CHAPTER III

SVM AND MARKOV MODEL APPROACH FOR PROTEIN SECONDARY STRUCTURE PREDICTION

3.1 Support Vector Machine

Support Vector Machine is a type of classification method, which estimates a classification function. This method is based on the work of Vladimir Vapnik in statistical learning theory [30]. Like other learning machines, such as multi-layer perceptrons and radial basis function networks, the Support Vector Machine (SVM) is capable of classification and non-linear regression. However, SVM is attractive because it is an extremely well developed statistical learning theory based on the principle of structural risk minimization that guarantees the lowest error. Accordingly, SVM can provide a good generalization performance despite the fact that it does not incorporate problem-domain knowledge [21]. The SVM approach is not only theoretically provided, but also superior in practical applications, especially in Bioinformatics [31]. There are many applications of SVM in Bioinformatics such as; detection of remote protein homologies [32], recognition of translation initiation sites [33] and prediction of protein secondary structure [34, 35].

The main idea of a support vector machine is to construct a hyperplane as a decision surface in such a way that the margin of separation between positive and negative examples is maximized. In the process of training the SVM, a nonlinear Φ-function that maps the input vectors to a higher dimensional space is chosen. This function may be informed by the designer's knowledge of the problem domain or be chosen by using polynomials, Gaussians, or yet other basis functions. So, the dimensionality of the mapped spaces, so-called feature spaces, can be arbitrarily high. The mapping function is represented in forms of a positive definite kernel function, *k(x,x')*, which is easier to calculate the inner product in the feature spaces. The inner product is specified by the following from:

$$k(x,x') = (\Phi(x) \cdot \Phi(x')) \tag{3.1}$$

We do not need to know Φ, because the mapping is not performed explicitly. A common kernel function performing the efficient mapping from input spaces into a high dimensional feature space for complex classification problems is the Gaussian radial basis function (RBF) shown in equation (3.2)

$$k(x,x') = exp\left(-\gamma\|x-y\|^2\right) \tag{3.2}$$

Currently, the theory of kernalizing the input data for machine learning approach is still in active research area. RBF kernel function in our learning models was used, after almost all kernel functions currently developed had been tested and compared the results.

In basic formulation, SVM finds a linear decision function, *f(x)=sign(***w***.x+b)*, that minimizes the prediction error on the training set and promises the best generalization performance. To construct the optimum margin hyperplane for the soft-margin SVMs, we solve the following optimization problem:

$$\text{minimize} \qquad \Gamma(\mathbf{w},\xi)=\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{m}\xi_i \tag{3.3}$$

$$\text{subject to} \qquad y_i\cdot(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1-\xi_i,\ \xi_i\geq 0, \quad \mathrm{i}=1,\ldots,\mathrm{m}$$

This constrained optimization problem is solved by introducing Lagrange multipliers $0\leq\alpha_i\leq C$ and a Lagrangian:

$$L(\mathbf{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{m}\alpha_i\left(y_i\cdot(\mathbf{x}_i\cdot\mathbf{w}+b)-1\right) \tag{3.4}$$

At the saddle point, we get:

$$\sum_{i=1}^{m}\alpha_i y_i=0 \tag{3.5}$$

and

$$\mathbf{w}=\sum_{i=1}^{m}\alpha_i y_i\mathbf{x}_i \tag{3.6}$$

To find the multipliers α_i, we substitute (3.5) and (3.6) in to (3.4), and solve the following optimization problem:

$$\text{maximize} \qquad \mathbf{w}(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i x_j) \tag{3.7}$$

$$\text{subject to} \qquad 0\le\alpha_i\le C,\ \ \mathrm{i}=1,\ldots,\ \mathrm{m},\ \text{ and } \sum_{i=1}^{m}\alpha_i y_i=0 \tag{3.8}$$

where x_i represents an input vector, y_i is either +1 or –1 depending on whether x_i being in positive or negative class, m is the number of the training data, and $C\ge 0$ is the regularization parameter which selecting by the user. This parameter is used to balance the trade-off between margin and classification error represented by slack variables ξ_i.

The hyperplan vector $\mathbf{w}$ is represented in terms of the training examples (x_i, y_i) and their Lagrangian multipliers(α_i), which are calculated during the optimization process:

$$\mathbf{w}=\sum_{i\in I}\alpha_i y_i \mathbf{x}_i \tag{3.9}$$

The hyperplane decision function can thus be written as:

$$f(x)=sign\left(\sum_{i=1}^{m}y_i\alpha_i\cdot k(x,x_i)+b\right) \tag{3.10}$$

3.2 Markov Model

Markov Model - named after the Russian mathematician *Andrei A. Markov* (1856–1922) - is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. It is a process that generates a sequence in which the probability of the occurrence of state *i* depends only on the previous occurrence at the state *i*-1. The definition of the model can be written as follows:

$$P(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i \mid x_{i-1}) \tag{3.11}$$

This model for the production of any sequence is described by transition probabilities:

$$a_{st} = P(x_i = t \mid x_{i-1} = s) \tag{3.12}$$

These parameters of the model are typically estimated from large sets of trusted examples. For instance, the transition probability a_{st} for amino acid T followed amino acid S can be estimated as the observed frequency of two connected residues in database of known protein structure, such as Protein Data Bank (PDB). This method of estimating models is called maximum likelihood estimation [36].

For representative models of the secondary structure of proteins, we derive three Markov models for the three classes of secondary structure, H(Helix), E(Strand) and C(Coil). The transition probabilities for each model are calculated using the equation:

$$a^{i}_{st} = \frac{c^{i}_{st}}{\sum_{t'} c^{i}_{st'}} \tag{3.13}$$

where c^{i}_{st} is frequency of amino acid T followed S for sequences in class $i \in \{H,E,C\}$.

The model we have mentioned so far is technically called "a first-order discrete time Markov model", because the probability at state *i* depends only on the state *i*-1. A general from of Markov model, n^{th}-order Markov model, can be explained by the following equation:

$$P(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i \mid x_{i-1}, \ldots, x_{i-n}) \tag{3.14}$$

Thus, the transition probabilities of a third-order Markov chains are:

$$a_{qrst} = P(x_i = t \mid x_{i-1} = s, x_{i-2} = r, x_{i-3} = q) \tag{3.15}$$

where *t* is amino acid residue at position *i* and *s*, *r* and *q* are amino acid residues at the previous positions, *i*-1, *i*-2 and *i*-3, respectively. We can use the same process to deal with any order of the models.

This method of data representation is considered to be a pattern representation of connected amino acid. For example, the third-order Markov chain is represented for the pattern of connected four residues so-called 4-mers, and so the forth order is represented for 5-mers, respectively.

The Markov models can be applied to capture the information embedded in biological sequences in forms of transition a matrix, so-called Makov transition matrix. The Markov transition matrix has been extensively used for sequence analysis and given satisfactory results in many applications such as the method for identifying splice sites and translational start sites [33, 37] and the method for detection of eukaryotic promoters [38]. Furthermore, it is the core algorithm of the popular GenMark program [39].

For protein sequence, the transition matrix is constructed by define the columns to be positions on fixed windows on the sequences. Each entry of the matrices contains a transitional probability of adjacent residues. In our experiment, we apply the Markov transition matrix as a new encoding scheme to produce the input vectors for the learning models.

3.3 Multi-Step Markov model

If one considers the basis structure of α-helix conformation, one can see the H-bond linkage of residues $i+1$ and $i\pm4$ is the major potential force to formulate the helix structure class. With this phenomenon, we can model that linkage property with an extended Markov Chain, so-called multi-step Markov Chains.

The multi-step Markov Model can be considered to be a jumping chains whose the probability of the occurrence of state i depends only on the m previous occurrence at the state i-m. The definition of the model can be written as follows:

$$P(x_i \mid x_{i-m}, x_{i-2m}, \ldots, x_1) = P(x_i \mid x_{i-m}) \tag{3.16}$$

Therefore, a general from of multi-step Markov model, multi-step n^{th}-order Markov model, can be explained by the following equation:

$$P(x_i \mid x_{i-m}, x_{i-2m}, \ldots, x_{i-nm}, \ldots, x_1) = P(x_i \mid x_{i-m}, x_{i-2m}, \ldots, x_{i-nm}) \quad (3.17)$$

Thus, the transition probability of a multi-step third-order Markov chains becomes:

$$a_{qrst} = P(x_i = t \mid x_{i-m} = s, x_{i-2m} = r, x_{i-3m} = q) \quad (3.18)$$

where m is the number of jumping step, t is amino acid residue at position i and s, r and q are amino acid residues at the previous position, i-m, i-$2m$ and i-$3m$, respectively.

The transition matrix of multi-step Markov model can be constructed in the same way as the single-step model. Instead of using the window connected residues, the windows of jumping step of closest residues is used to be the data representation. After the Markov Transition Matrices have been created, the processes of input vector construction can be followed by the same procedure as the single-step Markov Model.

3.4 Vectors Representation of Protein Sequences

The orthogonal binary vector , a conventional encoding scheme of protein sequences, have been used since the first Neural Network models of protein secondary structure prediction was introduced by Qian and Sejnowski [8]. Five years later, the extension of this encoding scheme, that incorporates the profile information from multiple sequence alignment, was introduced [9]. Recently, the most efficient alignment profile comes from a position specific scoring matrix (PSSM) generated from PSI-BLAST program. The encoding scheme based on PSSM was introduced by David T. Jones [10]. Even though the input vectors generated from those encoding schemes can be used to represent the feature of protein sequences, they contain the enormously high dimensional vector spaces. For instance, if we use a window size of only 15 residues, the size of input vector takes about 15×20 = 300 dimensions per pattern. Consequently, the performance of learning models is deteriorated by that bad property. In other words, the larger input dimensions grow, the more it increases a number of degrees of freedom. So that, with the limited training patterns, the learning models is prone to be ill generalization.

As the limitation of the conventional encoding scheme, we have introduced a new method that efficiently used to represent the input feature of protein sequences. Instead of using the orthogonal binary vector in forms of a sparse matrix format, we adopted the n^{th}-order Markov transition matrices to produce the input vectors. The transition matrices are generated by defining the columns to be positions on fixed windows of protein sequences. (see Figure 3.2) The size of windows can be varied depending on the performance of each classifier on specific secondary structure classes. Each entry of the matrices contains the transitional probability of residues estimated from the database of known protein structure. To estimate these parameters, a set of known protein structures from the DSSP, the database of secondary structure assignments for all protein entries in the Protein Data Bank [5] was used. This database consists of 18,947 currently known protein structures. For preventing a bias of the data set that used to validate our model, we removed those sequences in the validation set from the DSSP database. Then, with this database, the Markov transition matrices are constructed separately for each secondary structure class, H (helix), E (sheet) and C (coil) (see Figure 3.3, 3.4, 3.5 and algorithms follows), and are used to create the input vectors for each classifier of those classes. The following algorithms are the step by step to construct the Markov transition matrices and Input Vectors.

Algorithm 3.1: Construction of Markov Transition Matrix and Input Vectors

1) The patterns of sequences from the DSSP database, which is used to estimate the free parameters of the transition matrices, are prepared by using the sliding window.
2) Assigning the secondary structure classes (H, E or C) depends on the structural class of the middle residue on to each pattern.
3) For each class of sequence patterns, the transition matrices are created by observing the frequency of transition on connected residues at each position in the sequences.
4) The transition probabilities for each matrix's element are computed using equation:

$$a^i_{st} = \frac{c^i_{st}}{\sum_{t'} c^i_{st'}} \tag{3.19}$$

   where c^i_{st} is the frequency of amino acid T followed S for class $i \in \{H,E,C\}$
5) The sequence patterns of data sets (training and validating sets) are prepared in the same way using a sliding window.

6) To construct the input vectors, log-odds ratio of the transition probabilities obtained from the Markov transition matrices are calculated by:

$$b_{st}^{i} = \log\left(\frac{a_{st}^{i}}{\sum_{i} a_{st}^{i}}\right), \quad i \in \{H, E, C\} \tag{3.20}$$

7) The log-odds ratios at the corresponding positions on sequence patterns are used to be the input vectors for learning models.

Using the same algorithms (on steps 1 to 4), the first, second, third and fourth-order Markov transition matrices are produced by considering two, three, four and five adjacent residues respectively. The input vectors created from those models take into account the dependencies between adjacent amino acid residues at each position on the sliding window. Whereas these vectors can effectively represent the informative local interaction features of protein sequences, they utilize only $w - n$ dimensions of vector spaces; where w and n are size of window and order of Markov model respectively.

The property of 4th-order transition matrix is an array of 5 dimensions. Each dimension represents the amino acid position on sliding window. Value of each element in the array is the number of amino acid patterns that connected in five residues (5-mers). For example, in array of helix's structure, the number of patterns that have helix structure in the middle residue are counted and accumulated into the array. The patterns of other structures that repeatedly occur in data set will be counted by the same procedure.

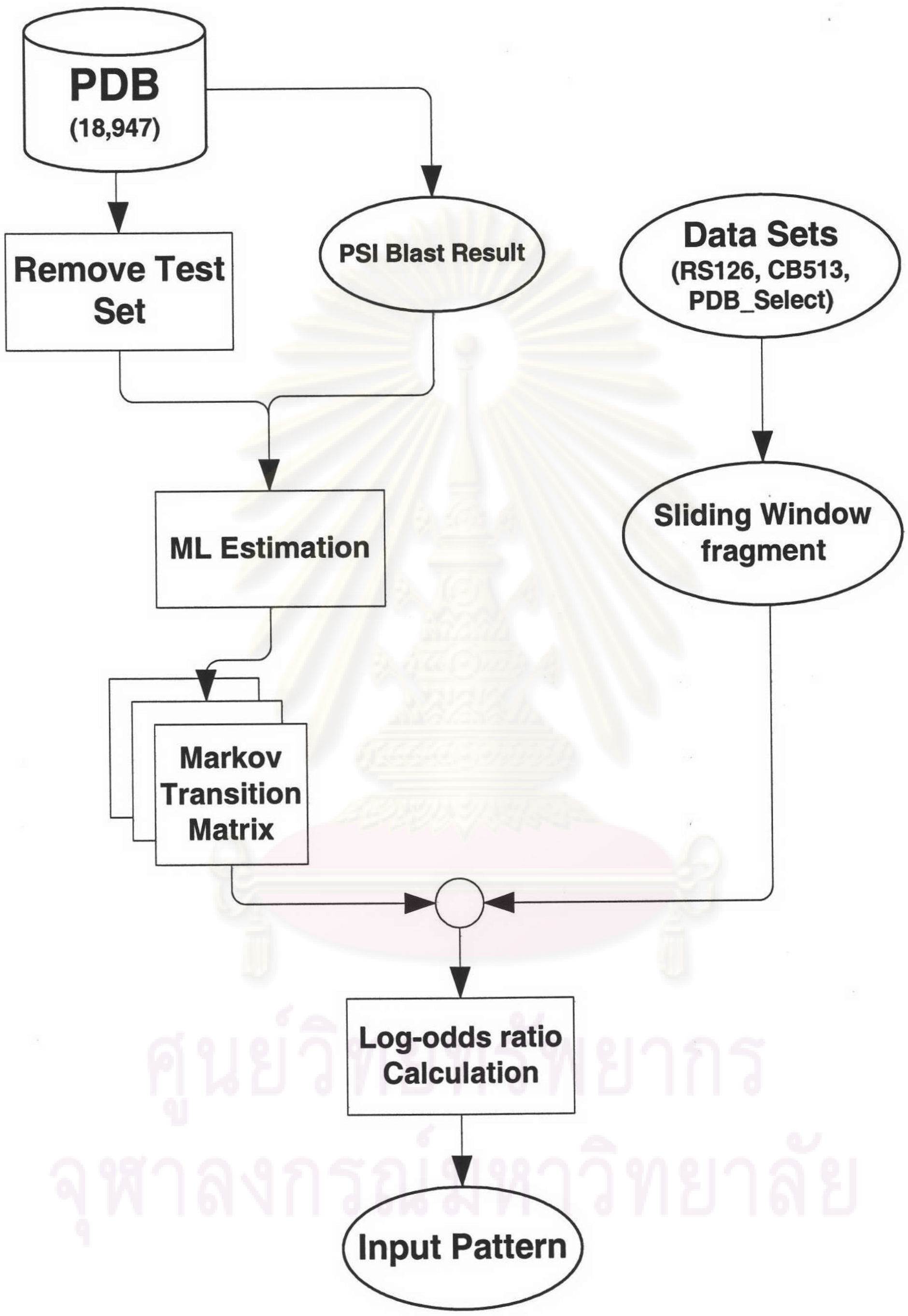


Figure 3.1 Data flow of input patterns preparation process.

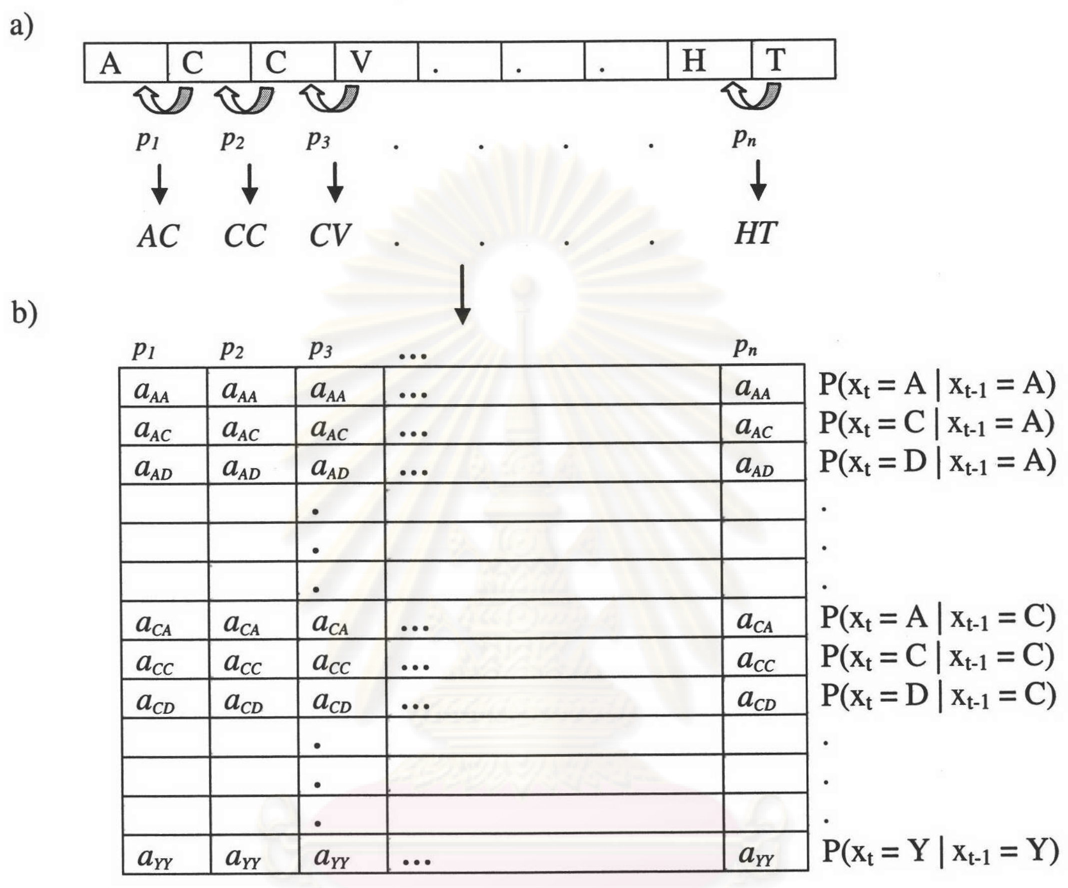

a)
A C C V . . . H T
p_1 p_2 p_3 . . . . p_n
AC CC CV . . . . HT
b)
p_1 p_2 p_3 … p_n
a_{AA} a_{AA} a_{AA} … a_{AA} $P(x_t = A \mid x_{t-1} = A)$
a_{AC} a_{AC} a_{AC} … a_{AC} $P(x_t = C \mid x_{t-1} = A)$
a_{AD} a_{AD} a_{AD} … a_{AD} $P(x_t = D \mid x_{t-1} = A)$
a_{CA} a_{CA} a_{CA} … a_{CA} $P(x_t = A \mid x_{t-1} = C)$
a_{CC} a_{CC} a_{CC} … a_{CC} $P(x_t = C \mid x_{t-1} = C)$
a_{CD} a_{CD} a_{CD} … a_{CD} $P(x_t = D \mid x_{t-1} = C)$
a_{YY} a_{YY} a_{YY} … a_{YY} $P(x_t = Y \mid x_{t-1} = Y)$

Figure 3.2 Markov transition Matrix
a) Patterns of protein sequences are prepared using sliding window. At each position on the window, the frequency of transition on connected residues is observed.
b) The 1st order Markov transition matrix is constructed using the transition probabilities estimated by maximum likelihood estimation.

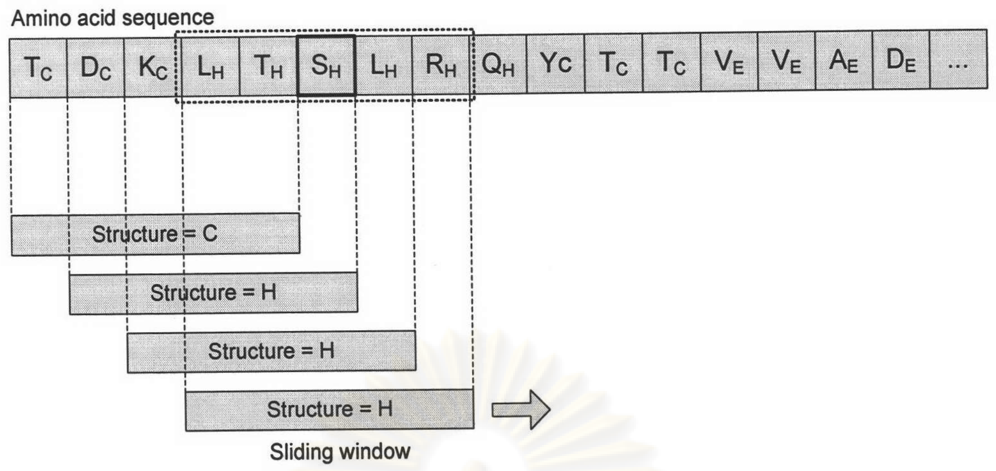
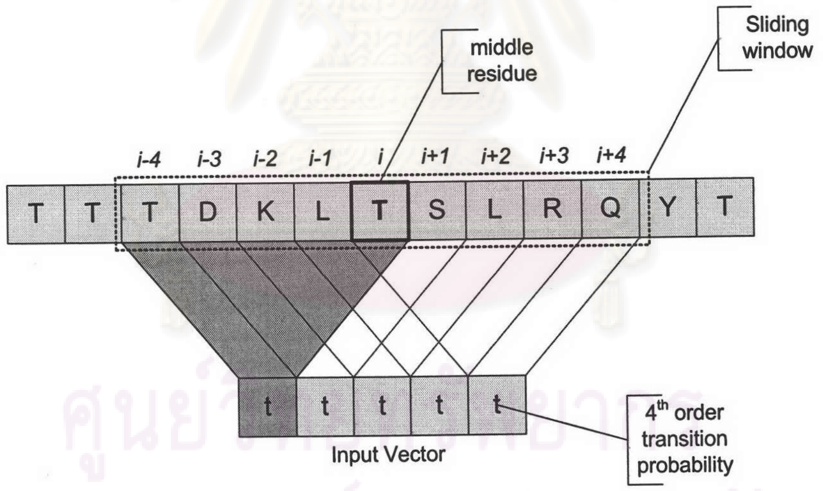


Figure 3.3 An amino acid pattern constructed by sliding window; the pattern is represented for the structural class of the middle residue of each window.



Sliding window
middle residue
i-4 i-3 i-2 i-1 i i+1 i+2 i+3 i+4
T T T D K L T S L R Q Y T
t t t t t
Input Vector
4th order transition probability

Figure 3.4 The input vector created from 4th order Markov Chain; the size of input dimension, s, is determined by the window's size, w (dot box), that is $s = w - n$; where n is the order of Markov Chain.

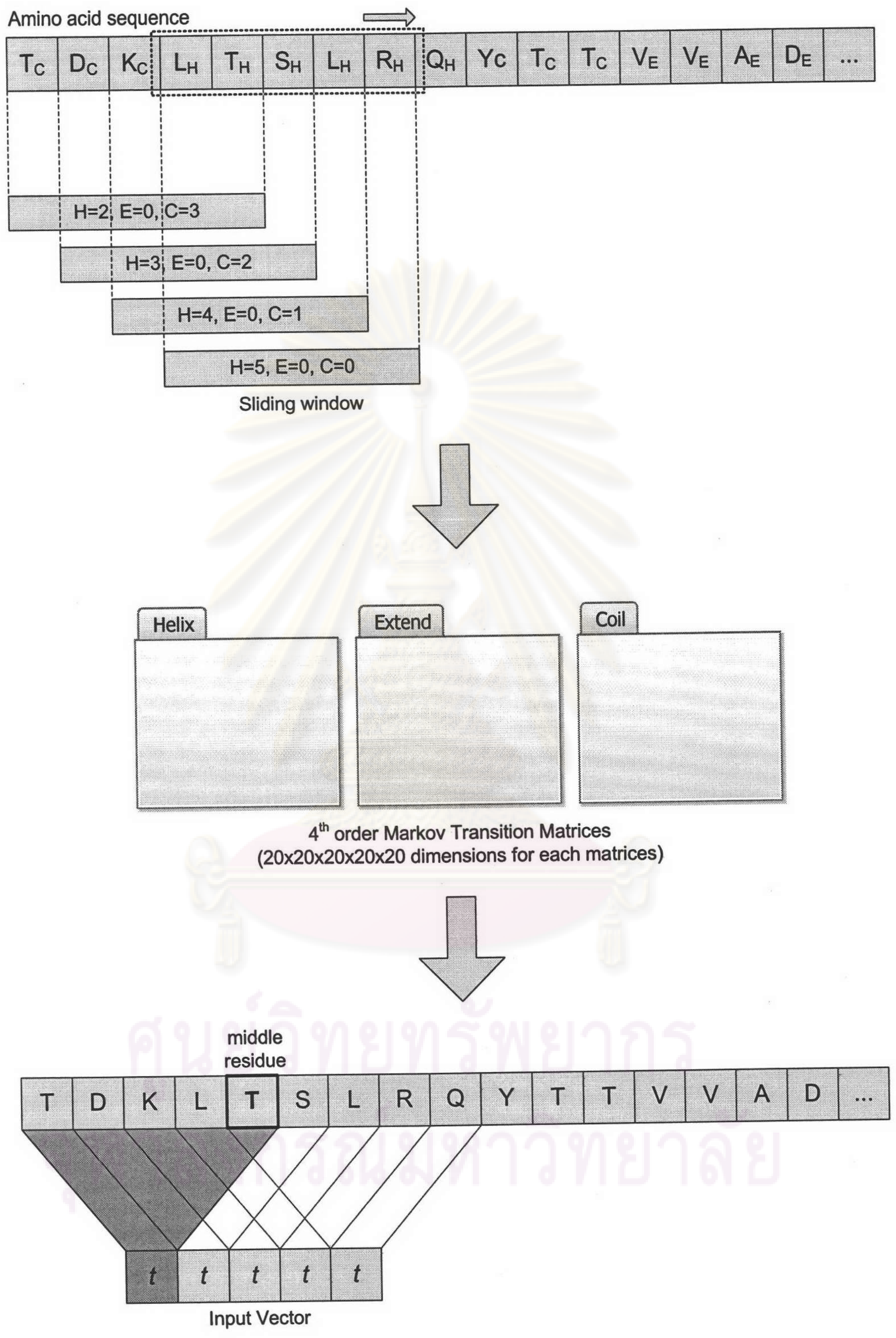
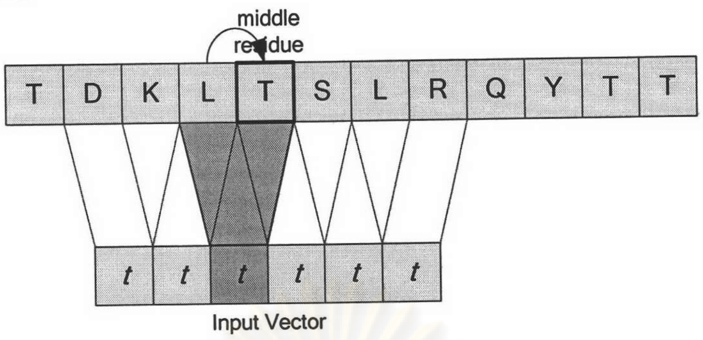
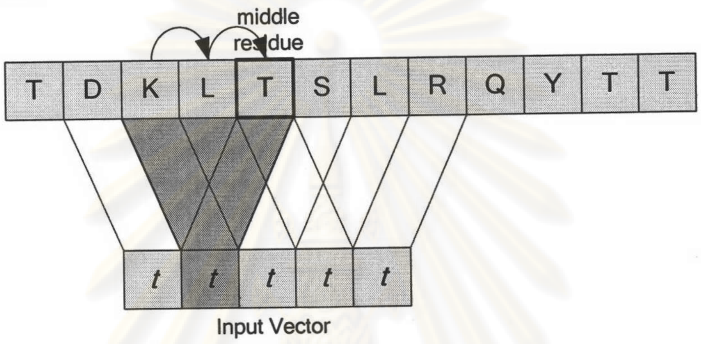

Amino acid sequence
T$_C$ D$_C$ K$_C$ L$_H$ T$_H$ S$_H$ L$_H$ R$_H$ Q$_H$ Y$_C$ T$_C$ T$_C$ V$_E$ V$_E$ A$_E$ D$_E$ ...
H=2, E=0, C=3
H=3, E=0, C=2
H=4, E=0, C=1
H=5, E=0, C=0
Sliding window
Helix
Extend
Coil
4th order Markov Transition Matrices
(20x20x20x20x20 dimensions for each matrices)
middle residue
T D K L T S L R Q Y T T V V A D ...
t t t t t
Input Vector

Figure 3.5 The constructing process of Input Vectors;
using the 4th order Markov Transition Matrices constructing with normalization and interpolation methods.

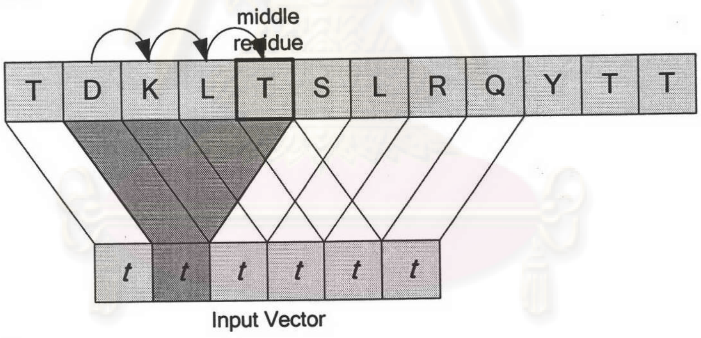
a) 1st order



b) 2nd order



c) 3rd order



d) 4th order

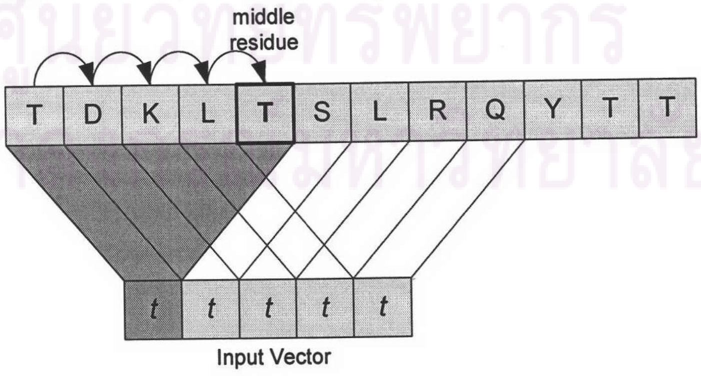


Figure 3.6 The input vector that created from high-order Markov chain; the 1st 2nd 3rd and 4th orders Markov Chain, a), b), c), and d), respectively.

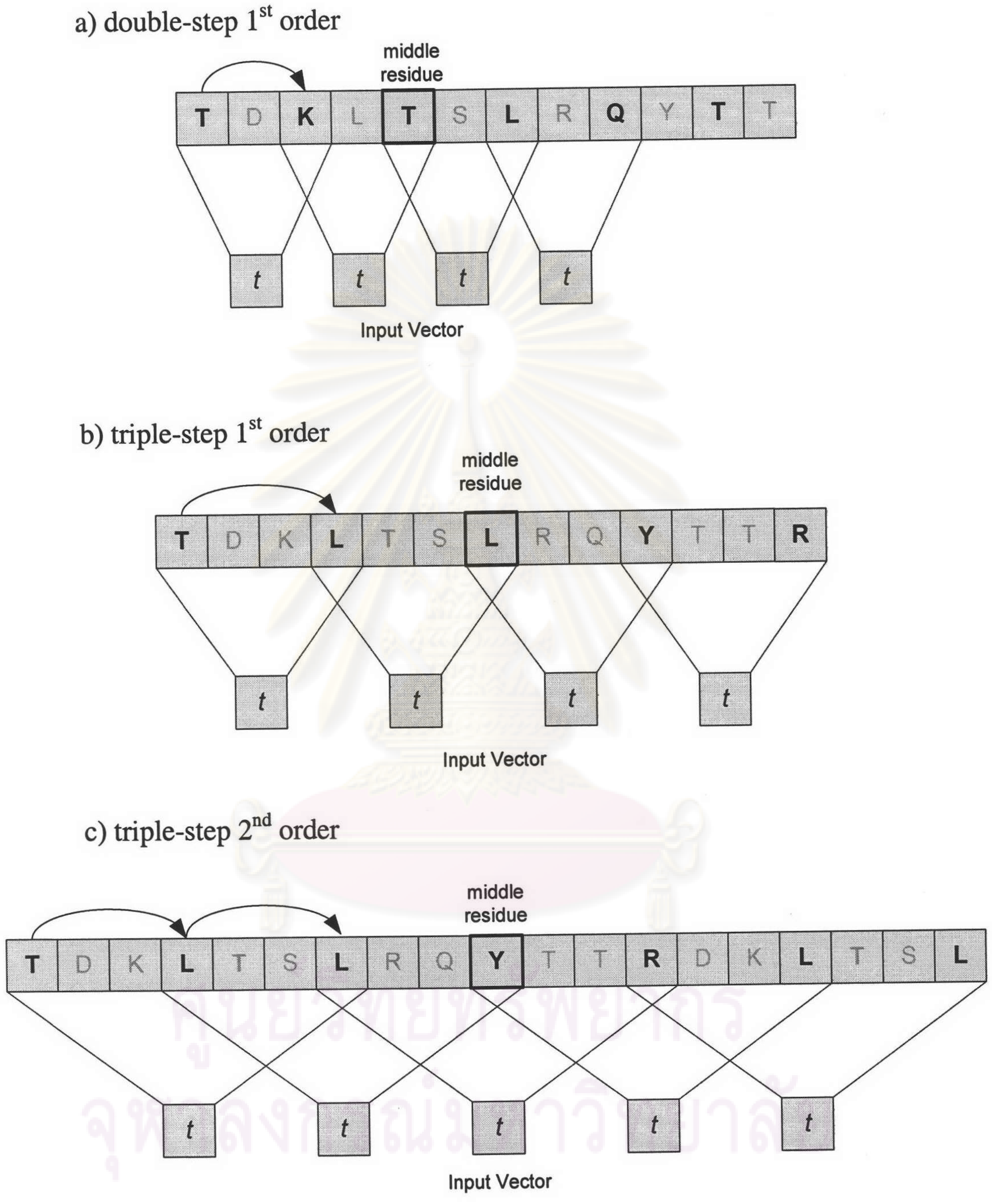


Figure 3.7 The input vector constructed from multi-step Markov Chain;
the double-step 1st order, triple-step 1st order, and triple-step 2nd order Markov Chain, a), b), and c), respectively.

3.5 Normalization and Interpolation of Transition Matrix

Due to the limitation of estimated data, uncompleted transition matrix may occur. That is the zero value in the matrix. The zero value means that there is no amino acid pattern match in the estimated set. Therefore, to remove the zero value, the normalization and interpolation have been used.

The normalization is means that the consideration of structure is not only focused on the middle residue of the sliding window but also considered for structure of all residues in the window. For each sliding window, we can see that there are some level of probability to be in class of Helix, Extend, and Coil structure. With this concept, then, we apply the scoring technique to score the pattern of amino acid by counting the number of structure occurred in those sliding windows. After that, the scores are accumulated into the transition matrices of helix, extend, and coil, respectively.

The normalization technique is useful for reducing the zero value of those matrices though. However, those matrices are still not completed non-zeros. Therefore, the interpolation of score is applied to the transition data. By the fact that shorter patterns are most likely to occur in the dataset more than longer patterns, if we reduce the length of patterns at the zero-point and those reduced patterns are found in the dataset, the zero value can be eliminated. With this concept, we focus on the patterns that zero points occurred and reduce the five dimension array down to four dimension array. Then, the non-zero values of four dimensional arrays are used instead. In case of some zero values still appears in matrices. The same reduction method is also applied by reducing the array dimension until the zero values are all eliminated. With the normalization and interpolation techniques, the completed transition matrices with no zero values can be obtained and used as an informative data source to construct the input vector of the learning model.

3.6 Incorporation of the Evolutionary Information

From the observation on evolution, it is unlikely to exchange the amino acids at positions that will change the structure, as a change of structure usually results in a loss of function. Thus, the residue exchange patterns extracted from a protein family are highly indicative of the specific structural details for that family. The protein family sequence can be obtained by using PSI-BLAST search [40]. PSI-BLAST has recently been shown to be able to extend sequence search results into the twilight zone of sequence similarity. The method iterates BLAST searches, using outputs of the first search as inputs for the next. This allows more distant homologues to be found. In this research, we use the homologues sequences obtained form PSI-BLAST search to be an additional estimated data together with the normal estimated set. Those sequences the evolution information can be incorporated into the predicting model.

Using only the estimated sequences form known structure sequences from PDB still has a limitation if the higher order of Markov Transition Matrix is used. In case of 4th-order matrix, it seems to run into a problem of not enough data to estimate and complete those matrices. To include the homologues and distant homologues sequences from PSI-BLAST into the estimated set, not only the more informative estimated data can be obtained, but they also help to overcome this limitation.

3.7 Network Structure of Learning Model

For our prediction model, a three-layer SVM network is simply constructed. These layers are the first-layer, Sequence to Structure, the second-layer, structure to structure, and final layer, multi-class ternary classification. (see Figure 3.9)

First-Layer: Sequence to Structure

The first-layer, Sequence to Structure, is composed of three binary classifiers for each class of secondary structures. The input to this layer is the vectors of log-odds ratio derived form the Markov Transition Matrices constructed from the algorithms previously defined. The network is presented with a window of consecutive residues; if the network is predicting the i^{th} residue, then it is presented with the log-odds ratio of transition probability of residues *i-n* to *i+n*. This means that the window size of input sequence is *2n+1* residues which the same size to input dimensions.

In this layer, the SVM classifiers take the input and predict membership of secondary structure class. Whereas SVMs are only formulated as binary classifiers, the ternary prediction cannot be performed immediately. We train three binary classifiers that specific to the structural classes – Helix, Extend and Coil. Those classifiers decide between H/~H, E/~E and C/~C, respectively.

Second-Layer: Structure to Structure

The second-layer, Structure to Structure, is also composed of three binary classifiers. In this layer, each classifier takes same inputs from the decision values of the first-layer. The input vectors are presented in from of window of contiguous residues having three elements from each classifier per-residue. The idea of "Structure to Structure" network was introduced and used in PHD system - the Neural Networks based method for protein secondary structure prediction [9]. We apply this powerful technique into our SVM based method.

Third Layer: Multi-class Ternary Classification

On the final layer, we make full multi-class ternary classification by combining the outputs of three binary classifiers with voting strategy. This simple and useful technique is also known as the "one versus all others" strategy.

In our experiment, we create two predicting models. The first one uses only the first-layer, then, immediately passes the outputs to the ternary classifier on the final layer. We call this model as "single layer network model". Another model uses fully cascading of the first and second layers. Then, the outputs are combined with final layer. The later is call "double layer network model". The results of those models are discussed on the next section.

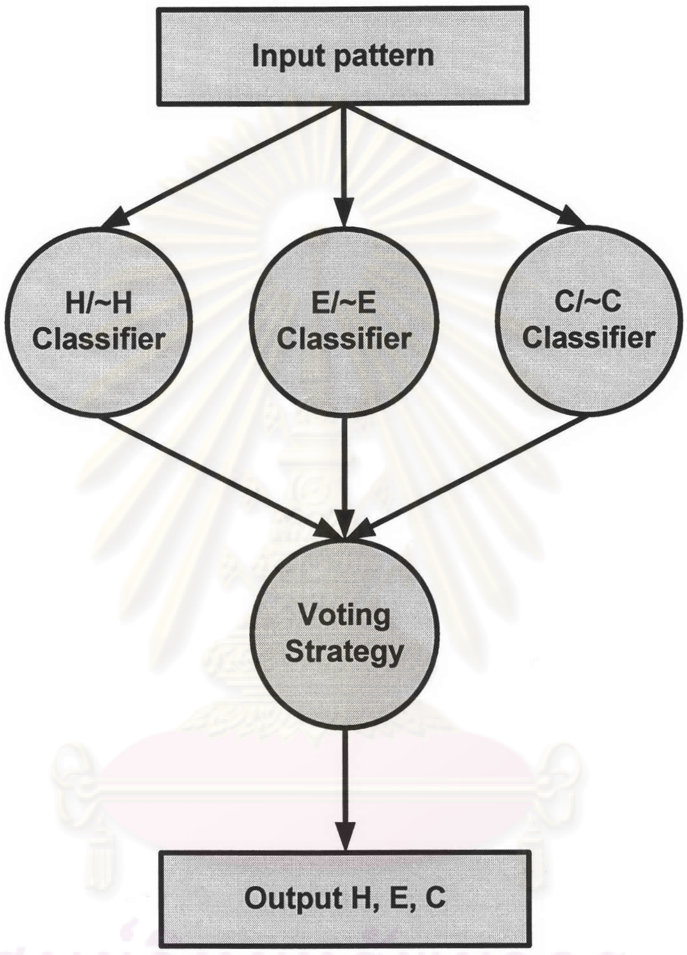


Figure 3.8 The single layer network model; the combining output of three classifiers is passed directly to the ternary classifier to make the decision by voting strategy (winner-take-all).

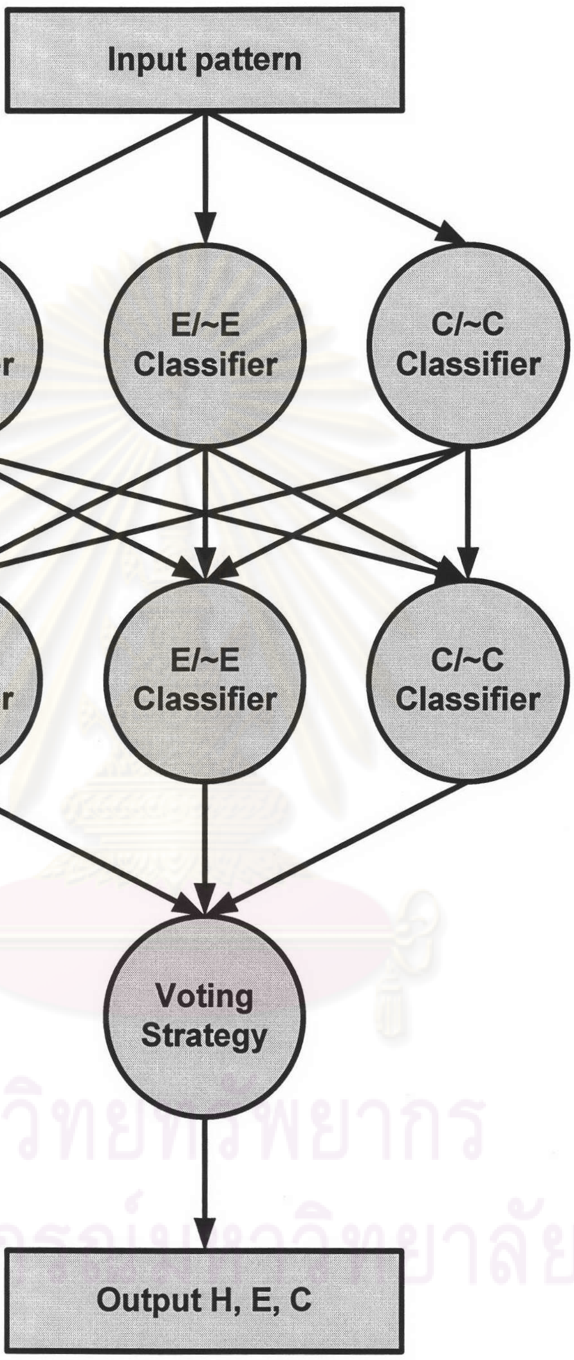


Figure 3.9 The double layer network model; the combining output of three classifiers from first layer is passed to the second layer (middle layer) for reclassifying process, then the combining output of second layer is passed to the ternary classifier at final layer to make the final decision of the tree structural classes.

3.8 Reliability Index and Filtering of the Predictions

Reliability indexes (*Ri*) have been used successfully throughout many models of secondary structure prediction. It particularly helps when the prediction is to be used by an expert, as they allow the user to concentrate their attention on areas that are known to be predicted with high reliability. The *Ri* offers an excellent tool for focusing on key regions having high prediction accuracy.

Our reliability index is based on the technique developed by Rost and Sander [9]. It measures the distance of the successful classifier to the optimal separating hyperplane and it is defined as follows.

$$Ri = integer(d / 0.5) \tag{3.21}$$

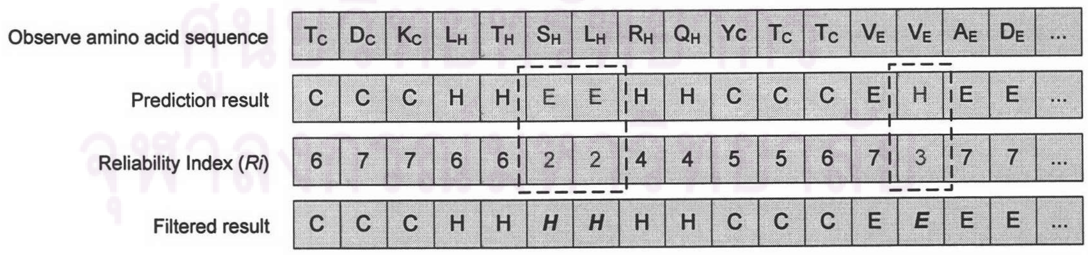
The value of index is truncated in between [0...9]. If the value of *Ri* > 9 then *Ri* is set to 9, where *d* is the remainder or distance of the two decision function of the classifiers with largest output. The value of *d* is computed from the following equation.

$$d = \textit{maximal}\ (\text{output}) - \textit{second largest}\ (\text{output}) \tag{3.22}$$

This reliability index can also be applied to improve the predicting result by using a filtering algorithm. The prediction accuracy of residues with higher *Ri* values is much better than those with lower *Ri* values. The filtering algorithm tries to automatically adjust and finish the output received from the ternary classifier by modifying the result of residue *Ri* lower than a threshold (*t*) to be the result of closest edge with *Ri* higher than the threshold. By this algorithm, the longer connected secondary structure segments can be produced and they significantly improve the over all performance of the predicting model.

Algorithm 3.2: Filtering Algorithm

1) Calculate the *Ri* of all predicting result on protein sequences.
2) Find the *Ri* that lower the threshold (*t*).
3) Mask those points to be *uncertainty* points.
4) Find left and right edges of *uncertainty* points.
5) At both sides of edges, replace the structure of *uncertainty* points with structure of connected edge point, and reset the points to *certainty* point.
6) Repeat steps 4) and 5) until all *uncertainty* points are reset.



Observe amino acid sequence	T_C	D_C	K_C	L_H	T_H	S_H	L_H	R_H	Q_H	Y_C	T_C	T_C	V_E	V_E	A_E	D_E	...
Prediction result	C	C	C	H	H	E	E	H	H	C	C	C	E	H	E	E	...
Reliability Index (*Ri*)	6	7	7	6	6	2	2	4	4	5	5	6	7	3	7	7	...
Filtered result	C	C	C	H	H	***H***	***H***	H	H	C	C	C	E	***E***	E	E	...

Figure 3.10 A filtered result of the prediction model. The uncertainty results (dot boxes) which *Ri* is lower the threshold are replaced by the results of the closest edge on either the left or the right hand side.

3.9 Parameters optimization on SVM

In our experiment, a Gaussian Radial Basis (RBF) kernel with $\gamma = 0.05$ and the regularization parameter R = 1 is used for all classifiers after many trial processes for optimization have been attempted. The problem of imbalance in class memberships of training sets is dealt with the weighting term (w) of the regularization parameter [41]. This requires that the two regularization parameters are specified.

$$R_0 = R \tag{3.23}$$

$$R_1 = w\,R \tag{3.24}$$

$$w = N_0 / N_1 \tag{3.25}$$

where R_0, R_1 and N_0, N_1 are regularization parameters and a number of patterns in class 0 and class 1, respectively; i.e. for the classifier of class Helix (H/~H), class 0 means patterns of non Helix structures (~H) and class 1 means patterns of Helix structure (H). With this from of weighted regularization, the penalties associated with misclassifying of the two classes are different. It means that adjusting the optimum separating hyperplane is more sensitive to the SVs for the smaller class in the training examples.