CHAPTER 2

BACKGROUND

2.1 Biological Background

In this section, we give a description of three protein functions of interest. The three protein functions are analyzing by using different computational-based methods. In general, proteins sequence is an ordered of amino acids. Each protein function has different characteristics.

2.1.1 Human Leukocyte Antigen gene

The human leukocyte antigen system or human lymphocyte antigen (HLA) is the molecular name of a group of molecules in the human major histocompatibility complex (MHC) region on human chromosome 6 which encode the cell-surface antigen-presenting proteins [4] (shown in Figure 2.1). The HLA are a class of proteins found on the surface membranes of cells which serve the purpose of presenting possible antigens to T and B cells.

The MHC contains a group of molecules that play a crucial role in immune recognition and for the tolerance of tissue grafting. In mice and humans, the MHC molecules have also been found to influence body odors, body odor preferences, and mate choice [5,6]. These sequences are also some of the most polymorphic regions of the genome and are known to play a central role in controlling immunological self and non-self recognition [7]. There are different types of HLA, e.g. HLA-I and HLA-II. These two gene types are important in the matching of tissues and organs for donation

and organ transplantation under outdated immunesuppression protocols. In addition, the major HLA antigens are essential elements for immune function. The two different classes have different functions. The principle function of HLA-I, is to present virally induced peptides on the surface of the cell by linking to the T-Cell receptor of a Cytotoxic (CD8) T Cell. This allows the identification of viruses. The role of HLA-II, by initiating a molecular immune response, is the reason they are only present on "immunologically active" cells (B lymphocytes, macrophages, etc.) and not on all tissues [8].

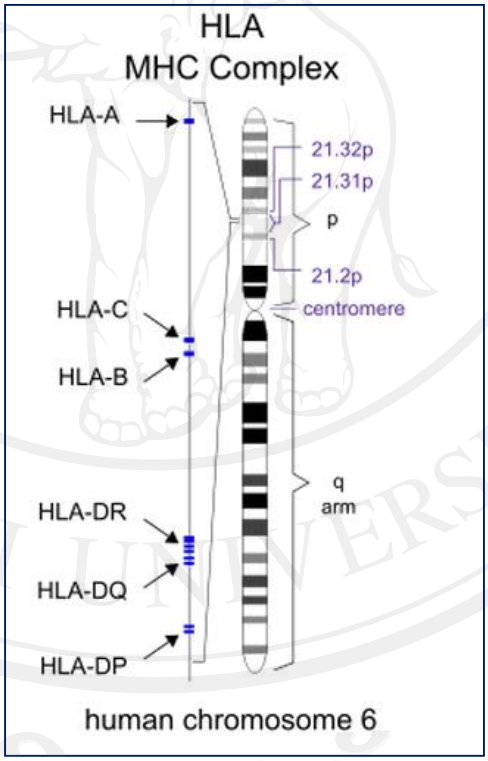

Figure 2.1 - The HLA region of Chromosome 6 [4]

2.1.2 Human Immunodeficiency Virus type 1

The Human Immunodeficiency Virus type 1 (HIV-1) enters target cells using interaction between envelope glycoprotein (gp120) with the cellular CD4 receptor and two main coreceptors, CC-chemokine receptor 5 (CCR5) and/or CXC-chemokine receptor 4 (CXCR4) [9]. Based on the ability of the coreceptor usage, HIV-1 variants can be classified as CCR5 tropic (R5), CXCR4 tropic (X4), or dual-mixed tropic (R5X4) [10]. R5 variants are generally established in early stage infections, while X4 variants generally emerge in later stages and have been associated with a faster CD4 decline and progression to AIDS [11, 12]. Therefore, predicting the emergence of X4 variants has potential value for understanding pathogenesis, monitoring disease progression and making treatment decisions.

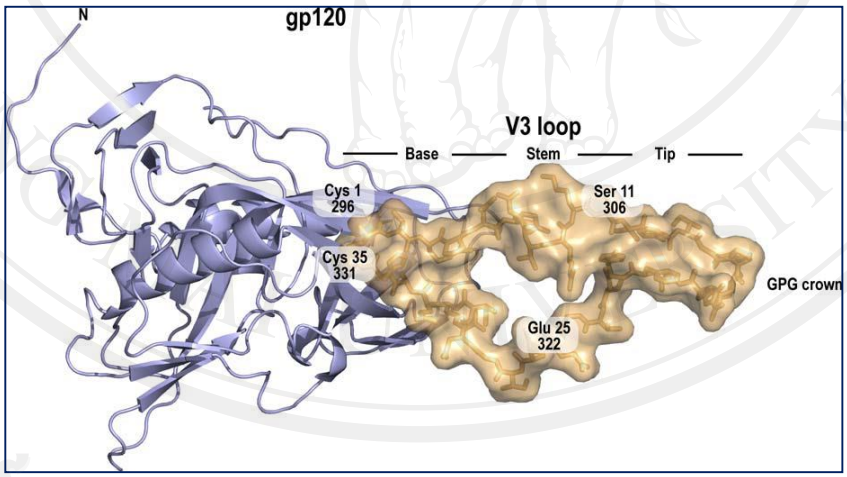

Figure 2.2 - The V3 loop region [9]

2.1.3 Protein crystallization

The ability to obtain experimentally measured 3D folding structures using X-ray crystallography is dependent on the availability of high quality protein crystals.

Since it is expensive and time consuming to produce such crystals, being able to computationally determine whether or not a protein will be able to crystallize has become a key step in determining protein folding structure. Furthermore, determining a proteins structure provides understanding of the proteins properties and function which can be utilized in the field of drug design [13]. Although X-ray crystallography is generally considered to be the most reliable and accurate approach to produce 3D structures, it is a very complex, time-consuming, laborious and expensive process. In addition, for this method to work a crystallized form of the protein is required which can be difficult to produce. At present, approximately 87 % of the protein structures deposited in the Protein Data Bank (PDB) was characterized using the X-ray crystallography method [14]. In addition, since many proteins have complex structures, the current experimental protocol to produce 3D structures has only a 30% success rate [15].

2.2 Sequence Classification

Sequence classification has a broad range of applications such as genomic analysis, protein functions etc [16]. Generally, a sequence is an ordered list of an alphabet of symbols $\{E_1, E_2, E_3, \dots, E_n\}$, a simple symbolic sequence is an ordered list of the symbols from the alphabet, such as DNA or protein sequences.

Generally, sequence classification methods can be divided into three large categories.

- The first category is *feature based classification*, which transforms a sequence into a feature vector and then applies computational-based methods. Feature selection plays an important role in these kinds of methods.

- The second category is *sequence distance based classification*. The distance function measuring the similarity between two sequences determines the quality of the classification. The method of choosing the similarity measure is the important step.
- The last category is *model based classification*, such as using Hidden Markov Model (HMM) and other statistical models to classify sequences.

In this chapter, we mainly focus on the representative methods in the three categories. Some methods may be related to more than one category. For example, the SVM method can be interpreted as either category 1 or category 2. Thus, the character of SVM method can be classified as category 4.

2.2.1 Feature based classification

Many conventional classification methods, such as decision trees and neural networks, are designed for classifying feature vectors. Practically, we can solve such problem by transforming a sequence into a suitable vector of features through feature selections.

In this case, the simplest way is to assign each element as a feature. For example, a sequence CACG can be transformed as a vector $\{A, C, C, G\}$. In fact, the sequential data cannot be captured by such transformation. To keep the order of the elements in a sequence, a short sequence segment of k consecutive symbols, called a k-gram, is usually selected to be the representation of sequential data. This particular method is known as a spectrum kernel. By using k-grams as features, sequences can be classified by computational-based method, such as the SVM method [18,19] or decision trees [20]. More detail about k-gram based feature selection methods for

sequence classifications can be referred in [21] and in the next section. In the case of protein sequences, the amino acid and dipeptide compositions are well-known features which correlate to physicochemical properties (PCP).

2.2.2 Sequence distance based classification

Sequence distance based methods define a distance function to measure the similarity between a pair of sequences. Generally, we first choose the suitable distance function; we then apply some existing methods, such as *k*-NN classifier or SVM method. The *k*-NN classifier is a popular and lazy learning method [23]. This method is very effective for a variety of problem domains. However, the *k*-NN classifier cannot classify some datasets which are highly complex or overlap. More detail about this drawback and how to solve for it can be found in Chapter 3. Given a labeled sequence data set *D*, a positive integer k, and a new sequence s to be classified, the *k*-NN classifier finds the k nearest neighbors of s in *D*, kNN(s), and returns the dominating class label in kNN(s) as the label of s.

In the *k*-NN classifier, choosing a distance measure is the crucial process, since it determines the *k*-NN classifier's performance. One well-known distance measure is the Euclidean distance. For two vectors *x* and *y*, the Euclidean distance is defined by:

$$dist(x,y) = \sqrt{\sum_{i=1}^{L}(x[i]-y[i])^2} \quad \textbf{(2.1)}$$

In our work, we examined the *k*-NN classifier based the string kernel. More detail about such methods and how to use them in protein function prediction can be found in Chapter 3.

2.2.3 Model based classification

This category of sequence classification methods is based on generative models, which assume sequences in a class are generated by an underlying model *M*. Generally, given a class of sequences, *M* is used to assign the probability distribution of the sequences in the class. Usually, some assumption is used to define *M*, and the probability distributions are described by a set of parameters. In a supervised learning setup, the learning process tries to learn all of the parameters. In the testing process, an unknown sequence is assigned to the probable class with the highest likelihood by using the best parameter from the learning process. This particular process is called the probabilistic graphical model. There are two main types, i.e. generative and discriminative model [24-27]. The graphical model can be apply in many problems, including biological sequence classification [28,29].

2.2.4 Support vector machine

Using machine learning, there are techniques called kernel methods which are used to construct a maximum-margin separating hyperplane between two separated classes. This particular kernel method is known as a Support Vector Machine (SVMs). The SVM is one of the best-known and most frequently used kernel methods [30], since the kernel method offers applicable tools to process, analyze, and compare many types of data, and outperforms other methods in many cases [31-33]. Vapnik

introduced the kernel method with the principle of structure risk minimization in statistical learning theory [34, 35]. In general, a data set is formally represented as

$$D = \{ (x_i, y_i) \mid x_i \epsilon \mathbb{R}^n, y_i \epsilon \{1, -1\} \}; i = 1, 2, \dots, n \quad \textbf{(2.2)}$$

where x_i is the *i*-th input vector and y_i is the class of x_i. Each x_i is an *n*-dimensional vector. Principally, the idea of the kernel method is to construct a maximum-margin hyperplane separating the classes of *x*.

In general, when training data sets are nonlinearly separable vectors, the basic idea is to retain the simplicity of linear methods by using mapping functions to map the original data set into a higher dimensional space, called feature space, where linear methods can classify them. The mapping function $\Phi(x)$ is performed by defining the inner product between each pair of data points in the data set of the feature space through the kernel function. Thus, if $\Phi(x)$ denotes the mapping function, the kernel function can be expressed as a similarity measurement between the training data set, which is defined as:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle = \Phi(x)^{\mathsf{T}} \Phi(x') \quad \textbf{(2.3)}$$

One of the most widely used kernels for sequence classification is a spectrum kernel or string kernel (or k-gram [21]), which transforms a sequence into a feature vector [37]. The kernel function of strings was first proposed by Watkins [38]. In 2002, Lodhi et al. introduced a powerful string subsequence kernel [39] for text classification. Moreover, Leslie *et al* [40] showed that the spectrum kernel can effectively be applied to protein classification. Saunders *et al*. also reported on the computational advantages of the spectrum kernel for its fast and simple calculation. If

a suitable data structure is used, the prediction can be done in linear time [41]. We will discuss the spectrum kernel in Section 2.4.

One disadvantage of kernel based methods is that they are hard to be interpreted and hard for users to gain knowledge besides a classification result. We used the string kernel to transform HLA genes, and then applied a *k*-NN classifier to predict its class [1]. More detail of this work can be found in next part.

2.3 Feature Selection Methods

The objectives of feature selection are manifold, the most important ones being: (a) to avoid overfitting and improve model performance, i.e. prediction performance in the case of supervised classification and better cluster detection in the case of clustering, (b) to provide faster and more cost-effective models and (c) to gain a deeper insight into the underlying processes that generated the data.

The feature selection methods perform a search through the space of feature subsets, in general, and must be addressed with four processes: 1) Selecting a starting point in the space of feature subsets for beginning the search affecting the direction, 2) Searching the space of feature subsets. There are two main procedures, i.e. forward and backward algorithms, 3) Evaluating strategy. This is concerned with how many feature subsets are evaluated (more details of the category of evaluating strategy can be seen below), and 4) the stopping criterion. A feature selector must decide when to stop searching through the space of feature subsets. Depending on the evaluation strategy, a feature selector might stop adding or removing features when none of the alternatives improves the merit of a current feature subset.

Table 2.1 A taxonomy of feature selection techniques [44].

Model search	Advantage	Disadvantage	Examples
Filter	-Independent of the classifier -Better computational complexity than wrapper methods	-Ignores interaction with the classifier	Information gain, gain ration, Euclidean distance [45]
Wrapper	-Interacts with the classifier -Model feature dependencies	-Risk of overfitting -Classifier dependent selection	Genetic algorithm [46], Sequencial forward selection [47], Sequencial backward selection [47]
Embedded	-Better computational complexity than wrapper methods -Interacts with the classifier -Model feature dependencies	-Classifier dependent selection	Decision tree [48], Logistic model tree [49], Random forests [50]

In the context of classification, feature selection techniques or evaluating strategy can be organized into three categories which depend on how they combine the feature selection search with the construction of the classification model: filter methods, wrapper methods and embedded methods. Table 2.1 and 2.2 provide a common taxonomy and characteristics of feature selection methods (some existing methods), showing for each technique the most prominent advantages and disadvantages, as well as some examples of the most influential techniques [44].

Table 2.2 The characteristics of feature selection techniques.

Model search	Criterion	Search	Assessment
Filter	Relevance	Top rank	Statistical test
Wrapper	Usefulness	All subset	Cross-validation
Embedded	Usefulness	Guide by learning process	Cross-validation

2.3.1 Filter techniques

Filter techniques assess the relevance of features by looking at only the intrinsic properties of the data. The ideal of this technique is shown in Figure 2.3. In most cases a feature relevance score is calculated by using a statistical test, such as a T-test, and low-scoring features are removed. Afterwards, this subset of features is presented as input to the classification algorithm. Filter techniques face the problem of finding a good feature subset (FS) independently of the model selection step.

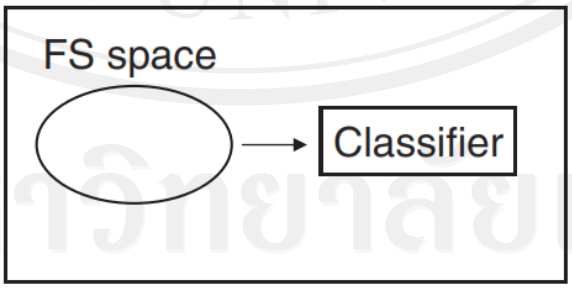

Figure 2.3 - The system flowchart of filter techniques [44].

2.3.2 Wrapper techniques

Wrapper methods embed the model hypothesis search within the feature subset search. In this setup, a search procedure in the space of possible feature subsets is defined, and various subsets of features are generated and evaluated. The evaluation of a specific subset of features is obtained by training and testing a specific classification model, rendering this approach tailored to a specific classification algorithm. The ideal of this technique is shown in Figure 2.4.

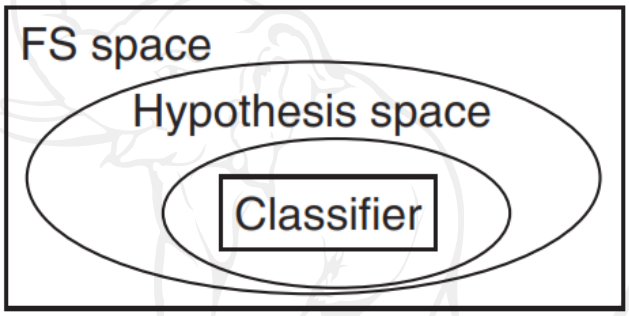

Figure 2.4 - The system flowchart of wrapper techniques [44].

2.3.3 Embedded techniques

The search for an optimal subset of features is built into the classifier construction, and can be seen as a search in the combined space of feature subsets and hypotheses. Just like wrapper approaches, embedded approaches are thus specific to a given learning algorithm. Embedded methods have the advantage that they include the interaction with the classification model, while at the same time being far less computationally intensive than wrapper methods

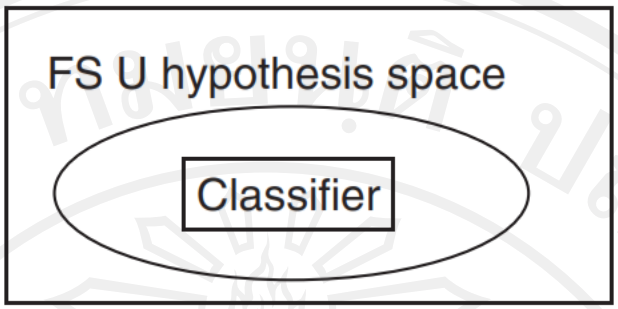

Figure 2.5 - The system flowchart of embedded techniques [44].

2.4 Support Vector Machine

In a previous section we described the main idea of the SVM method. This classifier can be categorized as both a sequence distance and a feature based classification. The SVM classifier is a well-known method in non-linear problems. The SVM method constructs a separating hyperplane maximizing the margin between the two data sets which are sets of vectors in an *n*-dimensional space [30]. Intuitively, a good separation or classification occurs when the hyperplane has the largest distance to the neighboring data points of both classes, since the larger margin leads to a lower generalization error of the classifier and also ensures that it can identify the particular class of each data point [36,37].

2.4.1 Notation

To easily understand SVM, we will be considering a linear classifier for a binary classification problem given a training data D. A linear classifier can be represented as:

$$h_{w,b}(x) = g(w^T x + b) \quad \textbf{(2.4)}$$

This classifier has $w,\ b$ as its parameters. Here, $h(z) = 1\ if\ z \geq 0$, and $h(z) = -1\ otherwise$.

2.4.2 Functional and geometric margins

We first formalize the notions of the functional and geometric margins. Given a training data set D, we define the *functional margin* of (w, b) with respect to *D*.

$$\gamma_i = y_i(w^T x + b) \quad \textbf{(2.5)}$$

The goal is to seek the smallest of the function margins of the individual training examples. Denoted by $\hat{\gamma}$, this can therefore be written:

$$\hat{\gamma} = argmin_i\ \gamma_i \quad \textbf{(2.6)}$$

2.4.3 The optimal margin classifier

How can we find the one that achieves the maximum geometric margin? We can represent the following by defining an optimization problem. In general, we now have the following optimization problem:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \quad \textbf{(2.7)}$$

$$s.t. \qquad y_i(\ w^T x_i + b\ ) \geq\ 1, i = 1,2, \dots, m$$

Note that: in the learning process of SVMs, a hard-margin separation, as represented above, is usually performed, even though this kind of misclassification is unavoidable in many practical problems. To deal with this problem, soft-margin separation is introduced to mitigate these errors by finding a maximum margin separator which allows misclassifying a training data set [36,37]. Currently, we know

that the simplest way to permit errors in the maximum margin linear classifier is to introduce "slack" variables ε_i for the classification/margin constraints in the optimization problem.

In order to solve this constrained optimization problem, we introduce Lagrange multipliers $\alpha_i \geq 0$. When we construct the Lagrangian for our optimization problem we have:

$$L(w, b, \alpha) = \frac{1}{2}\|\mathrm{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i(\mathrm{w}^T x_i + b) - 1) \quad \textbf{(2.8)}$$

Let's go on the dual form of the problem by setting $\frac{\partial L(w,b,\alpha)}{\partial w} = 0$ and $\frac{\partial L(w,b,\alpha)}{\partial b} = 0$. In the classification problem with a training data set *D*, we can predict the class of unknown data x_{n+1} by using a linear decision function determined by the kernel function of the inner product between feature vectors. The decision function can then be expressed as follows:

$$f(x_{n+1}) = \sum_{i=1}^{n} y_i \alpha_i K(, x_i, x_{n+1}) + b \quad \textbf{(2.9)}$$

2.4.4 Kernel function

In general, the basis of protein classification is to represent protein sequences as vectors in a high-dimensional feature space via a string-based feature map. We then train the SVM, a large-margin linear classifier, on the feature vectors representing our training sequences. Recall, since SVMs are a kernel-based learning algorithm, so we intend to introduce a suitable kernel for the protein classification

In Section 2.2.4, we gave a description of the most widely used kernels for sequence classification which is a spectrum kernel or string kernel, which transforms

a sequence into a feature vector [18-19, 38-40]. The idea behind the spectrum kernel approach is based on the similarity of two strings containing common subsequences. The spectrum kernel is a convolution kernel specialized for the string comparison problem. For a number $sk \geq 1$, the sk-spectrum of a sequence x is all the possible subsequences of length sk that it contains. Given the alphabet A, a sequence x is transformed into a feature space by a transformation function or feature mapping function

$$\Phi_{\mathrm{sk}}(x) = (\phi_a(x))_{a \in A^{sk}} \tag{2.10}$$

where $\phi_a(x)$ is the number of times a occurs in x. The kernel function is the inner product of the features vectors:

$$K_{\mathrm{sk}}(x, x') = \langle \Phi_{\mathrm{sk}}(x), \Phi_{\mathrm{sk}}(x') \rangle. \tag{2.11}$$

For another variant of the kernel, we can assign to the a-th coordinate a binary value of 0 if a does not occur in x, 1 if it does occur.

In Chapter 3, we were interested in HLA gene prediction by using the string kernel to transform HLA genes, and then classify them by using the *k*-NN classifier.

2.5 Scoring Card Method

The scoring card method SCM is an efficient and generalized method for creating various kinds of dipeptide scoring cards for predicting protein functions from whole sequences. The principle hypothesis of the SCM method uses amino acid and dipeptide compositions which play an important role in serving as significantly effective features. The description of the SCM is given in a general purpose algorithm without using heuristics or specific domain knowledge. The SCM method can be

applied to other prediction problems without significant modifications. Of course, the generic score matrix of dipeptides can be further customized and utilized with other complementary features for improved prediction accuracy [51].

The system flowchart of the SCM method with propensity analysis is shown in Figure. 3.1. The description of the SCM consists of the following parts: 1) creation of data sets for both training and an independent test, 2) establishment of an initial scoring matrix for the propensity of dipeptides using a statistical approach, 3) optimized solubility scoring matrix of dipeptides, 4) prediction of protein solubility, and 5) propensity analysis of amino acids.

The procedure of the SCM method is briefly described below. More details about the SCM can be found in.

2.5.1: Prepare a training dataset consisting of two subsets for positive and negative classes.

2.5.2: Generate an initial scoring card consisting of 400 propensity scores of dipeptides by using a coarse-tofine approach. The initial scoring card is created by using a statistical approach based on the dipeptide composition

I) Calculate the numbers of 400 dipeptides in each class.

II) Normalize the dipeptide composition by dividing the numbers using the total numbers of dipeptides in each class.

III) Obtain the propensity scores of individual dipeptide by subtracting the score of the negative class from that of the positive class.

IV) Normalize the scores of all dipeptides into the range [0, 1000].

2.5.3: Derive the propensity score of each amino acid A by averaging the 20 scores of dipeptides AX and XA where X can be any amino acid. If the acid composition (i.e.,

percentages) of a certain protein has a high correlation with the CSM (crystallizable scoring matrix) of amino acids, this protein is easy to predict as a crystallizable protein.

2.5.4: Optimize score card (optimized CSM) by using an intelligent genetic algorithm (IGA) [52]. In the chromosome representation, the 400 real-valued variables are encoded in a chromosome of IGA, which is in the range [0, 1000]. The IGA algorithm for obtaining the optimized score card is described as follows:

Step 1: (Initialization) Randomly generate Npop individuals including the initial SSM. In this study, Npop = 40.

Step 2: (Evaluation) Compute fitness values of all individuals where Ibest is the best individual in the population.

Step 3: (Selection) Use a rank-based selection to select Ps*Npop individuals to establish a mating pool. In this study, Ps = 1.0.

Step 4: (Crossover) Perform the intelligence crossover operation [15] for each individual with Ibest to find the best two individuals among two parents and two children as the new children (the elitist strategy).

Step 5: (Mutation) Use a real-valued mutation operator to randomly mutate individuals with a mutation probability Pm (= 0.01). Mutation is not applied to Ibest to prevent the best fitness value from deteriorating.

Step 6: (Termination test) If a given termination condition is satisfied, stop this algorithm, otherwise, go to Step 2.

The fitness function of the IGA is to maximize the prediction accuracy in terms of the area under the ROC curve (AUC) [53] and maximize the Pearson's correlation coefficient (the R value) between the initial and optimized scores of amino acids, described as follows [51] (W_1=0.9 and W_2=0.1 in this study):

$$Max\ Fit(\text{Scard}) = W_1 \times AUC + W_2 \times R. \quad \textbf{(2.12)}$$

2.5.5: The prediction of a sequence P based on the scoring function $S(P)$ and a threshold value determined by maximizing the prediction accuracy of training dataset.

$$S(P) = \sum_{i=1}^{400} w_i S_i,. \quad \textbf{(2.13)}$$

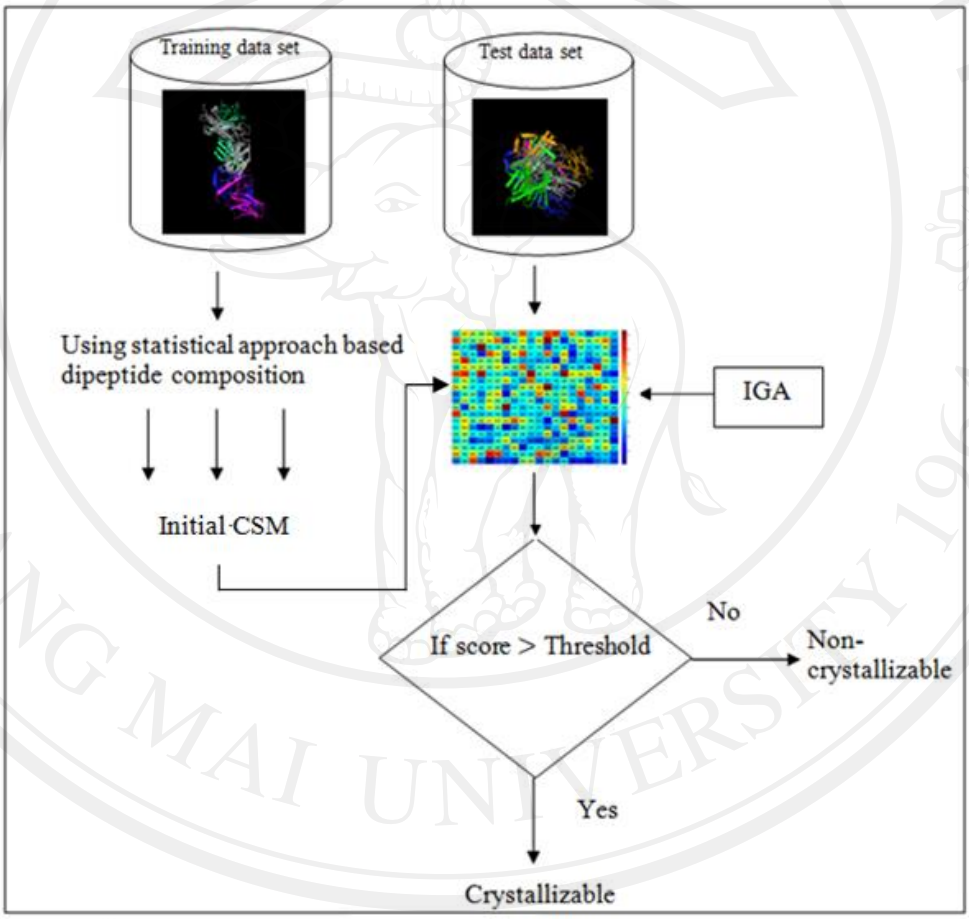

Figure 2.6 - The system flowchart of the scoring card method (SCM) [3]

where w_i is the frequency of the dipeptide composition of *P*, which is in the range [0,1], S_i is the score of the *i*-th dipeptide, and i= 1,..,400. *P* is classified as the positive class when *S(P)* is greater than the threshold value; otherwise, *P* is the negative class.

2.6 Logistic Model Trees

In this section, the logistic model tree method, or LMT for short, is presented. It combines logistic regression models with tree induction, and thus is an analogue of model trees for classification problems. The term of regression sometimes refers to a particular kind of parametric model which especially refers to a numeric target variable, and sometimes to the process of estimating a target variable in general (as opposed to a discrete one). For tree induction, its objective is to find a subdivision of the instance space into corrected regions.

2.6.1 The model

A logistic model tree basically consists of a standard decision tree structure with logistic regression functions at the leaves [49]. As in ordinary decision trees, a test on one of the attributes is associated with every inner node. For a nominal attribute with k values, the node has k child nodes, and instances are sorted down one of the k branches depending on the value of the attribute. For numeric attributes, the node has two child nodes and the test consists of comparing the attribute value to a threshold: an instance is sorted down the left branch if its value for that attribute is smaller than the threshold and sorted down the right branch otherwise.

More formally, a logistic model tree consists of a tree structure that is made up of a set of inner or non-terminal nodes N and a set of leaves or terminal nodes T. Let $D = D_1 \times ... \times D_m$ denote the whole instance space, spanned by all attributes $V = \{v_1 \times ... \times v_m\}$. Then the tree structure gives a disjoint subdivision of D into regions D_t, and every region is represented by a leaf in the tree:

$$D = \bigcup_{t \in T} D_t\,, \qquad D_t \cap D_{t'} = \emptyset, for\ t \neq t' \tag{2.14}$$

Unlike ordinary decision trees, the leaves $t \in T$ have an associated logistic regression function f_t instead of just a class label. The regression function f_t takes into account an arbitrary subset $V_t \subset V$ of all attributes present in the data, and models the class membership probabilities.

where

$$F_j(x) = \alpha_0^j + \sum_{v \epsilon V_t} \alpha_v^j \cdot v \tag{2.15}$$

Or, equivalently,

$$F_j(x) = \alpha_0^i + \sum_{k=1}^{m} \alpha_{v_k}^i \cdot v_k \tag{2.16}$$

if $\alpha^i_{v_k} = 0$ for $v_k \notin V_t$. The model represented by the whole logistic model tree is then given by

$$f(x) = \sum_{t \in T} f_t(x) \cdot I(x \in D_t) \tag{2.17}$$

where $I(x \in D_t)$ is indicator function, $I(x \in D_t) = 1$ if $x \in D_t$ and $I(x \in D_t) = 0$ otherwise.

In general, the tree induction works in a divide-and-conquer algorithm: a classifier for a set of examples is built by performing a split and then building classifiers for the binary class classification. Building a tree with a very small dataset is usually not a good idea, so there is a simpler model which is better to solve such problems, this is the logistic regression model. In general, we can have only a few datasets, especially if part of our dataset is encountered at lower levels in the tree

which is smaller and smaller. For this reason, we prefer to build a linear logistic model instead of using the tree growing procedure recursively. This is one motivation for the logistic model tree algorithm.

2.6.2 Building Logistic Model Tree

There is a straightforward approach for growing logistic model trees following the standard model tree. This would first involve building a standard classification tree, i.e. the algorithm selects the attribute which gives the largest decrease in standard deviation. By binary splitting the tree, nominal attributes are converted to binary ones (that are treated as numeric) before the tree growing starts. For this, the average value of the target variable is calculated for every nominal value of the attribute, and the nominal values are sorted according to these averages. If the nominal attribute has k values, it is replaced by $k-1$ binary attributes; the i^{th} being zero if the nominal value is among the first i in the ordering and one otherwise. After that, we build a logistic regression model at every node. Note we initially need a logistic regression model at every node of the tree, because every node is a 'candidate leaf' during pruning. In this approach, the logistic regression model would be built in isolation on the local training examples at a node based on LogitBoost [54] (to obtain the parameter shown in Figure. 3.2), not taking into account the surrounding tree structure.

LogitBoost (J classes)

1. Start with weights $w_{ij} = 1/n,\ i = 1, \ldots, n,\ j = 1, \ldots, J,\ F_j(x) = 0$ and $p_j(x) = 1/J\ \ \forall j$

2. Repeat for $m = 1, \ldots, M$:

   (a) Repeat for $j = 1, \ldots, J$:

   i. Compute working responses and weights in the jth class

$$z_{ij} = \frac{y^*_{ij} - p_j(x_i)}{p_j(x_i)(1 - p_j(x_i))}$$

$$w_{ij} = p_j(x_i)(1 - p_j(x_i))$$

   ii. Fit the function $f_{mj}(x)$ by a weighted least-squares regression of z_{ij} to x_i with weights w_{ij}

   (b) Set $f_{mj}(x) \leftarrow \frac{J-1}{J}(f_{mj}(x) - \frac{1}{J}\sum_{k=1}^{J} f_{mk}(x)),\ \ F_j(x) \leftarrow F_j(x) + f_{mj}(x)$

   (c) Update $p_j(x) = \frac{e^{F_j(x)}}{\sum_{k=1}^{J} e^{F_k(x)}}$

3. Output the classifier $\underset{j}{\text{argmax}}\ F_j(x)$

Figure 2.7 - LogitBoost algorithm [54]

The LMT would be constructed as follow:

1: Start building a logistic model f_n at node n by running Logitboost based on five fold cross-validation on D_n, including more variables or features in the model by adding simple regression f_{mj} fit to F_j^n (linear model for class j at node n), shown in step 2(b).

2: Split node n and build refining the logistic models at child nodes t and t' by proceeding the Logitboost on the smaller dataset D_t, and adding more simple regression to F_j^n from the F_j^t. The simple linear regression F_j^t are trained from the variables of the set of dataset D_t.

3: Splitting of the child nodes continues in this fashion until some stop criterion is met (the stop criterion is described later). Figure 3.3 summarizes this scheme for building the logistic model tree.

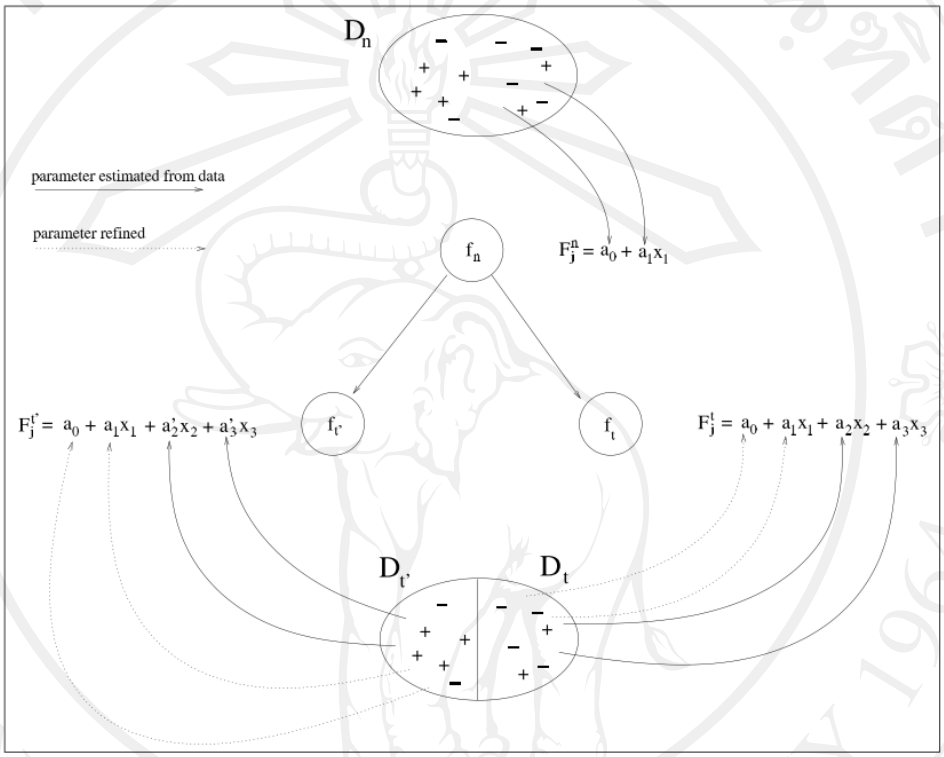

Figure 2.8 - Building logistic models

2.6.3 Splitting Criterion

In general, two criterions are used, i.e. entropy and information gain. The entropy characterizes the impurity of a variable. And the information gain is the expected reduction in entropy derived by partitioning the dataset. In the case of LMT, the global impurity overall for the classes is measured. And then selects the split that gives the largest decrease in the global impurity based on the C4.5 [55] splitting criterion) The LMT's splitting would be described in the following way:

1: Calculating entropy for each subset of dataset $D_i \subset D$ according to the selected variable. We can calculate the entropy as follows:

$$E(D_i) = \sum_{j=1}^{k} -p(C_j)\log_2 p(C_j) \quad \textbf{(2.18)}$$

where $p(C_j)$ is the probability the relative frequency of class *j* in D_i at the ith case. For example, in the crystallization protein, C_j consists of crystallizable and non-crystallizable proteins.

2: And then calculate information gain of dataset $D_v \subset D_i$ which is denoted as follows:

$$G(D_i, V) = E(D_i) - \sum_{v \in V} \frac{|D_v|}{|D|} E(D_v) \quad \textbf{(2.19)}$$

$$Gain\ ratio\ (D_i, V) = G(D_i, V)/E(D_i) \quad \textbf{(2.20)}$$

3: The optimum variable that is used to spilt is selected from the largest decrease $Gain\ ratio\ (D_i, V)$.

2.6.4 Stopping Criterion

Tree growing stops for one of three reasons:

1: A node is not split if it contains less than 15 examples.

2: A logistic model is only built at a node if it contains at least 5 examples, because the five fold cross-validation is used to perform training dataset.

3: A particular split is only considered if there are at least 2 subsets that contain 2 examples each. This is a heuristic used by the C4.5 algorithm to avoid overly fragmented splits. Furthermore, a split is only considered if it achieves a minimum

information gain (for C4.5-style splitting) or a minimum decrease in impurity (for splitting on the response). When no such split exists, we stop growing the tree.

2.6.5 Pruning the Tree

In order to generate a sequence of smaller and smaller trees, each of which is a candidate for the appropriately-fit final tree, the method of "cost-complexity" pruning is used. The idea of LMT pruning based on CART [56]:

$$R_\alpha(T) = R(T) + \alpha\left|\tilde{T}\right| \quad \textbf{(2.21)}$$

The complexity parameter α determines the relative penalty when we assign to a complex model and depends on the domain in question. It means that if $\alpha = 0$, there is no penalty for large trees, and the initial tree T_{max} minimizes R_α. On the other hand, $\alpha \to 1$ means the minimizer of R_α become smaller and smaller, and at $\alpha = 0$ it will consist of a single leaf.