1

CURRENT METHODS FOR PROTEIN SECONDARY-STRUCTURE PREDICTION BASED ON SUPPORT VECTOR MACHINES

HAE-JIN HU, ROBERT W. HARRISON, PHANG C. TAI, AND YI PAN

Department of Computer Science (H-J.H., R.W.H., Y.P.) and Department of Biology (R.W.H., P.C.T.), Georgia State University, Atlanta, Georgia

The desire to understand protein structure has produced many approaches over the last four decades since Blout et al. (1960) attempted to correlate the sequence information of amino acids with their structural elements (Casbon, 2002). Instead of costly and time-consuming experimental approaches, effective prediction methods have been developed continuously. With the help of growing databases and the evolutionary information available from multiple-sequence alignments, resources for secondary-structure prediction became abundant. Also, progress in machine learning technology provided various advanced tools for prediction. Among the many machine learning approaches, support vector machine (SVM) methods are the most recent to be used for structure prediction. SVMs perform successfully, but compared with other machine learning approaches, there is no systematic review in the SVM approach when applied to secondary-structure prediction. Therefore, this study focuses mainly on methods of predicting secondary structure based on support vector machines. The organization of this chapter is as follows. In Section 1.1, traditional secondary-structure prediction approaches are described. In Section 1.2, various SVM-based prediction methods are introduced. In Section 1.3, the performance of SVM methods is evaluated, and in Section 1.4, problems with the SVM approach and efforts to overcome them are discussed.

Knowledge Discovery in Bioinformatics: Techniques, Methods, and Applications, Edited by Xiaohua Hu and Yi Pan

1.1 TRADITIONAL METHODS

1.1.1 Statistical Approaches

The first attempts to predict secondary structure took place in the 1970s. These include the Chou–Fasman algorithm (Chou and Fasman, 1974) and the GOR method (Garnier et al., 1978), both based on empirical and statistical methods. In the Chou–Fasman algorithm, the conformational parameters P_α, P_β, and P_c for each amino acid are obtained by analyzing the x-ray-determined structures of 15 proteins including 2473 amino acid residues. These parameters represent a given amino acid's tendency to be found in α-helix, β-sheet, or coil, and they contain the physicochemical properties of amino acids. Based on these parameters, the authors established empirical rules for secondary-structure prediction. This relatively simple algorithm has been criticized (Kyngas and Valjakka, 1998) since it is based on a small database for statistical analysis; the original algorithm consists of 15 proteins with 2473 amino acid residues, and the revised version (Chou, 1989) has a database of 64 proteins with 11,445 amino acid residues.

The second statistical approach, that of Garnier, Osguthorpe, and Robson (*GOR I*; Garnier et al., 1978), is based on information theory. This method applies a sliding window of 17 residues and calculates the probabilities of eight neighboring residues on each side of the sliding window for predicting a residue in the middle. This algorithm has been revised during the past 20 years, and in 2002, version GOR V was developed (Kloczkowski et al., 2002). Similar to the Chou–Fasman algorithm, the first version of GOR I used a small database of 26 proteins with about 4500 residues. However, during the revision process, the database size was extended to 267 proteins for GOR IV and to 513 proteins for GOR V. A major improvement of GOR V over earlier versions was achieved by including evolutionary information through use of PSI-BLAST multiple-sequence alignments into the GOR method (Kloczkowski et al., 2002). According to the authors of GOR V, the average prediction accuracy of secondary structure is $Q_3 = 73.5\%$ on the CB513 data set (Cuff and Barton, 1999) using a jackknife test. In the test, each protein in the data set is singled out in turn for an independent test while the remaining proteins are used for training. This process is repeated until all proteins are selected for testing.

1.1.2 Machine Learning Approaches

Nearest-Neighbor Method The nearest-neighbor approach predicts the secondary structure of the central residue of a segment based on the secondary structure of homologous proteins from a database with known three-dimensional structures (Nishikawa and Ooi, 1986). In other words, this method matches each segment of the query sequence against all sequences in the database. Once nearest neighbors (homologous segments) are found based on similarity criteria, the secondary structure of the central residue is determined using the frequencies ($f_{\text{helix}}, f_{\text{sheet}}, f_{\text{coil}}$) of secondary-structure states at the central position of its neighbors (Biou et al., 1988; Levin and Garnier, 1988). Even though the basic idea is simple, it requires adjusting lots of factors, such as

similarity measures, window size of querying sequence, or the number of nearest neighbors. Therefore, there have been many studies with various ways of applying different parameters with various results (Yi and Lander, 1993; Salamov and Solovyev, 1995; Levin, 1997). For example, Yi and Lander (1993) matched their 19 residue segments against the database of 110 proteins with known tertiary structure. Based on the local structural environmental scoring metric of Bowie et al. (1991), 50 nearest neighbors are found. This metric contains environmental parameters such as secondary-structure state, polarity, and accessible surface area. The score of matching a residue R_i with a local structural environment E_i was defined as

$$\text{Score}(R_i, E_j) = \log_{10} \frac{P(R_i|E_j)}{P(R_i)} \tag{1.1}$$

where $P(R_i|E_j)$ is the probability of finding residue i in environment j, and $P(R_i)$ is the probability of finding residue i in any environment (Yi and Lander, 1993). The secondary structure predicted for a test residue was selected as the state of maximum frequency, $\max(f_{\text{helix}}, f_{\text{sheet}}, f_{\text{coil}})$, of secondary-structure states at the central position of its 50 neighbors. The authors tested various scoring schemes with different numbers of environment classes and obtained the optimal system. It consists of 15 environmental classes, including three secondary structures combined with five different accessibility or polarity classes based on the mutation matrix of Gonnet et al. (1992). To combine the results from six scoring schemes, the authors adopted the neural network for jury decision and attained an accuracy of $Q_3 = 68\%$ with a jackknife test.

Salamov and Solovyev (1995) revised Yi and Lander's scheme. Their improvements were first, changing the scoring scheme by considering the N- and C-terminal positions of α-helices and β-sheets and by taking beta turns as unique types of secondary structure. Second, the authors restricted the database to a smaller subset of proteins that are close to a test sequence in general properties, thus reducing the computation time. Third, by applying multiple-sequence alignments and a jury decision process, they achieved $Q_3 = 72.2\%$ accuracy on a 126-protein data set using a jackknife test.

In 1997, Levin modified his own method, called SIMPA (Levin and Garnier, 1988), by applying a larger database of known structures, the Blosum62 substitution matrix, and a regularization algorithm (Zimmermann, 1994). However, both prediction algorithms are the same. The algorithm compares every residue in a window with each residue of the same window size in a database of known structures. Based on the Blosum62 scoring matrix, the match score is calculated. If the match score is smaller than a cutoff value, the peptide is not considered. Otherwise, the conformation observed is allocated to the test residue with its similarity score. The prediction of a test sequence is made based on the highest score at each residue location by applying a regularization algorithm. The regularization algorithm restricts the minimum length of helix and strand to four and two residues, respectively. By including the evolutionary information, the updated version of SIMPA96 reached $Q_3 = 71.4\%$ accuracy in a jackknife test (Levin, 1997).

Hidden Markov Model The hidden Markov model (HMM) is a probabilistic finite-state machine applied to model stochastic sequences. In HMMs, domain information

can be included in the topology of the HMM, while other information is learned by training the emission and transition probabilities on data (Won et al., 2005). Because of this merit, HMMs have been applied widely in computational biology, such as in gene finding, sequence alignment, and protein structure prediction (Krogh et al., 1994; Bystroff et al., 2000).

A HMM consists of a set of states, emission probabilities related to each state, and transitions that connect states. Each state has symbols characterizing an amino acid residue or a secondary-structure type. If the symbols are the set of 20 amino acids denoted by $\mathbf{A}$ and the set of HMM parameters are represented by $\boldsymbol{\pi}$, HMM assigns a probability to a given sequence $\mathbf{s} = (s_1, s_2, \ldots, s_m)$ of length m:

$$\sum_{s \in A^m} P(s|\pi) = 1 \tag{1.2}$$

If the HMMs emit class labels as well as symbols from the 20 amino acids, a sequence $\mathbf{s}$ can be associated with a corresponding class labels $\mathbf{t} = (t_1, t_2, \ldots, t_m)$. If we let the set of states be represented by Q and denote a sequence of states $\mathbf{q} = (q_1, q_2, \ldots, q_m)$, the chance of a sequence $\mathbf{s}$ having class labels $\mathbf{t}$ can be written as the sum over all possible paths through the states:

$$P(s, t|\pi) = \sum_{q \in Q^m} P(s, t, q|\pi) \tag{1.3}$$

For a given sequence $\mathbf{s}$ and the corresponding class labels $\mathbf{t}$, the maximum likelihood (ML) set of parameters can be found based on the Baum–Welch algorithm (Rabiner, 1989):

$$\pi^{\mathrm{ML}} = \arg \max_{\pi} \; P(s, t|\pi) \tag{1.4}$$

Among the studies using HMM, Bystroff et al. (2000) introduced a new HMM model, HMMSTR, for general protein sequences based on basic local structural motifs called *I-sites*. These structural motifs are a set of sequence motifs of length 3 to 19 obtained from a nonredundant database of known structures. Each motif has information about the sequence and structure and was expressed as a chain of Markov states. By merging the linear chains of states (motifs) based on sequence and structure similarity, an HMM graph with branched topology was formed. The interesting feature of HMMSTR is that it models local motifs common to all protein families instead of modeling individual protein families. Based on this advanced HMM model, the authors attained $Q_3 = 74.3\%$ accuracy in secondary-structure prediction.

In 2005, Won et al. designed a new scheme to evolve a HMM with a genetic algorithm (GA). The authors applied a hybrid GA that adopts traditional GA operators to search the space of HMM topologies combined with the Baum–Welch algorithm to optimize the transition and emission probabilities. With this method, they achieved $Q_3 = 75\%$ accuracy with a fivefold cross-validation test.

Neural Network Methods Inspired by neurons in the brain, artificial neural networks are parallel information-processing structures frequently used in classification problems. For secondary-structure prediction, feedforward network architecture is used successfully. The feedforward network is organized with a layered structure, including input layer, output layer, or zero or more hidden layers, such as that shown in Figure 1.1. Each layer has one or more processing units called *nodes* or *neurons*. Each node in a layer is connected to nodes in a preceding layer by weights. In secondary-structure prediction with neural networks, the inputs are sliding windows with a 13- to 17-residue sequence. In the input layer each node takes a feature value representing the amino acid type. Based on the node's input value, the output value is calculated in each node. If we let the nodes of the preceding layer be i ($i = 1, 2, \ldots, M$), a node of the current layer be j, and the weight between nodes i and j be w_{ji}, the total weighted input to j, X_j, can be calculated using the formula (Eidhammer et al., 2004)

$$X_j = \sum_{i=1}^{M} w_{ji} a_i \tag{1.5}$$

where a_i is the activity level of node i in the preceding layer. The activity level of node j, a_j, can be computed based on some function of the total weighted input. Typically, the sigmoid function is applied as follows:

$$a_j = \frac{1}{1 + e^{-X_j}} \tag{1.6}$$

This value is sent to all lines connected out from the node j.

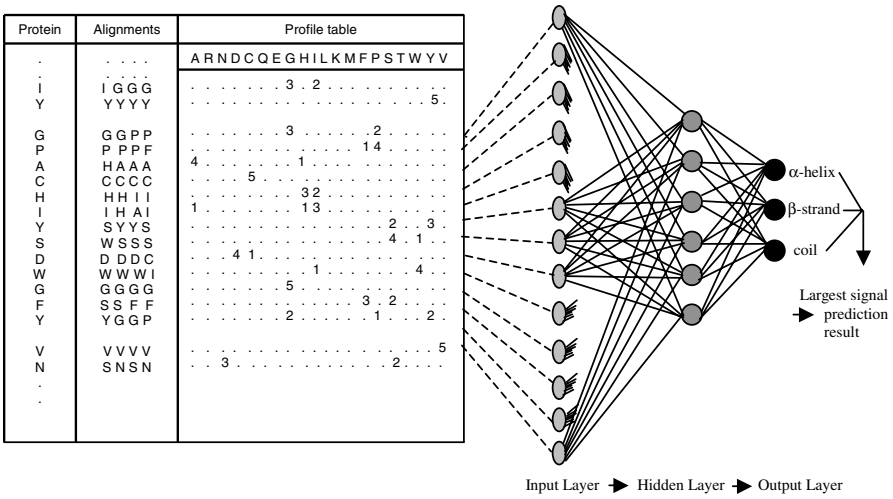

FIGURE 1.1 Neural network architecture. As an encoding profile for training the neural network, frequencies of each amino acid at each position are adopted. These frequencies are obtained from multiple-sequence alignments. The input signal is propagated through the network with one input layer, one hidden layer, and one output layer. (Based on Rost and Sander, 1993a.)

The neural network is trained using the supervised learning method. Here the training process is finding the appropriate value for the weight of each line in the network to make as many correct predictions as possible. In supervised learning, a training data set is formed as encoded feature vectors combined with correct class labels, such as helix, sheet, or coil. Before training the network, the weights are initialized to small random values. Once all training data have been fed into the network, the network generates output based on its current weights. By comparing the output with the correct class label, an error is computed for each node of the output layer. The classical back-propagation algorithm (Rumelhart et al., 1986) can be used to propagate the errors to the previous layers and to adjust the weights.

Rost and Sander's PHD (1993b) is one of the most popular methods for predicting secondary structure. PHD uses a two-layer feedforward neural network and incorporates the evolutionary information based on multiple sequence alignments. The initial version (Rost and Sander, 1993b) gives $Q_3 = 70.8\%$ accuracy for globular proteins with a seven-fold cross-validation test. Since then there have been many approaches to improving this result (Riis and Krogh, 1996; Chandonia and Karplus, 1999; Cuff and Barton, 1999; Petersen et al., 2000; Pollastri et al., 2002).

The PSIPRED method by Jones (1999) is another successful approach for predicting secondary structure. In PSIPRED, a greatly simplified two-stage neural network was used based on the position-specific scoring matrices generated by PSI-BLAST (Figure 1.2). The author set up a new cross-validation scheme by screening the training and testing sets based on a structural similarity criterion. Instead of removing a protein from a training set that has a high degree of sequence similarity to the testing set, the author discarded a protein with a fold similar to that of any of the testing set. Based on this test, the author attained an accuracy between $Q_3 = 76.5$ and 78.3% on 187 test proteins.

In 1999, Chandonia and Karplus (1999) used an enlarged database of 258 to 681 proteins to train the neural networks with more informative patterns. In addition, by applying second-level networks called *juries* they obtained $Q_3 = 74.9\%$ average accuracy on 681 proteins when tested with 15-fold cross-validation.

Petersen et al. (2000) combined 800 neural network predictions based on new methods called output expansion and balloting procedure. In the *output expansion process*, the secondary structures of a residue and its neighbors are predicted at the same time. The central idea of this approach is that these additional outputs give more information to the neural networks for optimizing the weights. To combine the results of multiple predictions, the authors adopted a statistical method called the *balloting scheme*. The balloting method is a variety of weighted-average scheme based on a mean and standard deviation. The authors reported that this balloting scheme enhances the performance better than straight averaging. An accuracy of $Q_3 = 80.2\%$ was claimed when the RS126 data set is used as an independent test set.

In 2002, Pollastri et al. introduced the ensembles of a bidirectional recurrent neural network (BRNN), which are the next version (SSpro2.0) of their previous BRNN architecture (Baldi et al., 1999). SSpro2.0 achieves a Q_3 value of 78.1% on an RS126 independent test set.

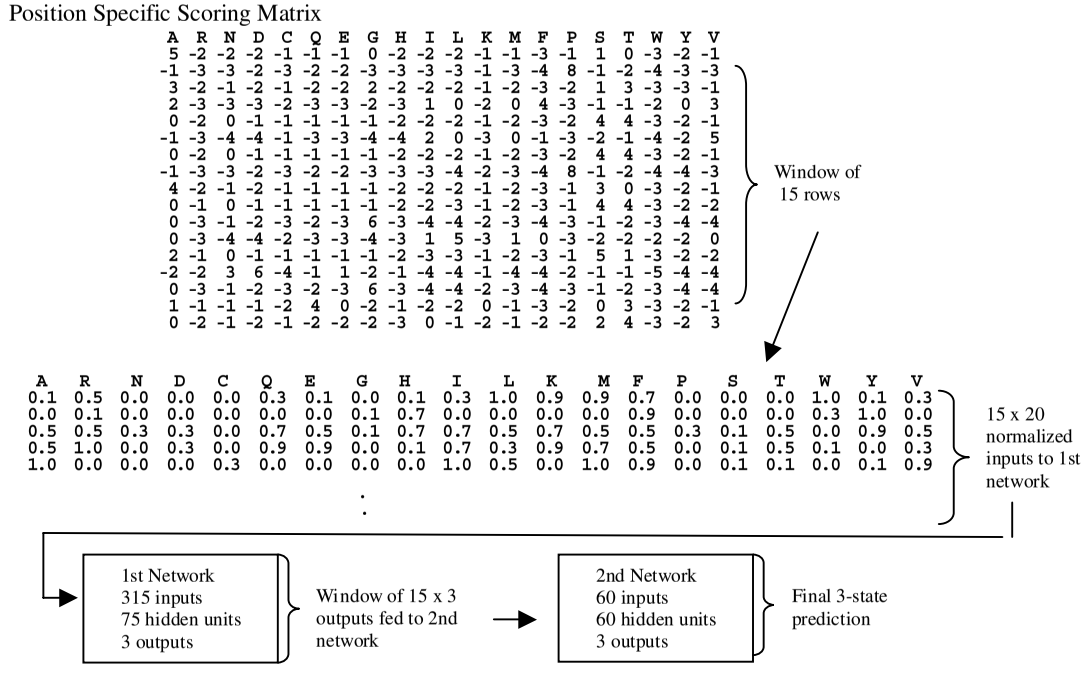

FIGURE 1.2 PSIPRED method, describing how the PSSMs are processed. (Based on Jones, 1999.)

Kernel-Based Methods Recently, a number of kernel-based learning schemes, including the support vector machine (SVM) method, kernel Fisher discriminant (KFD), and kernel principal component analysis (KPCA), were introduced (Müller et al., 2001). The difference in these approaches is that they used different algorithms to handle the high dimensionality of kernel feature space. Among these schemes, the SVM method is the most widely used machine learning approach. Therefore, in the next section several SVM-based secondary-structure prediction methods are discussed in detail.

1.2 SUPPORT VECTOR MACHINE METHOD

1.2.1 Introduction to SVM

SVM is a modern learning system designed by Vapnik and his colleagues (Vapnik and Cortes, 1995). Based on statistical learning theory, which explains the learning process from a statistical point of view, the SVM algorithm creates a hyperplane that separates the data into two classes with the maximum margin. Originally, it was a linear classifier based on the optimal hyperplane algorithm developed by Vapnik in 1963. However, by applying the kernel method to a maximum-margin hyperplane, in 1992 Vapnik and his colleagues proposed a method to build a nonlinear classifier. In 1995, Cortes and Vapnik suggested a soft margin classifier, which is a modified maximum margin classifier that allows for misclassified data. If there is no hyperplane that can separate the data into two classes, the soft margin classifier selects a hyperplane that separates the data as cleanly as possible with a maximum margin.

SVM learning is related to recognizing the pattern from the training data (Burges, 1998; Cristianini and Shawe-Taylor, 2000). Namely, a function f: $R_N \rightarrow \{\pm 1\}$ is estimated based on the training data, which have an N-dimensional pattern $\mathbf{x}_i$ and class labels y_i. By imposing a restriction called *structural risk minimization* (SRM) on this function, it will correctly classify the new data $(\mathbf{x},y)$ that have the same probability distribution $P(\mathbf{x},y)$ as the training data. SRM is used to find the learning machine that yields a good trade-off between low empirical risk (mean error over the training data) and small capacity (a set of functions that can be implemented by the learning machine). In the linear soft margin SVM, which allows some misclassified points, the optimal hyperplane can be found by solving the following constrained quadratic optimization problem:

$$\min_{w,b,\varepsilon} \tfrac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{l} \varepsilon_i \tag{1.7}$$

$$\text{subject to} \qquad y_i(\mathbf{w} \bullet \mathbf{x}_i + b) \geq 1 - \varepsilon_i, \qquad \varepsilon_i > 0, \quad i = 1, \ldots, l \tag{1.8}$$

where $\mathbf{x}_i$ is an input vector, $y_i = +1$ or -1 based on whether $\mathbf{x}_i$ is in a positive or a negative class, l is the number of training data, $\mathbf{w}$ is a weight vector perpendicular to the hyperplane, and b is a bias that moves the hyperplane parallel to itself. C is a cost

factor (penalty for misclassified data) and ε is a slack variable for misclassified points. The resulting hyperplane decision function is

$$f(x) = \text{sign}\left[\sum_{i=1}^{\text{SV}} \alpha_i y_i \, (\mathbf{x} \bullet \mathbf{x}_i) + b\right] \tag{1.9}$$

where, α_i is a Lagrange multiplier for each training data. The points $\alpha_i > 0$ lie on the boundary of the hyperplane and are called *support vectors*. In Eqs. (1.8) and (1.9) it is observed that both the optimization problem and the decision function rely on the dot products between each pattern.

In nonlinear SVM, the algorithm first maps the data into high-dimensional feature space via a kernel function and constructs the optimal separating hyperplane there using the linear algorithm. The common kernel functions are the following:

$$K(\mathbf{x}, \mathbf{x}') = \begin{cases} (\mathbf{x} \bullet \mathbf{x}' + 1)^p & (1.10) \\ e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} & (1.11) \\ \tanh(k\mathbf{x} \bullet \mathbf{x}' - \delta). & (1.12) \end{cases}$$

Equation (1.10) is a polynomial, Eq. (1.11) is a Gaussian radial basis function (RBF), and Eq. (1.12) is a two-layer sigmoidal neural network kernel. Based on one of the kernel functions above, the final nonlinear decision function has the form

$$f(x) = \text{sign}\left[\sum_{i=1}^{\text{SV}} \alpha_i y_i K(\mathbf{x} \bullet \mathbf{x}_i) + b\right] \tag{1.13}$$

SVM$^{\text{light}}$ (Joachims, 1999) and LIBSVM (Chang and Lin, 2001) are widely used software implementations of SVM.

The SVM algorithm has the following outstanding features. First, it can avoid overfitting effectively with the use of structural risk minimization. Second, the formulation can be simplified to a convex quadratic programming (QP) problem; the training will converge to a global optimum. Note that the global optimum is the best solution for a given kernel and training data sets. Different qualities of results will be found with different kernels and data. Third, for a given data set, information can be condensed while training without losing useful information (Hua and Sun, 2001). Since this SVM has outperformed most other learning systems in most pattern recognition problems (Hua and Sun, 2001), it has gradually been applied to pattern classification problems in biology.

One recent study adopting this SVM learning machine for secondary-structure prediction adopted frequency profiles with evolutionary information as an encoding scheme for SVM (Hua and Sun, 2001). Another approach applied two layers of SVM with a weighted cost function for balanced training (Casbon, 2002). Also, there were methods that incorporated PSI-BLAST PSSM profiles as an input vector and that applied new tertiary classifiers (Kim and Park, 2003; Hu et al., 2004). Based on the results of these studies, the success of SVM methods depends on the proper choice of

encoding profile, kernel function, and tertiary classifier. In the following section we investigate how these factors are optimized in each study.

1.2.2 Encoding Profile

In the studies above, the SVM was trained with different sequence and structural information using a sliding window scheme, a method that has been used in many other secondary-structure prediction schemes, including a later version of GOR methods, neural network methods, and nearest-neighbor algorithms. In the sliding window method, a window becomes one training pattern for predicting the structure of the residue at the center of the window. In each training pattern, information about local interactions among neighboring residues is embedded. The feature value of each amino acid residue in a window represents the weight (costs) of each residue in a pattern.

Orthogonal Encoding In orthogonal encoding, each residue has a unique binary vector, such as (1,0,0,. . .), (0,1,0,. . .), (0,0,1,. . .), and so on. Each binary vector is 20-dimensional. In this method, the weights of all residues in a window are assigned to 1. As a simple example, if we take a window size as 5, the dimension of one input pattern becomes:

$$\text{one vector dimension} = (20 \text{ binary bits})\underbrace{(5 \text{ residues})}_{\text{window size}} = 100$$

Therefore, the amino acid segment STAAD can be written as the following vector based on the indices of Eq. (1.14).

$$\underbrace{(0,0,\ldots,1,0,0,0,0)}_{\text{S (16)}},\underbrace{(0,0,\ldots,1,0,0,0)}_{\text{T (37)}},\underbrace{(1,0,0,\ldots)}_{\text{A (41)}},\underbrace{(1,0,0,\ldots)}_{\text{A (61)}},\underbrace{(0,0,0,1,0,\ldots)}_{\text{D (84)}}$$

Hydrophobicity Encoding Hu et al. (2004) examined hydrophobicity encoding as one of their encoding profiles. Among the many different hydrophobicity measures, they adopted the Radzicka and Wolfenden scale (Radzicka and Wolfenden, 1988). These index values are as follows:

$$\text{amino acids } [\cdot] = \{\text{A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V}\} \quad (1.14)$$

$$\begin{aligned}\text{hydrophobicity index } [\cdot] = \{&1.81, -14.92, -6.64, -8.72, 1.28, -5.54, -6.81,\\ &0.94, -4.66, 4.92, 4.92, -5.55, 2.35, 2.98, 4.04,\\ &-3.40, -2.57, 2.33, -0.14, 4.04\}\end{aligned} \quad (1.15)$$

The hydrophobicity matrix is formulated based on the values above, and by using the following function:

$$\text{hydrophobicity matrix}[i][j] = \frac{\text{abs}(\text{hydrophobicity index}[i] - \text{hydrophobicity index}[j])}{20.0} \quad (1.16)$$

The denominator, 20, is used to convert the data range into [0,1] since SVM feature values are within this range. According to the function above, hydrophobicity matrix[2][3] means the absolute value of the difference of the hydrophobicity indices of two amino acids: for example, R (−14.92) and N (−6.64). With the range adjustment, it becomes 0.414. Based on this method, a 20×20 hydrophobicity matrix can be formulated.

BLOSUM Encoding In BLOSUM coding (Hu et al., 2004) each amino acid is represented by values from the BLOSUM62 amino acid replacement cost matrix. The BLOSUM62 matrix represents the ''log-odds'' scores for the likelihood that a given amino acid pair will interchange with another. It is expected that this would partially account for the structural conservation of residue upon replacement.

Hybrid Encoding Since each of these coding schemes captures different aspects of the properties of amino acids, Hu et al. (2004) tested combinations of two encodings, such as the following: orthogonal matrix + hydrophobicity matrix, BLOSUM62 matrix + hydrophobicity matrix, and orthogonal matrix + BLOSUM62 matrix.

Frequency Matrix Encoding In this coding (Hua and Sun, 2001), the frequency of occurrence of 20 amino acid residues at each position in the multiple sequence alignment is calculated for each residue. This encoding was initially applied in Rost and Sander's neural network approach (Rost and Sander, 1993a) (Figure 1.1).

PSSM Encoding PSSM coding applies the position-specific scoring matrix (PSSM) generated by PSI-BLAST (Kim and Park, 2003; Hu et al., 2006). In this coding the individual profiles were used to reflect detailed conservation of amino acids in a family of homologous proteins. This scheme was originally adopted by Jones (1999) to perform the prediction of protein secondary structure with his neural network (Figure 1.2). According to the author, PSI-BLAST is a very effective sequence query method, due to three aspects. First, the alignments generated by PSI-BLAST are based on pairwise local alignments. The previous study (Frishman and Argos, 1997; Salamov and Solovyev, 1997) reported that by using reliable local alignments, the prediction accuracy could be improved. Next, based on the iterated profiles, the sensitivity of PSI-BLAST was enhanced. Finally, the author tried many automatic multiple-sequence alignments. Among them, the PSI-BLAST alignments performed best.

1.2.3 Kernel Functions

The choice of kernel function is critical to the success of SVM. Most studies (Hua and Sun, 2001; Casbon, 2002; Kim and Park, 2003; Hu et al., 2004) adopted the Gaussian radial basis function (RBF) kernel after testing the common kernel functions from Section 1.2.1. Recently, an approach to designing new kernel functions based on a substitution matrix for protein secondary-structure prediction was developed (Vanschoenwinkel and Manderick, 2004). In another approach, Altun et al. (2006a,b) designed hybrid kernels that combined a substitution matrix–based kernel,

an edit kernel, or a self-organizing map (SOM)–based kernel with the RBF kernel. Even though both approaches are not very successful in outperforming the best secondary-structure prediction methods, these are decent examples to give some idea of designing new kernel functions. These approaches are introduced in detail in this section.

Substitution Matrix–Based Kernels Substitution matrix–based kernels were developed by Vanschoenwinkel and Manderick (2004). The authors introduced a pseudo inner product (PI) between amino acid sequences based on the Blosum62 substitution matrix values. PI is defined as follows:

Definition 1.1 Let M be a 20×20 symmetric substitution matrix with entries $M(a_i, a_j) = m_{ij}$, where a_i and a_j are components of the 20-tuple $\mathbf{A} = (\mathrm{A}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{G}, \mathrm{H}, \mathrm{I}, \mathrm{K}, \mathrm{L}, \mathrm{M}, \mathrm{N}, \mathrm{P}, \mathrm{Q}, \mathrm{R}, \mathrm{S}, \mathrm{T}, \mathrm{V}, \mathrm{W}, \mathrm{Y}) = (a_1, \ldots, a_{20})$. Then the inner product of two amino acid sequences $\mathbf{s}$, $\mathbf{s}' \in \sum^n$ with $\mathbf{s} = (a_{i1}, \ldots, a_{in})$ and $\mathbf{s}' = (a_{j1}, \ldots, a_{jn})$, with $a_{ik}, a_{jk} \in \mathbf{A}, i, j \in \{1, \ldots, 20\}$ and $k = 1, \ldots, n$, is defined as

$$\langle \mathbf{s} | \mathbf{s}' \rangle = \sum_{k=1}^{n} M(a_{ik}, a_{jk}) \tag{1.17}$$

By applying the PI above, the authors defined a substitution matrix–based distance function between amino acid sequences as follows:

Definition 1.2 Let $\mathbf{s}, \mathbf{s}' \in \sum^n$ be two amino acid sequences with $\mathbf{s} = (a_{i1}, \ldots, a_{in})$ and $\mathbf{s}' = (a_{j1}, \ldots, a_{jn})$, and let $\langle \mathbf{s} | \mathbf{s}' \rangle$ be the inner product as defined by Eq. (1.17); then the substitution distance d_{sub} between $\mathbf{s}$ and $\mathbf{s}'$ is defined as

$$d_{\text{sub}}(\mathbf{s}, \mathbf{s}') = \sqrt{\langle \mathbf{s} | \mathbf{s} \rangle - 2\langle \mathbf{s} | \mathbf{s}' \rangle + \langle \mathbf{s}' | \mathbf{s}' \rangle} \tag{1.18}$$

Based on both the PI and the substitution distance d_{sub} above, three different kernel functions were developed:

$$K_{\text{PIK}}(\mathbf{s}, \mathbf{s}') = (\langle \mathbf{s} | \mathbf{s}' \rangle + c)^d \tag{1.19}$$

$$K_{\text{SRBK}}(\mathbf{s}, \mathbf{s}') = \exp[-\gamma d_{\text{sub}}(\mathbf{s}, \mathbf{s}')^2] \tag{1.20}$$

$$K_{\text{NSDK}}(\mathbf{s}, \mathbf{s}') = -[d_{\text{sub}}(\mathbf{s}, \mathbf{s}')]^\beta \qquad \text{with} \quad 0 < \beta \leq 2 \tag{1.21}$$

The pseudo inner product kernel (K_{PIK}) has the form of a polynomial kernel. However, instead of the inner product in Eq. (1.10), PI was used. The substitution radial basis kernel (K_{SRBK}) is formulated based on the radial basis kernel of Eq. (1.11) and the substitution distance of Eq. (1.18). The negative substitution distance kernel (K_{NSDK}) is created based on the negative distance kernel (K_{ND}) and the substitution distance of Eq. (1.18). K_{ND} has the form

$$K_{\text{ND}}(\mathbf{s}, \mathbf{s}') = - \parallel \mathbf{s} - \mathbf{s}' \parallel^\beta \qquad \text{with} \quad 0 < \beta \leq 2 \tag{1.22}$$

The authors reported that the kernel functions based on the substitution matrix outperformed the counterparts that do not apply the matrix when tested with a sixfold cross-validation test based on a CB513 data set (Cuff and Barton, 1999). Also, those kernel functions performed slightly better (about 1.5% higher accuracy) than did the radial basis kernel function.

Hybrid Kernels Altun et al. (2006a) designed three different hybrid kernels by combining a substitution matrix (SM)–based kernel, an edit kernel, or a self-organizing map (SOM)–based kernel with an RBF kernel. An example and the algorithm of the hybrid kernel of an SM-based kernel and an RBF kernel (SVM_{SM+RBF}) are given in Figure 1.3, which explain how a sequence segment is given to the hybrid kernel for finding distances. The data encoding given to SVM_{SM+RBF} is shown in Figure 1.4. The data input for each sequence is the PSSM encoding of the sequence and the sequence combined.

The second hybrid kernel is a combination of an edit kernel and an RBF kernel, $SVM_{EDIT+RBF}$ (Altun et al., 2006a). The edit kernel was devised by Li and Jiang (2004) to predict translation initiation sites in eukaryotic mRNAs with SVM. It is based on the string edit distance, which contains biological and probabilistic information. The edit distance is the minimum number of edit operations (e.g., insertion, deletion, substitution) that transform one sequence to another. These edit operations can be considered as a series of evolutionary events. In nature, evolutionary events

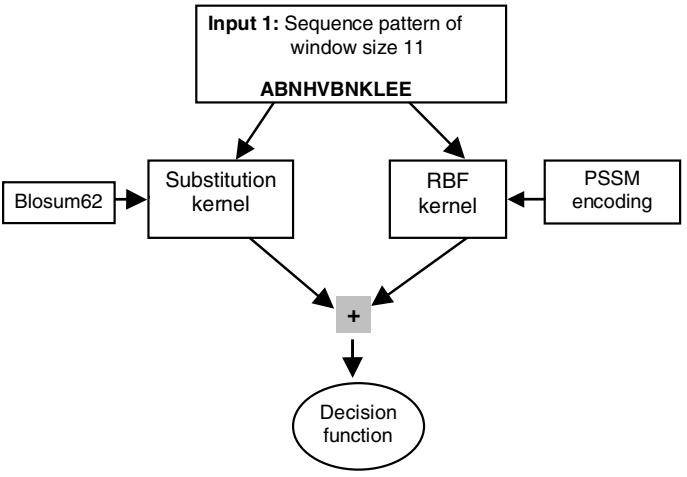

FIGURE 1.3 Example of the SVM_{SM+RBF} algorithm.

FIGURE 1.4 Data encoding for SVM_{SM+RBF} and $SVM_{EDIT+RBF}$.

happen with different probabilities. The authors (Li and Jiang, 2004) defined the edit kernel as follows:

$$K(x, x') = e^{-\gamma \bullet \text{edit}\,(x,x')} \tag{1.23}$$

$$\text{edit}\,(x, x') = -\tfrac{1}{2}\left[\sum_i \log P(x_i|x'_i) + \sum_i \log P(x'_i|x_i)\right] \tag{1.24}$$

where the edit distance is the average of the negative log probability of mutating x into x' and that of mutating x' into x. They modified the 1-PAM matrix to get an asymmetric substitution cost matrix (SCM) for the edit kernel above. Altun et al. combined this edit kernel with the RBF kernel. An example of $\text{SVM}_{\text{EDIT+RBF}}$ is given in Figure 1.5, which explains how a sequence segment is given to the hybrid kernel for finding distances.

The third hybrid kernel is the combination of a self-organizing map (SOM)–based kernel and an RBF kernel, $\text{SVM}_{\text{SOM+RBF}}$ (Altun et al., 2006b). Self-organizing maps are a data visualization technique invented by T. Kohonen, which reduce the dimensions of data through the use of self-organizing neural networks (Kohonen, 1997) and provide a way of representing multidimensional data in lower-dimensional spaces (usually, two dimensions). In SOM, usually a two-dimensional grid of neurons or nodes is used where the grid forms the output space and input data patterns form the input space. Each node has a specific topological position (an x,y coordinate in the grid) and contains a vector of weights of the same dimension as that of the input vectors. SOM initializes its nodes or clusters by random sampling of the data. Then the network of artificial neurons is trained by providing information about the input. When applied this algorithm, vectors that are similar to each other in the initial space remain close on the two-dimensional grid. The input sequence patterns (PSSM

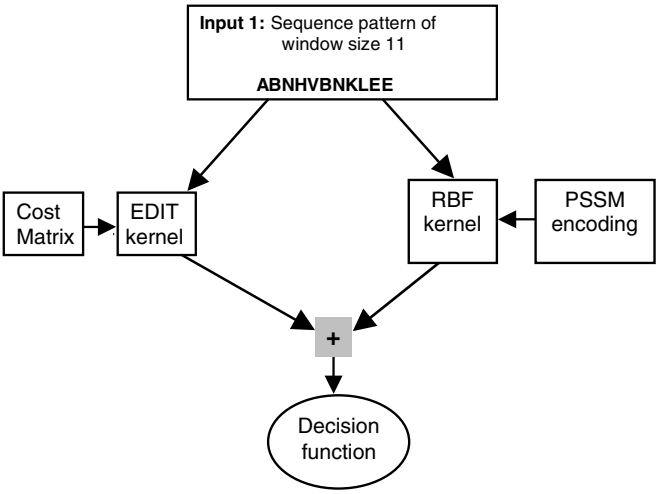

FIGURE 1.5 Example of an $\text{SVM}_{\text{EDIT+RBF}}$ algorithm.

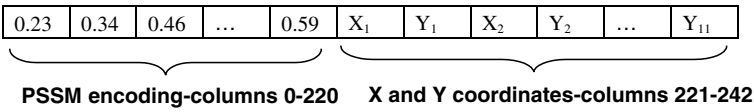

FIGURE 1.6 Data encoding for $SVM_{SOM+RBF}$.

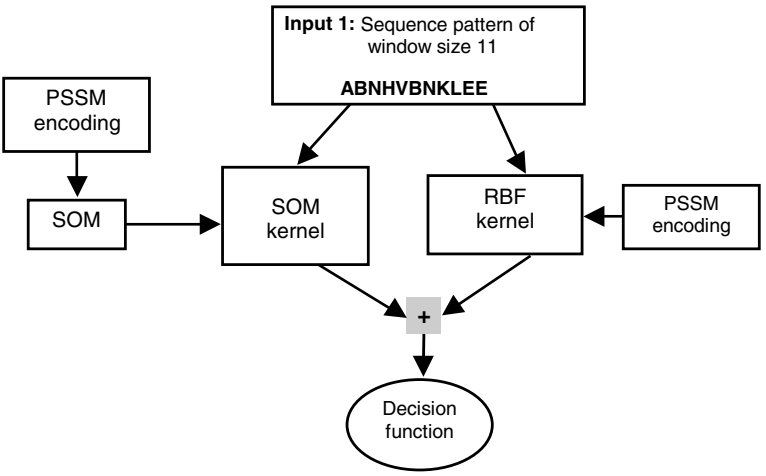

FIGURE 1.7 Example of an $SVM_{SOM+RBF}$ algorithm.

encoding per residue) have been given to SOM and an output value of x and y data points on the grid have been assigned to each residue. Later, these x and y coordinates are used in a new encoding scheme. In this encoding scheme, x and y coordinates and the PSSM encoding of 11 sequence samples are combined. The new encoding scheme is shown in Figure 1.6.

Each encoded input is sent to the hybrid kernel of SOM and RBF. The PSSM encoding part is sent to the RBF kernel, and the x- and y-coordinate part is sent to the SOM kernel (Figure 1.7). The SOM kernel calculates the distance based on the following distance function:

$$d_{\text{som}}(x, x') = \sum_{i=1}^{11} (x_i - x_i')^2 + (y_i - y_i')^2 \tag{1.25}$$

The algorithmic details of the SOM–RBF hybrid kernel are shown in Figure 1.7. The authors reported that $SVM_{SOM+RBF}$ produced results similar to those of the SVM_{SM+RBF} and SVM_{RBF} methods.

1.2.4 Tertiary Classifier Design

SVM is a binary classifier. To apply this binary classifier to three class problems (helix, sheet, coil) of secondary-structure prediction, many studies have attempted to

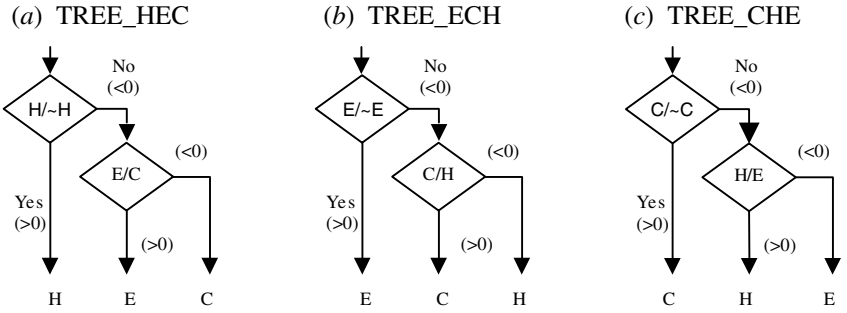

FIGURE 1.8 Tree-based tertiary classifiers. (Based on Hua and Sun, 2001.)

design effective ways of combining the output from binary classifiers. Some methods depend on the result from the six SVM binary classifiers, including three one-versus-rest classifiers (''one'' positive class; ''rest'' negative class): H/∼H, E/∼E, and C/∼C; and three one-versus-one classifiers: H/E, E/C, and C/H. For example, the classifier H/E is constructed on training samples having helices and sheets and classifies the testing sample as a helix or sheet. Some methods depend either on three one-versus-the rest binary classifiers or on three one-versus-one binary classifiers.

Instead of combining the result of binary classifiers, Nguyen and Rajapakse applied direct multiclass SVM for secondary-structure prediction (Nguyen and Rajapakse, 2003). In this section, several tertiary classifiers based on binary classifiers and multiclass SVMs are introduced.

Tree-Based Tertiary Classifier This method (Hua and Sun, 2001) is based on three one-versus-rest binary classifiers (H/∼H, E/∼E, and C/∼C) and three one-versus-one classifiers (H/E, E/C, and C/H). With these classifiers, three cascade tertiary classifiers, TREE_HEC (H/∼H, E/C), TREE_ECH (E/∼E, C/H), and TREE_CHE (C/∼C, H/E), were created. These tree-based classifiers are shown in Figure 1.8.

Simple Voting Tertiary Classifier (SVM_VOTE) In this method (Hua and Sun, 2001), all six binary classifiers are combined by using a simple voting scheme in which the testing sample is predicted to be state i (i is among H, E, or C) if the largest number of the six binary classifiers classify it as state i. If a testing sample has two classifications in each state, it is considered to be a coil. Examples are given in Table 1.1. In the first example case, there are three votes for class C (C/∼C: +2.0,

TABLE 1.1 SVM_VOTE Scheme

Example Cases	H/∼H	E/∼E	C/∼C	H/E	E/C	C/H	Final Class
1	−1.3	−2.7	+2.0	−1.4	−2.3	+1.9	C
2	−0.5	+1.1	−2.6	−0.6	−1.1	+1.8	C
3	+1.9	−0.2	+1.4	+2.2	+1.0	−0.4	H
4	−2.5	+1.7	+2.3	−0.3	+2.5	−0.1	E

TABLE 1.2 SVM_MAX_D Scheme

Example Cases	H/∼H	E/∼E	C/∼C	Final Class
1	−1.3	−2.7	+2.0	C
2	−0.5	+1.1	+0.6	E
3	+1.9	+0.2	−1.4	H
4	−2.5	+1.7	+2.3	C
5	+1.1	−0.1	+1.9	C

E/C: −2.3, and C/H: +1.9). Therefore, the final class was assigned as C. In the second example case, there are two votes for class E (E/∼E: +1.1 and H/E: −0.6) and class C (E/C: −1.1 and C/H: +1.8), respectively. In this case, the final class is assigned as C.

SVM_MAX_D In this classifier (Hua and Sun, 2001), three one-versus-rest classifiers (H/∼H, E/∼E, and C/∼C) are combined for handling the multiclass case. The class of a testing sample (H, E, or C) was assigned to the one that presents the largest positive distance from the optimal separating hyperplane. For example, if the distance values of each one-versus- the rest classifiers (H/∼H, E/∼E, and C/∼C) are −1.7, 1.2, and 2.5, respectively, the negative distance of the H/∼H binary classifier does not give any information for the decision. Only two positive values (1.2,2.5) are compared. Finally, the class for the test sample is assigned to a coil because 2.5 is larger between the two values. More examples are given in Table 1.2.

Tertiary Classifier Combined with a Neural Network (SVM_NN) The outputs of the six binary classifiers are fed into the neural network (NN) (Hua and Sun, 2001). The NN has six units in the input layer, 20 units in the hidden layer, and three units in the output layer. The size of the hidden layer was optimized with various tests. All the parameters, including weights and bias, are decided during the training procedure.

Combined Classifier of All the Results of the Tertiary Classifiers (SVM_JURY) SVM_JURY combines the results of tertiary classifiers using a jury technique (Hua and Sun, 2001). The jury technique is just a majority vote of tertiary classifiers. This technique was used initially in the PHD method to reduce the bad effects of incomplete optimization of a single network by combining the results of a set of networks. Based on the tertiary classifiers that participated in the decision, different versions of SVM_JURY can be created. For example, Hua and Sun (2001) combined all tertiary classifiers, including SVM_MAX_D, SVM_TREEs, SVM_VOTE, and SVM_NN, for a jury decision.

Directed Acyclic Graph–Based Tertiary Classifier The directed acrylic graph (DAG) tertiary classifier (Kim and Park, 2003) is based on three one-versus-one classifiers (H/E, E/C, and C/H) (Figure 1.9). Many test results show that one-versus-one classifiers are more accurate than one-versus-rest classifiers, due to the fact that the one-versus-rest scheme often needs to handle two data sets with very different

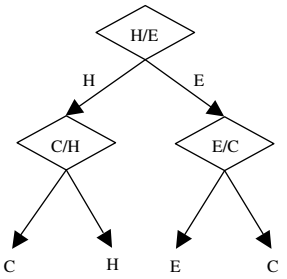

FIGURE 1.9 DAG scheme. (Based on Kim and Park, 2003.)

sizes (i.e., unbalanced training data) (Chang and Lin, 2001; Heiler, 2002). In this scheme, if the testing point is predicted to be H (not E) from the H/E classifier, the C/H classifier is used. If the point is predicted from the H/E classifier, not to be a helix ($\sim$H), the E/C classifier is used to determine if it is a sheet or coil. This DAG scheme is illustrated in Figure 1.9.

SVM_Representative This method is similar to the SVM_MAX_D classifier in that the maximum distance is used for the decision. But unlike the SVM_MAX_D classifier, this scheme (Hu et al., 2004) combines the three one-versus-one binary classifiers (H/E, E/C, and C/H), which give more information than the one-versus-one binary classifier provides. In a one-versus-one classifier, both positive and negative values are meaningful to assign a final class, but in a one-versus-rest classifier, the negative value does not provide specific information for the decision. In this scheme, regardless of whether the distance values are positive or negative, the classifier with the absolute maximum distance is chosen as the representative classifier for the final decision of the class. For example, if the distance values of the one-versus-one classifiers (H/E, E/C, and C/H) are -1.7, 0.4, and -2.5, respectively, the binary classifier with the highest absolute value, here the C/H classifier, can be chosen for deciding the final class. Once this representative classifier is selected, the final class is assigned based on the value of this classifier. In this example, since the value of the C/H classifier shows negative, the final class is assigned as a helix.

Hu et al. (2004) designed this tertiary classifier to combine the result of one-versus-one binary classifiers that are trained with an orthogonal and Blosum62 matrix hybrid encoding profile. The authors claimed that the improved Q_3 accuracy was based on this new tertiary classifier, which captured the information effectively from the binary classifiers. Their scheme was overestimated while combining the results of a seven-fold cross-validation test by failing to exclude preknowledge from the test data. Therefore, for a fair comparison the result was not included in the performance comparison with other typical machine learning approaches.

SVM_Maxpoint The first step of the SVM_Maxpoint scheme (Hu et al., 2006) is to divide all the value pairs into eight different cases based on the possible sign of each classifier. Since there are three classifiers and each one can have a positive or a

TABLE 1.3 Description of SVM_Maxpoint

	Cases in Which Values from Three One-Versus-One Binary Classifiers Fall			Classes Assigned Based on the Signs to the Left		
Possible Cases	H/E	E/C	C/H	H/E	E/C	C/H
1	—	—	—	E	C	H
2	—	—	+	E	C	C
3	—	+	—	E	E	H
4	+	—	—	H	C	H
5	—	+	+	E	E	C
6	+	—	+	H	C	C
7	+	+	—	H	E	H
8	+	+	+	H	E	C

negative sign, the three value pairs can fall into one of eight cases in Table 1.3. If we write a result class based on each sign in the Table 1.3, the second part of the table can be obtained. Since the sign is already used for assigning classes as in Table 1.3, the only thing to be considered now are the absolute decision function values, which indicate the distance from the hyperplane. As can be observed from Table 1.3, except for cases 1 and 8, all cases consist of two of the same class and one different class. Based on this observation, SVM_Maxpoint is formulated as follows:

- *Cases 1 and 8*: Assign the class that has the largest absolute decision function value.
- *Cases 2 through 7*: Add the absolute values of the decision function of two of the same class and compare this added value with the absolute decision function value of one different class. Assign the class that has the larger value.

Example: In case 2, E $= -1.2$, C $= -0.3$, and C $= 1.0$. Then the E point $= 1.2$ and the C point $= 0.3 + 1.0 = 1.3$. Since the C point is greater than the E point, the final class is C.

Multiclass SVM In multiclass SVM, instead of combining the results of binary classifiers [Figure 1.10 (*a*)], all classes are considered in one step [Figure 1.10 (*b*)]. Nguyen and Rajapakse (2003) applied the multiclass SVM method for secondary-structure prediction. The authors examined two multiclass SVM algorithms proposed by Vapnik and Weston (Vapnik, 1998; Weston and Watkins, 1999) and by Crammer and Singer (2000). Even though the authors reported that Vapnik and Weston's multiclass SVM showed better performance than other schemes, including combined binary SVMs, the accuracy improvement is trivial (less than 0.5%). Considering the mathematical complexity of the multiclass SVM, it is not clear that this scheme is more suitable for protein secondary-structure prediction than are combined binary SVMs.

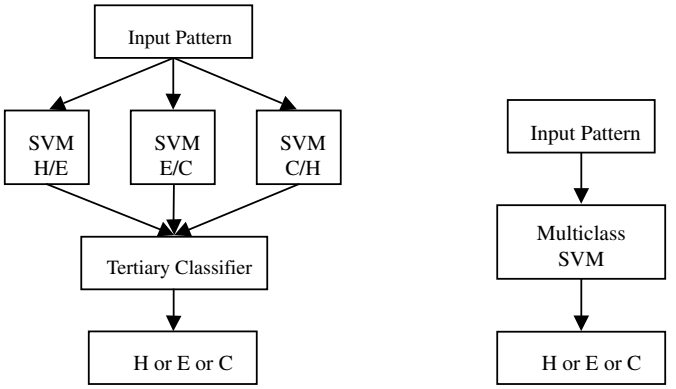

FIGURE 1.10 Combined scheme of binary classifiers and multiclass SVM: (*a*) combined scheme of one-versus-one binary classifiers; (*b*) multiclass SVM.

1.2.5 Accuracy Measure of SVM

There are several standard evaluation methods of secondary-structure prediction; Q_3, Matthew's correlation coefficient, and segment overlap measure (SOV) are widely used assessing methods.

Q_3 Q_3, one of the most commonly used performance measures in protein secondary-structure prediction, refers to the three-state overall percentage of residues predicted correctly. This measure is defined as

$$Q_3 = \frac{\sum\limits_{i\in\{\mathrm{H,E,C}\}} \text{no. of residues predicted correctly}}{\sum\limits_{i\in\{\mathrm{H,E,C}\}} \text{no. of residues in class } i} \times 100 \tag{1.26}$$

Based on Eq. (1.26) the per-residue accuracy for each type of secondary structure (Q_H, Q_E, Q_C) can be obtained as

$$Q_I = \frac{\text{no. of residues correctly predicted in state } I}{\text{no. of residues in state } I} \times 100 \qquad I \in \{\mathrm{H,E,C}\} \tag{1.27}$$

Matthew's Correlation Coefficient, C_i Matthew's correlation coefficient, another measure used in protein secondary-structure prediction, shows how closely the prediction is correlated with the results (Casbon, 2002). Matthew's correlation coefficient is given by

$$C_i = \frac{\mathrm{TP}\times\mathrm{TN} - \mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FN})(\mathrm{TP}+\mathrm{FP})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \qquad i \in \{\mathrm{H,E,C}\} \tag{1.28}$$

where, TP, FP, FN, TN are the number of true positives, false positives, false negatives, and true negatives for class i, respectively, and for clarity the dependency on i has been dropped on the right-hand side. This coefficient value falls on the range between -1 and 1, with 1 showing complete agreement, -1 complete disagreement, and 0 showing that the prediction was uncorrelated with the results.

Segment Overlap Measure Segment overlap measure was developed by Rost et al. (1994) and modified by Zemla et al. (1999) to evaluate the quality of a prediction in a more realistic manner. While all the previous measures are general statistics that can be used in any classification problem, SOV is the specially designed measure for secondary-structure prediction. In secondary-structure prediction, it is important that the continuous structural elements are predicted to exist (Casbon, 2002). Even though Q_3 shows high accuracy value, if a continuous element is not predicted as existing continuously, this cannot be a good prediction. By including this knowledge, SOV became a measure suitable for evaluation of the secondary-structure segment rather than individual residues. SOV is calculated as (Zemla et al., 1999)

$$\text{SOV} = \frac{1}{N} \sum_{i \in \{\text{H,E,C}\}} \sum_{s(i)} \left[\frac{\min \text{ov}(s_1,s_2) + \delta(s_1,s_2)}{\max \text{ov}(s_1,s_2)} \times \text{len}(s_1) \right] \times 100 \qquad (1.29)$$

where N is the normalization value and a sum of $N(i)$ over all three states $= \sum_{i \in \{\text{H,E,C}\}} N(i)$

$$N(i) = \sum_{s(i)} \text{len}(s_1) + \sum_{s'(i)} \text{len}(s_1)$$

$S(i)$ is the set of all overlapping pairs of segments (s_1,s_2) in state $i = \{(s_1,\ s_2) :\ s_1 \cap s_2 \neq \emptyset, \text{in state } i\}$

$S'\ (i)$ is the set of of segments (s_1, s_2) in state i for which there is no overlapping, $= \{s_1\ : \forall s_2,\ s_1 \cap s_2 = \emptyset, \text{in state } i\}$

$\text{len}(s_1)$ is the number of residues in segment $s_1 = e(s_1) - b(s_1) + 1$

$b(s_1)$ is the position at which segment s_1 begins and $e(s_1)$ is the position at which segment s_1 ends

$\text{minov}(s_1, s_2)$ is the length of the actual overlap $= \min(e(s_1), e(s_2)) - \max\ (b(s_1), b(s_2)) + 1$

$\text{maxov}(s_1, s_2)$ is the total extent of the segment expressed as $\max(e(s_1), e(s_2)) - \min(b(s_1), b(s_2)) + 1$

$$\delta(s_1, s_2) = \min\{(\text{maxov}(s_1, s_2) - \text{minov}(s_1, s_2)), \text{minov}(s_1, s_2), \text{int}(\text{len}(s_1)/2), \text{int}(\text{len}(s_2)/2)\}$$

The quality of the matching of each segment pair is taken as a ratio of the overlap of the two segments $\text{minov}(s_1, s_2)$ and the total extent of the pair $\text{maxov}(s_1, s_2)$. The

definition of δ and the normalization factor N differs between SOV94 (Rost et al., 1994) and SOV99 (Zemla et al., 1999).

1.3 PERFORMANCE COMPARISON OF SVM METHODS

Table 1.4 is a performance comparison among typical machine learning approaches with different SVM methods. As can be noted, SVM methods and typical machine learning approaches show comparable performance.

TABLE 1.4 Accuracy Comparison Based on the RS126 Data Set

Method[a]	Q_3 (%)	Q_H (%)	Q_E (%)	Q_C (%)	SOV94 (%)	SOV99 (%)
NNSSP (Salamov and Solovyev, 1995)	72.2	—	—	—	—	—
PHD(93) (Rost and Sander, 1993b)[b]	70.8	72.0	66.0	72.0	73.5	—
PHD(94) (Rost and Sander, 1994)[b]	71.6	—	—	—	—	—
NNetwork_CK (Chandonia and Karplus, 1999)	76.6	—	—	—	—	—
NNetwork_P (Petersen et al., 2000)	80.2	—	—	—	—	—
SSpro2.0 (Pollastri et al., 2002)	78.1	82.4	66.2	81.3	—	—
SVMfreq (Hua and Sun, 2001)[b]	71.2	73.0	58.0	75.0	74.6	—
SVMpsi (Kim and Park, 2003)[b]	76.1	77.2	63.9	81.5	79.6	72.0
SVMpssm_RP (Hu et al., 2006)[b]	75.0	68.7	59.9	86.1	—	66.5
SVMpssm_MP (Hu et al., 2006)[b]	76.4	72.1	62.5	85.1	—	68.3

[a]NNSSP is the prediction result of the nearest-neighbor method by Salamov and Solovyev (1995); PHD(93) is the neural network prediction result obtained by Rost and Sander (1993b); PHD(94) is the neural network prediction result obtained by Rost and Sander (1994); NNetwork_CK is the neural network prediction result of Chandonia and Karplus (1999); NNetwork_P is the neural network prediction result of Petersen et al. (2000); SSpro2.0 is the neural network prediction result of Pollastri et al. (2002); SVMfreq is the prediction result of SVM by Hua and Sun (2001); SVMpsi is the prediction result of SVM obtained by Kim and Park (2003); SVMpssm_RP is the prediction result of SVM with PSSM encoding and SVM_Representative tertiary classifier by Hu et al. (2006); SVMpssm_MP is the prediction result of SVM with PSSM encoding and SVM_Maxpoint tertiary classifier by Hu et al. (2006).
[b]The combined results of a sevenfold cross-validation based on RS126 are shown. In other cases, RS126 is used as an independent test set.

1.4 DISCUSSION AND CONCLUSIONS

SVM schemes are successful and promising machine learning methods for secondary-structure prediction. With strong generalization ability, they outperformed most machine learning methods. However, similar to neural network methods, the SVMs are black box models. They could not generate logical models that explain the predictions they make. In the area of bioinformatics, the ability to explain the underlying principles of the decision is especially crucial, since based on this principle, ''wet experiments'' are guided. In addition, the rules extracted can be applied to make decisions in the future. Therefore, poor comprehensibility has been considered a big challenge to the success of SVM. To overcome this drawback, recently, much research has attempted to extract rules from an SVM decision (Mitsdorffer et al., 2002; Barakat and Diederich, 2004; He et al., 2006a,b). With the help of these efforts, SVM is gradually obtaining comprehensibility. Even though it will take more time and effort to obtain rules that are biologically meaningful, considering the achievements so far, the future of the SVM approach for secondary-structure prediction is optimistic.

REFERENCES

Altun, G., H. Hu, D. Brinza, R. Harrison, A. Zelikovsky, and Y. Pan (2006a). Hybrid SVM kernels for protein secondary structure prediction, *IEEE International Conference on Granular Computing* (GrC'06), Atlanta, GA.

Altun, G., H. Hu, D. Brinza, R. Harrison, A. Zelikovsky, and Y. Pan (2006b). SOM kernel based SVM for protein secondary structure prediction, Technical Report, Computer Science Department, Georgia State University, Atlanta, GA.

Baldi, P., S. Brunak, P. Frasconi, G. Pollastri, and G. Soda (1999). Exploiting the past and the future in protein secondary structure prediction, *Bioinformatics*, 15, pp. 937–946.

Barakat, N., and J. Diederich (2004). Learning-based rule-extraction from support vector machine, *Third Conference on Neuro-computing and Evolving Intelligence* (NCEI'04).

Biou, V., J. F. Gibrat, J. M. Levin, B. Robson, and J. Garnier (1988). Secondary structure prediction: combination of three different methods, *Protein Eng.*, 2, pp. 185–191.

Blout, E., C. de Loz, S. Bloom, and G. D. Fasman, (1960). Dependence of the conformation of synthetic polypeptides on amino acid composition, *J. Am. Chem. Soc.*, 82, pp. 3787–3789.

Bowie, J. U., R. Luthy, and D. Eisenberg (1991). A method to identify protein sequences that fold into a known three-dimensional structure, *Science*, 253, pp. 164–170.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition, *Data Min. Knowledge Discovery*, 2, pp. 121–167.

Bystroff, C., V. Thorsson, and D. Baker (2000). HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins, *J. Mol. Biol.*, 301, pp. 173–190.

Casbon, J. (2002). *Protein Secondary Structure Prediction with Support Vector Machines*, M. Sc. thesis, University of Sussex, Brighton, UK.

Chandonia, J. M., and M. Karplus (1999). New methods for accurate prediction of protein secondary structure, *Proteins: Struct. Funct. Genet.*, 35, pp. 293–306.

Chang, C. C., and C. J. Lin (2001). LIBSVM: a library for support vector machines, software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Chou, P. Y. (1989). *Prediction of Protein Structure and the Principles of Protein Conformation*, Plenum Press, New York.

Chou, P. Y., and G. D. Fasman (1974). Prediction of protein conformation, *Biochemistry*, 13, pp. 211–215.

Crammer, K., and Y. Singer (2000). On the learnability and design of output codes for multi-class problems, *Comput. Learn. Theory*, pp. 35–46.

Cristianini, N., and J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines*, Cambridge University Press, New York.

Cuff, J. A., and G. J. Barton (1999). Evaluation and improvement of multiple sequence methods for protein secondary structure prediction, *Proteins: Struct. Funct. Genet.*, 34, pp. 508–519.

Eidhammer, I., I. Jonassen, and W. R. Taylor (2004). *Protein Bioinformatics: An Algorithmic Approach to Sequence and Structure Analysis*, Wiley, Hoboken, NJ.

Frishman, D., and P. Argos (1997). 75% accuracy in protein secondary structure prediction, *Proteins*, 27, pp. 329–335.

Garnier, J., D. J. Osguthorpe, and B. Robson (1978). Analysis and implications of simple methods for predicting the secondary structure of globular proteins, *J. Mol. Biol.*, 120, pp. 97–120.

Gonnet, G. H., M. A. Cohen, and S. A. Benner (1992). Exhaustive matching of the entire protein sequence database, *Science*, 256, pp. 1143–1445.

He, J., H. Hu, R. Harrison, P. C. Tai, and Y. Pan (2006a). Rule generation for protein secondary structure prediction with support vector machines and decision tree, *IEEE Trans. Nano-Biosci.*, 5(1), pp. 46–53.

He, J., H. Hu, R. Harrison, P. C. Tai, and Y. Pan (2006b). Transmembrane segments prediction and understanding using support vector machine and decision tree, *Expert Syst. Appl.* (Special Issue on Intelligent Bioinformatics Systems), 30(1), pp. 64–72.

Heiler, M. (2002). *Optimization Criteria and Learning Algorithms of Large Margin Classifiers*, University of Mannheim, Mannheim, Germany.

Hu, H., Y. Pan, R. Harrison, and P. C. Tai (2004). Improved protein secondary structure prediction using support vector machine with a new encoding scheme and an advanced tertiary classifier, *IEEE Trans. NanoBiosci.*, 3(4), pp. 265–271.

Hu, H., P. C. Tai, J. He, R. W. Harrison, and Y. Pan (2006). A novel tertiary classifier for protein secondary structure prediction based on support vector machine and a PSSM profile, Technical Report, Computer Science Department, Georgia State University, Atlanta, GA.

Hua, S., and Z. Sun (2001). A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach, *J. Mol. Biol.*, 308, pp. 397–407.

Joachims, T. (1999). *Making Large Scale SVM Learning Practical*, MIT Press, Cambridge, MA.

Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices, *J. Mol. Biol.*, 292(2), pp. 195–202.

Kim, H., and H. Park (2003). Protein secondary structure prediction based on an improved support vector machines approach, *Protein Eng.*,16, pp. 553–560.

Kloczkowski, A., K. L. Ting, R. L. Jernigan, and J. Garnier (2002). Combining the GOR V algorithm with evolutionary information for protein secondary structure prediction from amino acid sequence, *Proteins: Struct. Funct. Genet.*, 49, pp. 154–166.

Kohonen, T. (1997). *Self-Organizating Maps*, Springer-Verlag, New York.

Krogh, A., M. Brown, I. S. Mian, K. Sjolander, and D. Haussler (1994). Hidden Markov models in computational biology: applications to protein modeling, *Biology*, 235, pp. 1501–1531.

Kyngas, J., and J. Valjakka (1998). Unreliability of the Chou–Fasman parameters in predicting protein secondary structure, *Protein Eng.*, 11(5), pp. 345–348.

Levin, J. M. (1997). Exploring the limits of nearest neighbor secondary structure prediction, *Protein Eng.*, 10(7), pp. 771–776.

Levin, J. M., and J. Garnier (1988). Improvements in a secondary structure prediction method based on a search for local sequence homologies and its use as a model building tool, *Biochim. Biophys. Acta*, 955(3), pp. 283–295.

Li, H., and T. Jiang (2004). A class of edit kernels for SVMs to predict translation initiation sites in eukaryotic mRNAs, *J. Comput. Biol.*, 12, pp. 702–718.

Nishikawa, K., and T. Ooi (1986). Amino acid sequence homology applied to the prediction of protein secondary structure, and joint prediction with existing methods, *Biochim. Biophys. Acta*, 871, pp. 45–54.

Nguyen, M. N., and J. C. Rajapakse (2003). Multi-class support vector machines for protein secondary structure prediction, *Genome Inf.*, 14, pp. 218–227.

Mitsdorffer, R., J. Diederich, and C. Tan (2002). Rule-extraction from Technology IPOs in the U.S. stock market, ICONIP'02, Singapore.

Müller, K. R., S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf (2001). An introduction to kernel-based learning algorithms, *IEEE Trans. Neural Networks*, 12(2), pp. 181–201.

Petersen, T. N., C. Lundegaard, M. Nielsen, H. Bohr, J. Bohr, S. Brunak, P. G. Gippert, and O. Lund (2000). Prediction of protein secondary structure at 80% accuracy, *Proteins: Struct. Funct. Genet.*, 41, pp. 17–20.

Pollastri, G., D. Przybylski, B. Rost, and P. Baldi (2002). Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles, *Proteins: Struct. Funct. Genet.*, 47, pp. 228–235.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition, *Proc. IEEE*, 77, pp. 257–286.

Radzicka, A., and R. Wolfenden (1988). Comparing the polarities of the amino acids: side-chain distribution coefficients between the vapor phase, cyclohexane, 1-octanol, and neutral aqueous solution, *Biochemistry*, 27, pp. 1664–1670.

Riis, S. K., and A. Krogh (1996). Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments, *J. Comput. Biol.*, 3, pp. 163–183.

Rost, B., and C. Sander (1993a). Improved prediction of protein secondary structure by use of sequence profiles and neural networks, *Proc. Natl. Acad. Sci. USA*, 90, pp. 7558–7562.

Rost, B., and C. Sander (1993b). Prediction of protein secondary structure at better than 70% accuracy, *J. Mol. Biol.*, 232, pp. 584–599.

Rost, B., and C. Sander (1994). Combining evolutionary information and neural networks to predict protein secondary structure, *Proteins*, 19, pp. 55–72.

Rost, B., C. Sander, and R. Schneider (1994). Redefining the goals of protein secondary structure prediction, *J. Mol. Biol.*, 235, pp. 13–26.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning representations by back-propagating errors, *Nature*, 323, pp. 533–536.

Salamov, A. A., and V. V. Solovyev (1995). Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments, *J. Mol. Biol.*, 247, pp. 11–15.

Salamov, A. A., and V. V. Solovyev (1997). Protein secondary structure prediction using local alignments, *J. Mol. Biol.*, 268, pp. 31–36.

Vanschoenwinkel, B., and B. Manderick (2004). Substitution matrix based kernel functions for protein secondary structure prediction, *Proc. ICMLA*.

Vapnik, V., and C. Cortes (1995). Support vector networks, *Mach. Learn.*, 20, pp. 273–293.

Vapnik, V. (1998). *Statistical Learning Theory*, Wiley, New York.

Weston, J., and C. Watkins (1999). Multi-class support vector machines, in M. Verleysen, Ed., *Proc. ESANN'99*, De Facto Press, Brussels, Belgium.

Won, K. J., T. Hamelryck, P. B. Adam, and A. Krogh (2005). Evolving hidden Markov models for protein secondary structure prediction, *Proc. IEEE Congress on Evolutionary Computation*, pp. 33–40.

Yi, T. M., and E. S. Lander (1993). Protein secondary structure prediction using nearest-neighbor methods, *J. Mol. Biol.*, 232, pp. 1117–1129.

Zemla, A., C. Venclovas, K. Fidelis, and B. Rost (1999). A modified definition of SOV, a segment-based measure for protein secondary prediction assessment, *Proteins: Struct. Funct. Genet.*, 34, pp. 220–223.

Zimmermann, K. (1994). When awaiting ''bio'' champollion: dynamic programming regularization of the protein secondary structure predictions, *Protein Eng.*, 7, pp. 1197–1202.