STUDIA INFORMATICA 2014
Volume 35 Number 1 (115)

Piotr FABIAN, Katarzyna STĄPOR
Politechnika Śląska, Instytut Informatyki

DICTIONARY SUPPORTED PROTEIN SECONDARY STRUCTURE PREDICTION

Summary. This paper describes a method of predicting the secondary structure of proteins, based on dictionaries of subsequences. These subsequences are derived from records available in the PDB database. Depending on the construction of the learning set, accuracies of up to 79% have been achieved. Dictionaries use hashing functions, which make them fast and capable of storing large sets of substrings.

Keywords: Proteins, secondary structure prediction

PRZEWIDYWANIE STRUKTURY DRUGORZĘDOWEJ BIAŁEK METODĄ SŁOWNIKOWĄ

Streszczenie. W artykule opisano sposób przewidywania struktury drugorzędowej białek, oparty na słownikach podciągów. Sekwencje te są pobierane z danych dostępnych w bazie danych PDB. W zależności od konstrukcji zestawu uczącego, osiągnięto dokładność do 79%. Do szybkiego dostępu do słowników zawierających dużą liczbę podciągów zastosowano funkcje mieszające.

Słowa kluczowe: białka, przewidywanie struktury drugorzędowej

1. Introduction

Proteins are biochemical compounds consisting of one or more polypeptides which are single linear polymer chain of amino acids bonded together by peptide bonds between the carboxyl and amino groups of adjacent **amino acid residues**. To characterize the structural topology of proteins, the concept of protein structure hierarchy with four levels: primary, secondary, tertiary, and quaternary structures have been proposed [17].

The **protein secondary structure** (**pss**) is the characterization of a protein with respect to certain local structural conformations of amino acid sequence (**primary structure**) - like α - helices (**H**), β -sheets (**E**) and others known together as coils (**C**). Protein secondary structure plays an important role in characterizing protein structures as it provides a basis for tertiary structure and function prediction, and many other [17].

The usual goal of **protein secondary structure prediction** (**pssp**) is to classify a pattern of residues in amino acid sequences to one of the predefined protein secondary structures, usually the above mentioned three: *H, E* or *C*. In some applications, more classes of secondary structures are defined - e.g. eight in DSSP [6]. ***D****efine* ***S****econdary* ***S****tructure of* ***P****roteins* is a program developed by Kabsch and Sander to standardize secondary structure assignment. The DSSP algorithm takes as input coordinates of all atoms in a protein. Then, it analyzes the 3-dimensional structure of the compound and assigns a class to each element of the protein chain. The rules used in this assignment are arbitrary (e.g. they should decide, what is the maximum allowed angle in a "sheet" and when to begin treating it as a "helix"). But as DSSP became a standard, we use results of it as our input data, reduced to three classes. Protein secondary structure prediction is now one of the most important problems in **structural biology**.

Since the 1970s many researchers have developed varieties of approaches. Early methods of secondary structure prediction were based on making predictions only on the basis of information coming from a single residue, either in the form of the statistical tendency to appear in an α-helix, β-strand or coil region [1, 3] or in the form of the explicit biological expert rules [8]. However, the accuracy of these methods is much below 70%. Second generation methods apply the connection architecture, taking into account local interactions by means of an input sliding window with encoding. It is generally assumed that 65% (being the average accuracy here) of a secondary structure depends on local interactions [14]. Third generation methods started exploiting the information coming from homologous sequences. Jones [4] first proposed incorporating position-specific scoring matrices (PSSM) obtained using PSI-BLAST program into neural networks. The basic observation is that the secondary structure within a family of evolutionary related proteins is more conserved than the sequences. As a result, most of the state-of-the-art methods, including PHDpsi [12], Porter [11], SVM [18, 10], etc., are all based on PSSM matrix and machine learning methods (classification methods) (see for example [16] for an introduction of classification algorithms). The Q3 accuracy of the newest methods remains at approximately 80% and further improvements are very difficult. The estimated theoretical limit of the accuracy of secondary structure assignment from the experimentally determined 3D structure is 88% of the Q3 accuracy [13] which is deemed the upper bound for secondary structure prediction.

The Q3 measure is a simple measure of similarity of two strings. A more sophisticated measure is SOV, defined by Zemla [19], described later. We use both measures.

This article describes a new approach to predict the secondary structure of proteins, using a dictionary of known structures. In general, dictionary methods use a (large) set of items, which may be words, words with translation, sequences of symbols etc. These methods are used in many domains: text translation (the dictionary contains a number of phrases with translations), speech synthesis, cryptography (the dictionary contains a set of patterns to test as potential passwords etc.). Dictionary methods use a large set of solved problems and try to find a solution in this set. Even if the problem to be solved is not present in the dictionary, the algorithm may find similar problems and use their solutions to generate the answer.

In our method of prediction, we do not analyze neither physical or chemical properties of the compound. The secondary structure prediction is based on similarities to known proteins only. It is assumed, that the local primary structure determines the secondary structure and the influence of more distant amino acids may be ignored. In fact, this property is satisfied only for very long chains of amino acids. And even in such cases there are derogations from this rule. However, taking into account the entire primary structure would be very complex and time consuming and most methods predicting the secondary structure make a similar assumption - only amino acids close to the predicted position are analyzed.

For the dictionary method we assume, that a *sufficiently* long substring of the primary structure also leads to the same secondary structure subsequence. Therefore, in order to predict the secondary structure, we build a dictionary of primary structure substrings and their corresponding secondary structures. The size of the dictionary depends on the available training set. We choose a maximum length $l_{\max}$ of subsequences and fill the dictionary with strings of this fixed maximum length and shorter. In our solution, we have used $l_{\max}=13$ and $l_{\max}=11$. The implementation supports strings of arbitrary length. But introducing longer strings does not improve prediction results. Even a string composed of 11 amino acids is usually found in one protein only in the whole PDB database. A longer string would occur in the same protein only and would not change the result of prediction.

We have solved the problem of storing all such substrings from the whole PDB database and retrieving the information about secondary structure in a reasonable time. Lengths of chains of amino acids making up proteins are expressed in hundreds, and in some cases even thousands. The PDB database contains about 85000 proteins. So, dictionaries are created for tens of millions of subsequences.

Dictionaries contain records composed of two elements: the primary subsequence and corresponding secondary structure for the middle element of the subsequence. If there are many identical subsequences, the information about the number of occurrences of all possible secondary structure classes is stored.

The prediction algorithm tries to match the examined protein against all subsequences in the dictionary, starting with longest strings. If an exact, longest match is found, the corresponding, most frequently occurring secondary structure element is selected. If not, the search continues in shorter strings and in similar strings. Similarity means, that one primary structure element may remain unmatched. Because the dictionary contains strings of the lengths down to one element, the algorithm always finds a matching string. At two ends of the sequence, a padding pattern is added to complete the desired length of strings.

Our implementation was collecting substrings up to 13 elements long. Memory usage for $l_{\max}$=13 is less than 1 GB. We use a string matching function, which finds exact matches and matches with one differing pair of amino acids.

2. Methods and algorithms

Our assumptions are (of course) simplifying the reality and we may encounter several problems applying them to real data. In general, we try to decompose an unknown amino acid sequence into shorter sequences, which possibly may be found in available databases. Then we check secondary structure patterns for these short sequences and build a solution for the whole protein.

Following problems may occur:

1. We assume, that a protein is composed of up to 20 different amino acids, for which standard symbols are used (*A*, *C*, *D*, *E*, *F*, *G*, *H*, *I*, *K*, *L*, *M*, *N*, *P*, *Q*, *R*, *S*, *T*, *V*, *W*, *Y*). However, the PDB database contains proteins with undefined elements - not matching any of these 20. In these cases, we use an additional symbol (*X*) for these undefined elements. Unfortunately, these undefined elements sometimes create longer chains. An extreme case is the protein **1D8S**, composed of 3082 undefined amino acids in the primary sequence but having an observed secondary structure. Other proteins may contain shorter, but still unacceptable long sequences of undefined elements. In these cases, we break the sequence, leave out the undefined part of it and create two subsequences from the remaining parts.

2. This problem is related to the previous one. After leaving out a part of the chain, we of course leave out the corresponding secondary structure too. But this "cutting" does not reflect the true shape of the created subchains. Especially, when there was a helix at the removed subsequence, it may be continued in the second subsequence (that we take into our dictionary). A problem occurs, when there is only one position left with the class "H" assigned. I.e., the second string begins in this case its secondary structure with a helix (at the first position) and then it is changing to another class. It is not possible - the helix is at least three elements long.

3. Short substrings (shorter than about 7 amino acids), often correspond to different secondary structures. So, a dictionary record for a given primary structure subsequence should contain multiple versions of the secondary structure. In practice, we use only three classes for the secondary structure (*H*, *E* and *C*) and it is enough to count, how many times each of these three classes occurred for a given substring.

4. Due to the limited size of the training set, the dictionary of substrings does not contain all possible combinations of amino acids. After limiting the maximum length of the substring to $l_{\max}$, the total number of different substrings of this length is still $20^{l\max}$, and with added shorter sequences approximately 20 times bigger. For example, for six-elements and shorter substrings, the size of the dictionary would be about a billion (10^9) of records. But even using all known proteins from the PDB database, we get only less than 10^8 different six-element substrings. For longer sequences, our dictionary is much smaller than the number of possible sequences. Because of that, it is very probable, that we will not match a substring of a desired length from the analyzed protein in our dictionary. In this case, we try to find shorter matches and agree to non-exact matches.

5. As we do not analyze chemical or physical properties of amino acids, we treat them as "equally different". It means, that in our algorithm a difference at a given position of the primary structure is always treated in the same way with no regard which amino acid is at this position.

Following sections describe all steps of the method we have used.

2.1. The method

The general method for predicting the secondary structure consists of two stages: building the dictionary, shown in Fig. 1 and the prediction stage, shown in Fig. 3. Details are described in subsections.

2.2. Building the dictionary

The dictionary is built on the basis of selected proteins from the PDB database. The entire database contains about 85 thousands of proteins with known spatial structure. Our algorithm may use even the whole database to build the dictionary. In some cases, it results in a dictionary created from about 50 million individual entries.

Input data:

- ProtSet, a set of records (prim. str., sec. str)

Output data:

- *D*, a set of subdictionaries D_1, D_3, …, D_{13} with subsequences

Algorithm:

1. Slide a window of the length 13 through the primary sequence
2. Add a pair (substring of the primary sequence and a corresponding element of the secondary sequence) to D_{13}
3. Store all entries in a hash tables

Fig. 1. The learning stage – building the dictionary
Rys. 1. Faza uczenia – budowanie słownika

The dictionary uses a family of hash functions to place all strings in a number of hash tables. Each hash table creates one subdictionary D_i, $i = 1, 3, 5, \dots, l_{\max}$. The subdictionary D_i contains strings of the length *i*. The size of hash tables depends on *i*. For $i = 1$ it may be even as small as 20; for $i > 3$ we have used hash tables with 1.5 millions entries. Each record placed in a dictionary is composed of two elements: the primary string and a set of three counters counting occurrences of three possible secondary structure classes (*H*, *E*, *C*) for the middle element of the primary string. We have used only odd values for the length of strings; this makes selecting the middle element obvious.

The hash function does not provide unique values and each element of the hash table is capable of holding multiple records, stored in lists. The average number of records in these lists depends on the subdictionary. For shorter strings, lists are shorter. The total number of records stored finally in hash tables is less than the number of strings available in the learning set – repetitions are stored in counters mentioned earlier.

Substrings are generated from the training set using a sliding window. Each chain of the length *n* generates *n* pairs (substring, secondary structure class). For a given length *l* of the window, $\lfloor l/2 \rfloor$ additional neutral symbols "X" are added at the beginning and at the end of the chain. Then, the sliding window is moved from left to right generating pairs. The secondary structure class applies to the middle element in the window. An example is shown in the fig. 2.

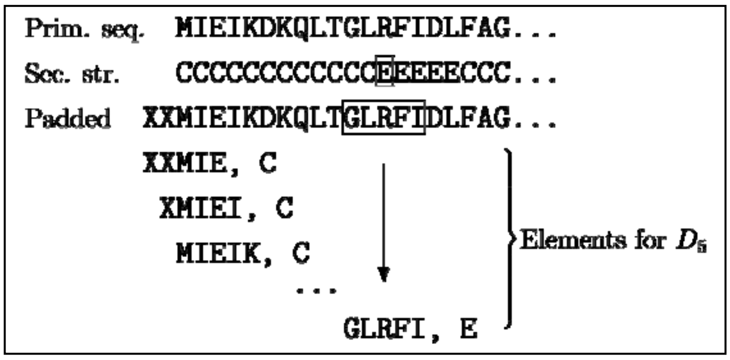
Prim. seq. MIEIKDKQLTGLRFIDLFAG...
Sec. str. CCCCCCCCCCCCEEEEEECCC...
Padded XXMIEIKDKQLTGLRFIDLFAG...
XXMIE, C
XMIEI, C
MIEIK, C
...
GLRFI, E
Elements for D_5

Fig. 2. A sliding window applied to the primary sequence
Rys. 2. Okno przesuwane nad sekwencją pierwszorzędową

2.3. The hash table

The method uses 6 (for $l_{\max} = 11$) or 7 (for $l_{\max} = 13$) hash tables of the size depending on the length of subsequences. For subsequences longer than 3 elements, we use about 1.5 million of entries in the hash table. When filled with the whole PDB database, this gives a load factor of up to 99.2% and average list lengths up to 5. These values are acceptable and provide a short time both for searching and updating the tables. The total amount of memory used for all dictionaries is below 1 GB. In some systems it may be crucial to keep the memory usage below 2 GB, as they don't let to allocate more memory for a single process (e.g. 32-bits versions of MS Windows).

2.4. The prediction stage

The prediction of secondary structure uses information from the dictionary built in the first stage. The general idea is shown in the Fig. 3. Each position of the structure *ps* is analyzed. For each position, subdictionaries D_i, $i = 13, 11, \ldots, 1$ are used to match a substring extracted from *ps*, from the position $ps[k-2i]$ to $ps[k+2i]$. If a match is found, the corresponding best secondary structure class is retrieved from the dictionary. If an exact match is not found, another try is made to find an approximate match with one non-matching position. If not successful, a smaller value of *i* is taken. The last subdictionary D_1 contains all twenty possible elements, so this algorithm always finds a match. The sequence *ps* is additionally padded with a sequence of $\lfloor i/2 \rfloor$ elements "X" at the beginning and the end, which is not shown in the code.

Input data:

- *D*, a dictionary of subsequences and secondary structures; this dictionary consists of many subdictionaries D_1, D_3, …, D_{13} created for subsequences of different lengths
- *ps*, the primary structure to be analyzed or the *structural code*

Output data:

- *ss*, the secondary structure for *ps*

Algorithm:

1. Take the subdictionary for longest subsequences (D_{13})
2. Slide a window of the length equal to the length of entries in the subdictionary, through the *ps*
3. For each alignment of the window, check if such a subsequence exists in the subdictionary
4. If a match was found, take as a result for the *ss* the corresponding element from the dictionary
5. If a match was not found, try again with one "wild character" (one non-matching position in the substring)
6. Repeat steps 2-5 with subdictionaries for shorter subsequences

Fig. 3. The prediction stage of the method
Rys. 3. Faza przewidywania opisywanej metody

2.5. Measures for evaluation of results

The traditional measure of the prediction quality is called Q_3, which is defined as the number of correctly predicted residues divided by the length of the chain. However, it was shown, that the evaluation should be more specific. The SOV measure has been introduced by Rost and Sander [15], modified and described by Zemla in [19]. SOV stands for Segment Overlap Measure. SOV takes into account segments of elements in the same conformation. It compares such segments in the observed structure (which we assume to be true) and the predicted structure (which we evaluate). SOV measures the extent to which two segments overlap and allow to have non-matching residues at end of these segments, without worsening the result. The SOV measure for a single secondary structure class is defined as (1):

$$\mathrm{SOV}(i) = \frac{1}{\mathrm{N}(i)} \cdot \sum_{\mathrm{S}(i)} \frac{\mathrm{minov}(s_1, s_2) + \delta(s_1, s_2)}{\mathrm{maxov}(s_1, s_2)} \cdot \mathrm{len}(s_1) \tag{1}$$

where s_1 and s_2 are the observed and predicted secondary structure segments in state *i* (i.e. *H*, *E* or *C*), len(s_1) is the number of residues in the segment s_1, minov(s_1, s_2) is the length of actual overlap of s_1 and s_2 (the extent for which both segments have residues in state *i*, e.g. *H*, maxov(s_1, s_2) is the length of the total extent for which either of the segments s_1 or s_2 has a residue in state *i*, $\delta(s_1, s_2)$ is an integer value defined as being equal to (2):

$$\delta(s_1, s_2) = \min\big((\mathrm{maxov}(s_1, s_2) - \mathrm{minov}(s_1, s_2)), \\ \mathrm{minov}(s_1, s_2), \lfloor \mathrm{len}(s_1)/2 \rfloor, \lfloor \mathrm{len}(s_2)/2 \rfloor\big). \tag{2}$$

The sum in equation (1) is taken over S, all the pairs of segments s_1*;* s_2, where s_1 and s_2 have at least one residue in state i in common. N(i) is the number of residues in state i defined as in the equation (3):

$$\mathrm{N}(i) = \sum_{\mathrm{S}(i)} \mathrm{len}(s_1) + \sum_{\mathrm{S}'(i)} \mathrm{len}(s_1)\,. \tag{3}$$

These two sums are taken over S and S'; S(i) is the number of all the pairs of segments s_1, s_2, where s_1 and s_2 have at least one residue in state i; in common S'(i) is the number of segments s_1 that do not produce any segment pair.

3. Experiments and results

We have used the dictionary method on several training and testing sets:

- a set of almost all proteins from the PDB database; the input was the primary structure, output: secondary structure,
- the B302 set, containing 302 selected proteins; input and output – as above,
- the same set B302, but the input was so called *structural code*, output: secondary structure.

We have used data from the PDB database, processed by the DSSP algorithm and made available as a set of records with additional DSSP information [5, 6] to train and test the method. It was necessary to adapt this data for our needs. DSSP contains analyzed protein chains with regard to the position of single atoms. Depending on the relative configuration of atoms, eight classes for the secondary structure are defined in DSSP. These eight classes (*G*, *H*, *I*, *E*, *B*, *S*, *T*, -) are mapped into three (*H*, *E*, *C*): *G*, *H*, *I* into *H* (helix), *E*, *B* into *E* (beta turn), *S*, *T*, - into *C*. We have retrieved all available records: about 85000 proteins described in this number of text files (Q3 2012), occupying about 8 GB. The set has been compacted into a 100 MB long file containing primary and secondary structures along with symbols of the proteins and the date of adding the proteins to the database. The set B302 is a set of 302 proteins selected from the whole PDB set to perform tests on secondary structure prediction. The B302 set contains proteins, that are pairwise not similar.

Experiments with the whole PDB database with about 85000 proteins (primary structure → secondary structure): in experiments involving the whole PDB set, we have divided it into a training and testing part, performing a part of a *k*-fold experiment. We also divided the set into two parts according to the date when proteins were added to the database. This simulated a real situation, when an unknown sequence is analyzed using all known proteins. The best Q3 value in these experiments was 79.1%. We expected to get better results when the size of the learning set grows. This trend was visible when proteins from one year were analyzed using all "older" proteins. When the testing set was bigger, results were more random.

Experiments with B302 (primary structure → secondary structure): The next test used the B302 set. In this case we got much worse results (SOV: 33.5%, Q3: 49.9%). This may be caused by the fact, that the dictionary was too small – with too little examples.

BIBLIOGRAPHY

1. Chou P. Y., Fasman G. D.: Conformational parameters for amino acids in helical, beta-sheet, and random coil regions calculated from proteins, Biochemistry, 13(2), 1974, p. 211–222.
2. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: Introduction to Algorithms. MIT Press, 1990.
3. Garnier J., Osguthorpe D. J., Robson B.: Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins, Journal of Molecular Biology 120, Academic Press, 1978, p. 97–120.
4. Jones D. T.: Protein secondary structure prediction based on position-specific scoring matrices, Journal of molecular biology 292(2), Academic Press, 1999, p. 195–202.
5. Joosten R., te Beek T., Krieger E., Hekkelman M., Hooft R., Schneider R., Sander C., Vriend, G.: A series of pdb related databases for everyday needs, Nucleic Acids Research vol. 39, Oxford University Press, 2011, p. D411–D419.
6. Kabsch W., Sander C.: Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features, Biopolymers 22(12), 1983, p. 2577–2637.
7. Kabsch W., Sander C.: A database of secondary structure assignments (and much more) for all protein entries in the protein data bank (pdb), website http://swift.cmbi.ru.nl/gv/dssp/, access: 2012.
8. Lim V.: Algorithms for prediction of α-helical and β-structural regions in globular proteins. Journal of Molecular Biology 88(4), Elsevier, 1974, p. 873–894.
9. Lin H., Sung T., Ho S., Hsu W.: Improving protein secondary structure prediction based on short subsequences with local structure similarity, BMC Genomics 11 Suppl 4, BioMed Central, 2010, p. S4.
10. Nguyen M. N., Rajapakse J. C.: Two-stage multi-class support vector machines to protein secondary structure prediction. Pacific Symposium on Biocomputing, 2005, p. 346–357.

11. Pollastri G., McLysaght A.: Porter: a new, accurate server for protein secondary structure prediction. Bioinformatics 21(8), 2005, p. 1719–1720.
12. Przybylski D., Rost B.: Alignments grow, secondary structure prediction improves. Proteins: Structure, Function, and Genetics 46(2), 2002, p. 197–205.
13. Rost B.: Rising accuracy of protein secondary structure prediction. Dekker, 2003, p. 207–249.
14. Rost B., Sander C.: Prediction of protein secondary structure at better than 70% accuracy. Journal of Molecular Biology 232(2), 1993, p. 584–599.
15. Rost B., Sander C., Schneider R: Redefining the goals of protein secondary structure prediction. Journal of Molecular Biology 235(1), 1994, p. 13–26.
16. Stąpor K.: Metody klasyfikacji obiektów w wizji komputerowej. Wydawnictwo Naukowe PWN, Warszawa 2011.
17. Tramontano A.: Protein structure prediction: concepts and applications, Wiley-VCH, 2006.
18. Ward J. J., McGuffin L. J., Buxton B. F., Jones, D. T.: Secondary structure prediction with support vector machines. Bioinformatics 19(13), 2003, p. 1650–1655.
19. 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, p. 220–223.

Omówienie

W artykule przedstawiono słownikową metodę przewidywania struktury drugorzędowej białek. Struktura drugorzędowa opisuje lokalną, przestrzenną konfigurację łańcucha aminokwasów. Ze względu na dużą złożoność obliczeniową wyznaczania tej konfiguracji bezpośrednio z fizycznych własności cząstek (np. minimalizacji energii całej cząstki), stosowane są metody heurystyczne bazujące na danych uzyskanych doświadczalnie. Opisana tu metoda słownikowa polega na zebraniu najczęściej występujących struktur drugorzędowych dla krótkich ciągów aminokwasów, a następnie zastosowaniu tych danych do wygenerowania struktury drugorzędowej dla nieznanego białka. Słownik zawiera podciągi o różnych długościach, aż do 13 elementów. Wybór konkretnego podciągu jako biorącego udział w generacji ostatecznego wyniku zależy m.in. od częstości występowania tego podciągu w bazie wejściowej. W eksperymentach zastosowano bazę zawierającą około 85000 protein. Przeprowadzono eksperymenty z różnymi podzbiorami protein, uzyskując wyniki do 79% poprawnie przewidzianych elementów struktury drugorzędowej.

Adresy

Piotr FABIAN: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, Piotr.Fabian@polsl.pl.

Katarzyna STĄPOR: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, Katarzyna.Stapor@polsl.pl.