A Combination of Support Vector Machines and Bidirectional Recurrent Neural Networks for Protein Secondary Structure Prediction

Alessio Ceroni, Paolo Frasconi, Andrea Passerini, and Alessandro Vullo

Machine Learning and Neural Networks Group
Dipartimento di Sistemi e Informatica
Universitá di Firenze, Italy
Phone: +39 055 4796 361
Fax: +39 055 4796 363
Web: http://www.dsi.unifi.it/neural/
Email: {aceroni,paolo,passerini,vullo}@dsi.unifi.it

Abstract. Predicting the secondary structure of a protein is a main topic in bioinformatics. A reliable predictor is needed for example by threading methods to improve the prediction of tertiary structure. Moreover, the predicted secondary structure content of a protein can be used to assign the protein to a specific folding class and to estimate its function.

We discuss here the use of support vector machines (SVMs) for the prediction of secondary structure. We show the results of a comparative experiment with a previously presented work. We measure the performances of SVMs on a significant non-redundant set of proteins. We present for the first time a direct comparison between SVMs and feed-forward neural netwoks (NNs) for the task of secondary structure prediction. We exploit the use of bidirectional recurrent neural networks (BRNNs) as a filtering method to refine the predictions of the SVM classifier. Finally, we introduce a simple but effective idea to enforce constraints into secondary structure prediction based on finite-state automata (FSA) and Viterbi algorithm.

1 Introduction

Proteins are polypeptide chains carrying out most of the basic functions of life at the molecular level. These linear chains fold in complex 3D structures whose shape is responsible of proteins' behavior. Each ring of the chain consists of one of the 20 amino acid existing in nature. Therefore, a single protein can be represented as a sequence of letters from a 20 elements alphabet called the *primary structure* of the protein. All the amino acids share a common part, formed by a carboxylic acid ($COOH$) and an amino group (NH_2) attached to a carbon atom (commonly referred to as C_α). The carboxylic acid and the amino group of two amino acids can merge into a peptide bond to form the links of the polypeptide chain (see Figure 1). Each amino acid is distinguished by a different R-group, attached to the C_α athom, which entails the amino acid dimensions and chemical properties. For this reason, different sequences will create different folded structures.

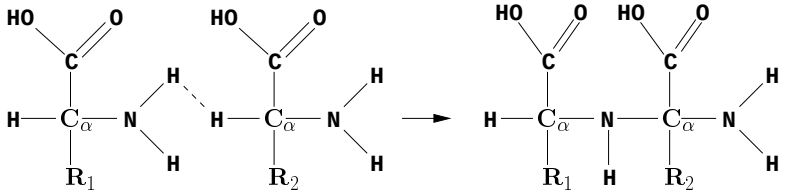

Fig. 1. Two free amino acids with their unspecified R-groups, and the dipeptide formed by their bonding.

Proteins are synthesized inside cells. The instructions used by the cell to build a protein are written in the DNA. DNA's double helix is made by two long chains composed by four different nucleotides. The DNA contains genes, sequences of nucleotides *codifying* for proteins. Each triplet of nucleotides in a gene correspond to an amino acid of the encoded protein: the 64 possible configurations of three nucleotides encode the 20 symbols alphabet of amino acids (it's a redundant code), plus two symbols to identify start and end of the coding sequence. Decoding is performed by specific parts of cell's nucleus: the piece of DNA corresponding to a gene is *transcripted* into mRNA (a disposable chain used to transfer informations inside cell's nucleus) which is in turn *translated* into the protein chain.

All the observed proteins present local regularities in their 3D structure, formed and maintained by hydrogen bonds between atoms. These regular structures are referred to as the protein's *secondary structure*. The most common configurations observed in proteins are called *alpha helices* and *beta strands*, while all the other conformations are referred to as *coils*. They are traditionally identified using a single letter code: H (alpha helix), E (beta strand) and C (coil). An alpha helix is found when two amino acids spaced three positions along the sequence are hydrogen bonded: a sequence of amino acids involved in a alpha helix will form a cork screw like 3D structure (Figure 2a). On the contrary, a beta strand is a straight conformation of amino acids hydrogen bonded to the components of another strand in the protein (Figure 2b), forming a planar aggregation called *beta sheet*. A group of adjacent amino acids sharing the same conformation are members of a *segment* of secondary structure. Segments of secondary structure are well defined and stable aggregations of amino acids which strongly influence the chain's folding and which usually carry out specifical functions inside the protein, like a list of words in a particular language forming a meaningful phrase. From the knowledge of the position of every atom of the protein molecule it is possible to compute the secondary structure at each position in the sequence.

[1].

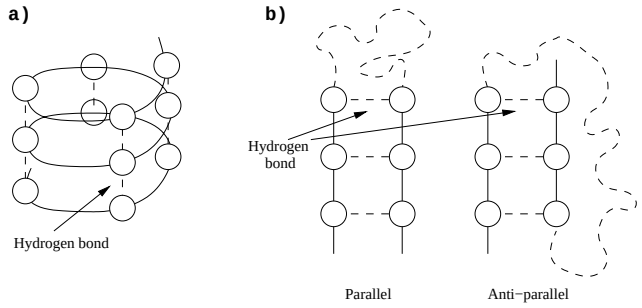

Fig. 2. Conformation of a) alpha helix and b) beta strands along the chain of a protein.

Thanks to several genome sequencing projects, the entire DNA sequence of many organisms has been experimentally determined. Inside each genome the positions of genes have been discovered using specific signals, particular sequences of nucleotides used by cells during transcription. From these identified genes the proteins' primary sequences have been extracted. Unfortunately our knowledge often stops here. The proteins' 3D (*tertiary*) structure, essential to study their functions, remains almost unknown. There exist physical methods to estimate the coordinates of each atoms of a protein, but they need the protein to be crystallized, a time consuming process which, at the moment, is impossible to automate and serialize. Even if the number of proteins whose primary sequence is known counts in the number of millions, only few thousands of them have been crystallized and their 3D structure deposited in the Protein Data Bank [2]. Unfortunately, neither alternative approaches based on nuclear magnetic resonance cannot be applied at the ge-

nomic scale. It is therefore becoming increasingly important to predict protein's tertiary structure *ab initio* from its amino acid sequence, using insights obtained from already known structures.

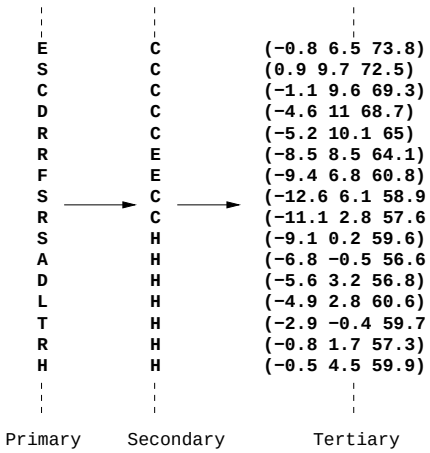

Fig. 3. Primary structure (amino acids), secondary structure (helices, strands and coils) and tertiary structures (C_α coordinates) of a protein.

The tertiary structure of a protein mostly depends on its primary structure and the environment where the protein folds. It is also known that proteins having similar primary structures tend to fold in similar ways. Therefore, the simplest approach to predict the tertiary structure of a query sequence is to align it to a database of known structures using string similarity techniques to search for close parents. This way of deriving proteins' tertiary structures is known as *comparative modeling*. Proteins that cannot be studied with this simple strategy are predicted using *threading* algorithms. These methods estimate the proteins' structure combining small pieces of other proteins which share local substructures (*domains*) with the query chain. Those building blocks are found performing a structure to structure comparison, using an estimation of the protein secondary structure. Therefore, reliable methods to estimate protein's secondary structure are fundamental for tertiary structure prediction. Moreover, the predicted secondary structure content of a protein can be used to identify its folding family [3, 4] and to estimate its functions.

The first attempt to apply machine learning techniques to the prediction of secondary structure [5] employed a standard multi-layer perceptron (MLP) with a single hidden layer, and used as inputs a window of amino acids in one-hot code. The accuracy of this method, measured as the proportion of amino acids correctly assigned to one of the three secondary structure classes (three-state accuracy or Q_3) was well below 70%. Although it is true that primary structure contains all the informations needed for the correct folding of a protein, unfortunately the configuration of every position of the sequence is influenced by the whole content of amino acids of the protein, which can contain thousands of them. Therefore, the same group of linked amino acids can appear in different conformations if its context varies (such misleading patterns are called *chameleons*).

The introduction of evolutionary information expressed by *multiple alignment profiles*, represented a major contribution to the solution of the problem and allowed a significant improvement of the reported accuracy to about 72% [6]. A multiple alignment is a collection of sequences of amino acids from different proteins, realized using a maximum local alignment algorithm [7]. The procedure searches in a large database of known primary structures for all the pieces of sequences similar to part of the protein to predict. The rationale behind this approach is that secondary structure is more conserved than primary structure, therefore similar primary structures will lead to secondary structures that are only slightly different [8]. Once the multiple alignment has been computed, the profile is obtained by counting the frequency of each amino acid at every position

in sequence. This profile is then used instead of the one-hot code as a representation of the amino acids in each position of the sequence.

A major drawback of using a MLP on a window of profiles is given by the relative independence between the predictions of adjacent positions in the sequence. On the contrary, the secondary structure of a protein is defined as a collection of segments composed by many consecutive amino acids. To quantify the capability of a classifier to correctly predict entire segments of secondary structure a measure of Segment OVerlap (SOV) is used [9]. A common approach that can improve both SOV and Q_3 is to add a second structure to structure classifier to filter the predictions of the secondary structure. Jones [10] used neural networks (NNs) for both stages, feeding the filtering network with a window of predictions output by the first stage. Thanks to this solution and to an increasing availability of training data, this architecture achieves the best performances so far with an accuracy of 78% and a SOV of 73.5%. A different approach has been presented by Baldi et al [11] and refined by Pollastri et al [12] which uses bidirectional recurrent neural networks (BRNNs) for secondary structure prediction, trained with profiles as inputs. BRNNs does not suffer of the limitations of a feed-forward neural network classifier, so they do not need a filtering stage. This architecture achieves results equivalent to Jones' work. Lately, Hua and Sun [13] proposed the use of support vector machines (SVMs) for secondary structure prediction. The authors claim the superiority of this model, supported by an high value of SOV without the use of a filtering stage.

Given the work of Hua and Sun, we decided to explore the use of SVMs for the prediction of secondary structure. In section 2 we briefly explain the preparation of the data used during this work. In section 3 we test the use of SVMs for the prediction of secondary structure. The section start with a description of the architecture used. We present here the results of an experiment run to replicate the claims made by Hua and Sun. Then, we apply the algorithm to a bigger and more representative dataset, posing more attentions on model selection. Finally, we compare SVMs and NNs on the same data. In section 4 we explore the use of bidirectional recurrent neural networks as a second stage classifier to filter the predictions of the SVM. This model is briefly explained at the beginning of the section and then experimental results are presented. In section 5 we present a novel method to enforce the prediction with constraints on secondary structure given in the form of a finite-state automaton (FSA). This method uses Viterbi algorithm to align the FSA to the sequence of probabilities output by the predictor, and it proves to be a simple but general method for embedding prior knowledge in the prediction of sequences. Finally, in section 6 we draw some conclusions about the results presented in this work, and we outline future directions of research inspired by these results.

2 Datasets

The first set of experiments is run to replicate the results of Hua and Sun [13]. In their work the authors used the publicly available dataset CB513 released by Cuff and Barton [14]. This dataset is composed by 513 chains with low similarity, so that test results are not biased. A 7-fold cross-validation is adopted to estimate the accuracy of the classifier. Evolutionary information is derived from multiple sequence alignments, obtained from the HSSP database [15]. Secondary structure labels are assigned using the DSSP program [1].

The remaining of the experiments are performed using a significant fraction of the current representative set of non homologous Protein Data Bank chains (PDB Select [16]). We extracted the sequences from the April 2002 release, listing 1779 chains with a percentage of homology lower than 25%. From this set we retained only high quality proteins on which the DSSP program does not crash, determined only by X-ray diffraction, without any physical chain breaks and resolution threshold lower than 2.5 Å. The final dataset contains 969 chains, almost 184,000 amino acids, splitted in a training set of 490 chains, a validation set of 163 chains and a test set of 326 chains. Multiple alignments are generated using PSI-BLAST [17] applied to the Swiss-Prot+TrEMBL non-redundant database [18].

3 Support Vector Machines for Secondary Structure Prediction

The most successful predictors of secondary structure so far employ neural networks as classifiers. Lately, Hua and Sun [13] presented an architecture based on SVMs, claiming the superiority of this model as demonstrated by the high value of SOV reached.

In this section we present our result about the use of SVMs for secondary structure prediction. We show the claimed value of SOV cannot be reached just implementing the classifier with SVMs. We experiment SVMs on a bigger and more representative dataset to further exploit the potentiality of this model. Finally, we perform a direct comparison with neural networks, using the same data for both models, showing that there is no clear superiority of SVMs in this task.

3.1 SVM Classifier

Kernel machines and in particular support vector machines are motivated by Vapnik's principle of structural risk minimization in statistical learning theory [19]. In the simplest case, the SVM training algorithm starts from a vector-based representation of data points and searches a separating hyperplane that has maximum distance from the dataset, a quantity known as the *margin*. More in general, when examples are not linearly separable vectors, the algorithm maps them into a high dimensional space, called *feature space*, where they are almost linearly separable. This is typically achieved via a kernel function that computes the dot product of the images of two examples in the feature space. The decision function associated with an SVM is based on the sign of the distance from the separating hyperplane:

$$f(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) \tag{1}$$

where $\mathbf{x}$ is the input vector, $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the set of support vectors, $K(\cdot, \cdot)$ is the kernel function, and y_i is the class of the i-th support vector (+1 or -1 for positive and negative examples, respectively).

In their standard formulation SVMs output hard decisions rather than conditional probabilities. However, margins can be converted into conditional probabilities in different ways, both in the case of binary classification [20, 21] and in the case of multi-class classification [22]. The method used in this paper extends the algorithm presented by Platt [21], where margins from equation 1 are mapped into conditional probabilities using a logistic function, parameterized by an offset B and a slope A:

$$P(C_i = 1|\mathbf{x}) = \frac{1}{1 + e^{Af(\mathbf{x})+B}}. \tag{2}$$

Parameters A and B are adjusted according to the maximum likelihood principle, assuming a Bernoulli model for the class variable. This is extended here to the multi-class case by assuming a multinomial model and replacing the logistic function by a softmax function [23]. More precisely, assuming Q classes, we train Q binary classifiers, according to the one-against-all output coding strategy. In this way, for each point $\mathbf{x}$, we obtain a vector $[f_1(\mathbf{x}), \cdots, f_Q(\mathbf{x})]$ of margins, that can be transformed into a vector of probabilities using the softmax function:

$$P(C = q|\mathbf{x}) = \frac{e^{A_q f_q(\mathbf{x})+B_q}}{\sum_{r=1}^{Q} e^{A_r f_r(\mathbf{x})+B_r}}, \; q = 1 \ldots Q. \tag{3}$$

The softmax parameters A_q, B_q are determined as follows. First, we introduce a new dataset $\{(f_1(\mathbf{x}_i), \ldots, f_Q(\mathbf{x}_i), \mathbf{z}_i), i = 1, \ldots, m\}$ of examples whose input portion is a vector of Q margins and output portion is a vector $\mathbf{z}$ of indicator variables encoding (in one-hot) one of Q classes. As suggested by Platt for the two classes case, this dataset should be obtained either using a hold-out strategy, or a k-fold cross validation procedure. Second, we perform a search of the parameters A_q and B_q that maximize the log-likelihood function under a multinomial model:

$$\ell = \sum_i \sum_{q=1}^{Q} z_{q,i} \log P(C_i = q|\mathbf{x}) \tag{4}$$

where $z_{q,i} = 1$ if the i-th training example belongs to class q and $z_{q,i} = 0$ otherwise.

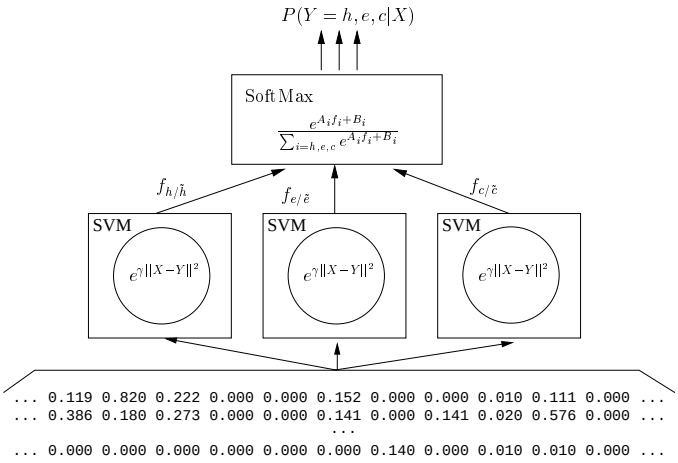

Fig. 4. Architecture of the SVM classifier for secondary structure prediction, composed of three one-against-all SVMs with gaussian kernel combined using a softmax.

3.2 Experiments on CB513

We now run a set of experiments to replicate the results of Hua and Sun [13] on the CB513 dataset. Our secondary structure predictor is constituted by three one-against-all SVM classifiers with gaussian kernel combined using a softmax (Figure 4). We used the same parameters and the same inputs as in [13] in the attempt to replicate their best results, even if we could not retrieve the same lists of proteins for each fold of the cross-validation.

Our experimental results show a significant difference with respect to the SOV obtained in [13]. This evidence supports our belief that the expected value of SOV reached by an SVM predictor

	Q_3	SOV
Our work	73.2	68.5
Hua and Sun	73.5	76.2

Table 1. Results of the experiments on the CB513 dataset.

should not be much different compared to a feed-forward neural-network approach, because both methods are *local*. There is no reason to expect that distinct models trained to predict a single position in the protein sequence and that achieve similar accuracy measured by Q_3 should behave completely different when their performance is measured on segments.

3.3 Experiments on PDB select: SVM vs NN

The CB513 is quite an old dataset which comprises very few proteins if compared to the present size of the Protein Data Bank. Moreover, the criteria used by Cuff and Barton [14] to check for redundancies inside the dataset has been subsequently replaced by different measures. It is then advisable to test the SVM classifier on a more representative dataset to better exploit its

capabilities. Also, in the experiments on the CB513 datasets we used fixed values of the γ parameter for every one-against-all classifiers. This is not advisable, since the optimal value of γ is strongly affected by the patterns to classify. Therefore, we now perform a model search to find the optimal value of γ for each one-against-all classifier at various dimensions of the input window, using a validation set to estimate the error, and fixing the value of C to 1. The model search is performed on a small part of the training set, because it would take too much time otherwise.

Classifier	$w = 9$	$w = 11$	$w = 13$	$w = 15$	$w = 17$	$w = 19$
$H/\bar{H}$	15.7	15.2	15.0	**14.9**	15.3	16.3
$E/\bar{E}$	16.4	16.0	15.7	**15.5**	15.6	15.7
$C/\bar{C}$	23.0	22.8	**22.7**	22.9	23.1	23.1

Table 2. Lowest errors achieved by each one-against-all classifier on the PDB select dataset optimizing the value of γ at various dimensions w of the input window.

The results of the model search (Table 2) show a saturation in the performances of the three classifiers for large windows. SVMs are capable of dealing with highly dimensional data, but they seems unable to use the higher quantity of information contained in such richer inputs. It seems possible that augmenting the input window have the effect of increasing the quantity of noise more than the quantity of information, resulting in poorer performances of the classifier. This can be a possible indication of the fact that even richer models could never exploit the information contained in the whole protein sequence for the prediction of its structure.

¿From the experiments on the CB513 dataset we have seen that SVMs do not guarantee a particularly high SOV. Moreover, these models require computationally expensive procedures for training. Therefore, we want to establish if they are somehow superior to neural networks in the task of secondary structure prediction. In this work we present for the first time a direct comparison between the two models on the same dataset. We use a NN architecture based on the work of Riis and Krog [24], which employed a four layers feed-forward neural network: an input layer where a window of amino acids is fed, a code layer used for adaptively search an encoding of each amino acid, an hidden layer and an output layer where the prediction is taken. The code layer has been introduced to limit the number of weights in the network, therefore preventing overfitting. A different coding is possible because the representation of amino acids with profiles is extremely sparse and also amino acids can be clustered in different overlapping categories according to their chemical properties. The neural networks is then forced to adaptively search an optimal encoding by sharing the weights between every group of 20 neurons in the input layer encoding a single amino acid and the corresponding k neurons in the code layer. Riis and Krog suggested to use $k = 3$ neurons for each amino acid in the code layer. We employ here a single classifier with multiple linear outputs combined using a softmax function to estimate the probability of each of the three secondary structure classes.

Method	Q_3	SOV	Time	Space
SVM	76.5	68.9	3 days	210 Mb
NN	76.7	67.8	2 hours	30 kb

Table 3. Performances of the SVM architecture compared to the NN architecture on the PDB select dataset. Running time and size of the trained model are reported for both architectures.

Given the optimal parameters found, we applied our classifiers to a single split of the dataset as described in section 2. The same input window is used for both the SVM and the NN architecture. The best NN model is searched by varying the size of the hidden layer and using the accuracy on the

validation set as a scoring function. The NN is trained with back-propagation and early stopping to avoid overfitting. The results of the experiments are shown in table 3. Both architectures reach a satisfying value of accuracy, quite comparable to the most recent works, but they suffer from a very low measure of SOV, as we expected for the reasons explained before. The results show no clear advantage of the SVM over the NN architecture. Moreover the SVM architecture have much higher time and space complexity: the time complexity of the training procedure of an SVM is in the order of a quadratic function of the number of examples (the training set we used contains more than 90000 data points), and its model consists of almost 60% of the training examples. Given the very high number of examples and the possibility of using a validation-set, the prediction of secondary structure seems a task fitted for neural networks.

4 Filtering Predictions with Bidirectional Recurrent Neural Networks

Our experiments with SVM and NN architectures confirmed that a local classifier trained on single positions of the sequence cannot achieve a high value of SOV. The SOV is a very important measure to assess the quality of a classifier, since most of the uses of secondary structure predictions rely on the correct assignment of segments. It is then necessary to adopt an architecture which can correlate predictions on adjacent amino acids, to somehow *smooth* the final predicted sequence. In this work we explore the use of bidirectional recurrent neural networks as a filtering stage to refine the predictions of the local classifier.

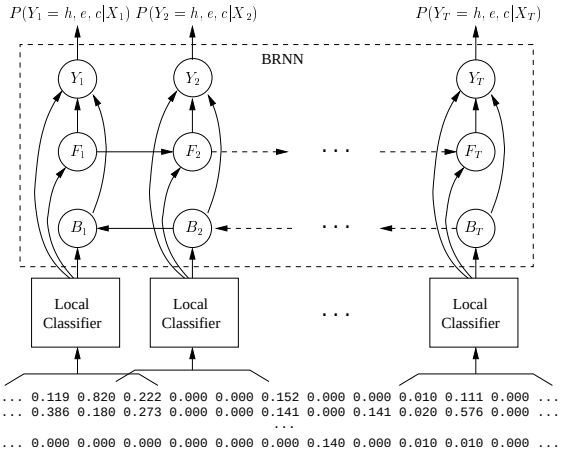

Fig. 5. Two stages architecture. Local classifier can be either SVM based or NN based. The bidirectional recurrent neural network is unfolded over the chain.

BRNNs are recurrent neural networks where two set of states F and B are recursively copied forward and backward along the sequence (Figure 5). BRNNs can develop complex non-linear and non-causal dynamics that can be used to correct output-local prediction by trying to capture valid segments of secondary structure. However, the problem of vanishing gradients [25] prevents from learning global dependencies, so it is impossible for the BRNN to model the whole conformation of the protein. The filtering BRNN have three inputs for each position of the sequence, corresponding to the probabilities calculated by the softmax function in the first stage classifier (Figure 5).

We used early stopping to control overfitting during the training phase. We tested the BRNN on both the predictions of SVM and NN architectures. These experiments (Table 4) clearly show the efficiency of the BRNN when used for filtering the predictions of a local classifier, reaching state-of-the-art accuracy and a very high value of SOV. The performances of this solution are

equivalent to the architecture based on a BRNN with profiles as input [12], even if the filtering BRNN has a much simpler architecture and its easier to train.

	Q_3	SOV
SVM+BRNN	77.9	74.1
NN+BRNN	77.8	74.2

Table 4. Performances of the BRNN used as a filtering stage applied to both the predictions of SVM and NN architectures.

5 Enforcing Constraints using the Viterbi Decoder

A close observation of the outputs of the two stages classifier shows the presence of *inconsistencies* in the predicted sequences. The DSSP program recognize the presence of alpha helices and beta strands from specific patterns in the hydrogen bonds between the amino acids of the protein. These way of labeling the secondary structure of a protein imposes some constraints on observable sequences:

- alpha helix segments must be at least 4 Ålong,
- beta strands must be at least 2 Ålong.

some additional facts enrich the list of constraints:

- a sequence must start and finish with a coil,
- between an alpha helix and a beta strand (and viceversa) there must be a coil.

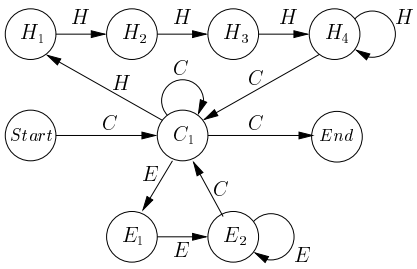

Fig. 6. Finite-state automaton representing every possible allowed sequence of secondary structure.

We present here a simple but effective method to enforce these constraints in the output of a classifier. All the previous facts known about physical chains can be expressed using a finite-state automaton (FSA, Figure 6), which represents every possible allowed sequence in our minimal secondary structure grammar. The outputs of the two stages classifier are the probabilities $P(H|X_t, t)$, $P(E|X_t, t)$ and $P(C|X_t, t)$ that the amino acid in position t of the sequence is in one of the three secondary structure classes, given the input X_t and the position t. We would like our constraints satisfying method to output the best possible sequence from the grammar defined by our FSA, using as scoring function its overall probability as estimated by the classifier:

$$P(Y|I) = \prod_{t=1}^{T} P(y_t|X_t, t)\,,\ Y = \{y_1 y_2 \dots y_T\}\ \ y_t \in \{H, E, C\}. \tag{5}$$

This request strictly resemble problem 2 of hidden Narkov models [26]: we have the probabilities of observations, we have a state model of our data and we want the best sequence of states. A finite-state automaton can be thought of as a degenerated hidden Markov model, where each state emits a single symbol with probability 1 and all the transitions have the same probability (it can be set to 1 because we don't need the probabilities of all the transitions coming out of a state sum to 1). Therefore, we can employ the *Viterbi algorithm* to align our model to the sequence, using the probabilities of observations estimated by the classifier (Algorithm 1). The algorithm searches an optimal path on a trellis whose nodes are (s, t), being s the corresponding state of the FSA and t the position in the sequence. Each node of the trellis has two attached variables: $score(s, t)$ is the score of the best sequence ending at this node, and $last(s, t)$ is the preceding state in the best sequence ending at this node. We define $symbol(s_i, s_j)$ as the symbol emitted during the transition from state s_i to s_j, and $parents(s)$ as the set states which have a transition ending in state s.

Algorithm 1 The Viterbi decoder.

Init the trellis:
for all (s, t) **do**
  $score(s, t) \leftarrow -\infty$
end for

Forward recursion:
$score(start, 0) \leftarrow 0$
for $t = 1$ to T **do**
  for all s_i **do**
    for all $s_j \in parents(s_i)$ **do**
      if $score(s_j, t-1) + \log P(symbol(s_i, s_j)|X_t, t) > score(s_i, t)$ **then**
        $score(s_i, t) \leftarrow score(s_j, t-1) + \log P(symbol(s_i, s_j)|X_t, t)$
        $last(s_i, t) \leftarrow s_j$
      end if
    end for
  end for
end for

Backward recursion:
$previous \leftarrow end$
for $t = T$ to 1 **do**
  $this \leftarrow previous$
  $previous \leftarrow last(this, t)$
  $y_t \leftarrow symbol(previous, this)$
end for
$Y \leftarrow \{y_1 y_2 \dots y_T\}$

We use log-probabilities because of numerical problems. The score of the ending state of the sequence is the log-probability of the best sequence Y. This algorithm can be easily extended to a FSA with more than one starting state and more than one ending state. Moreover, the probabilities used to calculate the scores do not need to come from a particular type of classifier: we can use the predictions of the second stage BRNN, but also the predictions of the first stage SVM and NN.

Target	..CEEEEECCCCCC..	..CHHHHHHC..
Predicted	..CE*H*EEECC*H*C*H*C..	..CH*CC*HH*E*C..
Corrected	..CEEEEECCCCCC..	..CHHHHHHC..

Table 5. Errors corrected by the Viterbi decoder.

A visual observation of the predictions shows the kind of errors the *Viterbi decoder* is able to correct (Table 5). In Table 6 we show the performances of the Viterbi decoder applied to the predictions of our classifiers. The Viterbi decoder can strongly increase the value of SOV of the classifiers, even improving the predictions of the filtering stage, and sometimes it can also improve the overall accuracyQ_3.

	Q_3	SOV
SVM	76.5	68.9
SVM+VD	76.9	73.5
NN	76.7	67.8
NN+VD	77.2	73.6
SVM+BRNN	77.9	74.1
SVM+BRNN+VD	78.0	74.7
NN+BRNN	77.8	74.2
NN+BRNN+VD	78.0	75.2

Table 6. Performances of the Viterbi decoder applied to the various classifiers presented in this paper, compared to their original results.

6 Conclusions

In this paper we have presented a two stages architecture for secondary structure prediction. In section 3 we have explored the use of SVMs for the prediction of secondary structure. We found that SVMs do not guarantee a high value of SOV, contrarily to a recent claim by Hua and Sun [13]. Moreover, we have found that SVMs are not superior to NN for secondary structure prediction, running for the first time an experiment to compare both models on the same data. Given the need of a filtering stage to refine the predictions of the local classifier and increase the value of SOV, we have explored the use of BRNNs for such task. We have demonstrated that a two stages architecture composed by a local classifier, either SVM based or NN based, and a filtering BRNN can reach state-of-the-art performances. Finally, we have introduced a Viterbi decoder to enforce constraints derived from prior knowledge into secondary structure predictions. The Viterbi decoder is capable of finding the best sequence of predictions from a predefined grammar given the probabilities estimated by a classifier. We have demonstrated the Viterbi decoder is able of increasing the value of SOV of our two stages architecture, and to output sequences which are consistent with the constraints on secondary structure.

We have demonstrated that SVMs are not superior to other types of classifier for the problem of predicting secondary structure, as a consequence of the high number of available examples. The superiority of SVMs is given by the possibility of working in high dimensionality spaces defined by kernels. The use of a gaussian kernels does not constitute an improvement over the implementation of NNs with sigmoid activation functions. The capabilities of SVMs would be really exploited if a more complex kernel using richer inputs is implemented. An example could be a kernel running directly on multiple alignments, without the need of calculating profiles which constitutes a loss in information. We have demonstrated the Viterbi decoder is very effective for aligning finite-state automata to sequences of probabilities: it is able of correcting isolated errors, resulting in high values of SOV. However, the Viterbi decoder cannot correct completely misclassified segments of secondary structure. A solution to this problem would require the creation of a richer finite-state automaton, comprising constraints on secondary structure segments, maybe automatically discovered from observed structures.

References

1. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers **22** (1983) 2577–2637
2. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The Protein Data Bank. Nucleic Acids Research **28** (2000) 235–242
3. Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., Thornton, J.M.: CATH - a hierarchic classification of protein domain structures. Structure **5** (1997) 1093–1108
4. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. Journal of Molecular Biology **247** (1995) 563–540
5. Qian, N., Sejnowski, T.J.: Predicting the secondary structure of globular proteins using neural network models. Journal of Molecular Biology **202** (1988) 865–884
6. Rost, B., Sander, C.: Prediction of protein secondary structure at better than 70% accuracy. Journal of Molecular Biology **232** (1993) 584–599
7. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology **147** (1981) 195–197
8. Abagyan, R.A., Batalov, S.: Do aligned sequences share the same fold? Journal of Molecular Biology **273** (1997)
9. Zemla, A., Venclovas, C., Fidelis, K., Rost, B.: A modified definition of SOV, a segment-based measure for protein secondary structure prediction assessment. Proteins **34** (1999) 220–223
10. Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology **292** (1999) 195–202
11. Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., Soda, G.: Exploiting the past and the future in protein secondary structure prediction. Bioinformatics **15** (1999) 937–946
12. Pollastri, G., Przybylski, D., Rost, B., Baldi, P.: Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins **47** (2002) 228–235
13. Hua, S., Sun, Z.: A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. Journal of Molecular Biology **308** (2001) 397–407
14. Cuff, J.A., Barton, G.J.: Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins **34** (1999) 508–519
15. Schneider, R., de Daruvar, A., Sander, C.: The HSSP database of protein structure-sequence alignments. Nucleic Acids Res. **25** (1997) 226–230
16. Hobohm, U., Sander, C.: Enlarged representative set of protein structures. Protein Science **3** (1994) 522–524
17. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. Nucleic Acids Res. **25** (1997) 3389–3402
18. Bairoch, A., Apweiler, R.: The Swiss-Prot protein sequence data bank and its new supplement TrEMBL. Nucleic Acids Research **24** (1996) 21–25
19. Vapnik, V.: Statistical Learning Theory. John Wiley, New York (1998)
20. Kwok, J.: Moderating the outputs of support vector machine classifiers. IEEE Transactions on Neural Networks **10** (1999) 1018–1031
21. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press (1999)
22. Passerini, A., Pontil, M., Frasconi, P.: From margins to probabilities in multiclass learning problems. In van Harmelen, F., ed.: Proc. 15th European Conf. on Artificial Intelligence. (2002)
23. Bridle, J.: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fogelman-Soulie, F., Hérault, J., eds.: Neuro-computing: Algorithms, Architectures, and Applications. Springer-Verlag (1989)
24. Riis, S.K., Krogh, A.: Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. Journal of Computational Biology (1996)
25. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks **5** (1994) 157–166
26. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE **77** (1989) 257–286