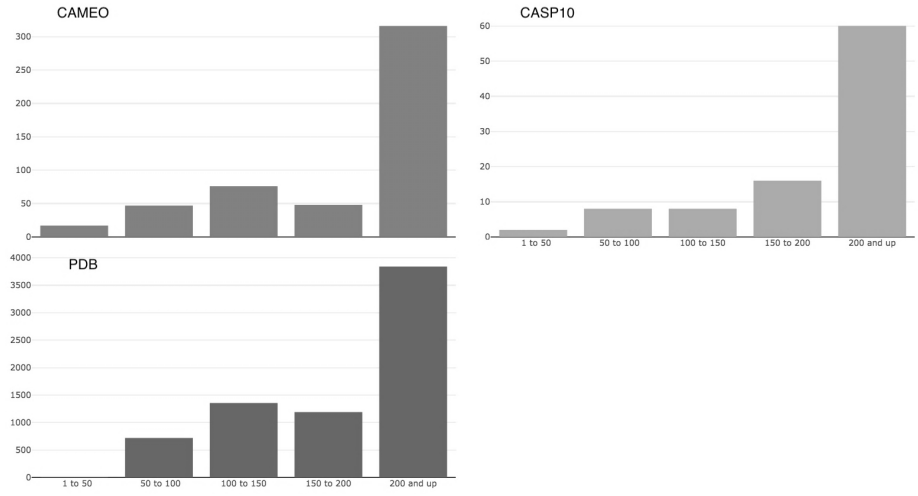

Figure 5.3 Protein length distribution in training, test and validation sets

Results

Training, Validation and Evaluation Datasets

Publicly available datasets are used to train, validate and evaluate the performance of our method. High resolution X-ray crystal structures from the Protein Data Bank (PDB) [55] are used to construct the training and validation data sets while CASP [4] and CAMEO [56] (http://www.cameo3d.org) are used for further validation. Figure 5.3 Protein length distribution in training, test and validation sets shows the protein length distribution for training, testing and validation sets.

We use Pisces protein sequence culling server (http://dunbrack.fccc.edu) [57] to extract sequences from PDB, filter for high resolution and reduce redundancy. Parameters selected for culling are (i) proteins sharing less than 25\% sequence identity (ii) resolution better than 1.8 Angstroms (iii) R-value up to 0.30. In total, 7,119 proteins are retrieved from PDB with an average length of 349 residues. The original dataset is then undersampled to create a 50/50 class balanced set, containing 181,060 examples. The effect of class imbalance is very detrimental to classification performance. In cases of an extreme ratio of imbalance, undersampling has been shown to perform on a par with oversampling without the risk of overfitting [29]. Undersampling has the additional advantage of reducing training times given that the training set is smaller in size.

Table 5.3 Distribution of disordered regions by length on the three main datasets used

Dataset	Number of Fragments			
	1-5	6-15	16-25	>25
CASP10	21	41	11	3
CAMEO	143	114	27	11
PDB	768	657	127	37

The balanced dataset was randomly partitioned into ten equally sized subsets and a ten-fold cross-validation was performed to determine the optimal parameters for (a) convolutional network architecture and (b) encoding reduced protein alphabet (Section 3.4). At each step of the cross validation, one subset is selected and used as validation set while the remaining nine are used as training set. This process is repeated until all subsets are validated, results for each of the parameters tested are shown in Table 5.4 and Table 5.5.

CASP10 is the latest dataset available from the series experiments, which released specific targets for protein disorder prediction. The 94 available targets are used for initial validation and as an independent benchmark set. Finally, to further assess and compare our method, we tested it against CAMEO 6 months targets released from August 26, 2017 to February 18th, 2018 (504 targets, categorized in 3 groups). Since CAMEO targets were released after the construction of our PDB training set, there is no sequence overlap between the two set. However, CASP10 targets were already present in PDB at the time of extraction. To prevent any redundancy between sets, we use BLASTClust [59] to filter and remove sequences from the PDB training set sharing at least 25\% identity with sequences in the CASP10 set.

Metrics and evaluation criteria

Disorder data is characterized by high class imbalance, disordered residues account for less than 5% of the data in the PDB set (training and test). Since disordered residues are relatively rare compared to ordered ones, they are harder to predict. Performance metrics should account for this imbalance and reward correct prediction of disordered residues higher than the correct prediction of ordered ones [41]. We selected a subset of the metrics commonly used for the assessment of disorder data [46] [61] [71] that take into account the nature of the imbalanced data: (i) specificity (ii) sensitivity (iii) balanced accuracy (iv) Matthews correlation coefficient and (v) AUC.

Binary metrics

$$\text{Specifity} = \frac{TN}{TN + FP} \tag{2}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{Acc} = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} \tag{4}$$

$$\text{MCC} = \frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

True positives (TP) and true negatives (TN) are the numbers of correctly predicted disordered and ordered residues. False positives (FP) and false negatives (FN) are the numbers of incorrectly predicted disordered and ordered residues.

Statistical metrics

The Receiver Operating Characteristic (ROC) curve is a plot that compares the true positive rate against the false positive rate under various threshold values for a binary classifier. ROC curve represents a monotonic function describing the balance between the true positive and false positive rates of a predictor [72]. For a set of probability thresholds (from 0 to 1), a residue is considered as a positive example (disordered) if its predicted probability is equal to or greater than the threshold value. The area under the curve (AUC) is used as an aggregate measure of the overall quality of a prediction method. AUC has a minimum value 0, a random value 0.5 and a perfect value 1.

Programs to compare

To benchmark our method we selected the following programs: Espritz [6], Disopred3 [50], IUPred [73] and ngram-sAlpha [39]. Given that our predictor is sequence-based, we compared our results with similar methods and we leave out clustering, template and meta based approaches. Espritz is an ensemble of sequence-only and multiple sequence alignments disorder prediction methods. The sequence-only method has three different versions, depending on the initial set used for training (X-ray, NMR, Disprot). We used X-ray trained version since it is the one that performs best among the three. Disopred3 runs a PSI-BLAST search for each of the residues in a 15-residue window. The profile is then used as input to a neural network classifier which outputs a probability estimate of the residue being disordered. IUPred method is based on estimating the capacity of polypeptides to form stabilizing contacts. It has two prediction

modes: IUPred (Long) and IUPred (Short). Each mode optimizes predictions for either long or short disordered regions. Finally, ngramsAlpha is our previously published predictor based on n-grams frequencies and reduced protein alphabets.

Parameter and model selection

In order to select the best performing model, we experimented with two of the components of our method while leaving the remaining parameters constant. In particular, we tested several network architectures and reduced amino acid alphabets and analyzed their effect on the model predictive value. We performed a ten-fold cross-validation, using the mean AUC across validation batches as the primary metric to compare performance. Values for parameters such as dropout and learning rate, optimizer, and window size have been selected after performing a hyperparameter search across a reduced size training set and are left constant.

Alphabet selection

Using reduced alphabets has two main advantages: (i) cluster residues with similar biochemical properties providing additional information to the original sequence and (ii) reduce the amino acid space from 20 to 3 residues, reducing, in turn, the model complexity and amount of data required for training.

We tested six different alphabets from the literature and analyzed which performed better in the context of our classification problem. We used the (2-conv-local) network architecture across all runs. A modified version of the network using the full

amino acid alphabet as input features (no alphabet translation step) is included for comparison. The effect of alphabet selection is shown in Table 5.4. Across the ten validation batches, we found that alphabets 1, 2, and 6 achieved better overall performance than alphabets 3, 4 and 5. Results also show that all six alphabets outperformed the model where no alphabet reduction was applied. This highlights the benefit of the dimensionality reduction step before training our models.

Despite being created with different objectives, the three alphabets cluster the same residues in group B, differing in the composition of groups J and U (Table 5.1). Group B contains most of the residues usually associated with ordered regions [74], which are hydrophobic and uncharged. The composition of group J and U differ among the three alphabets, containing disorder-promoting residues (polar/charged) and ambiguous residues (associated either with ordered or disordered regions). We selected alphabet 6 for our final model implementation based on the results shown in Table 5.4.

Table 5.4 Alphabet cross validation

Alphabet	**AUC Value of 10 cross Validation Batch Datasets**										
	1	2	3	4	5	6	7	8	9	10	mean
Alphabet 1	87.5%	89.2%	86.4%	88.5%	88.8%	87.2%	87.4%	87.5%	87.5%	87.7	87.8%
Alphabet 2	83.0%	86.3%	82.8%	86.6%	83.9%	82.8%	83.5%	84.3%	84.5%	84.6%	84.4%
Alphabet 3	**87.7%**	88.8%	**87.6%**	**89.3%**	88.5%	87.1%	87.4%	**88.0%**	**88.0%**	87.8%	88.0%
Alphabet 4	81.8%	86.0%	83.9%	86.2%	85.0%	81.4%	84.1%	83.8%	83.8%	85.1%	84.4%
Alphabet 5	85.2%	87.1%	84.0%	87.4%	85.1%	83.8%	86.0%	84.8%	85.3%	85.7%	85.5%
Alphabet 6	87.3%	**89.5**%	87.4%	89.0%	**88.9%**	**87.5%**	**87.5%**	87.6%	87.6%	**88.0%**	**88.2%**
NoAlphab	82.1%	85.5%	82.9%	86.0%	83.0%	81.7%	82.9%	82.6%	83.6%	83.5%	83.4%

Convolutional network architecture

To test the relationship between network architecture and performance, we trained five different networks models and evaluated their predictive value. We adapted models successfully used in the DNA space to predict DNA-protein binding and function ([68], [69]) hoping they would also perform well in the 3-letter reduced amino acid space. Our models differ in the number of kernels (50, 64, 128), the number of convolutional layers (1, 2) and max-pooling layer implementation (global vs. local). We found that the number of convolutional layers does not seem to have a great impact on performance. Models with a higher number of convolution kernels and local pooling implementation achieved better overall classification performance. Based on the results shown in Table 5.5, we selected 128-ker-local model.

Table 5.5 Model cross validation

Model	AUC Value of 10 cross Validation Batch Datasets										
	1	2	3	4	5	6	7	8	9	10	mean
64-ker-local	88.1%	**89.6%**	87.6%	**89.6%**	88.4%	87.4%	**87.9%**	88.0%	88.2%	**87.9%**	88.3%
128-ker-local	**88.4%**	89.5%	**87.8%**	89.6%	88.4%	**87.7%**	87.8%	88.2%	**88.4%**	87.9%	**88.4%**
64-ker-global	87.5%	88.2%	85.8%	89.0%	87.4%	86.3%	86.6%	87.0%	87.3%	86.8%	87.2%
128-ker-global	87.4%	88.5%	86.4%	89.1%	87.6%	85.8%	86.7%	87.6%	87.0%	86.8%	87.3%
2-conv-local	87.9%	89.1%	87.3%	89.2%	88.4%	87.4%	87.8%	87.8%	87.8%	87.7%	88.1%

Method performance

Figure 5.4, Figure 5.5, Table 5.6 and Table 5.7 compare the performance of our method against Disopred3, Espritz, IUPred, and ngramAlpha. It is worthwhile to mention

that -- of the listed methods -- Disopred is the only to make use of additional evolutionary information through sequence profiles (performing PSI-BLAST [75] searches for each input protein). This added evolutionary information gives the method an extra advantage in performance but comes at the cost of execution time. The other three methods are similar in nature to ours, using sequence-only information to make disorder/order predictions. All methods were downloaded and ran locally in a Linux server using default parameters.

Table 5.6 Performance of predictors on CASP10 dataset. Metrics showed: balanced accuracy (B.Acc), Sensitivity (Sens), Specificity (Spec) Mattehews correlation coefficient (MCC), and Area under the ROC curve (AUC)

Method	sequence profile	B.Acc	Sens	Spec	MCC	AUC
Disopred3	yes	0.64	0.32	0.97	0.32	0.86
cnnAlpha	no	0.75	0.64	0.85	0.31	0.85
Espritz	no	0.72	0.54	0.89	0.30	0.82
ngramAlpha	no	0.72	0.61	0.83	0.26	0.79
UIPred (short)	no	0.63	0.31	0.95	0.26	0.66
UIPred (long)	no	0.57	0.17	0.96	0.15	0.60

Table 5.7 Performance of predictors on CAMEO dataset. Metrics showed: balanced accuracy (B.Acc), Sensitivity (Sens), Specificity (Spec) Mattehews correlation coefficient (MCC), and Area under the ROC curve (AUC)

Method	sequence profile	B.Acc	Sens	Spec	MCC	AUC
Disopred3	yes	0.72	0.48	0.96	0.43	0.86
cnnAlpha	no	0.75	0.61	0.88	0.36	0.83
Espritz	no	0.75	0.64	0.88	0.35	0.81
ngramAlpha	no	0.73	0.56	0.89	0.33	0.79
UIPred (short)	no	0.71	0.47	0.94	0.36	0.80
UIPred (long)	no	0.64	0.35	0.93	0.27	0.73

In terms of balanced accuracy (B.Acc), our method outperforms all others on the two independent validation datasets. With respect to area under the ROC curve (AUC) and MCC, our method performs much better than the predictors not using sequence profiles (such as IUpred and Espritz) and nears the performance of Disopred3 for AUC on both validation sets.

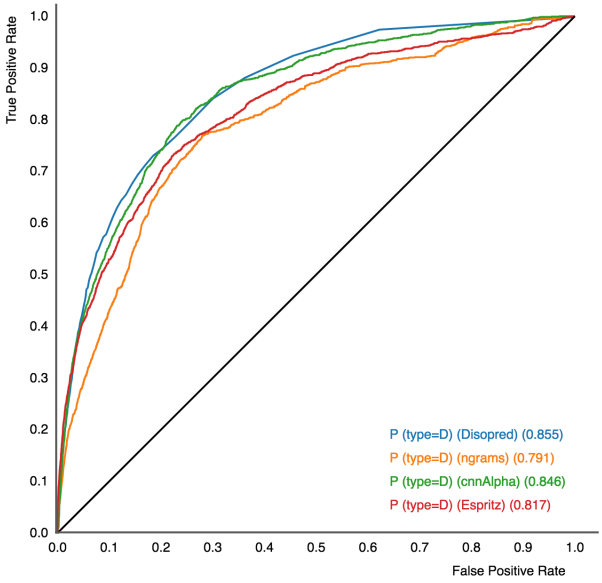

Figure 5.4 ROC curve for the evaluation set targets comparing the performance of the top four models (CASP)

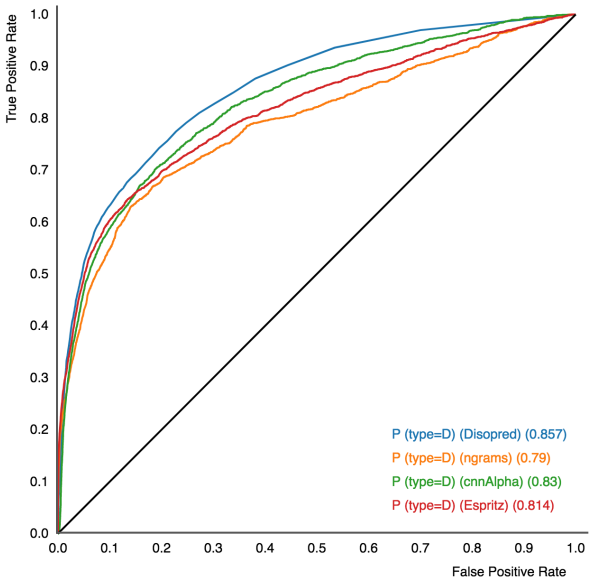

Figure 5.5 ROC curve for the evaluation set targets comparing the performance of the top four models (CAMEO)

The performance of the method was also evaluated on disordered regions of various lengths for the CASP10 dataset and compared with the other top performant methods. The percentage of residues correctly predicted to be disordered is reported in Table 5.8. While Espritz performs better on short length disorder regions, Disopred3 and cnnAlpha achieve better results on mid and long disordered regions.

Table 5.8 Predictors recall by region length in CASP10

Method	<10AA	10-30AA	>30AA
cnnAlpha	0.40	**0.42**	0.46
Espritz	**0.43**	0.39	0.33
Disopred3	0.26	0.32	**0.47**

Finally, we evaluate the speed at which our method performs predictions on a large scale. We created a script that takes as input a FASTA file of target proteins, performs predictions and saves the results into a file. The average execution time needed to perform predictions in a standard Linux server (4 CPUs/4GB memory) for the CASP10 dataset (94 proteins, 25,370 residues) was 0.37 seconds per protein.

Discussion

This paper presents cnnAlpha, a new convolutional neural network-based method for protein disorder prediction using sequence information. We demonstrated that our combination of amino acid alphabet reduction strategy and convolutional neural networks leads to an approach which can successfully compete with more elaborated and computationally expensive sequence-based algorithms. The source code for an R/Shiny application with the model implementation of our predictor can be found at https://github.com/mauricioob/shiny-pred.

CNNs are good at learning rich higher-order sequence features, such as secondary motifs and local sequence context. We believe that the reduction in dimension from 20 to 3 letter amino acid alphabet helped the convolutional layer to better detect these relationships and patterns. The reduction in dimensionality and our undersampling approach to the class imbalance problem have the additional advantage of reducing the amount of data required by the training sets. This, in turn, made our models faster to train and allow us further experimentation in parameter setting.

Overall, our method outperforms similar sequence-only algorithms across both evaluation data sets and nears the performance of sequence-based methods using additional evolutionary information (sequence profiles). Being several orders of magnitude faster than sequence profile based approaches, our method is suitable for high-throughput predictions at the proteomic scale. The high specificity of cnnAlpha also ensures a low false positive rate on high-throughput contexts, making it even more suitable for this task.

6. SHINY-PRED: A SERVER FOR THE PREDICTION OF PROTEIN DISORDERED REGIONS

In this chapter, we present the paper submitted and published by F1000Research [76].

Abstract

Intrinsically disordered proteins or intrinsically disordered regions (IDR) are segments within a protein chain lacking a stable three-dimensional structure under normal physiological conditions.

Accurate prediction of IDRs is challenging due to their genome wide occurrence and low ratio of disordered residues, making them a difficult target for traditional classification techniques. Existing computational methods mostly rely on sequence profiles to improve accuracy, which is time consuming and computationally expensive.

The shiny-pred application is an ab initio sequence-only disorder predictor implemented in R/Shiny language. In order to make predictions, it uses convolutional neural network models, trained using PDB sequence data. It can be installed on any operating system on which R can be installed and run locally. A public version of the web application can be accessed at https://gmu-binf.shinyapps.io/shiny-pred

Introduction

Experimental structure resolution of intrinsically disordered proteins/intrinsically disordered regions (IDP/IDRs) is complex, lengthy and expensive, leading to a variety of computational approaches being developed [4]. Over 60 computational protein disorder

prediction servers are currently available, although not all publicly. Methods can be classified in one of the following categories [5]: (i) Ab initio or sequence-based, (ii) clustering, (iii) template based, and (iv) meta or consensus.

shiny-pred is an ab initio predictor, which means it relies exclusively on amino acid sequence information to make disordered predictions. It uses prediction models based on convolutional neural networks and reduced protein alphabets. Currently, there are three available models, each one built using the same training protein data from PDB [55] but differing on the convolutional neural network architecture. Since it doesn't rely in sequence profiles to make predictions, it is fast to be used in proteome-wide disorder scenarios. It performs at the same level or outperforms other state-of-the-art sequence-only methods, achieving accuracy levels of 0.76 and AUC of 0.85 on the publicly available CASP10 dataset [54], at faster speeds.

Methods

Implementation

shiny-pred is written in the R programming language [77] and the shiny web application framework is implemented using the Shiny R package v1.1 [78]. Currently, three convolutional neural network models are made available by our application: (i) cnn-64-ker-local, is a one layer convolutional network (step size 1 and window size of 32) with 64 kernels and local max pooling model; (ii) cnn-128-ker-local, implements one convolutional layer (step size 1 and window size of 32) with 128 kernels and local max pooling model; and (iii) cnn-2-conv-local implements two convolutional layers (64 and

32 kernels) with local max pooling. The models were created, trained and accessed using the Keras R package v2.1.6 [70].

Operation

Our tool has two operation modes; predicting disordered residues in protein sequences (prediction) and benchmarking the predictor performance against sequences with known disorder information (benchmark). The mode is selected automatically based on the format of the input sequences. Users can either upload a sequence file, type/paste a sequence into the text area or select pre-loaded examples from a list.

When in prediction mode, the amino acid sequences are expected to be in FASTA format. In benchmark mode, input sequences in FASTA format are expected to have an additional line containing the disorder information (D=disorder, O=ordered). Multiple sequences can be submitted at once; several examples for different types of submissions (prediction and benchmark modes) are made available as examples. In both modes, the application will show a result panel, where for each input sequence a graph with the probability of disorder per residue is plotted.

Prediction mode

The workflow for protein disorder prediction is:

(i) Input the target sequences (in FASTA format) in the text area;

(ii) Select the model to use for the prediction (default is cnn-128-ker-local) and submit the sequence for prediction;

(iii) Visualize and download results.

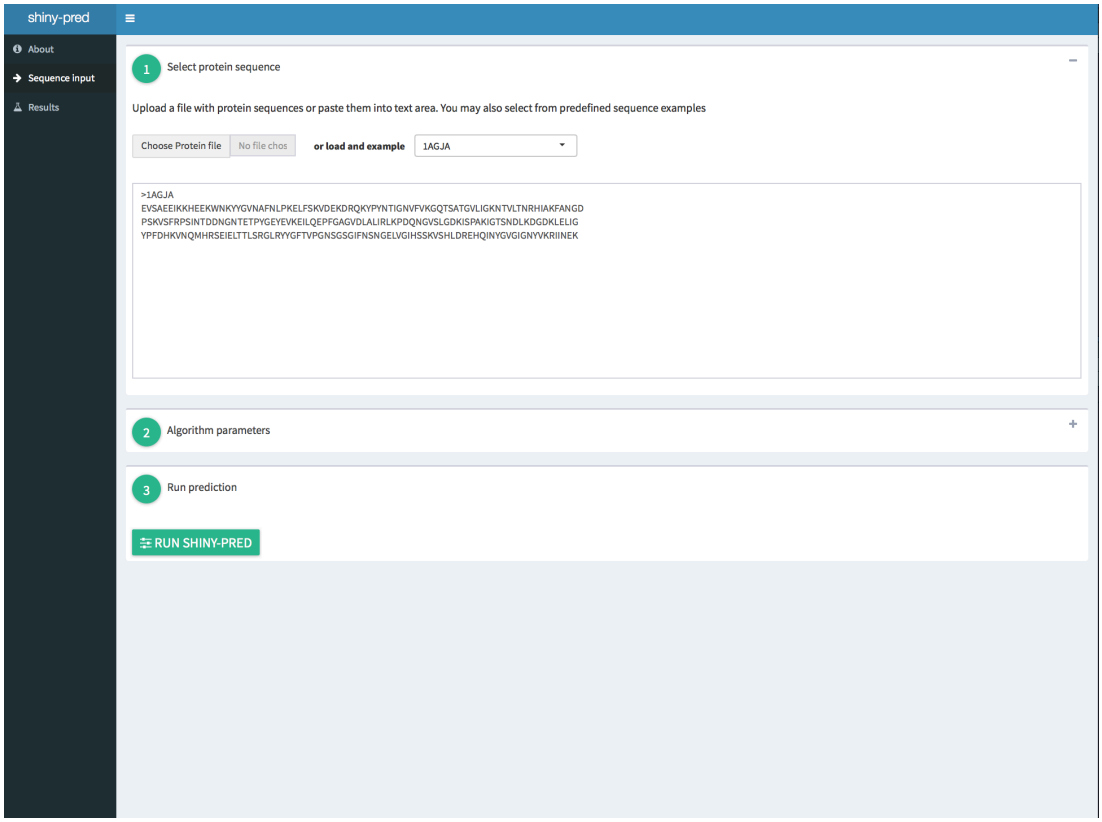

Figure 6.1 Input sequence format (prediction mode)

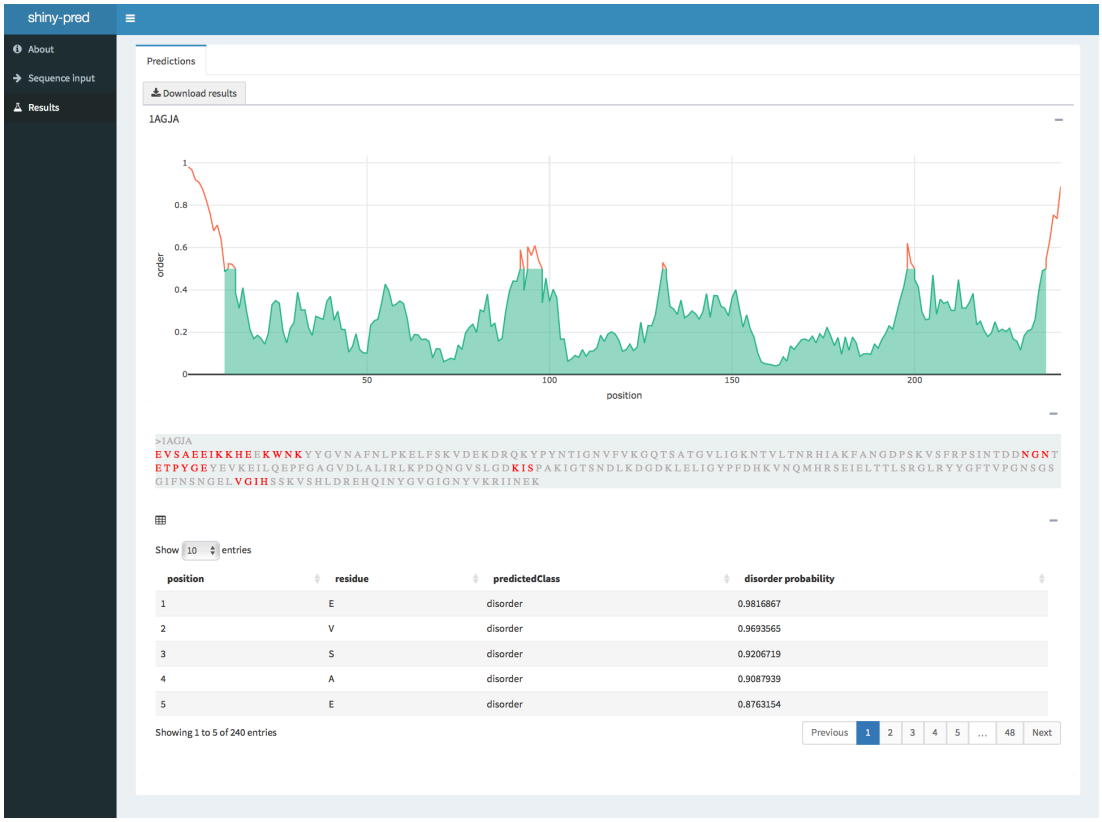

Figure 6.2 Prediction mode results

Benchmark mode

In benchmark mode, input sequences are expected to have an extra line with the actual disorder information to be used as benchmark. Result tables will populate two extra columns (actual class and match) with the actual disorder information and if the prediction was accurate for the current residue. An extra panel (Benchmark) shows the ROC curve along with other common binary metrics (sensitivity, specificity, balance accuracy, and Matthews correlation coefficient).

Use Cases

We use shiny-pred to predict disordered regions within the publicly available CASP10 benchmark dataset. The dataset contains 94 target sequences, each one annotated with the disorder/order information at the residue level. The annotated dataset is provided as an example ('CASP_all') and it can be selected from the example selection list on the 'Sequence Input' tab.

Figure 6.3 shows the input panel after the dataset is selected and loaded. Predictions per sequence can be viewed and downloaded from the 'Results' tab while the 'Benchmark' tab provides a summary of the performance using binary and statistical metrics.

Figure 6.4 shows the server performance for the input dataset, achieving a AUC value of 0.85 and balance accuracy of 0.75.

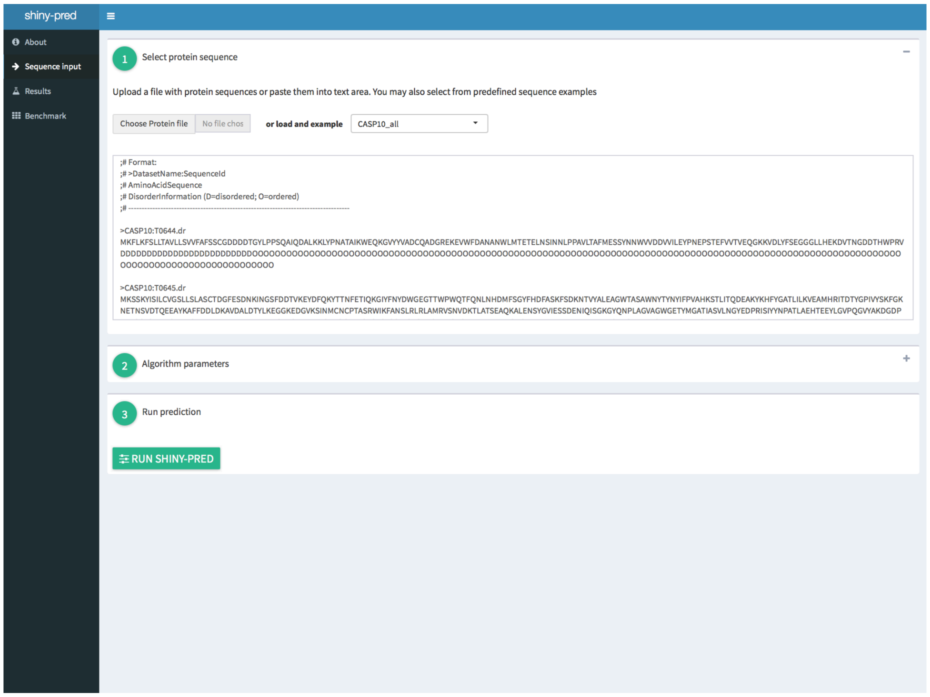

Figure 6.3 Input sequence format (benchmark mode)

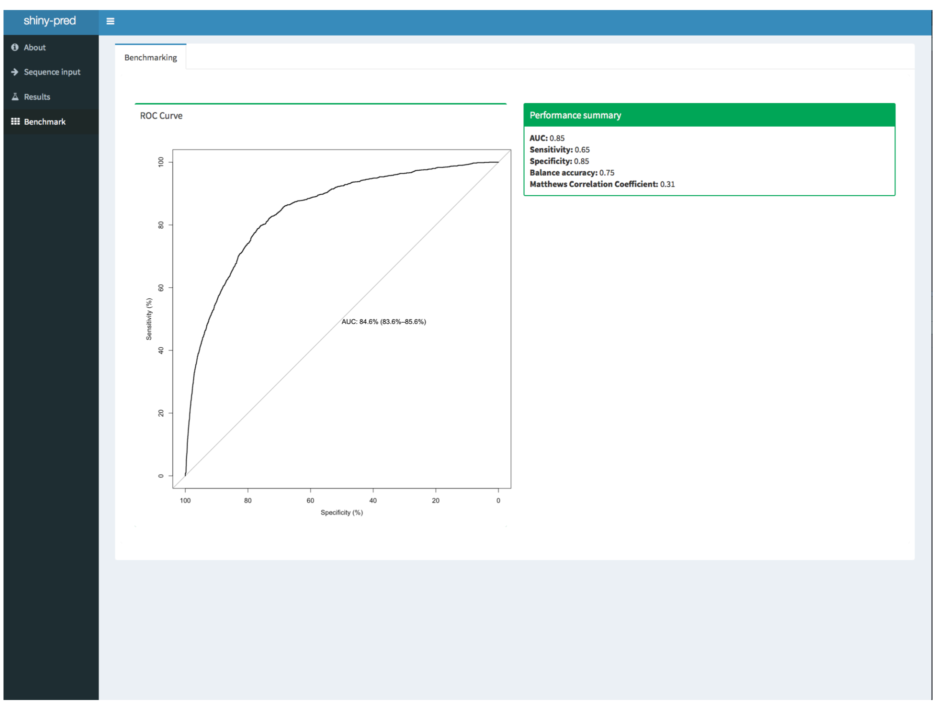

Figure 6.4 Predictor benchmarking

Summary

This article presents shiny-pred, a sequence-only ab initio web application for predicting protein disorder. It's based on reduced amino acid alphabets and convolutional neural networks, being fast and accurate, it is suitable for large proteome-wide experiments.

Software availability

Software available from: https://gmu-binf.shinyapps.io/shiny-pred

Source code available from: https://github.com/mauricioob/shiny-pred

Archived source code (publication): https://doi.org/10.5281/zenodo.2567259

License: GNU public license (GPL-3)

7. CONCLUSIONS

Two algorithms for the detection of intrinsically disordered regions in proteins were proposed in this dissertation work. Both approaches showed promising results, in particular, the one based on Convolutional Neural Networks achieved state-of-the-art results on the CASP benchmark datasets. After performing literature reviews of the current types of predictors and presenting the foundations of the methods to used, the specific objectives of this dissertation were to (i) develop an n-gram based approach to classify individual residues in protein sequences into one of the following classes (disorder, order) (ii) explore advanced machine learning methods to improve on the accuracy and prediction capabilities of our original n-gram based algorithm and (iii) develop a parameterized online resource for the prediction of disordered residues of a protein chain from its amino acid sequence-based on our best performant method.

In order to achieve the objective (i), a machine-learning based approach was developed using n-gram frequencies over reduced amino acid alphabets as features. Our approach calculates a position-dependent disorder score for each residue in the analyzed sequence and outputs a prediction of ordered/disordered based on a cut-off value. We benchmarked the performance of our method against existing independent methods (Espritz, Disopred, IUPred) using proteins from the CASP experiments, as well as a large subset of proteins extracted from PDB database. Despite underperforming when

compared to existing methods over the CASP datasets, n-gram frequencies when combined with alphabet mapping have shown a strong predictive power.

To achieve the objective (ii), we expanded on our first implementation losing the feature generation (n-grams frequencies) and training a Convolutional Neural Network directly on the translated reduced amino acid protein sequence. Our method showed promising outcomes when compared to algorithms of its same class (sequence-only ab initio).

For accomplishing goal (iii), an online prediction server based on our most performant algorithm was developed, based on the work in this dissertation. The server was built using the R/Shiny framework and is currently publicly available.

Finally, in order to improve the accuracy and performance of both methods, we suggest exploring the following series of modifications to be implemented and tested:

- Apply second tier refinements, making use of the predictions made on the first tier may improve accuracy
- Test other reduced alphabets in literature
- Usage of n-gram patterns which may increase frequency counts over smaller windows
- Explore different training datasets, combining proteins from PDB and DisProt and modifying our definition of disorder.

REFERENCES

[1] P. Tompa, “Intrinsically disordered proteins: a 10-year recap,” *Trends Biochem. Sci.*, vol. 37, no. 12, pp. 509–516, Dec. 2012.

[2] B. Xue, A. K. Dunker, and V. N. Uversky, “Orderly order in protein intrinsic disorder distribution: disorder in 3500 proteomes from viruses and the three domains of life,” *J. Biomol. Struct. Dyn.*, vol. 30, no. 2, pp. 137–149, Jun. 2012.

[3] S. Ratna, “Peptide Classification with Machine Learning,” *Teknos*. [Online]. Available: https://www.teknos.org/home/2018/1/31/classification-and-prediction-of-antimicrobial-peptides-using-n-gram-representation-and-machine-learning. [Accessed: 02-May-2019].

[4] B. He, K. Wang, Y. Liu, B. Xue, V. N. Uversky, and A. K. Dunker, “Predicting intrinsic disorder in proteins: an overview,” *Cell Res.*, vol. 19, no. 8, pp. 929–949, Aug. 2009.

[5] J. D. Atkins, S. Y. Boateng, T. Sorensen, and L. J. McGuffin, “Disorder Prediction Methods, Their Applicability to Different Protein Targets and Their Usefulness for Guiding Experimental Studies,” *Int. J. Mol. Sci.*, vol. 16, no. 8, pp. 19040–19054, Aug. 2015.

[6] I. Walsh, A. J. M. Martin, T. D. Domenico, and S. C. E. Tosatto, “ESpritz: accurate and fast prediction of protein disorder,” *Bioinformatics*, vol. 28, no. 4, pp. 503–509, Feb. 2012.

[7] X. Deng, J. Eickholt, and J. Cheng, “PreDisorder: ab initio sequence-based prediction of protein disordered regions,” *BMC Bioinformatics*, vol. 10, p. 436, Dec. 2009.

[8] X. Deng, J. Eickholt, and J. Cheng, “A comprehensive overview of computational protein disorder prediction methods,” *Mol. Biosyst.*, vol. 8, no. 1, pp. 114–121, Jan. 2012.

[9] T. Ishida and K. Kinoshita, “Prediction of disordered regions in proteins based on the meta approach,” *Bioinforma. Oxf. Engl.*, vol. 24, no. 11, pp. 1344–1348, Jun. 2008.

[10] T. Ishida and K. Kinoshita, “PrDOS: prediction of disordered protein regions from amino acid sequence,” *Nucleic Acids Res.*, vol. 35, no. Web Server issue, pp. W460-464, Jul. 2007.

[11] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[12] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A Training Algorithm for Optimal Margin Classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, New York, NY, USA, 1992, pp. 144–152.

[13] P. Domingos and M. Pazzani, “On the Optimality of the Simple Bayesian Classifier under Zero-One Loss,” p. 28.

[14] “Random forests - classification description.” [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed: 30-Apr-2019].
[15] “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278–282 vol.1.
[16] A. Dalyac, “Tackling Class Imbalance with Deep Convolutional Neural Networks.”
[17] Y. Bengio, “Learning Deep Architectures for AI,” *Found. Trends® Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
[19] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” p. 9.
[20] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
[21] T. M. Press, “Perceptrons,” *The MIT Press*. [Online]. Available: https://mitpress.mit.edu/books/perceptrons. [Accessed: 05-Mar-2019].
[22] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.
[23] B. C. Csáji and H. T. Eikelder, *Approximation with Artificial Neural Networks*. 2001.
[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, p. 533, Oct. 1986.
[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[26] “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.” [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 05-Mar-2019].
[27] S. Albelwi and A. Mahmood, “A Framework for Designing the Architectures of Deep Convolutional Neural Networks,” *Entropy*, vol. 19, no. 6, p. 242, Jun. 2017.
[28] T. Chen, R. Xu, Y. He, and X. Wang, “A Gloss Composition and Context Clustering Based Distributed Word Sense Representation Model,” *Entropy*, vol. 17, pp. 6007–6024, 2015.
[29] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018.
[30] J. Wang and W. Wang, “A computational approach to simplifying the protein folding alphabet,” *Nat. Struct. Mol. Biol.*, vol. 6, no. 11, pp. 1033–1038, Nov. 1999.
[31] C. Branden, “Introduction to protein structure. By C Branden and J Tooze. pp 302. garland publishing, New York. 1991 ISBN 0–8513–0270–3 (pbk),” *Biochem. Educ.*, vol. 20, no. 2, pp. 121–122, Apr. 1992.

[32] T. Li, K. Fan, J. Wang, and W. Wang, “Reduction of protein sequence complexity by residue grouping,” *Protein Eng.*, vol. 16, no. 5, pp. 323–330, May 2003.
[33] L. B. Mekler, *Specific selective interaction between amino acid residues of polypeptide chains.* 1969.
[34] L. R. Murphy, A. Wallqvist, and R. M. Levy, “Simplified amino acid alphabets for protein fold recognition and implications for folding,” *Protein Eng.*, vol. 13, no. 3, pp. 149–152, Mar. 2000.
[35] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten, “Data mining in bioinformatics using Weka,” *Bioinformatics*, vol. 20, no. 15, pp. 2479–2481, Oct. 2004.
[36] K. Chen, L. Kurgan, and J. Ruan, “Optimization of the Sliding Window Size for Protein Structure Prediction,” in *2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB ’06*, 2006, pp. 1–7.
[37] X. Deng, J. Gumm, S. Karki, J. Eickholt, and J. Cheng, “An Overview of Practical Applications of Protein Disorder Prediction and Drive for Faster, More Accurate Predictions,” *Int. J. Mol. Sci.*, vol. 16, no. 7, pp. 15384–15404, Jul. 2015.
[38] J. Bacardit, M. Stout, J. D. Hirst, A. Valencia, R. E. Smith, and N. Krasnogor, “Automated Alphabet Reduction for Protein Datasets,” *BMC Bioinformatics*, vol. 10, no. 1, p. 6, Jan. 2009.
[39] M. Oberti and I. I. Vaisman, “Identification and Prediction of Intrinsically Disordered Regions in Proteins Using N-grams,” in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics*, New York, NY, USA, 2017, pp. 67–72.
[40] V. N. Uversky, “Wrecked regulation of intrinsically disordered proteins in diseases: pathogenicity of deregulated regulators,” *Front. Mol. Biosci.*, vol. 1, Jul. 2014.
[41] G. Hu, Z. Wu, K. Wang, V. N. Uversky, and L. Kurgan, “Untapped potential of disordered proteins in current druggable human proteome,” *Curr. Drug Targets*, Jul. 2015.
[42] N. Rezaei-Ghaleh, M. Blackledge, and M. Zweckstetter, “Intrinsically disordered proteins: from sequence and conformational properties toward drug discovery,” *Chembiochem Eur. J. Chem. Biol.*, vol. 13, no. 7, pp. 930–950, May 2012.
[43] A. K. Dunker, S. E. Bondos, F. Huang, and C. J. Oldfield, “Intrinsically disordered proteins and multicellular organisms,” *Semin. Cell Dev. Biol.*, vol. 37, pp. 44–55, Jan. 2015.
[44] C. Y.-C. Chen and W. I. Tou, “How to design a drug for the disordered proteins?,” *Drug Discov. Today*, vol. 18, no. 19–20, pp. 910–915, Oct. 2013.
[45] Y. Cheng *et al.*, “Rational drug design via intrinsically disordered protein,” *Trends Biotechnol.*, vol. 24, no. 10, pp. 435–442, Oct. 2006.
[46] J. K. Vries, X. Liu, and I. Bahar, “The relationship between N-gram patterns and protein secondary structure,” *Proteins Struct. Funct. Bioinforma.*, vol. 68, no. 4, pp. 830–838, Sep. 2007.
[47] I. Vaisman and A. Srinivasan, “Identification and Prediction of Intrinsically Disordered Regions in Proteins Using n-Grams.” 2015.

[48] O. Noivirt-Brik, J. Prilusky, and J. L. Sussman, “Assessment of disorder predictions in CASP8,” *Proteins Struct. Funct. Bioinforma.*, vol. 77, no. S9, pp. 210–216, Jan. 2009.
[49] B. Monastyrskyy, K. Fidelis, J. Moult, A. Tramontano, and A. Kryshtafovych, “Evaluation of disorder predictions in CASP9,” *Proteins*, vol. 79, no. S10, pp. 107–118, 2011.
[50] J. J. Ward, L. J. McGuffin, K. Bryson, B. F. Buxton, and D. T. Jones, “The DISOPRED server for the prediction of protein disorder,” *Bioinformatics*, vol. 20, no. 13, pp. 2138–2139, Sep. 2004.
[51] L. P. Kozlowski and J. M. Bujnicki, “MetaDisorder: a meta-server for the prediction of intrinsic disorder in proteins,” *BMC Bioinformatics*, vol. 13, p. 111, May 2012.
[52] T. K. Ho, “The Random Subspace Method for Constructing Decision Forests,” *IEEE Trans Pattern Anal Mach Intell*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[53] J. R. Quinlan, “Induction of Decision Trees,” *Mach Learn*, vol. 1, no. 1, pp. 81–106, Mar. 1986.
[54] B. Monastyrskyy, A. Kryshtafovych, J. Moult, A. Tramontano, and K. Fidelis, “Assessment of protein disorder region predictions in CASP10,” *Proteins*, vol. 82, no. 0 2, pp. 127–137, Feb. 2014.
[55] H. M. Berman *et al.*, “The Protein Data Bank,” *Nucleic Acids Res.*, vol. 28, no. 1, pp. 235–242, Jan. 2000.
[56] J. Haas *et al.*, “The Protein Model Portal--a comprehensive resource for protein structure and model information,” *Database J. Biol. Databases Curation*, vol. 2013, p. bat031, 2013.
[57] G. Wang and R. L. Dunbrack, “PISCES: a protein sequence culling server,” *Bioinforma. Oxf. Engl.*, vol. 19, no. 12, pp. 1589–1591, Aug. 2003.
[58] R. Linding, L. J. Jensen, F. Diella, P. Bork, T. J. Gibson, and R. B. Russell, “Protein disorder prediction: implications for structural proteomics,” *Struct. Lond. Engl. 1993*, vol. 11, no. 11, pp. 1453–1459, Nov. 2003.
[59] “NCBI News: Spring 2004|BLASTLab.” [Online]. Available: https://www.ncbi.nlm.nih.gov/Web/Newsltr/Spring04/blastlab.html. [Accessed: 17-Apr-2017].
[60] S. Wang, J. Ma, and J. Xu, “AUCpreD: proteome-level protein disorder prediction by AUC-maximized deep convolutional neural fields,” *Bioinformatics*, vol. 32, no. 17, pp. i672–i679, Sep. 2016.
[61] T. Huang *et al.*, “A Sequence-based Approach for Predicting Protein Disordered Regions,” *Protein Pept. Lett.*, vol. 20, no. 3, pp. 243–248, Jan. 2013.
[62] R. Kohavi, “A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, San Francisco, CA, USA, 1995, pp. 1137–1143.
[63] A. K. Dunker *et al.*, “The unfoldomics decade: an update on intrinsically disordered proteins,” *BMC Genomics*, vol. 9 Suppl 2, p. S1, Sep. 2008.
[64] C. J. Oldfield and A. K. Dunker, “Intrinsically disordered proteins and intrinsically disordered protein regions,” *Annu. Rev. Biochem.*, vol. 83, pp. 553–584, 2014.

[65] T. Di Domenico, I. Walsh, A. J. M. Martin, and S. C. E. Tosatto, “MobiDB: a comprehensive database of intrinsic protein disorder annotations,” *Bioinforma. Oxf. Engl.*, vol. 28, no. 15, pp. 2080–2081, Aug. 2012.
[66] D. Piovesan *et al.*, “DisProt 7.0: a major update of the database of disordered proteins,” *Nucleic Acids Res.*, vol. 45, no. D1, pp. D219–D227, 04 2017.
[67] F. L. Sirota, H.-S. Ooi, T. Gattermayer, G. Schneider, F. Eisenhaber, and S. Maurer-Stroh, “Parameterization of disorder predictors for large-scale applications requiring high specificity by using an extended benchmark dataset,” *BMC Genomics*, vol. 11, no. 1, p. S15, Feb. 2010.
[68] H. Zeng, M. D. Edwards, G. Liu, and D. K. Gifford, “Convolutional neural network architectures for predicting DNA–protein binding,” *Bioinformatics*, vol. 32, no. 12, pp. i121–i127, Jun. 2016.
[69] D. Quang and X. Xie, “DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences,” *Nucleic Acids Res.*, p. gkw226, Apr. 2016.
[70] J. Allaire and F. Chollet, *R Interface to “Keras.”* 2018.
[71] S. Wang, J. Ma, and J. Xu, “AUCpreD: proteome-level protein disorder prediction by AUC-maximized deep convolutional neural fields,” *Bioinforma. Oxf. Engl.*, vol. 32, no. 17, pp. i672–i679, Sep. 2016.
[72] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[73] Z. Dosztányi, V. Csizmok, P. Tompa, and I. Simon, “IUPred: web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content,” *Bioinforma. Oxf. Engl.*, vol. 21, no. 16, pp. 3433–3434, Aug. 2005.
[74] A. K. Dunker *et al.*, “Intrinsically disordered protein,” *J. Mol. Graph. Model.*, vol. 19, no. 1, pp. 26–59, Feb. 2001.
[75] S. F. Altschul *et al.*, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, Sep. 1997.
[76] M. Oberti and I. Vaisman, “shiny-pred: a server for the prediction of protein disordered regions,” *F1000Research*, vol. 8, p. 230, Feb. 2019.
[77] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2018.
[78] W. Chang, *shiny: Web Application Framework for R*. 2018.

BIOGRAPHY

Mauricio Oberti received his bachelor's degree in computer science from the Catholic University, Montevideo, Uruguay (2003). He was awarded a master's degree in computer science from Johns Hopkins University, Maryland, USA (2008). He is currently a doctoral candidate at George Mason University and an Associate Director at the Novartis Institutes for BioMedical Research (NIBR).