Amino Acids (2012) 42:2195–2205
DOI 10.1007/s00726-011-0959-2

ORIGINAL ARTICLE

# SOMPNN: an efficient non-parametric model for predicting transmembrane helices

**Dong-Jun Yu · Hong-Bin Shen · Jing-Yu Yang**

Received: 28 December 2010 / Accepted: 7 June 2011 / Published online: 22 June 2011


**Abstract** Accurately predicting the transmembrane helices (TMH) in a helical membrane protein is an important but challenging task. Recent researches have demonstrated that statistics-based methods are promising routes to improve the TMH prediction accuracy. However, most of existing TMH predictors are parametric models and they have to make assumptions of several or even hundreds of adjustable parameters based on the underlying probability distribution, which is difficult when no a priori knowledge is available. Besides the performances of these parametric predictors significantly depend on the estimated parameters, some of them need to exploit the entire training dataset in the prediction stage, which will lead to low prediction efficiency and this problem will become even worse when dealing with large-scale dataset. In this paper, we propose a novel SOMPNN model for prediction of TMH that features by minimal parameter assumptions requirement and high computational efficiency. In the SOMPNN model, a self-organizing map (SOM) is used to adaptively learn the helices distribution knowledge hidden in the training data, and then a probabilistic neural network (PNN) is adopted to predict TMH segments based on the knowledge learned by SOM. Experimental results on two benchmark datasets show that the proposed SOMPNN outperforms most existing popular TMH predictors and is promising to be extended to deal with other complicated biological problems. The datasets and the source codes of SOMPNN are available at http://www.csbio.sjtu.edu.cn/bioinf/SOMPNN/.

**Keywords** Membrane protein · Transmembrane helix prediction · Non-parametric model · Self-organizing map · Probabilistic neural network · SOMPNN

D.-J. Yu · J.-Y. Yang
School of Computer Science, Nanjing University of Science and Technology, 200 Xiaolingwei Road, Nanjing 210094, China

H.-B. Shen (✉)
Department of Automation, Shanghai Jiao Tong University, and Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China
e-mail: hbshen@sjtu.edu.cn

## Introduction

Membrane proteins play a vital role in various cellular processes in any living organism, which are encoded by 20–35% of genes (von Heijne 2006; Hu et al. 2007). Membrane embedded alpha helical proteins constitute the majority of ion channels, transporters, and receptors, which account for approximately 40% of all membrane proteins, are infamously difficult targets for high resolution structural studies. Although the last few decades have witnessed major advances in determining high resolution structures of membrane proteins using primarily X-ray crystallography or nuclear magnetic resonance (NMR), they are just the tip of the iceberg, where the total solved membrane protein structures only represent less than 2% of known protein structures in the protein data bank (PDB). Thus, developing powerful computational models is critical for speeding up the process of determining membrane protein structures (Frishman 2010; Gromiha 2010; Shen and Chou 2007b).

Transmembrane helices (TMH) prediction is an important subtask in integral membrane protein structure prediction. Accurately predicting the TMHs in a membrane protein can help to understand its structural and functional characteristics. To date, many TMH predictors have been developed, where the early stage models are mainly based

on calculating amino acid hydrophobic scores or statistical tranmembrane preferences (Wimley and White 1996; Gromiha 1999; Ponnuswamy and Gromiha 1993), whereas the recent ones tend to use statistics-based machine learning techniques to improve the TMH prediction accuracy. For example, TMHMM (Krogh et al. 2001) and PHOBIUS (Kall et al. 2004) are based on hidden Markov model (HMM), evidence theoretical $K$ nearest neighbor algorithm is used in MemBrain (Shen and Chou 2008), and support vector machine (SVM) has also been extensively applied in the literature (Yuan et al. 2004; He et al. 2006). Although much progress has been made in developing statistics-based models for predicting TMHs, challenges remain. One of the difficult things is that most, if not all, of the predictors are based on parametric models, which need to estimate several to even hundreds of adjustable parameters and their qualities significantly affect the final prediction performance. For example, a complex three-stage scheme was applied to optimize the model parameters of HMM in TMHMM (Krogh et al. 2001). Until now, model parameters optimization is still a complicated problem and the rule for obtaining global optimization values is unclear. In addition, some existing predictors require exploiting the entire training dataset in the prediction stage, which will inevitably lead to high computation and storage cost. This disadvantage will be even severe when training space is exponentially increased as more and more structure-determined membrane proteins become available.

Non-parametric model requiring minimal a priori assumption, is a better alternative solution to these problems, which attracts more and more attention (Gibbons and Chakraborti 2003). Probabilistic neural network (PNN) is one of the popular non-parametric machine learning techniques and has been successfully used in many fields for its simplicity and effectiveness (Haykin 1999). However, the main limitation of PNN is that the entire training dataset is needed in the prediction stage, which makes it impractical in bioinformatics fields, in which large size information is common. In order to improve the computational efficiency of PNN for predicting TMHs, we propose a novel SOMPNN model by combining a self-organizing map (SOM) with the PNN model. In the proposed SOMPNN framework, SOM is used to adaptively learn the data distribution knowledge hidden in the training data and then the learned knowledge is encoded in the codebook vectors of the trained SOM. These codebook vectors, rather than the original entire training dataset, are then utilized by a PNN to perform prediction for unlabeled data.

The proposed SOMPNN for TMH prediction has several advantages. First, as a non-parametric model, it does not make any a priori assumptions of the underlying probability distribution and thus has very few parameters to be optimized. Second, it features by high computational efficiency, i.e., the prediction speed is not directly relevant to the size of the training dataset, but to the knowledge learned from the data set. This advantage is due to the learning ability of the SOM model. These two advantages make SOMPNN sufficiently flexible to suite for a wide variety of problems in dealing with large-scale bioinformatics information. Two benchmark datasets are used to demonstrate the prediction performance of SOMPNN in this study. The first one is composed of 120 helical transmembrane proteins that have been widely used by many other popular TMH predictors and the second dataset consists of 126 membrane proteins extracted from the most recent release of Mptopo database. All the experimental results show that SOMPNN outperforms other tested predictors, indicating SOMPNN is a promising algorithm for predicting TMH in membrane proteins.

## Materials

### Benchmark dataset

Two datasets were used in this study. The first one is a benchmark that was used by many other popular predictors, including TMHMM (Krogh et al. 2001), PHOBIUS (Kall et al. 2004), THUMBUP (Zhou and Zhou 2003), and SVMtm (Yuan et al. 2004). It consists of 120 helical transmembrane proteins and is thus referred to as DB120. The training subset, denoted as DB120_Train, and the testing subset, denoted as DB120_Test, includes 50 and 70 helical membrane proteins of known TMH regions, respectively. The Swiss-Prot accession codes of the proteins in DB120_Train and DB120_Test are listed in Tables 1 and 2, respectively.

In order to further demonstrate the efficacy of the proposed SOMPNN model for predicting TMHs, we constructed the other benchmark dataset containing the most recently solved membrane protein structures. This dataset is collected from the Mptopo membrane protein database (http://blanco.biomol.uci.edu/mptopo/), where only the

**Table 1** The Swiss-Prot accession codes of the 50 training proteins in DB120_Train

| | | | | | |
|---|---|---|---|---|---|
| P00396 | P00404 | P00415 | P00423 | P00430 | P00844 |
| P02699 | P02724 | P02945 | P03617 | P03621 | P03805 |
| P03806 | P03989 | P04038 | P06008 | P06009 | P07470 |
| P07471 | P10175 | P13183 | P26789 | P26790 | Q54397 |
| P07371 | O53898 | P77921 | P00804 | P02786 | P02916 |
| P08187 | P08188 | P08716 | P15877 | P17448 | P18783 |
| P19568 | P21865 | P22306 | P22519 | P23462 | P23889 |
| P24282 | P25525 | P25737 | P37310 | Q53068 | P36574 |
| P09130 | P08336 | | | | |

**Table 2** The PDB accession codes of the 70 testing proteins in DB120_Test

| | | | | | |
|---|---|---|---|---|---|
| 1AP9_A | 1AR1_A | 1AT9_A | 1BCC_C | 1EHK_A | 1EYS_L |
| 1EYS_M | 1EYS_H | 1FX8_A | 1IH5_A | 1IWG_A | 1JGJ_A |
| 1KQG_B | 1KQG_C | 1L7V_A | 1LGH_A | 1LGH_B | 1NEK_C |
| 1NEK_D | 1NKZ_A | 1OCC_D | 1OCC_G | 1OCC_J | 1OCC_K |
| 1OCC_L | 1OCC_M | 1OED_A | 1OED_B | 1OED_C | 1OED_E |
| 1OKC_A | 1PRC_M | 1PSS_L | 1PSS_M | 1PV7_A | 1PW4_A |
| 1Q90_D | 1QHJ_A | 1QLB_C | 1RC2_B | 1RHZ_A | 1RWT_A |
| 1SOR_A | 1U7G_A | 1UAZ_A | 1VF5_C | 1VF5_D | 1VGO_A |
| 1XIO_A | 1XQF_A | 1YCE_A | 1ZCD_A | 2A65_A | 2AHZ_A |
| 2B2J_A | 2B5F_A | 2BBJ_A | 2BL2_A | 2BRD_A | 2H8A_A |
| 2HI7_B | 2IRV_A | 2IUB_A | 2J7A_C | 2JO1_A | 2NQ2_A |
| 2ONK_C | 2PNO_A | 2Q7M_A | 2QTS_A | | |

helical membrane proteins with their 3D structures available are considered. To reduce the sequence homology bias, the sequence identity between any two proteins was reduced to less than 60% by using PISCES software (Wang and Dunbrack 2003). Finally, we obtain 126 transmembrane helical proteins that constitute the second benchmark dataset denoted as DB126, which is further randomly partitioned into the training part, denoted as DB126_Train, and the testing part, denoted as DB126_Test, both of which have 63 samples. Tables 3 and 4 show the detailed information of DB126_Train and DB126_Test.

## Construction of training vectors

Sliding window technique is used by most TMH predictors in encoding a residue, i.e., whether a residue belongs to a TMH or not is predicted based on its neighboring residues with a window centered at the residue. A residue can then be further represented by a feature vector, depending on what kind of encoding scheme is applied. Some predictors, e.g. TOP-PRED (Claros and von Heijne 1994), represent a residue based on the hydrophobicity scores of all amino acids in the windowed sequence. Some predictors, e.g. PHDhtm (Rost et al. 1995), use an amino acid composition (AAC) encoding scheme, where a residue is represented in a 21-dimensional vector, within which the first 20 components stand for 20 types of amino acids and the last one represents the break or uncommon amino acid. Recently emerged predictors tend to represent a residue by utilizing the evolution information obtained from multiple sequence alignment technique, which encodes a residue by the position specific scoring matrix (PSSM) scores of its neighboring residues (Shen and Chou 2008; Jones 2007). Because PSSM scheme provides a more complete description of the evolutionary characteristics of a protein sequence and has been demonstrated to be superior to other encoding system for detecting TMH, which is also used in this paper.

For a protein sequence with $n$ residues, its PSSM ($n$ rows and 20 columns) can be generated by using the PSI-BLAST to search against a protein database, such as SWISS-PROT. The PSSM matrix of a protein with $n$ residues is defined as:

$$\mathbf{P}^{\mathbf{pssm}} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,20} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,20} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,20} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,20} \end{pmatrix} \quad (1)$$

where $p_{i,j}$ represents the probability of residue $i$ in the protein being changed to amino acid type $j$. Intrinsically, $\mathbf{P}^{pssm}$ is a numerical description of a given protein sequence.

It is worth pointing out that the original PSSM scores of a protein are often needed to be preprocessed to facilitate the latter computation in specific studies. For example, the scores in a PSSM were normalized to the range between 0 and 1 using a standard logistic function to predict protein secondary structures (Campbell and Kurgan 2008; Mizianty and Kurgan 2011). Ou used PSSM profiles to

**Table 3** The Mptopo accession codes of the 63 training proteins in DB126_Train

| | | | | | |
|---|---|---|---|---|---|
| 3D_helix;104 | 3D_helix;105 | 3D_helix;106 | 3D_helix;107 | 3D_helix;113 | 3D_helix;115 |
| 3D_helix;116 | 3D_helix;117 | 3D_helix;118 | 3D_helix;119 | 3D_helix;120 | 3D_helix;123 |
| 3D_helix;140 | 3D_helix;141 | 3D_helix;142 | 3D_helix;143 | 3D_helix;147 | 3D_helix;151 |
| 3D_helix;167 | 3D_helix;178 | 3D_helix;179 | 3D_helix;181 | 3D_helix;187 | 3D_helix;191 |
| 3D_helix;192 | 3D_helix;199 | 3D_helix;205 | 3D_helix;206 | 3D_helix;208 | 3D_helix;209 |
| 3D_helix;210 | 3D_helix;212 | 3D_helix;214 | 3D_helix;217 | 3D_helix;224 | 3D_helix;226 |
| 3D_helix;228 | 3D_helix;243 | 3D_helix;249 | 3D_helix;261 | 3D_helix;29 | 3D_helix;34 |
| 3D_helix;44 | 3D_helix;45 | 3D_helix;46 | 3D_helix;50 | 3D_helix;53 | 3D_helix;54 |
| 3D_helix;56 | 3D_helix;59 | 3D_helix;60 | 3D_helix;61 | 3D_helix;62 | 3D_helix;66 |
| 3D_helix;67 | 3D_helix;69 | 3D_helix;72 | 3D_helix;79 | 3D_helix;83 | 3D_helix;85 |
| 3D_helix;86 | 3D_helix;93 | 3D_helix;99 | | | |

**Table 4** The Mptopo accession codes of the 63 testing proteins in DB126_Test

| | | | | | |
|---|---|---|---|---|---|
| 3D_helix;100 | 3D_helix;101 | 3D_helix;102 | 3D_helix;103 | 3D_helix;109 | 3D_helix;114 |
| 3D_helix;121 | 3D_helix;122 | 3D_helix;131 | 3D_helix;133 | 3D_helix;134 | 3D_helix;138 |
| 3D_helix;146 | 3D_helix;148 | 3D_helix;149 | 3D_helix;150 | 3D_helix;152 | 3D_helix;153 |
| 3D_helix;162 | 3D_helix;163 | 3D_helix;165 | 3D_helix;166 | 3D_helix;177 | 3D_helix;190 |
| 3D_helix;200 | 3D_helix;201 | 3D_helix;202 | 3D_helix;203 | 3D_helix;204 | 3D_helix;207 |
| 3D_helix;211 | 3D_helix;213 | 3D_helix;222 | 3D_helix;225 | 3D_helix;232 | 3D_helix;233 |
| 3D_helix;235 | 3D_helix;242 | 3D_helix;251 | 3D_helix;255 | 3D_helix;259 | 3D_helix;260 |
| 3D_helix;33 | 3D_helix;42 | 3D_helix;43 | 3D_helix;47 | 3D_helix;48 | 3D_helix;49 |
| 3D_helix;51 | 3D_helix;52 | 3D_helix;55 | 3D_helix;57 | 3D_helix;58 | 3D_helix;63 |
| 3D_helix;64 | 3D_helix;68 | 3D_helix;74 | 3D_helix;75 | 3D_helix;78 | 3D_helix;80 |
| 3D_helix;84 | 3D_helix;92 | 3D_helix;98 | | | |

generate 400D input vector as input features by summing up the same amino acid rows for identifying beta-barrel membrane proteins (Ou et al. 2008). In this study, we take the row-by-row normalization scheme adopted by Shen and Chou (2007a).

After the preprocessing, the feature matrix of the residue $i$ with a sliding window size $W$, $1+\frac{W-1}{2} \le i \le n-\frac{W-1}{2}$, is defined as:

$$\begin{pmatrix} p_{i-\frac{W-1}{2},1} & p_{i-\frac{W-1}{2},2} & \cdots & p_{i-\frac{W-1}{2},20} \\ p_{i-\frac{W-1}{2}+1,1} & p_{i-\frac{W-1}{2}+1,2} & \cdots & p_{i-\frac{W-1}{2}+1,20} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,20} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i+\frac{W-1}{2},1} & p_{i+\frac{W-1}{2},2} & \cdots & p_{i+\frac{W-1}{2},20} \end{pmatrix} \quad (2)$$

where $W$ is an odd number. Then, the feature vector of the residue $i$, denoted as $\mathbf{x}_i$, is obtained by concatenating the elements of the corresponding feature matrix row by row.

The feature vector of a residue is labeled as 'TMH' if its corresponding residue $i$ is a part of a TMH, and is otherwise labeled as 'NOT TMH'. We use '1' to represent 'TMH' and '0' to represent 'NOT TMH', then these labeled feature vectors of all the training proteins constitute the labeled training vector set $T_W$:

$$T_W = \{\mathbf{x}_i^{\omega_i}\}_{i=1}^{N},\ \omega_i \in \{0,1\} \quad (3)$$

These labeled training vectors are then used by a SOM to learn the knowledge of helices distribution. The learned knowledge is then utilized by a PNN to predict the probability of an unlabeled residue being a part of a TMH.

## Methods

### Probabilistic neural network

The probabilistic neural network (PNN) is originated from Parzen's probability density estimator (Specht 1990). Let $X_{\omega_k} = \left\{\mathbf{x}_1^{\omega_k}, \mathbf{x}_2^{\omega_k}, \ldots, \mathbf{x}_{M_{\omega_k}}^{\omega_k}\right\}$ be the training dataset of class $\omega_k$, where $M_{\omega_k}$ be the number of training samples in $X_{\omega_k}$, then the probability density function can be estimated as:

$$p(\mathbf{x}|\omega_k) = \frac{1}{(2\pi)^{d/2}\sigma^2 M_{\omega_k}} \sum_{j=1}^{M_{\omega_k}} \exp\left(-\frac{\left(\mathbf{x}-\mathbf{x}_j^{\omega_k}\right)^T\left(\mathbf{x}-\mathbf{x}_j^{\omega_k}\right)}{2\sigma^2}\right) \quad (4)$$

where $\sigma$ is the smoothing factor.

In the predicting stage, two strategies (optimal classification and probabilistic classification) can be used. When applying the optimal classification strategy, a testing sample $\mathbf{x}_t$ is classified to class $\omega_i$ if

$$p(\mathbf{x}_t|\omega_i) > p(\mathbf{x}_t|\omega_k), \quad \forall \omega_k \neq \omega_i \quad (5)$$

While applying a probabilistic classification strategy, the posterior probability of $\mathbf{x}_t$ belonging to class $\omega_i$ can be estimated as

$$P(\omega_i|\mathbf{x}_t) = \frac{p(\mathbf{x}_t|\omega_i)}{\sum_{k=1}^{C} p(\mathbf{x}_t|\omega_k)} \quad (6)$$

where $C$ is the number of classes.

As a non-parametric method, PNN is simple and works well in many classification problems. However, as PNN utilizes the entire training dataset in prediction stage and thus inevitably leads to huge computation and storage cost. This disadvantage greatly limits the application of PNN when the training dataset is large. To solve this problem, a SOM can be incorporated into PNN to construct the SOMPNN model that can reduce the computation and storage cost while retaining the simplicity and the effectiveness of PNN at the same time (Feng et al. 1998).

### Self-organizing map

The classical self-organizing map (SOM) consists of a regular output grid (usually one- or two-dimensional), onto which the distribution of input vectors is projected nonlinearly (Kohonen 1989). The mapping tends to preserve

the topological-metric relations between input vectors, *i.e.*, similar inputs fed into the SOM will give similar outputs.

A SOM can be represented as a set of output neurons, denoted as $\{i\}_{i=1}^{K}$, on the output layer, where $K$ is the number of neurons on the output layer. Each output neuron $i$ is fully connected to all the input neurons and contains a $d$-dimensional codebook vector $\mathbf{w}_i \in \mathbb{R}^d$ (also referred to as a prototype vector).

A SOM can be trained by iterative sequential or batch learning algorithms. In this paper, the batch learning algorithm is used, as the training dataset is large, to accelerate the training process of the SOM. In a batch learning algorithm, all the training samples are fed to the SOM before any adaptations are made. One round of batch learning is briefly described as follows.

1. The entire training dataset is first partitioned into Voronoi regions $\{V_j\}_{j=1}^{K}$ of the codebook vectors $\{\mathbf{w}_j\}_{j=1}^{K}$. More specifically, sample $\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^{N}$ is partitioned into Voronoi region $V_j$ if the best matching unit (BMU) of $\mathbf{x}$ is the output neuron $j$.
2. Let $n_j$ be the number of samples in the Voronoi region $V_j$, then the average of the samples contained in $V_j$, denoted as $\bar{\mathbf{x}}_j$, can be computed, i.e.

$$\bar{\mathbf{x}}_j = \frac{1}{n_j}\sum_{p=1}^{n_j} \mathbf{x}_p \tag{7}$$

   where $\mathbf{x}_p \in V_j, 1 \le p \le n_j$.
3. The codebook vectors are updated by using a simple linear weighted average scheme as follows

$$\mathbf{w}_i(t+1) = \frac{\sum_{j=1}^{K} h_{ji}(t) \cdot n_j \cdot \bar{\mathbf{x}}_j}{\sum_{j=1}^{K} h_{ji}(t) \cdot n_j} \tag{8}$$

The batch learning algorithm can effectively accelerate the leaning process and lessen the impact of the order of the sample data fed to the SOM. When the size of the training dataset is large, batch learning is preferred.

In this paper, the labeled training vectors, $T_W$ defined in Eq. (3), are used to learn a SOM. In addition, the majority voting mechanism is used to label each of the output neurons in the SOM from its corresponding BMUs in the training dataset.

Figure 1 shows the labeled trained SOM with $W = 13$ on DB120_Train, in which '0' represents 'NOT TMH' and '1' represents 'TMH'. There are 33 × 18 output neurons which are arranged in hexagonal lattice structure. From Fig. 1, it can be found that the training vectors in high-dimensional input space have been successfully mapped onto the two-dimensional output space of the SOM. Specifically, the SOM has learned to represent these high dimensional training vectors in the low two-dimensional space. In addition, the SOM can well separate classes ('0' or '1') of these training vectors, as shown in Fig. 1, which means the SOM has successfully learned the distribution characteristics (knowledge) of the helices hidden in the training vectors. Note that the learned knowledge, which will be further utilized by a PNN, is encoded in the codebook vectors of the trained SOM.

## SOMPNN

Let $X = X_{\omega_1} \cup X_{\omega_2} \cup \cdots \cup X_{\omega_C}$ be the training dataset, where $C$ is the number of classes, $X_{\omega_k} = \{\mathbf{x}_1^{\omega_k}, \mathbf{x}_2^{\omega_k}, \cdots, \mathbf{x}_{M_{\omega_k}}^{\omega_k}\}$ are the $M_{\omega_k}$ training samples in class $\omega_k$ $(1 \le k \le C)$. First, data set $X$ is used to train a SOM. After training process of a SOM, each output node of the trained SOM is labeled as class $\omega_1$,$\omega_2$,..., or $\omega_C$ based on the labeled training samples in $X$. Note that in this study, $C = 2$, i.e., there are two classes of residues, where $\omega_1 = 0$ and $\omega_2 = 1$ represent a 'NOT TMH' and a 'TMH' residue, respectively.

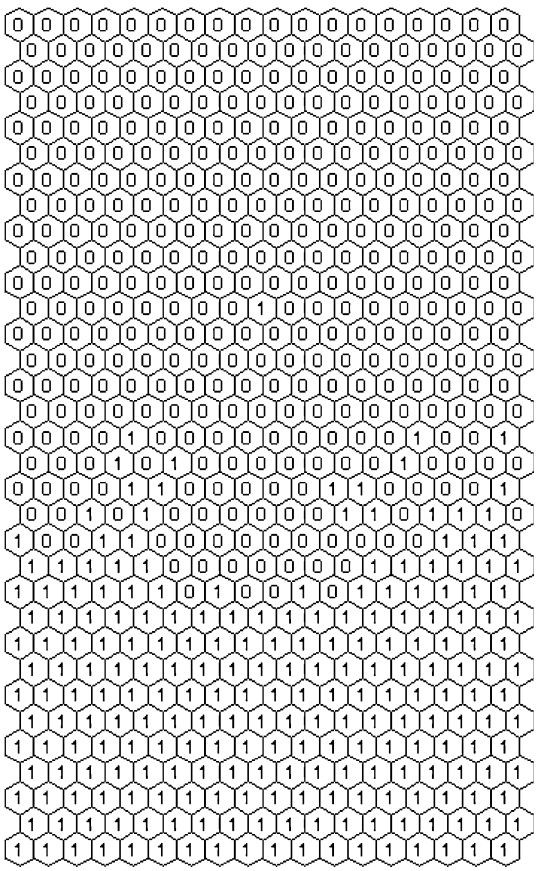

**Fig. 1** The labeled trained SOM with 33 × 18 output neurons arranged in hexagonal lattice structure for $W = 13$ on DB120_Train dataset

In a classic PNN, when estimating the probability density function of class $\omega_k$, all the training samples belonging to class $\omega_k$ are utilized. This computation procedure is inefficient and time-consuming when training set is large. To speed up the computation and reduce the noises in the training data, the codebook vectors of the trained SOM, rather than the original training samples, are used by a PNN to execute prediction task. In detail, suppose that the trained SOM has $N_{\omega_k}$ output nodes which are labeled as class $\omega_k$, $1 \le k \le C$. The codebook vectors of these $N_{\omega_k}$ nodes are $\mathbf{w}_j^{\omega_k}$, $1 \le j \le N_{\omega_k}$. In a SOMPNN, these $N_{\omega_k}$ codebook vectors, rather than the original $M_{\omega_k}$ training samples of class $\omega_k$ [as in Eq. (4)], are used to approximately estimate the probability density function of class $\omega_k$ as follows:

$$p(\mathbf{x}|\omega_k) = \frac{1}{(2\pi)^{d/2}\sigma^2 N_{\omega_k}} \sum_{j=1}^{N_{\omega_k}} \exp\left( -\frac{\left(\mathbf{x} - \mathbf{x}_j^{\omega_k}\right)^T \left(\mathbf{x} - \mathbf{x}_j^{\omega_k}\right)}{2\sigma^2} \right) \quad (9)$$

The posterior probability of a testing sample $\mathbf{x}_t$ belonging to class $\omega_k$, denoted as $P(\omega_i|\mathbf{x}_t)$, can be further obtained by using Eq. (6). As $N_{\omega_k} \ll M_{\omega_k}$, the computation cost of Eq. (4) can be greatly reduced by using Eq. (9). Take DB120 dataset for an example, there are 14,531 training vectors obtained from DB120_Train with $W = 13$. By using SOMPNN, only 594 ($33 \times 18$) codebook vectors of the SOM, rather that the original 14,531 training vectors, are used for prediction.

In the prediction stage, a testing protein sequence's PSSM as described in Eq. (1) is first computed by applying PSI-BLAST. To predict the probability of residue $i$ in the testing protein for being a part of a TMH, a testing local feature vector of residue $i$, denoted as $\mathbf{x}_i (\mathbf{x}_i \in \mathbb{R}^d, d = W \times 20)$, is built. Then, $\mathbf{x}_i$ is fed into the SOMPNN to estimate the probability of being a part of a TMH. It is worth pointing out that in the prediction stage, the local feature vector of a residue together with the knowledge (codebook vectors) learned by SOM are fed into the PNN to estimate the probability. There is no need to feed the local feature vector of a residue to the trained SOM again. After the probability of each residue has been estimated, the curve of residue-specific TMH probability of the testing protein can be obtained, as shown in Fig. 2. TMHs can then be further determined from this probability curve by setting an appropriate initial threshold and applying dynamic threshold segmentation (Shen and Chou 2008).

Figure 2 illustrates the workflow of the SOMPNN for TMH Prediction. In the training stage, the labeled training vectors constructed from the PSSMs of the training proteins are used to train a SOM until it converges. In the prediction stage, for a given testing protein, its PSSM is computed first. Then, the feature vector of each residue can be extracted. The probability of a residue being a part of a TMH can be estimated by feeding its feature vector and the codebook vectors (knowledge) of the trained SOM to a PNN.

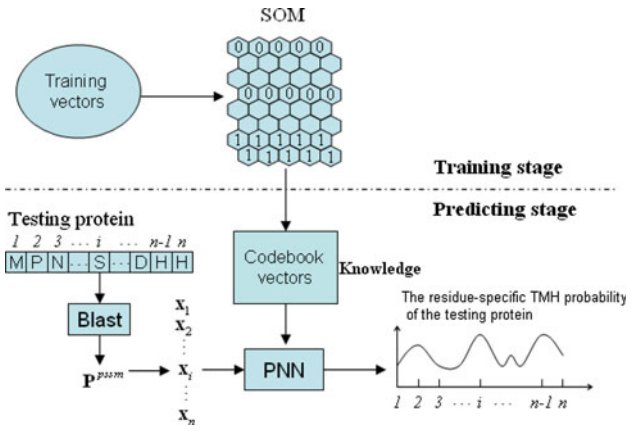


**Fig. 2** Workflow of a SOMPNN model for predicting TMH from amino acid sequence

## Results and discussions

To compare the performance of our predictor with other existing predictors, five widely used measures of accuracy are taken (Cuthbertson et al. 2005; Shen and Chou 2008).

$V_{\text{tmh}}$ is used to measure the TMH prediction success rate. $V_{\text{tmh}}$ measures the percentage of the observed TMHs in the testing dataset that have been correctly predicted. An observed TMH is considered to be predicted correctly if it has an overlap of at least nine residues with the prediction.

$V_{\text{p}}$ represents the protein prediction success rate, and measures the percentage of the proteins in the testing dataset that have been correctly predicted. A protein is considered predicted correctly if all the TMHs in this protein are correctly predicted and the number of predicted TMHs is equal to the observed number of TMHs in the protein.

To measure accuracy in predicting the location of the ends of the TMHs, the $N_{\text{score}}$ and $C_{\text{score}}$ are used. $N_{\text{score}}$ and $C_{\text{score}}$ are the number of N-terminal and C-terminal residues that do not match when aligning the predicted and observed TMHs, respectively.

Finally, the normalized RMSD is used to measure the distance between the observed and predicted TMH representation vectors. The TMH representation vector of an observed or a predicted protein is in the form of $\mathbf{p} = (p_1, p_2, \ldots, p_i, \ldots, p_n)$, in which $p_i$ is assigned to 1 if residue $i$ is a part of a TMH and is otherwise assigned to 0. *RMSD* is defined as:

**Table 5** The performance comparison between PNN and SOMPNN on DB120 and DB126 datasets

| Dataset | Model | $V_{tmh}$ | $V_p$ | $N_{score}$ | $C_{score}$ | *RMSD* | Prediction Time (s) |
|---|---|---|---|---|---|---|---|
| DB120 | PNN | 0.955 | 0.786 | 3.42 ± 3.74 | 3.51 ± 5.01 | 0.49 ± 0.15 | 8231 |
| | SOMPNN | 0.984 | 0.871 | 3.28 ± 3.73 | 3.23 ± 3.44 | 0.49 ± 0.12 | 161 |
| DB126 | PNN | 0.970 | 0.698 | 3.54 ± 4.35 | 3.75 ± 5.21 | 0.55 ± 0.18 | 5942 |
| | SOMPNN | 0.974 | 0.873 | 3.44 ± 3.37 | 3.67 ± 3.74 | 0.53 ± 0.12 | 113 |

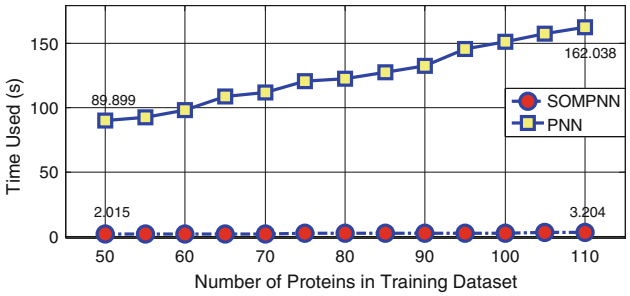


**Fig. 3** Speed performance comparison between PNN and SOMPNN for predicting TMHs in protein 1OED_A when the training samples are increased from 50 to 110

$$RMSD = \frac{\|\mathbf{p} - \mathbf{p}^0\|}{\|\mathbf{p}^0\|} = \frac{\sqrt{\sum_{i=1}^{n} (p_i - p_i^0)^2}}{\sqrt{\sum_{i=1}^{n} (p_i^0)^2}} \quad (10)$$

where $\mathbf{p}$ and $\mathbf{p}^0$ are the predicted and observed TMH representation vectors of a protein, respectively.

In all the tests, the configuration and parameters of the SOMPNN are fixed on both datasets to fairly compare the performance of the SOMPNN with other predictors. Specifically, a SOM with 33 × 18 output neurons arranged in hexagonal lattice structure (Fig. 1) was used, which was automatically determined by the standard batch learning algorithm (Kohonen 1989) provided in the SOM toolbox package, which is freely available at http://www.cis.hut.fi/somtoolbox/. The only smoothing factor parameter $\sigma$ in PNN is set to be 3.0. The slide window length $W$ is set to 13 in all experiments after tests on several other candidates.

## Comparison between PNN and SOMPNN

As stated in the aforementioned section, the purpose of incorporating a SOM module into PNN is to enhance its online prediction efficiency. Because training a SOM is very fast and can be conducted offline, thus we mainly compare the computational complexity in the prediction stage between PNN and SOMPNN. Let $M_0$ be the number of training vectors labeled as '0' ('NOT TMH'), and $M_1$ be the number of training vectors labeled as '1' ('TMH'), then the computational complexity of PNN in the prediction stage is $O((M_0 + M_1) \cdot d)$, where $d = W \times 20$ is the dimensionality of the training vector. While in a SOMPNN, the computational complexity in the prediction stage is $O((N_0 + N_1) \cdot d)$, where $N_0$ and $N_1$ are the numbers of output neurons of a trained SOM labeled as '0' and '1', respectively. As $N_0 \ll M_0$ and $N_1 \ll M_1$, hence the prediction efficiency of SOMPNN can be significantly enhanced over PNN. At the same time, the storage requirement is greatly reduced.

Table 5 illustrates the detailed quantitative comparison results between PNN and SOMPNN on the two benchmark datasets. The calculation configuration is Lenovo T400 laptop with Intel Core(TM) 2 Duo CPU and 1 GB memory. From Table 5, we can find that PNN needs 8,231 s (2.29 h) to predict the 70 proteins in DB120_Test and 5,942 s (1.65 h) for the 63 proteins in DB126_Test; whereas only 161 and 113 s are needed by SOMPNN for the two datasets, respectively, which means the computational efficiency has been improved by more than 50 times! The main reason for the low computational efficiency of PNN is it needs to explore all the training samples when predicting an unlabeled target and the sizes of training data in DB120_Train and DB126_Train are 14,531 and 14,135 respectively, where the large sizes come from the process of representing each amino acid residue in the sequence as a sample. It can be imagined that with more high-resolution membrane protein 3D structures being solved and added into the training knowledge, the size of training set will be exponentially increased accordingly, which can cause PNN extremely slow for dealing with post-genomic large-scale datasets. In order to further demonstrate this phenomenon. Figure 3 illustrates the speed performance comparison between PNN and SOMPNN, where the number of training proteins is gradually improved from 50 to 110 with a step size 5 and only one testing target of 1OED_A is applied. As clearly shown by Fig. 3, when there are 50 proteins in the training dataset, PNN needs 89.899 s to predict the TMHs in 1OED_A, however, SOMPNN only needs 2.015 s. With the size of training proteins is increased from 50 to 110, the prediction time of PNN is increased by 72.139 s, while only 1.189 s improvement was observed in SOMPNN, indicating the increasing rate of the prediction time of PNN is much higher than that of SOMPNN. Referring to Table 5, the prediction accuracy is also improved in SOMPNN on both datasets. The reasons for the improved performance of SOMPNN can be due to: (1) the codebook

vectors of the trained SOM encode the underlying data distribution knowledge hidden in the original training dataset; and (2) data in the training dataset is not noise free and if PNN is directly used, the prediction accuracy may be degraded by the noise (as PNN uses each sample in the training dataset). Thus, compared with the original training dataset, these codebook vectors that represent the underlying data distribution knowledge and are noise reduced, are more suitable to build a powerful PNN predictor.

To further demonstrate that the proposed SOMPNN is not fitted into the two specific testing sets, i.e., DB120_Test and DB126_Test, we also carried out fivefold cross-validation on the two benchmark datasets. Taking DB126 as an example, we first randomly partition 126 proteins into 5 groups; then in each cross-validation, 4 groups are used as training dataset and the retaining 1 group is used as testing dataset. Table 6 illustrates the fivefold cross-validation results of SOMPNN on DB126, which shows the good generalization ability of SOMPNN.

**Table 6** Fivefold cross-validation results of SOMPNN on DB126

| 5-fold cross-validation | $V_{tmh}$ | $V_p$ | $N_{score}$ | $C_{score}$ | *RMSD* |
|---|---|---|---|---|---|
| 1 | 0.985 | 0.885 | 3.03 ± 3.03 | 3.27 ± 3.39 | 0.49 ± 0.09 |
| 2 | 0.979 | 0.846 | 3.45 ± 3.43 | 4.28 ± 4.30 | 0.53 ± 0.12 |
| 3 | 0.990 | 0.923 | 3.94 ± 3.48 | 3.65 ± 3.55 | 0.54 ± 0.12 |
| 4 | 0.965 | 0.846 | 3.50 ± 3.08 | 4.28 ± 4.16 | 0.57 ± 0.12 |
| 5 | 0.928 | 0.682 | 4.09 ± 4.51 | 3.89 ± 4.25 | 0.54 ± 0.16 |
| Average | 0.969 | 0.836 | 3.60 ± 3.50 | 3.87 ± 3.93 | 0.53 ± 0.12 |

## Comparison with other popular TMH predictors

Tables 7 and 8 show the performance comparison of SOMPNN and several other popular TMH predictors on the two benchmark datasets, respectively. From the two tables, it is found that SOMPNN significantly outperforms all the tested predictors in terms of $V_{tmh}$ and $V_p$. And as for $N_{score}$, $C_{score}$, and *RMSD* measures, SOMPNN achieves the very comparable results with the MemBrain and outperforms all the others. Take the results obtained on DB120 as illustrated in Table 7 for example, the $V_{tmh}$ and $V_p$of SOMPNN are 98.4 and 87.1%, respectively, which are the best in all the listed predictors. The *RMSD* of SOMPNN is 0.49, which is very close to 0.48 of MemBrain and is approximately 14% better than the second-best performer among the tested predictors. In terms of the ability of predicting the ends of TMHs ($N_{score}$ and $C_{score}$), SOMPNN is a little worse than MemBrain but still remarkably better than other predictors.

## Combining non-parametric with parametric models

Developing a single independent TMH predictor that performs consistently the best for all situations is almost

**Table 7** The performance comparison of different TMH predictors on DB120

| TMH predictors | $V_{tmh}$ | $V_p$ | $N_{score}$ | $C_{score}$ | *RMSD* |
|---|---|---|---|---|---|
| THUMBUP | 0.854 | 0.486 | 7.33 ± 5.42 | 6.88 ± 5.36 | 0.69 ± 0.11 |
| SOSUI | 0.892 | 0.571 | 5.15 ± 4.42 | 5.27 ± 4.73 | 0.60 ± 0.17 |
| DAS-TMfilter | 0.907 | 0.657 | 6.52 ± 5.08 | 5.62 ± 5.58 | 0.65 ± 0.13 |
| TOP-PRED | 0.926 | 0.600 | 4.64 ± 4.04 | 4.66 ± 4.16 | 0.61 ± 0.15 |
| TMHMM | 0.910 | 0.657 | 4.71 ± 4.26 | 4.56 ± 4.17 | 0.57 ± 0.13 |
| PHOBIUS | 0.918 | 0.700 | 4.73 ± 4.29 | 4.42 ± 4.24 | 0.58 ± 0.14 |
| MemBrain | 0.979 | 0.857 | 3.19 ± 3.01 | 3.03 ± 2.74 | 0.48 ± 0.13 |
| SOMPNN | 0.984 | 0.871 | 3.28 ± 3.73 | 3.23 ± 3.44 | 0.49 ± 0.12 |
| SOMPNN + MemBrain | 0.981 | 0.871 | 2.96 ± 3.10 | 2.91 ± 2.73 | 0.46 ± 0.12 |

**Table 8** The performance comparison of different TMH predictors on DB126

| TMH predictors | $V_{tmh}$ | $V_p$ | $N_{score}$ | $C_{score}$ | *RMSD* |
|---|---|---|---|---|---|
| SOSUI | 0.858 | 0.651 | 5.27 ± 4.57 | 5.13 ± 4.65 | 0.59 ± 0.18 |
| DAS-TMfilter | 0.862 | 0.587 | 6.81 ± 6.11 | 6.62 ± 7.25 | 0.65 ± 0.15 |
| TOP-PRED | 0.903 | 0.556 | 5.05 ± 4.24 | 4.65 ± 3.97 | 0.62 ± 0.15 |
| TMHMM | 0.903 | 0.667 | 4.77 ± 3.91 | 4.45 ± 4.11 | 0.57 ± 0.16 |
| PHOBIUS | 0.903 | 0.683 | 4.68 ± 4.09 | 4.57 ± 4.21 | 0.57 ± 0.16 |
| MemBrain | 0.966 | 0.778 | 3.10 ± 3.40 | 3.61 ± 3.31 | 0.50 ± 0.16 |
| SOMPNN | 0.974 | 0.873 | 3.44 ± 3.37 | 3.67 ± 3.74 | 0.53 ± 0.12 |
| SOMPNN + MemBrain | 0.966 | 0.825 | 3.12 ± 3.39 | 3.43 ± 3.28 | 0.49 ± 0.14 |

impossible considering different prediction models have their own advantages and disadvantages at the same time (Chen et al. 2002). For example, Zhang et al.'s (2011) work also shows that there is no universally best predictor for the protein secondary structure prediction. Much work has been done in the literature to try to further improve the predictor's accuracy and reliability. For example, some researchers have found that fusing different but complementary features is a promising direction (Shen et al. 2009; Song et al. 2009). Remarkable efforts have been devoted to develop powerful ensemble predictors in the bioinformatics literature, which have been demonstrated as a promising route to deal with complicated biological data in many fields (Asur et al. 2007; Ishida and Kinoshita 2008; Martelli et al. 2003; Nanni and Lumini 2006; Song et al. 2007; Yanover et al. 2009; Mizianty and Kurgan 2009; Klammer et al. 2009; Chen and Kurgan 2007; Plewczynski 2010; Zhang et al. 2011).

Thus, in this paper, we also investigate the complementary characteristics of combining the non-parametric with parametric models for predicting TMHs by ensembling SOMPNN with another parametric predictor MemBrain (Shen and Chou 2008). In the fused predictor (SOMPNN + MemBrain), the probability of a residue for being a part of a TMH is obtained by fusing the probabilities predicted by SOMPNN and MemBrain with a simple average scheme. The results of SOMPNN + MemBrain on the two benchmark datasets are illustrated in the last rows of Tables 7 and 8 respectively, according to which the fused predictor outperforms all the state of the art predictors on the two benchmark datasets. Perhaps the most valuable achievement of the SOMPNN + MemBrain is the improved ability of predicting the ends of TMHs. For example, the $N_{score}$ of the fused predictor on DB126 is 2.96, which is about 7% better than that of MemBrain and 10% better than that of SOMPNN. While in terms of $C_{score}$, the fused predictor is about 4% better than MemBrain and 10% better than SOMPNN. The performance improvement obtained by SOMPNN + MemBrain indicates that it could be a promising direction to develop ensemble TMH predictor by fusing non-parametric and parametric statistical models.

### Case study

A most recent solved membrane protein structure of the spinach minor light-harvesting complex CP29 (PDB code: 3PL9) that has three TMHs is used as the case study for comparing different TMH predictors (Pan et al. 2011). Figure 4 gives the TMH probability curve of 3PL9 predicted by SOMPNN, and the predicted three TMHs are shown as the gray boxes. We also tested 3PL9 on several other popular predictors and the results are listed in Table 9. From Table 9, we can find that TMHMM missed all the three TMHs in 3PL9, both SOSUI and DAS-TMfilter missed two TMHs, PHOBIUS missed one TMH, and TOP-PRED, MemBrain and SOMPNN can correctly predict there are three TMHs in the spinach CP29 protein.

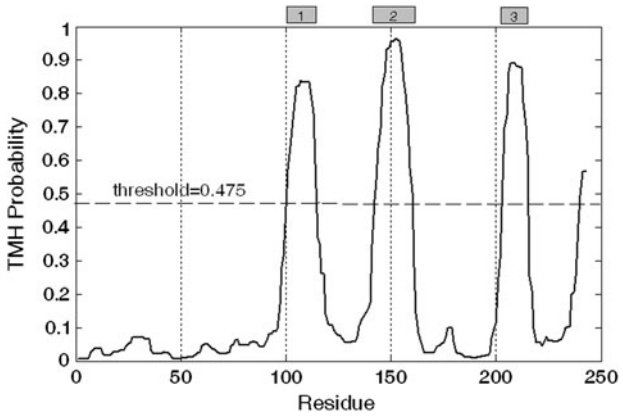


**Fig. 4** TMH probability curve of the spinach minor light-harvesting complex CP29 (PDB code: 3PL9) predicted by SOMPNN

**Table 9** TMH prediction results for 3PL9 by different TMH predictors

| Observed TMHs of 3PL9 | | 89–118 | 143–164 | 189–219 |
|---|---|---|---|---|
| Predicted TMHs | TMHMM | | | |
| | SOSUI | | 140–162 | |
| | DAS–TMfilter | | 147–153 | |
| | PHOBIUS | 102–121 | 141–160 | |
| | TOP–PRED | 102–122 | 138–158 | 201–221 |
| | MemBrain | 96–115 | 140–159 | 201–215 |
| | SOMPNN | 100–114 | 142–160 | 202–215 |

## Conclusions

In this paper, we presented SOMPNN, a non-parametric model with high prediction efficiency, for TMH prediction. Results based on two benchmark datasets have shown that SOMPNN is a new competitive TMH predictor. On the one hand, SOMPNN overcomes the need for problematic modeling assumptions required by parametric model methods; on the other hand, SOMPNN can perform prediction with high efficiency by exploiting the learning ability of SOM. These advantages make the SOMPNN sufficiently flexible to suite for a wide variety of problems in bioinformatics. As a new independent TMH predictor, the SOMPNN can complement the existing parametric predictors and help to further improve prediction accuracy.

Our future work will focus on improving SOMPNN's ability of predicting the ends of TMHs. Research has shown that there exist position-specific residue preferences around the helices and strands (Duan et al. 2008). We will integrate the position-specific residue preference feature into SOMPNN to further improve its ability of predicting TMH ends. In addition, current knowledge learned by SOM is encoded in the codebook vectors, which are obscure for human to understand. How to improve the interpretability of the leaned knowledge is also our future research interest.

**Acknowledgments** The authors wish to thank the three anonymous reviewers whose constructive comments are very helpful for strengthening the presentation of this paper. This work was supported by the National Natural Science Foundation of China (Grant No. 60704047), A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, A Foundation for the Author of National Excellent Doctoral Dissertation of PR China (Grant No. 201048), Shanghai Pujiang Program, and Innovation Program of Shanghai Municipal Education Commission (10ZZ17).

## References

Asur S, Ucar D, Parthasarathy S (2007) An ensemble framework for clustering protein-protein interaction networks. Bioinformatics 23(13):i29–i40

Campbell K, Kurgan L (2008) Sequence-only based prediction of b-turn location and type using collocation of amino acid pairs. Open Bioinf J 2:37–49

Chen CP, Kernytsky A, Rost B (2002) Transmembrane helix predictions revisited. Protein Sci 11(12):2774–2791

Chen K, Kurgan L (2007) PFRES: protein fold classification by using evolutionary information and predicted secondary structure. Bioinformatics 23(21):2843–2850

Claros MG, von Heijne G (1994) TopPred II: an improved software for membrane protein structure predictions. Comput Appl Biosci 10(6):685–686

Cuthbertson JM, Doyle DA, Sansom MS (2005) Transmembrane helix prediction: a comparative evaluation and analysis. Protein Eng Des Sel 18(6):295–308

Duan M, Huang M, Ma C, Li L, Zhou Y (2008) Position-specific residue preference features around the ends of helices and strands and a novel strategy for the prediction of secondary structures. Protein Sci 17(9):1505–1512

Feng M, Wenping W, Wai Wan T, Zesheng T, Shaowei X, Xin T (1998) Probabilistic segmentation of volume data for visualization using SOM-PNN classifier. Paper presented at the Proceedings of the 1998 IEEE symposium on Volume visualization. Research Triangle Park, NC

Frishman D (2010) Structural bioinformatics of membrane proteins. Springer, Berlin

Gibbons JD, Chakraborti S (2003) Nonparametric statistical inference, 4th edn. M. Dekker, New York

Gromiha MM (1999) A simple method for predicting transmembrane alpha helices with better accuracy. Protein Eng 12(7):557–561

Gromiha MM (2010) Protein Bioinformatics: From Sequence to Function. Elsevier Publishers

Haykin SS (1999) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall, Upper Saddle River, NJ

He JY, Hu HJ, Harrison R, Tai PC, Pan Y (2006) Transmembrane segments prediction and understanding using support vector machine and decision tree. Expert Syst Appl 30(1):64–72

Hu HJ, Holley J, He J, Harrison RW, Yang H, Tai PC, Pan Y (2007) To be or not to be: predicting soluble SecAs as membrane proteins. IEEE Trans Nanobiosci 6(2):168–179

Ishida T, Kinoshita K (2008) Prediction of disordered regions in proteins based on the meta approach. Bioinformatics 24(11): 1344–1348

Jones DT (2007) Improving the accuracy of transmembrane protein topology prediction using evolutionary information. Bioinformatics 23(5):538–544

Kall L, Krogh A, Sonnhammer EL (2004) A combined transmembrane topology and signal peptide prediction method. J Mol Biol 338(5):1027–1036

Klammer M, Messina DN, Schmitt T, Sonnhammer EL (2009) MetaTM—a consensus method for transmembrane protein topology prediction. BMC Bioinform 10:314

Kohonen T (1989) Self-organization and associative memory. 3rd edn. Springer, Berlin

Krogh A, Larsson B, von Heijne G, Sonnhammer EL (2001) Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. J Mol Biol 305(3):567–580

Martelli PL, Fariselli P, Casadio R (2003) An ENSEMBLE machine learning approach for the prediction of all-alpha membrane proteins. Bioinformatics 19(suppl 1):i205–i211

Mizianty MJ, Kurgan L (2009) Meta prediction of protein crystallization propensity. Biochem Biophys Res Commun 390(1): 10–15

Mizianty MJ, Kurgan L (2011) Improved identification of outer membrane beta barrel proteins using primary sequence, predicted secondary structure, and evolutionary information. Proteins 79(1):294–303

Nanni L, Lumini A (2006) An ensemble of K-local hyperplanes for predicting protein–protein interactions. Bioinformatics 22(10): 1207–1210

Ou YY, Gromiha MM, Chen SA, Suwa M (2008) TMBETADISC-RBF: discrimination of beta-barrel membrane proteins using RBF networks and PSSM profiles. Comput Biol Chem 32(3): 227–231

Pan XW, Li M, Wan T, Wang LF, Jia CJ, Hou ZQ, ZHANG JP, Zhao XL, Chang WR (2011) Structural insights into energy regulation of light-harvesting complex CP29 from spinach. Nat Struct Mol Biol 18(3):309–316

Plewczynski D (2010) Brainstorming: weighted voting prediction of inhibitors for protein targets. J Mol Model. doi:10.1007/s00894-010-0854-x

Ponnuswamy PK, Gromiha MM (1993) Prediction of transmembrane helices from hydrophobic characteristics of proteins. Int J Pept Protein Res 42(4):326–341

Rost B, Casadio R, Fariselli P, Sander C (1995) Transmembrane helices predicted at 95% accuracy. Protein Sci 4(3):521–533

Shen H, Chou JJ (2008) MemBrain: improving the accuracy of predicting transmembrane helices. PLoS One 3(6):e2399

Shen HB, Chou GC (2007a) Nuc-PLoc: a new web-server for predicting protein subnuclear localization by fusing PseAA composition and PsePSSM. Protein Eng Des Sel 20(11):561–567

Shen HB, Chou KC (2007b) Using ensemble classifier to identify membrane protein types. Amino Acids 32(4):483–488

Shen HB, Song J, Chou KC (2009) Prediction of protein folding rates from primary sequence by fusing multiple sequential features. J Biomed Sci Eng 2(3):136–143

Song J, Tan H, Mahmood K, Law RH, Buckle AM, Webb GI, Akutsu T, Whisstock JC (2009) Prodepth: predict residue depth by

support vector regression approach from protein sequences only. PLoS One 4(9):e7072

Song J, Yuan Z, Tan H, Huber T, Burrage K (2007) Predicting disulfide connectivity from protein sequence using multiple sequence feature vectors and secondary structure. Bioinformatics 23(23):3147–3154

Specht DF (1990) Probabilistic neural networks and the polynomial Adaline as complementary techniques for classification. IEEE Trans Neural Netw 1(1):111–121

von Heijne G (2006) Membrane-protein topology. Nat Rev Mol Cell Biol 7(12):909–918

Wang G, Dunbrack RL Jr (2003) PISCES: a protein sequence culling server. Bioinformatics 19(12):1589–1591

Wimley WC, White SH (1996) Experimentally determined hydrophobicity scale for proteins at membrane interfaces. Nat Struct Biol 3(10):842–848

Yanover C, Singh M, Zaslavsky E (2009) M are better than one: an ensemble-based motif finder and its application to regulatory element prediction. Bioinformatics 25(7):868–874

Yuan Z, Mattick JS, Teasdale RD (2004) SVMtm: support vector machines to predict transmembrane segments. J Comput Chem 25(5):632–636

Zhang H, Zhang T, Chen K, Kedarisetti KD, Mizianty MJ, Bao Q, Stach W, Kurgan L (2011) Critical assessment of high-throughput standalone methods for secondary structure prediction. Brief Bioinform. doi:10.1093/bib/bbq088

Zhou H, Zhou Y (2003) Predicting the topology of transmembrane helical proteins using mean burial propensity and a hidden-Markov-model-based method. Protein Sci 12(7):1547–1555