A Recursive Connectionist Approach for Predicting Disulfide Connectivity in Proteins

Alessandro Vullo
Department of Systems and Computer Science
University of Florence
50139 Firenze, Italy
vullo@dsi.unifi.it

Paolo Frasconi
Department of Systems and Computer Science
University of Florence
50139 Firenze, Italy
paolo@dsi.unifi.it

ABSTRACT

We are interested in the prediction of disulfide bridges in proteins, a structural feature that conveys important information about the protein conformation and that can therefore help towards the solution of the folding problem. We assume here that the disulfide bonding state of cysteines is known and we focus on the subsequent problem of disulfide bridges pairings assignment. In this paper, disulfide connectivity is modeled by undirected graphs. A graph-space search algorithm is employed to explore alternative disulfide bridges patterns and prediction consists of selecting the 'best' graph in the search space. The core of the proposed method is a recursive neural network architecture trained to score candidate graphs. We report experiments on previously published data showing that our algorithm outperforms the known alternative methods for most proteins. Furthermore, we assess the generalization capabilities testing the model on previously unpublished data.

General Terms

Protein folding, Disulfide connectivity prediction, Recursive Neural Networks.

1. INTRODUCTION

Predicting the 3-dimensional arrangement of proteins starting from their sequences is one of the most challenging open problem in structural genomics. Many of the available prediction methods try to solve intermediate sub-problems, the goal of each one being the prediction of a simplified structural feature. Examples of these representations include the secondary structure, residue solvent accessibility and coordination numbers, cysteines bonding state and connectivity, transmembrane proteins topologies and contact maps.

Among the existing proteins, those which contain cysteine (Cys) residues represent an important class. Their sequence is subject to post-translational covalent modifications and

cysteines occur either in oxidized or thiol form. Two oxidized cysteines form a covalent bond, known as disulfide bridge, which helps in stabilizing the native conformation because it contributes to the thermodynamic stability of the 3D structure [9]. Depending on their number and location, these bonds may connect very distant portion of the sequence. Therefore, they add strong structural constraints that can be very helpful towards the ab-initio prediction of the 3D structure. In the absence of an experimentally determined structure, sequence archives do not report reliable information relating either the oxidized form of cysteines or disulfide bridges. Prediction techniques are based on two steps. First, the disulfide-bonding state of each cysteine is predicted (a binary classification problem) [2, 5, 7]. Once candidate cystines are known, other algorithms can be used to predict the actual location of disulfide bridges. This paper focuses on the second task, that has received relatively scarce attention in the literature. To the best of our knowledge, there is only one published method [3, 4]. It is based on a weighted graph representation of disulfide bridges, where vertices are oxidized cysteines and undirected edges are labeled by the strength of interaction (contact potential) in the associated pair of cysteines. First, constrained optimization or neural network predictions are used to find an optimal set of weights. After a complete labeled graph is obtained, candidate bridges are located by finding the maximum weight perfect matching[1]. The problem can be solved in polynomial time using linear programming. The computation of contact potentials is nevertheless a time consuming process.

The method we propose in this paper is based on extended recursive neural networks (RNN) [6], a connectionist model which allows to formulate classification and regression tasks on structured data, like the graphs representing disulfide connectivity patterns. The network is trained to score candidate graphs according to a similarity metric with respect to the correct graph. During prediction, the score computed by the network is used to exhaustively explore the space of candidate graphs.

The paper is organized as follows. Next section gives a formal definition of the problem and describes the prediction algorithm. The recursive neural network architecture is then introduced. Subsequently, data preparation, the used performance measures, and the adopted experimental protocol are described in the given order.

---

[1]A perfect matching of a graph (V, E) is a subset $E' \subseteq E$ such that each vertex $v \in V$ is met by only one vertex.

2. PREDICTION ALGORITHM

A disulfide connectivity pattern has a simple representation in terms of an undirected graph $G = (V, E)$. The vertex set V represents the set of bonded cysteines and an edge $e \in E$ corresponds to a disulfide bridge between its adjacent cysteines. Admissible vertex and edge sets are constrained because an even number of intra-chain bonded cysteines is required and a cysteine can only be bridged to one and only one different cysteine. Thus, we have $|V| = 2B$, $|E| = B$ and $degree(v) = 1$ for any $v \in V$ (perfect matching), where B denotes the number of disulfide bonds in a chain.

The problem of predicting the correct connectivity pattern for a given disulfide bonded chain is simply formulated as finding the best possible candidate as given by a suitable scoring function. This function maps undirected graphs to real numbers. Let $G^\star = (V, E^\star)$ denote the target connectivity pattern. Let $\mathcal{G}$ be the set of candidate solutions and let $s(E, V) : \mathcal{G} \longmapsto [0, 1]$ be a scoring function mapping $G \in \mathcal{G}$ into [0,1] and satisfying the following assumptions:

1. $s(E, V) = 1$ iff $E = E^\star$;

2. for every pair (E_1, E_2) of edge sets,
$$|E_1 \cap E^\star| \geq |E_2 \cap E^\star| \Rightarrow s(E_1, V) \geq s(E_2, V)$$

The function $s(G)$ induces a partial order relation over the set of candidate pattern graphs sharing the same vertex set V. In other words, if $s(G) > s(G')$ then $G \succ G'$. Given $s(E, V)$ or an approximation of this function, a pattern can be predicted by a simple procedure enumerating all possible candidates and giving as output one graph with maximal score. The predicted pattern $\tilde{G} = (V, \tilde{E})$ is formally computed as:

$$\tilde{E} = \arg\max_{E \in \mathcal{E}} s(E, V) \quad (1)$$

where $\mathcal{E}$ is the set of possible edge sets satisfying the given perfect matching constraints. It can be easily shown that the function

$$s^\star(E) = \frac{|E \cap E^\star|}{|E|} \quad (2)$$

satisfies the required assumptions, thus can be effectively used in the procedure of Eq.1. The scoring function represents the fraction of correct pairs in the candidate solution. The definition of Eq.2 implies that $s : \mathcal{G} \longmapsto [0, 1]$ is neither injective nor onto [0, 1]. The codomain is the finite discrete set $\{s_0, \ldots, s_{B-2}, s_{B-1}\}$, with $s_i = i/B$, $i = 0 \ldots B - 2$ and $s_{B-1} = 1$. In this case, many different candidates project into the same value, but still we have that $G = G^\star$ if $s_{B-1} = 1$ and the search guarantees to find the target solution.

To analyze the computational complexity involved, first observe that we need to generate the whole set $\mathcal{E}$ of possible solutions. Given a chain with $|V| = 2B = n$ cysteines, the size of this set is

$$(n-1)!! = \prod_{i \leq n/2}(2i - 1) = \frac{n!}{2^{n/2}(n/2)!}$$

By the last equality, it holds that

$$\left(\frac{n}{4}\right)^{n/2} < (n-1)!! < \left(\frac{n}{2}\right)^{n/2}$$

thus $\Omega((\frac{\sqrt{n}}{2})^n)$ steps are necessary to compute $\mathcal{E}$. Each step requires the evaluation of the function s for the current candidate. Assuming s to be linear in $|V|$, it follows that the algorithm takes time at least $\Theta(n(\frac{\sqrt{n}}{2})^n)$. This computational complexity limits the application of the algorithm only to chains with few bridges, but this is not a severe problem, since most of the sequences concentrate in the range of 1-5 disulfide bonds (see Fig. 2). In the following we propose a connectionist model capable of learning $s(E, V)$ from examples of known disulfide bond patterns.

3. RECURSIVE NEURAL NETWORKS

A labeled graph on a set $\mathcal{X}$ is a pair $G = (V, E)$, with $E \subset V \times V$, and with a function $x : V \to \mathcal{X}$ that associates a label x_v to each vertex $v \in V$. The set of all finite size labeled graphs on $\mathcal{X}$ is denoted $\mathcal{X}^\#$. Classification or regression tasks for which the input portion consists of a labeled graph can be formulated as a mapping from $\mathcal{X}^\#$ to a set of categories (classification) or to real numbers (regression). Since graphs have variable size, regression in this case need to be represented as the composition of two functions, a mapping $F : \mathcal{X}^\# \to \Phi$ that transforms input graphs to an intermediate vectorial representation in a feature space Φ, and a mapping $g : \Phi \to \mathbb{R}$ from vectors in feature space to real numbers. Tools for implementing the second function have been well studied and include, for example, neural networks trained on examples of input output pairs. In these cases g also depends on some adjustable parameters θ (e.g. neural network connection weights). The first function, however, requires more attention. Two solutions have been proposed in the context of machine learning. One possibility is to use kernel machines [8]. In this case graphs are mapped into a high-dimensional feature space, whose components are associated with all the possible sub-constituents of the input graph space (i.e. each feature is associated with one particular subgraph in $\mathcal{X}^\#$). The vector-space representation does not necessarily need to be explicitly computed and it may even have infinite dimension, provided that one can compute dot products in feature space. For example, convolutional kernels can be used for this purpose in the case of discrete structures such as strings and labeled trees. A second strategy consists of mapping labeled graphs into a low-dimensional vector space but allowing the mapping F to depend on a second set of adjustable parameters, ϑ. In this case $\Phi = \mathbb{R}^n$. Recursive neural networks (RNN) are one tool for performing this mapping in a way that optimize the parameters ϑ of F together with the parameters θ of g, in a way that globally minimizes an error function. The theory, briefly reviewed here, is relatively simple to develop in the case of directed ordered acyclic graphs (DOAG) with bounded connectivity and that possess a supersource [6]. Ordered in this context refers to the existence of a total order defined on the set of children of each vertex. A supersource r is a vertex having the property that for every other vertex $v \in V$ there exists a directed path from r to v. Under these assumptions, F can be written recursively as follows. For each vertex v, we introduce a label in feature space $\phi_v \in \mathbb{R}^n$ computed as:

$$\phi_v = \begin{cases} 0 & \text{if } \mathrm{ch}_v = \emptyset \quad \text{(Base step)} \\ f(x_v, \phi_{\mathrm{ch}_v^1}, \ldots, \phi_{\mathrm{ch}_v^k}, \vartheta_v) & \text{otherwise} \quad \text{(Induction)} \end{cases} \quad (3)$$

where ch$_v$ denotes the ordered set of children of v, ch$_v^i$ denotes the i-th child of v, and k is the maximum outdegree in the class of graphs being considered. In RNNs, function $f(\cdot)$ is computed by a neural network with connection weights ϑ. The processing is said to be *stationary* if the dependency of weights on v is dropped (this can also be seen as a form of weight sharing or parameter tying). x_v, the label of vertex v, is assumed to be encoded through a real vector of m components. Thus, the network must have $m + kn$ input units and n output units. The computation in Equation 3 proceeds in a bottom-up fashion, from 'leaf' vertices to the supersource. Since we assume that G is acyclic, propagation order can be obtained by sorting topologically the vertex set. The adaptive mapping from graphs to features is simply accomplished by taking the label at the supersource: $F(G) = \phi_r$. It turns out that for each vertex v, vector ϕ_v is the encoding of the subgraph induced by v and all its descendants.

Training is performed using a set of pairs $\{(G_i, y_i), i = 1, \ldots m\}$ where $y_i \in \mathbb{R}$ is the desired output for graph G_i. Parameters θ and ϑ are adjusted by minimizing an error function that, usually, is a mean squared error:

$$C(\theta, \vartheta) = \frac{1}{2} \sum_{i=1}^{m} \Big(y_i - g(F(G_i)) \Big)^2 \qquad (4)$$

As shown in [6], a gradient descent algorithm can be obtained by propagating errors backward in structure (i.e. in a top-down fashion, starting from the supersource and following a reverse topological sort of G).

Graphs describing disulfide connectivity do not perfectly match the above framework since they are disconnected and unordered. However it is not difficult to devise a suitable transformation that converts a disulfide graph $G = (V, E)$ into a DOAG G' having a supersource. First, note that the set of vertices V can be ordered reading the protein sequence from left to right. Edges can be thus oriented so that the source vertex precedes in sequence the target vertex. In this way, G is converted into a directed graph. Moreover, additional sequential edges can be added to connect vertices that are adjacent in sequence. After doing so G is connected and has a supersource. In practice, if we use a model like the one described by Equation 3, the role played by sequential links is to propagate information from left to right. More precisely, the feature vector ϕ_v associated with each half-cystine v would summarize information about all the upstream half-cystines and candidate bridges. Note, however, that dependencies in biological sequences are not unidirectional from left to right. To propagate information in both direction we can duplicate G and transpose the edge set of the copy, as shown in Figure 1. In our implementation, the recursive computation described by Equation 3 is actually piecewise stationary: $\vartheta_v = \vartheta_u$ if and only if v and u are both in the original graph or in the transposed copy. This means that we have three sets of adjustable weights for function F: one for vertices linked upstream to downstream, one for vertices linked downstream to upstream, and a distinguished set of weights for the supersource.

4. DATASETS GENERATION AND INPUTS

In order to effectively compare our method with the existing alternative [3, 4], we extracted the same previously used set of sequences from the SWISS-PROT release no. 39, October 2000 [1] and with an identical filtering procedure. This
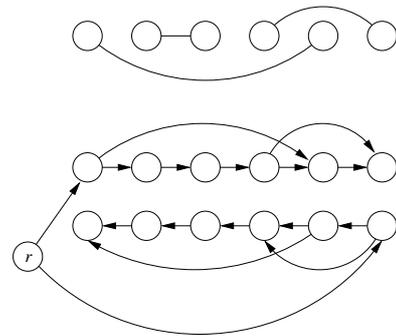

Figure 1: Transforming a disulfide graph (top) into a supersource DOAG (bottom).
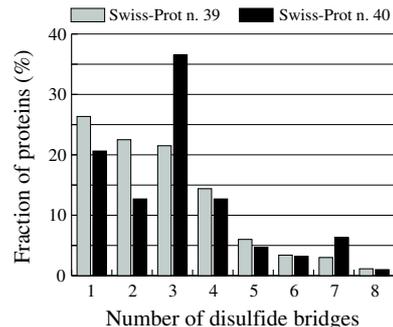

Figure 2: Sequence distribution in the Swiss-Prot data sets. Chains are grouped according to B

assures the presence of only high quality experimentally verified intra-chain disulfide bridge annotations. The resulting data set was splitted in four different subsets with low sequence similarity among each other ($\leq 30\%$) allowing us to perform an identical 4-fold cross validation procedure.

The set of available data was enriched by the new sequences appeared on the SWISS-PROT release no. 40, October 2001 (SP40) and the subsequent new entries introduced on the update of August 2002. This data set was used to further assess the generalization capability of the recursive model over unseen data. We obtained 83 instances from 1 to 24 bonds distributed as in Fig.2. The figure shows a comparison of the distribution of sequences between the new and the old dataset according to number of disulfide bonds, from 1 to 8 bridges. It is evident the difference in frequency distribution, especially in the range from 1 to 3 disulfide bonds, and performance on this new set has to be carefully interpreted.

The available data sets of protein sequences were used as the source for generating instances suitable to be processed by the neural network model. More precisely, for each chain in a set we collected the entire set of possible connectivity patterns. We ended up in a total of 55683 undirected graphs for the set SP39 and 4029 graphs for the updated SP40.

Input to these structures is represented by fixed size numerical labels, each one attached to a node in a graph and representing in a compact form the local environment of a bonded cysteine. In particular, each cysteine vertex was labeled by a 21-dimensional vector. For the first 20 com-

ponents, input in position i encodes for the frequency of appearance of the i-th amino-acid in a fragment of the sequence. The sequence fragment we considered is a window of dimension 21 and centered at the position of the cysteine corresponding to the vertex. Last component of the input vector encodes for the relative distance of the cysteine to the first and last position of the sequence and is given by $j(l-j)/l^2$, where j is the position of the cysteine along the sequence and l is the sequence length.

5. PERFORMANCE MEASURES

Consider the prediction problem on a set $\mathcal{D}$ of proteins and let $G_i = (V_i, E_i)$ and $G_i = (V_i, E_i^\star)$ be respectively the predicted and correct connectivity patterns for the i-th protein in the set $\mathcal{D}$. Model performances can be evaluated using two measures for the quality of predictions:

$$Q_p = \frac{\sum_{i=1}^{|\mathcal{D}|} \delta(E_i, E_i^\star)}{|\mathcal{D}|}, \ Q_c = \frac{\sum_{i=1}^{|\mathcal{D}|} |E_i \cap E_i^\star|}{\sum_{i=1}^{|\mathcal{D}|} |E_i|} \quad (5)$$

where $\delta(x, y) = 1$ if $x = y$ and 0 otherwise. The index Q_p is the fraction of correctly assigned connectivity patterns and estimates the predictive performance at protein level. It is a rather restrictive measure, because predictions are roughly partitioned into the correct and incorrect ones. By this, Q_p can be very small as the number B of bonds increases and it is unable to indicate the quality of a single prediction at a finer level. For this reason, it is reasonable to report, together with Q_p, the index Q_c, defined as the fraction of correctly predicted pairs.

Besides the given indexes, it also makes sense to compare the performances of a model with respect to those of a random predictor. In this case, Q_p and Q_c are represented respectively by the probability of a single pattern (drawn uniformly) and the probability of guessing the correct pair, as given by:

$$Q_p^{rnd} = \frac{1}{\prod_{i \leq B}(2i-1)}, \ Q_c^{rnd} = \frac{1}{2B-1} \quad (6)$$

The dependence of both equations on B indicates that a comparison with a random predictor can only be done grouping the chains according to the number of disulfide bonds.

6. EXPERIMENTS AND RESULTS

Several preliminary experiments were carried out to tune up the prediction system and choose the best architectural parameters. We selected a Bi-Recursive architecture with state vectors of dimension five (both for the forward and backward dynamics) and without hidden layers as a reasonable choice among the speed of operation and the accuracy of the model. In all the experiments we employed the adapted version of Back-Propagation Through Structure [6] for causal and non-causal dynamics as training procedure. We used the online stochastic approximation with the shared network weights updated immediately after the propagation of a single instance. During an epoch, for each chain in the training set the network is given all the candidate connectivity patterns as training instances. During a test phase, predictions are computed using the procedure as given by Eq.1, with the trained network acting as the evaluation function.

B	N.Chains	Q_p	ΔQ_p^{rnd}	Q_c	ΔQ_c^{rnd}
2	39 (35%)	0.61	1.8	0.61	1.8
3	36 (32%)	0.21	3.0	0.32	1.6
4	25 (23%)	0.10	10.5	0.23	1.6
5	11 (10%)	0.02	22.7	0.26	2.3
All	111	0.31		0.35	

Table 1: Bi-Recursive Neural Network based method. Cross-validation results on SP39. Indexes obtained training a model with all training instances of a fold split

B	N.Chains	Q_p	ΔQ_p^{rnd}	Q_c	ΔQ_c^{rnd}
2	39 (35%)	0.68	2.0	0.68	2.0
3	36 (32%)	0.22	3.1	0.37	1.8
4	25 (23%)	0.20	21.0	0.37	2.6
5	11 (10%)	0.02	22.7	0.26	2.3
All	111	0.35		0.42	

Table 2: Feed-forward Neural Network based method. Cross-validation results on SP39

In a first set of experiments, we compared our algorithm with the contact potentials method used in [4], with a similar 4-fold cross validation procedure and on the same data set taken from SP39. We excluded from training and testing phases those instances with a trivial prediction (one disulfide bond) and with non-manageable number of bonds ($B > 5$). Initially, we trained a single BiRNN model with all the training instances in a fold split. Results are as in Table 1 and can be compared with the results of the feed-forward NN based method (Table 2). The 2nd column reports the average number of chains for each different number of disulfide bonds. Averages are computed over all 4 test sets used in the cross validation procedure. Third and fifth columns report the accuracy of prediction as estimated by the indexes Q_p and Q_c (5) and fourth and sixth columns indicate the relative increment over performances of a random predictor ($\Delta Q_p^{rnd} = Q_p/Q_p^{rnd}$ and $\Delta Q_c^{rnd} = Q_c/Q_c^{rnd}$, with Q_p^{rnd} and Q_c^{rnd} given by 6). The first four rows show the predictions grouped according to the number of disulfide bonds, whereas the last one reports the total average number of test chains together with averaged Q_p and Q_c over a whole test set.

Results from tables 1 and 2 show that our recursive model obtains significantly better performances with respect to a random predictor. Nevertheless, it is not able to perform at least the same as the method based on the neural computation of contact potentials, except for the case of 5-bonds chains. One justification could lie in the differences among the various scoring functions, which depend on B, the number of disulfide bonds. Trying to represent all the possible order relations with a unique model introduces high nonlinearities and many local minima, thus limiting the network performance. Besides this, the recursive model uses very poor information as input, compared to that of the alternative method, where the computational structures are given inputs representing the propensity to interact of two cysteines potentially involved in a disulfide bond.

The former explanation can be given a more biological interpretation: the information relating "better" and "worse"

B	N.Chains	Q_p	ΔQ_p^{rnd}	Q_c	ΔQ_c^{rnd}
2	39 (35%)	0.71	2.1	0.71	2.1
3	36 (32%)	0.33	4.7	0.47	2.3
4	25 (23%)	0.10	10.5	0.29	2.0
5	11 (10%)	0.11	124.8	0.33	2.9
All	111	0.39		0.45	

Table 3: Bi-Recursive Neural Network based method. Cross validation results on SP39. Indexes obtained using different models for chains with different B

candidate patterns is hardly exchanged or transferred among the different B-bonds classes of chains. Furthermore, it suggested us to employ an alternative training strategy. The chains were grouped into different sets according to B and we trained and tested one different model for each set. We obtained significantly better results, as shown in Table 3. Each specialized network was able to capture specific commonalities shared by chains of the same type. They outperformed the existing method in all the cases except the class of 4-bonds chains, where we observed an improvement only on the prediction of pairs. Note the increase, both for Q_p and Q_c in the case of 3 and 5 bonds and the significant rate of increment with respect to a random predictor. Globally, the network ensemble increased pattern and pairs prediction respectively by 8 and 10 percentage points.

B	N.Chains	Q_p	ΔQ_p^{rnd}	Q_c	ΔQ_c^{rnd}
2	8 (20%)	0.53	1.7	0.53	1.7
3	22 (53%)	0.31	4.6	0.42	2.1
4	8 (20%)	0.00	0.0	0.16	1.1
5	3 (7%)	0.00	0.0	0.32	2.9
All	41	0.16		0.36	

Table 4: Bi-Recursive Neural Network based method. Results on SP40. Indexes obtained using different models for chains with different B

In a subsequent experiment, the networks of Table 3 were tested on a different set containing the recently added sequences appeared on the SwissProt release no. 40, August 2002 (SP40). We obtained 41 chains in the range from 2 to 5 bonds. This data was used to partially asses the robustness of a learned network function to a change in the test set and to simulate model behaviour in a realistic situation, in which we want to apply the model for the prediction on new data added to a biological sequences archive. For this task, we exploited the ensemble of 16 specialized networks (1 network for each B and for each fold) in the following way. For the B-bonded chains in a test-fold, their patterns are predicted four times using each time one of the B-specialized networks. The obtained Q_p and Q_c are then averaged over the four subsets and taken as the estimated prediction indexes. The results of this procedure, listed in Table 4, deviates from those shown in Table 3 and have to be carefully interpreted in the light of the amount of sequences listed in the 2nd column of each table. We can observe a rather different distribution of chains for the two datasets, especially for $B = 2, 4$. In these cases, the set SP40 contains a small number of test sequences, both in absolute and compared

to SP39. Failing or being successful in predicting only one pattern causes great oscillations both for Q_p and Q_c. This observation is confirmed by the results obtained for $B = 3, 5$. In the 3-bonds case, the number of test sequence approaches to that of SP39 and both indexes are similar to the previously obtained ones. For $B = 5$, only Q_p drops down, partly due to the intrinsic difficulty of choosing the correct pattern among 945 alternatives. The above considerations suggest that the prediction indexes we used are not the best choice for measuring model behaviour on a very small set or at chain level. In this case, validation should be enriched with complementary indexes. For the purpose of obtaining more accurate performance estimates, it could be better to adopt the leave-one-out validation procedure.

7. CONCLUSIONS

We have proposed and tested a novel machine learning method for predicting the disulfide connectivity patterns in cysteine-rich proteins. Performance is comparable or better with respect to other existing algorithms in literature. One obvious direction for further study is to combine cysteine bonding state predictors to a pairing algorithm like the one presented in this paper, in order to build a complete predictor of disulfide bridges. In this perspective, the use of neural networks for solving the pairing problem is potentially advantageous as it allows global optimization of the recursive network together with the parameters of the bonding state predictor.

Disulfide bridges can also be seen as a special (and important) case of residue contact. Therefore it may be important in the future to compare and combine predictors of disulfide bridges with predictors of contact maps whose performance is improving but still appears to be not satisfactory for long ranged interactions.

8. REFERENCES

[1] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL. *Nucleic Acids Res.*, 28:45–48, 2000.

[2] P. Fariselli et al. Role of evolutionary information in predicting the disulfide-bonding state of cysteine in proteins. *Proteins*, 36:340–346, 1999.

[3] P. Fariselli et al. Prediction of disulfide connectivity in proteins. *Bionformatics*, 17:957–964, 2001.

[4] P. Fariselli et al. A neural network-based method for predicting the disulfide connectivity in proteins. In *Proc. 6th Int. Conf. Knowledge Engineering Sys.*, 2002.

[5] A. Fiser and I. Simon. Predicting the oxidation state of cysteines by multiple sequence alignment. *Bionformatics*, 3:251–256, 2000.

[6] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, 9:768–786, 1998.

[7] P. Frasconi, A. Passerini, and A. Vullo. A two stage SVM architecture for predicting the disulfide bonding state of cysteines. In *Proc. 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002.

[8] B. Schoelkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[9] W. Wedemeyer et al. Disulfide bonds and protein-folding. *Biochemistry*, 39:4207–4216, 2000.