Computational Prediction of Transposon Insertion Sites

by

Maryam Ayat

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
January 2013

Thesis advisor

Mike Domaratzki

Author

Maryam Ayat

Computational Prediction of Transposon Insertion Sites

Abstract

Transposons are DNA segments that can move or transpose themselves to new positions within the genome of an organism. Biologists need to predict preferred insertion sites of transposons to devise strategies in functional genomics and gene therapy studies. It has been found that the deformability property of the local DNA structure of the integration sites, called V_{step}, is of significant importance in the target-site selection process. We considered the V_{step} profiles of insertion sites and developed predictors based on Artificial Neural Networks (ANN) and Support Vector Machines (SVM). We trained our ANN and SVM predictors with the Sleeping Beauty transposonal data, and used them for identifying preferred individual insertion sites (each 12 bp in length) and regions (each 100 bp in length). Running a five-fold cross-validation showed that (1) Both ANN and SVM predictors are more successful in recognizing preferred regions than preferred individual sites; (2) Both ANN and SVM predictors have excellent performance in finding the most preferred regions (more than 90% sensitivity and specificity); and (3) The SVM predictor outperforms the ANN predictor in recognizing preferred individual sites and regions. The SVM has 83% sensitivity and 72% specificity in identifying preferred individual insertion sites, and 85% sensitivity and 90% specificity in recognizing preferred insertion regions.

Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to thank my supervisor, Dr. Mike Domaratzki, for his help, insight and advice which were instrumental in the completion of my research, for his meticulous proofreading of my writing, and for providing me with financial support throughout my program.

Also, I would like to thank Dr. Brian Fristensky, for engaging me in a genomics project which opened my eyes to the numerous applications of Bioinformatics methods in tackling real-world biological problems.

My thanks also go to all my friends in the Bioinformatics Lab at the University of Manitoba, especially Justin Zhang, Abiel Roche Lima, Natalie Bjorklund, Graham Alvare, and Sherif Mamun. I cherish all the good times that we had together.

My final acknowledgments are to my family: Mehraneh, Narges, Soheil and Hossein, for their patience, constant encouragement and companionship. This work would not have been possible without their infinite supports.

Chapter 1

Introduction

A transposon is a short mobile DNA sequence that can insert itself into the genome of the cell and replicate. Transposons have been found in a variety of genomes, from bacteria to humans, and may have favorable or unfavorable effects. They may deliver specific genes to the host genomes and they can cause mutations. They may activate genes nearby their insertion sites, and increase the amount of protein made, or they may deactivate nearby genes, and prevent their ability to create protein. Consequently, transposons may cause some diseases (e.g., see Belancio et al. [2009]), and on the other hand, may be used for therapeutic purposes (e.g., see Aronovich et al. [2011] and Di Matteo et al. [2012]). Also, they may be used in experimental biology to help determine the function of genes and the effect of environmental factors on them (e.g., see Largaespada and Collier [2008]). In any case, the outcome of the integration of a transposon in a host genome depends on the insertion sites of the transposon.

Predicting hotspots, or most preferred insertion sites, of transposons helps gene

therapists and functional genomicists to assess the risks of adverse effects for activating or deactivating genes (see the review by Ivics and Izsvak [2010]). There are many works on finding the factors that affect preferences in transposon integration, but a limited number of them centered on predicting insertion sites, such as the works reviewed in Chapter 5. In some of these works, such as Liu et al. [2005] and Geurts et al. [2006] which this thesis extends, the identifying process of preferred insertion sites of transposons is based on the local features in the DNA sequence or structure. Liu et al. found that there is a relationship between the natural deformability property of target sites, which is described by a parameter called V_{step}, and the mechanism of the target-site selection of the Sleeping Beauty (SB) transposon. Based on the work done by Liu et al., and considering that the DNA-deformability is a main factor in the target-site selection process, Geurts et al. tried to predict insertion-site preferences of a transposon by finding similar V_{step} patterns. They studied several transposons, including the SB transposon. Except for the SB, Geurts et al. could not find any consistent V_{step} pattern for preferred insertion sites of other transposons. Indeed, neither of the works of Liu et al. and Geurts et al. led to a structured method for prediction, as described in Section 4.2.2.

We view the insertion site prediction problem as a classification problem, or in general terms, as a pattern recognition problem. Hence, in order to find a better (i.e., more structured) solution for the problem, we take advantage of machine learning methods. Whenever there are samples of input and output pairs at hand, but there is no way to specify a precise relationship between input and desired output pairs, machine learning methods can be a reasonable framework to extract such relation-

ships. We use the V_{step} profiles of the local sequence of insertion sites as the features that influence target-site selection, and make two types of predictor, one based on Artificial Neural Networks (ANNs) and the other based on Support Vector Machine (SVMs). However, using ANN and SVM methods for solving prediction problems in biology is not new. There are many efficient and accurate tools based on ANNs and SVMs, of which we explain a few in Chapter 3. ANNs and SVMs are two different methods for classification, yet they have analogies that make the comparison of their performance on the same problem more sensible. To train and evaluate the predictors, we use an SB trasposon integration dataset from Hackett [2011], and report the related sensitivities, specificities, and ROC curves.

We have organized the thesis material as follows. In the next chapter, Chapter 2, we explain briefly all the required background and definitions. Chapter 3 and Chapter 4 are devoted to the related work. In Chapter 3, we illustrate samples of prediction problems in biology and their successful ANN- and SVM-based solutions; then in Chapter 4, we focus on the insertion site prediction problem and explain the current solutions for it. In Chapter 5, we describe our solution methodology, i.e., the ANN and SVM predictors. Also, we report the evaluation results of the predictors, and present a comparing analysis. Chapter 6 is devoted to the conclusion.

Chapter 2

Background

As this thesis proposal has an interdisciplinary essence, we describe the required background in three separate subsections: biology, machine learning, and statistics.

2.1 Biology

DNA is the hereditary material in all living organisms. The information in DNA is stored as a code made up of four nucleotide bases: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). DNA bases pair up with each other, A with T and C with G, to form units called base pairs. For instance, human DNA consists of about three billion base pairs. As shown in Figure 2.1, DNA has a double helical structure as nucleotides are arranged in two long sequences that form a spiral. A subsequence is a partial or local DNA sequence of one or more nucleotides in length. For example, a TA dinuleotide subsequence is a partial sequence of DNA with two nucleotides T and A, in order.

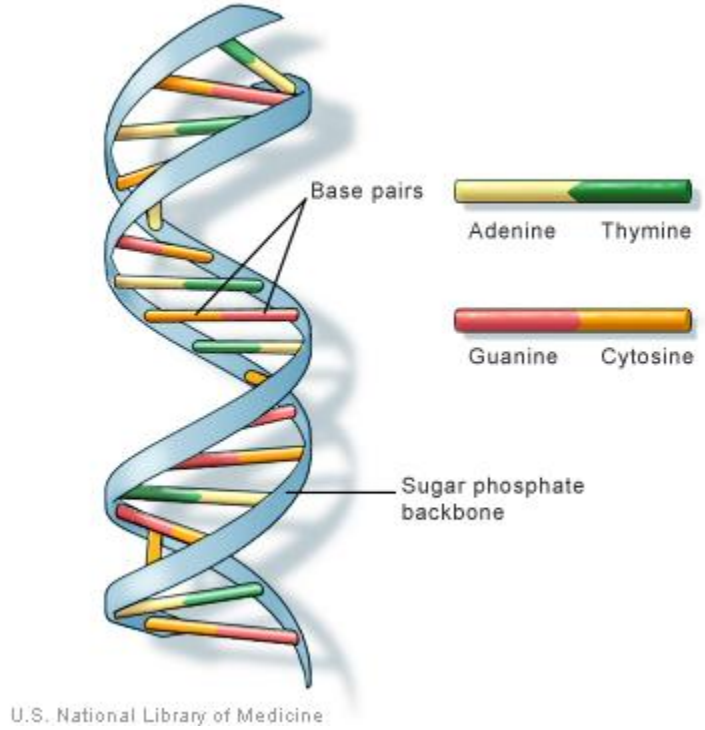

Figure 2.1: DNA 3D-structure: It is a double helix formed by base pairs. *Figure taken from the U.S. National Library of Medicine [2012], in the public domain.*

Transposons are segments of DNA that are capable of moving or transposing themselves to new locations within the genome of a cell, as shown in Figure 2.2. They may transpose either by copy-and-paste or cut-and-paste mechanism. A location in the genome that a transposon is inserted, or amenable to be inserted in, is called an insertion site, or a target site. SB (Liu et al. [2005]; Hackett et al. [2007]), piggyBac (Geurts et al. [2006]; Hackett et al. [2007]) and Drosophila P-element (Spradling et al. [2011]) are examples of transposons. The SB transposon, whose integration data has been used in this thesis, is a well characterized DNA transposon vector used in mammals, and has been a powerful tool in functional genomics to identify

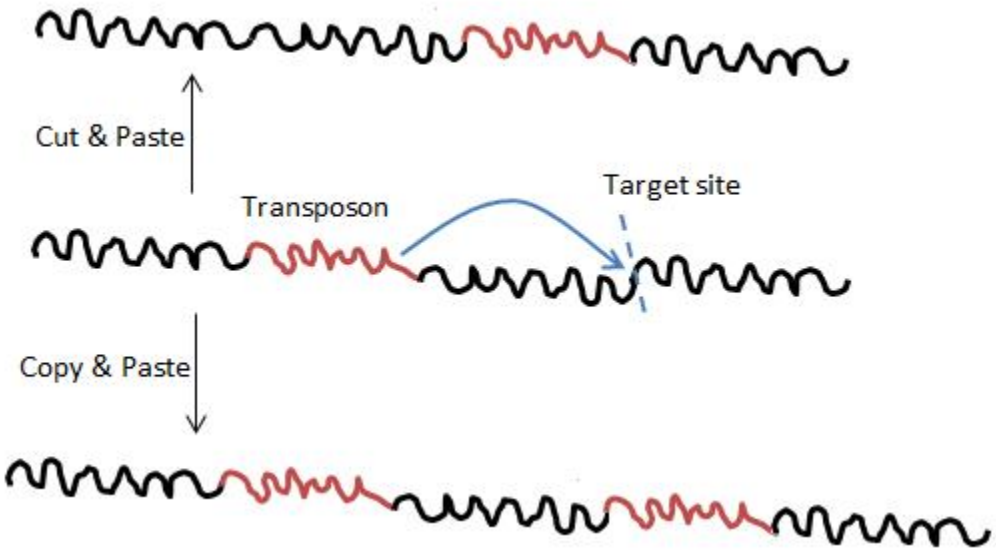

Figure 2.2: Transposon: insertion into a target site via copy-and-paste or cut-and-paste mechanisms.

genes in vertebrates, including fish and mammals (Hackett et al. [2007]). SB transposons transpose via a cut-and-paste mechanism that results in precise sequences moving from one site in DNA to another site that contains the TA dinucleotide (Liu et al. [2005]). For a comprehensive tutorial on transposons and their uses, see Broad Institute of MIT and Harvard [2012] or Hackett et al. [2007].

There are many factors that influence the target-site preferences of transposons. One of these factors is the natural deformation property or the V_{step} profile of the sequence around the target site. As Hackett et al. [2007] stated, V_{step} represents the physical relationships of any two planar base pairs in term of their relative displacements and angular orientation in the 3D-structure of DNA. V_{step} offers a measure of dimer deformability; the higher the V_{step} value, the more deformable the dimer step is,

Table 2.1: V_{step} values for dinucleotides.

dinocleotide	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
V_{step}	2.9	2.3	2.1	1.6	9.8	6.1	12.1	2.1	4.5	4.0	6.1	2.3	6.3	4.5	9.8	2.9

where a step refers to dinucleotides along a DNA sequence. More precisely, V_{step} is a composite parameter of the six standard base-pair step parameters: Shift, Slide, Rise, Twist, Roll, and Tilt. Table 2.1 shows the V_{step} values for all possible dinucleotides. For the details on V_{step} calculations, see Olson et al. [1998] and Liu et al. [2005].

Proteins are the molecules in the cell that execute almost all the cell functions. A typical protein molecule is a chain of tens to several thousand amino acid residues. In general, there are 20 kinds of amino acid molecules. The instructions to make proteins are coded in genes. A gene is made of a length of DNA. A segment of a gene that codes information for producing proteins is called an exon, and the segments between exons are called introns. Figure 2.3 shows that there is a route from genes in DNA to a protein: first, a process called transcription produces an intermediate product, named mRNA, from DNA; then, another process called translation generates the protein corresponding to the mRNA.

The N-terminus and C-terminus refer to the start and end of a protein, respectively. The start of a protein is terminated by an amino acid with a free amine group (-NH2), and the end of a protein is terminated by an amino acid with a free carboxyl group (-COOH). When an mRNA is translated, its correspondent protein is created from N-terminus to C-terminus.

Protein structure describes the various levels of organization of protein molecules. The three main levels of protein structure are primary, secondary, and tertiary structures. The primary structure refers to the polypeptide chain of amino acids. The

protein chain is linear and can be read as a sequence. Within the long protein chains there are regions in which the chains are organized into regular structures known as alpha-helices and beta strands or beta sheets. These are the secondary structures in proteins. In an alpha-helix, the protein chain is coiled like a spring; and in a beta sheet, the chains are folded so that they lie alongside each other. Other than alpha-helices and beta strands, the remainder of the chain is tight turns or loops (random coils) that connect one segment of secondary structure to the next. The tertiary structure of a protein is a description of how the whole chain, including the secondary structures, folds itself into its final 3-dimensional shape (Figure 2.4). The biochemical function of a protein is directly dependent on its three-dimensional structure. Prediction of a protein's 3D structure and function is desirable, but very complicated. There is not a one-one relationship between primary structure and tertiary structure of a protein, so primary structure cannot determine tertiary structure, but it can help in determining secondary structure. Then, a predicted secondary structure can be

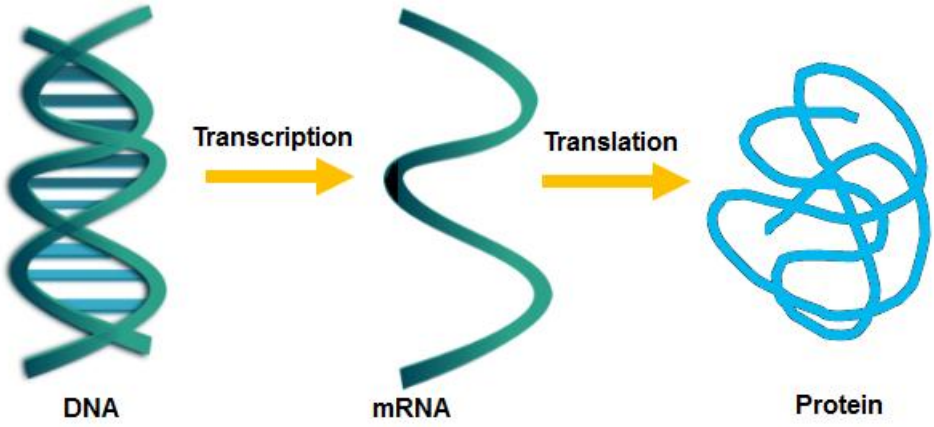

Figure 2.3: From the DNA code to the protein. *Figure is based on images created by Aguado and Lee, in the public domain.*

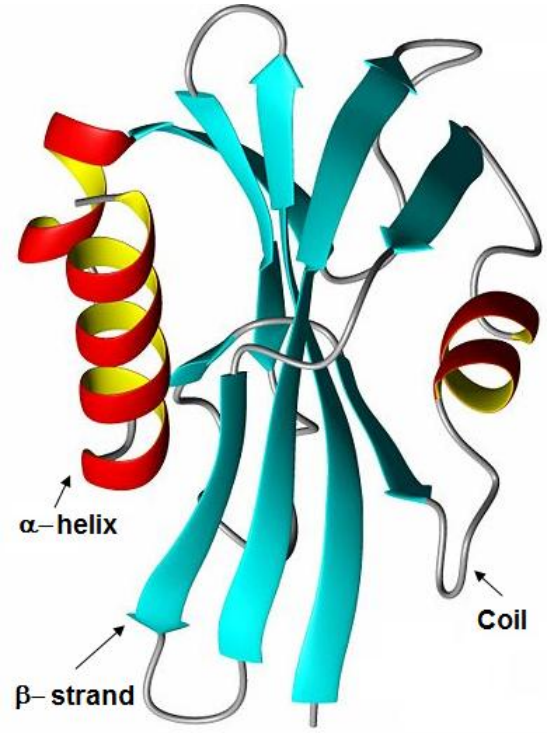

Figure 2.4: Protein Tertiary Structure. *Figure created by Volk [2007], in the public domain.*

assumed as a starting point for tertiary structure prediction.

A sequence alignment is a way of arranging two or more biological sequences, such as DNA or proteins, to identify regions of similarity. One way of describing patterns in the alignment of several DNA or protein sequences is to construct a Position-Specific Scoring Matrix (PSSM). For example, a PSSM for a multiple protein sequence alignment with any given sequence of fixed length n is a matrix with n columns corresponding to the residues in a sequence and twenty rows corresponding to twenty amino acids. Therefore, each residue position is represented by a 20-dimensional vector with integer values. These values represent effective frequencies of occurrence

at respective positions in a multiple alignment.

2.2 Machine Learning

Machine learning methods are computational models that can learn, from examples or past experiences, the theory behind a set of data. For this reason, they are suited for domains like biology where there are a large amount of data and noisy patterns, but no general theory for describing those data. Gene finding, protein structure/function prediction, and phylogenetic tree construction are instances of biological problems that take advantage of machine learning methods. The two machine learning techniques used in this thesis are ANN and SVM methods.

2.2.1 Artificial Neural Networks

The original idea of ANNs is inspired by information processing in the brain. Neurons are basic units of the brain that process and transmit information. They are connected through their synapses. An ANN consists of an interconnected group of artificial neurons (or units) in a layered architecture. Individual neurons respond to stimuli received from their input neurons and transmits their results to their output neurons. Similar to biological neurons, an activation function is assigned to an artificial neuron. Figure 2.5 shows the basic model of an artificial neuron. The basic neuron model consists of inputs $(x_0, x_1, ..., x_n)$, synaptic weights $(w_0, w_1, ..., w_n)$, output (y), and an activation function (f). The inputs, adjusted by the weight of their corresponding connections, are sent to the neuron and accumulated. The neuron

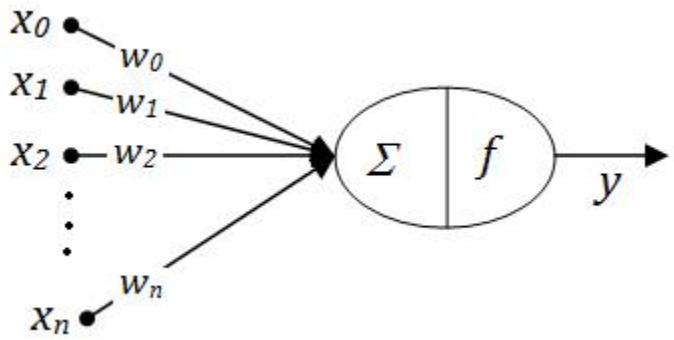

Figure 2.5: Basic neuron model

responds to the accumulated sum of inputs using its activation function:

$$y = f\left(\sum_i w_i x_i\right)$$

For example, suppose the activation function is:

$$f(x) = \begin{cases} 1 & x \geq \theta \\ 0 & x < \theta \end{cases}$$

In this case, if the accumulated sum in the neuron is greater or equal to the threshold θ, then the output y will be one (i.e., the neuron will fire). Otherwise, the output of the neuron will be zero.

Generally, every ANN has at least two layers: the input layer and the output layer. There may be one or more hidden layers between the input and output layers. Each layer consists of one or more neurons.

An ANN learns by examples, and like the brain, the learning process involves adjustments to the synaptic connections between neurons. Assume the input and output data can be represented as vectors of numerical data. The learning or training phase is an iterative process to produce the correct outputs for the given inputs using

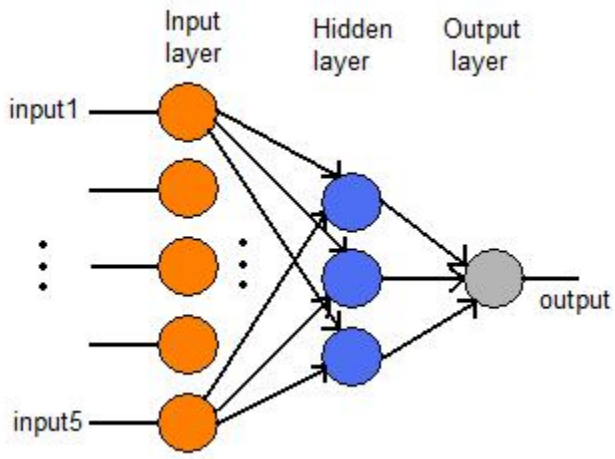

Figure 2.6: An example of a three-layer ANN. It has 5+3+1 neurons, and 15+3 connections, correspondingly.

a learning algorithm. We repeatedly feed the ANN an example (an input-output pair), compare the output on this example (the actual output) with the desired output and adjust the values of the connections (i.e., the weights) to generate better output (i.e., closer to the desired value) next time. An iteration, or the period during which each pattern is presented once to the network and the weights are updated, is called a learning epoch. Generally, training a network requires many epochs. After training, the ANN is ready to respond to new, unseen patterns.

Figure 2.6 shows an example of a multilayer ANN. Suppose this ANN receives as input a piece of DNA which is five nucleotides in length, and it predicts if the central residue is part of an intron or not. Correspondingly, it has five neurons in the input layer and one neuron in the output layer. Also, it has one hidden layer with three neurons. In the training phase, all the 18 connections are adjusted to appropriate values.

Radial Basis Function (RBF) Neural Network

An RBF neural network is a type of ANN for application to problems of supervised learning, such as classification and regression. RBF networks can be viewed as a mixture of competitive networks and back propagation networks. Competitive networks learn the local features of patterns, so they can handle exceptions, i.e., the patterns that do not follow the general rules. Back propagation networks learn the global features of patterns, so they can interpolate when a test pattern does not exactly match the training pattern. RBF networks learn specific classes of patterns (prototypes) while still being able to interpolate.

The RBF network consists of three layers: the input layer, the hidden layer, and the output layer. The input neurons receive the input data. The output layer consists of linear processing neurons. The activation function of the hidden neurons is a radial basis function. The most common radial basis function is the Gaussian function:

$$f(x) = e^{-\frac{(x-u)^2}{2\sigma^2}}$$

or

$$f(x) = e^{-\gamma(x-u)^2}$$

where x is an input vector, u is the position of the RBF center (i.e., the center of the hidden neuron), and σ (or its inverse, γ) defines the RBF width (i.e., the receptive field for the hidden neuron). The graph of a Gaussian function has a bell shape as shown in Figure 2.7 for the case of one-dimensional input space, with different values for the parameter σ. The main characteristic of a Gaussian RBF is that its response decreases monotonically by increasing the distance from its center. Actually, an RBF

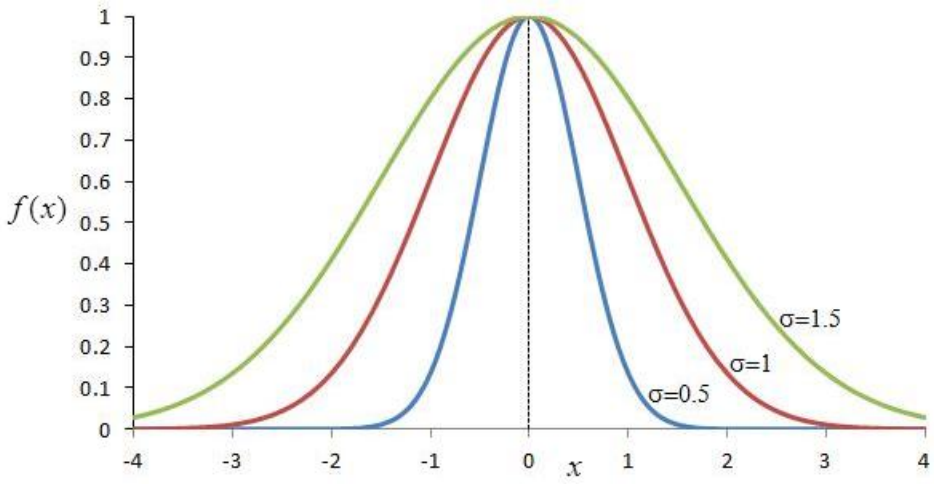

Figure 2.7: Gaussian radial basis function: $f(x) = e^{-(x-u)^2}/2\sigma^2$, where u=0 and σ=0.5, 1, or 1.5.

network positions the RBF neurons in the space described by the input variables. The Euclidean distance is computed from an input pattern to the center of each neuron, and an RBF is applied to the distance to compute the weight (influence) for each neuron. The further a neuron is from the point being evaluated, the less influence it has. Both parameters u and σ must be learned (or set) from training patterns during the learning process.

Learning in RBF networks has two phases: competitive learning followed by delta learning. During competitive learning, the network allocates the training patterns to the hidden neurons, so the network learns the weights between the input layer and the hidden layer. Indeed, RBF parameters in the hidden neurons are the synaptic weights of the input to the hidden layer. Once the weights of the input to the hidden layer are found, they are kept fixed while the network learns the weights between the hidden layer and the output layer using delta learning. After training, when an

unseen pattern is presented to the network, each hidden neurons generates a value that indicates the degree of closenesses of the input pattern to the prototype pattern stored at that neuron, and the output neurons perform the interpolation. The learning process in RBF networks is fast, which is their main advantage.

For additional information on different organization of ANNs, see Haykin [2009].

2.2.2 Support Vector Machines

An SVM is a classification method based on statistical learning theory. Assume we can represent each piece of data as a vector of numerical data in a high dimensional vector space. Suppose we are given a set of training examples from two different classes, and we have to classify each of the new, unseen examples to one of the two classes. An SVM constructs a hyperplane in the vector space to separate the two classes of training data. The best separating hyperplane has the largest distance (margin) to the nearest data point of any class (such a point is called a support vector). Having the hyperplane, we are able to classify unseen data. However, many real data sets may not be linearly separable. In this case, the SVM has two solutions to handle the problem: assuming a soft margin and using a kernel function. The soft margin assumption allows a few training data to fall on the wrong side of the hyperplane (i.e., it allows training errors). Also, the SVM uses a kernel function (explained later), and maps the training examples as points in a high dimensional space, i.e., the feature space, in a way that they are separable by a gap as wide as possible. Then, a hyperplane is passed through the gap to do the classification. Figure 2.8 illustrates the basic idea of SVM by an example.

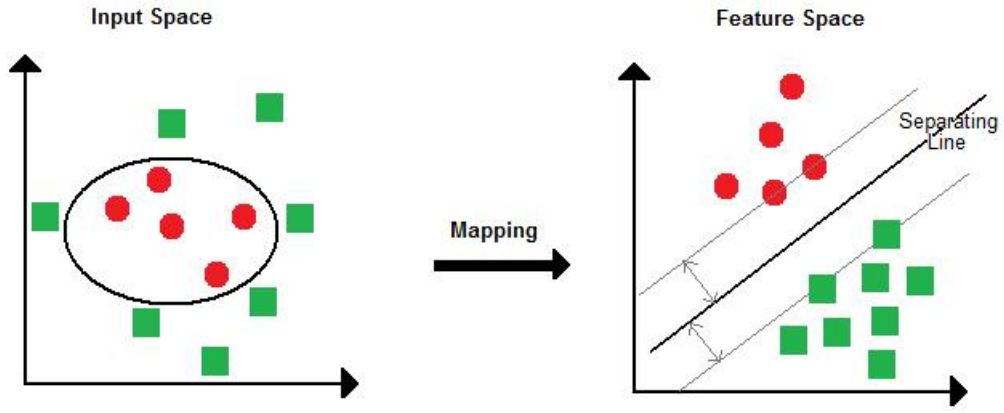

Figure 2.8: An example of a linear SVM: The two classes are represented by red circles and green squares. As the input samples are not linearly separable in the input space, they are mapped to the points in the feature space. The samples in the feature space are separable by the best separating line.

The Binary SVM Formulation

Given an implicit mapping function ϕ (or a kernel function K) and training data $(\mathbf{x}_i, y_i)$ from two classes such that $y_i = \pm 1$, an SVM finds an optimal hyperplane $\mathbf{w^T}\phi(\mathbf{x}) + \mathbf{b} = \mathbf{0}$ that separates the two classes, as shown in Figure 2.9 for the case of two-dimensional data. The learned hyperplane maximizes the margin while minimizing some measure of loss on the training data. The primal formulation of this binary SVM for a set of l training samples is:

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

$$\textit{subject to} \quad y_i(\mathbf{w}^T\phi(\mathbf{x}) + \mathbf{b}) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \quad i = 1, ..., l$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

where $\mathbf{w}$ is the normal vector of the hyperplane, C is the user specified misclassification penalty, $\sum_{i=1}^{l} \xi_i$ is the training error, $\mathbf{b}$ is the offset of the hyperplane from the origin, and K is defined by an inner product. An unseen sample $\mathbf{x}$ is classified as ± 1 by evaluating $sign(\mathbf{w}^T \phi(\mathbf{x}) + \mathbf{b})$.

This primal formulation cannot be solved directly, since the mapping function ϕ is unspecified and it is hard to compute. Indeed, the solution is obtained by moving to the dual formulation of the optimization problem. In the dual formulation, the kernel function K appears, which is simple to evaluate and bypasses the intensive computation of ϕ. This is because K simply computes the inner products between the images of all pairs of data in the feature space, without ever computing the coordinates of the data in that space. However, there are some suitable kernel functions that correspond to the inner product of the data after mapping. Therefore, one of these functions can be picked, instead of computing the inner products. A commonly used kernel function is a Gaussian RBF:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma(\mathbf{x}_i - \mathbf{x}_j)^2}$$

SVM for Regression

A version of an SVM for function estimation from a set of training data is called support vector regression (SVR). In epsilon-SVR, the goal is to find a function $f(\mathbf{x})$ that has at most ϵ deviation (where ϵ is the margin of tolerance) from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. Suppose $f(\mathbf{x})$ is a linear function in the form as

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + \mathbf{b}, \quad \mathbf{w} \in \mathbb{R}^n \; and \; \mathbf{b} \in \mathbb{R}$$

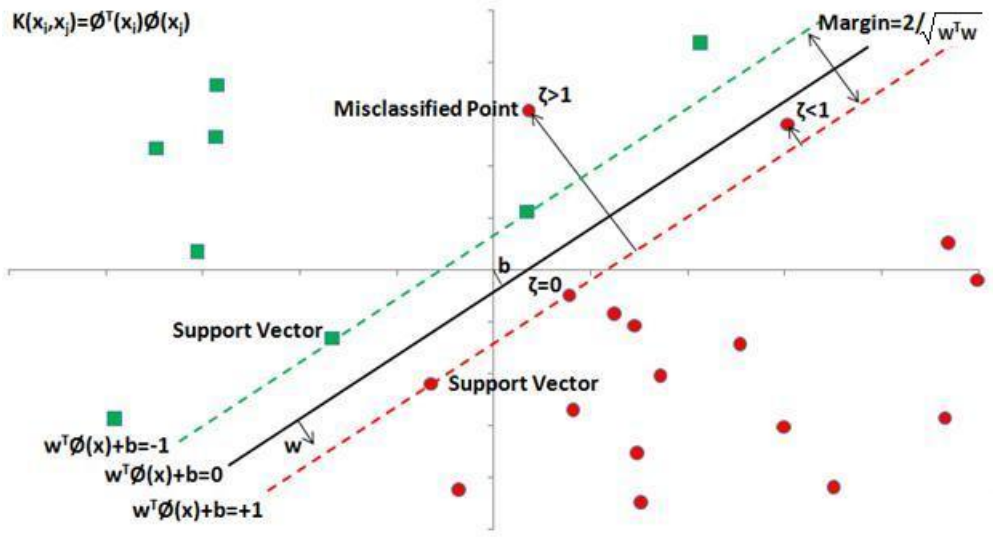

Figure 2.9: The Binary SVM learns an optimal hyperplane. Red circles have a label $y_i = +1$, while green squares have a label $y_i = -1$.

In this case, flatness means finding a small $\mathbf{w}$, or minimizing $\|\mathbf{w}\| = \langle \mathbf{w}, \mathbf{w} \rangle$. The epsilon-SVR uses the same principals as the SVM for classification with a few minor differences. The main difference is that ϵ is requested from the problem as a parameter.

For a comprehensive tutorial on the general idea and basic concepts of SVM models, see Noble [2006], and for additional information on the mathematical foundations of the SVM theory for classification and regression, see Vapnik [1998] and Smola and Scholkopf [2004], respectively.

2.3 Statistics

2.3.1 Measures

Sensitivity, Specificity, and Accuracy

There are three main statistical measures for expressing the estimated performance in a binary classification:

- Sensitivity (SN), measures the accuracy of positive classifications, and is defined as

  $SN = TP/(TP + FN)$,

  where TP and FN refer to the number of true positives and the number of false negatives, respectively.

- Specificity (SP), measures the accuracy of negative classifications, and is defined as

  $SP = TN/(TN + FP)$,

  where TN and FP refer to the number of true negatives and the number of false positives, respectively.

- Accuracy (ACC), represents the proportion of true results, both positive and negative, in the selected population, and is defined as

  $ACC = (TP + TN)/(TP + FP + TN + FN)$.

Correlation Coefficient

Pearson's correlation coefficient (r) is a measure of linear dependence between two variables X and Y:

$$r_{xy} = \frac{\sum_{i=1}^{n} X_i Y_i - \frac{\left(\sum_{i=1}^{n} X_i\right)\left(\sum_{i=1}^{n} Y_i\right)}{n}}{\sqrt{\left(\sum_{i=1}^{n} {X_i}^2 - \frac{\left(\sum_{i=1}^{n} {X_i}^2\right)^2}{n}\right)\left(\sum_{i=1}^{n} {Y_i}^2 - \frac{\left(\sum_{i=1}^{n} {Y_i}^2\right)^2}{n}\right)}}$$

where n refers to the number of pairs of data. The correlation coefficient shows that how closely one variable is related to another variable. The value of r falls in the range [-1,1]. A correlation coefficient of 0.00 means that there is no relationship between the two variables. The closer a correlation coefficient to +1 or -1, the stronger the relationship is between variables. In this thesis, we will compute the correlation coefficient between the observed and predicted values. In this case, if an X_i is the observed output value for an input, its corresponding Y_i represents the predicted output value for that input by a predictor.

Akaike Information Criterion

The Akaike information criterion (AIC), mentioned in Section 4.2.2, is a measure of the relative goodness of a statistical model. The AIC does not tell anything about how well a model fits the data in an absolute sense; rather it is a tool for comparison and selection among several statistical models. For example, the AIC for a least squares regression model is defined as

$$AIC = n \ln\left(\frac{RSS}{n}\right) + 2k$$

where n is the number of data points (observations), k is the number of parameters in the model, and RSS is the residual sums of squares (which is a measure of discrepancy between the actual data and the estimated model). If $n/k < 40$, then the AIC requires a bias-adjustment:

$$AIC = n \ln\left(\frac{RSS}{n}\right) + 2k + \frac{2k(k+1)}{n-k-1}$$

2.3.2 Cross-validation

The performance of a predictive model such ANN or SVM is estimated by using an evaluation method like cross-validation. In k-fold cross-validation, we partition the original dataset into k equal-sized subsets, randomly. Then, we train our model k times, each time leaving one of the k subsets for testing, and using all the remaining $k-1$ subsets for training. In this way, each of the k subsets is used exactly once as the test set:

$$dataset = \bigcup_{j=1}^{k} d_j$$

$$Fold_i : \quad testset = d_i, \quad trainset = dataset - d_i,$$

$$i = 1, ..., k.$$

The k results from the folds can be averaged (or combined) to produce the final evaluation result.

As a cross-validation result depends on how a given dataset is divided into folds, sometimes repeated cross-validation is applied. In repeated k-fold cross-validation, k-fold cross-validation runs for several times (e.g., N) using different split into folds, and the result will be the average of all N cross-validation results. In general, cross

validation has some bias and variance, like any other estimator. As it may have relatively high variance, using repeated cross-validation may reduce the variance. However, if a classifier has stable predictions for a given dataset (i.e., all iterations have the same outcome), then repeated cross-validation does not reduce the variance. In this case, the cross validation estimate will be unbiased and the variance of the estimated accuracy will be approximately $ACC(1 - ACC)/n$, where ACC is the accuracy of the k-fold cross-validation and n is the number of instances in the dataset (see Kohavi [1995]).

2.3.3 ROC Curve Analysis

A Receiver Operating Characteristic (ROC) curve is a plot of the True Positive Rate (TPR) versus the False Positive Rate (FPR) for different possible cut-offs of a binary classifier system. A cut-off specifies a level for discriminating positive and negative classes. For example, in the case of having a predictor that produces real valued outputs, using a cut-off on the outputs will generate binary values. A typical ROC curve is shown in Figure 2.10. Generally, ROC curve analysis is undertaken to:

1. Assess the overall discriminatory ability of a binary classifier, i.e., the ability of a classifier to correctly classify the samples into two groups, positive and negative. The area under the curve (AUC) is a common metric to compute the strength of a classification: An AUC close to 1 indicates a strong test, and an AUC close to 0.5 (i.e., random classifier performance) represents a weak test.

2. Recognize the best SN and SP, or the best trade-off in the TPR and the FPR. As a result, ROC curve helps in determining the best cut-offs that maximize

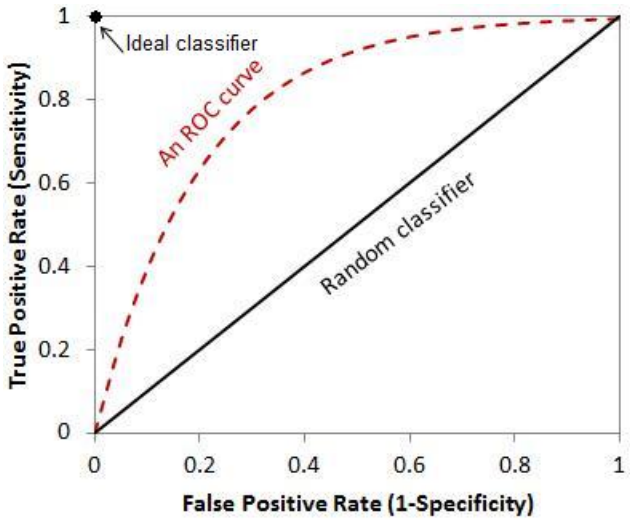

Figure 2.10: A typical ROC curve.

SN (TPR) and minimize 1-SP (FPR).

Chapter 3

Prediction Problems in Biology

Research on prediction problems for DNA and proteins includes gene prediction, protein secondary structure prediction, prediction of signal peptides and their cleavage sites, DNA-binding protein site prediction, restriction enzyme site prediction and the splice site prediction problems. These prediction problems occupy a large area in bioinformatics and there are efficient and accurate solutions based on machine learning methods such as SVMs and ANNs for them. As examples, we describe four of these problems and their known SVM-based and ANN-based solutions. However, it is worth mentioning that there may also be other machine learning solutions for such prediction problems, such as hidden Markov models (e.g., Eddy [1998]), probabilistic graphical models (e.g., Huelsenbeck and Ronquist [2001]), stochastic grammars (e.g., Knudsen and Hein [2003]), evolutionary algorithms (e.g., Notredame and Higgins [1996]), or even some hybrid methods (e.g., Li et al. [2005]). Studying and comparing these solutions is beyond the scope of this thesis, and the reason for focusing only on SVM and ANN solutions is to illustrate that these two machine learning methods

have been efficient for prediction problems that are similar to the subject of this thesis.

SVM models and ANNs are closely related to each other. Regardless of their problem solving techniques, they both use similar representation methods for input and output data, both have comparable fault tolerance, and both are appropriate for sequential data with sliding window methods, and for non-sequential data with fixed window methods. They are also analogous to each other in that they may be inappropriate in some problems such as gene prediction in which hidden Markov models may be successful (e.g., Majoros et al. [2003]). In gene prediction, using windows, fixed or sliding, is not applicable, since whole genome sequences must be processed for identifying the regions of DNA that encode genes.

3.1 Prediction of Signal Peptides and their Cleavage Sites

A signal peptide is a short sequence — 5 to 40 amino acids long — present at the N-terminus of a protein, which is cleaved off while the protein goes out of the cell membrane. The site at which a signal peptide ends is called the cleavage site. Signal peptides from different proteins do not have similar sequences; however, they have some structural similarities like having a hydrophobic core, a positively charged region in the N-terminus, and an uncharged region just before the cleavage site. Also, the identification of them is rather organism-specific. Effective identification of signal peptides and prediction of their cleavage sites have an important role in production

of proteins in recombinant systems, as Baldi and Brunak [2001] stated.

3.1.1 Using ANN

The work done by Nielsen et al. [1997] is the most successful solution for the signal peptide prediction problem. Considering the first 60-80 amino acids in the analysis, Nielsen et al. constructed two types of neural networks which provide a different score between 0 and 1 for each amino acid in the sequence. The idea of designing two types of networks arose due to Nielsen et al.'s two different approaches to the prediction problem:

(i) Classifying all amino acids in the sequence as belonging to the signal sequence or the mature protein; and

(ii) Classifying all amino acids in the sequence as cleavage site neighbors or non-cleavage site neighbors.

Nielsen et al. interpreted the output from the signal peptide/non-signal peptide networks, the S-score, as an estimate of the probability of the position's belonging to the signal peptide; and interpreted the output from the cleavage/non-cleavage networks, the C-score, as an estimate of the probability of the position's being the first in the mature protein. They combined these two scores in the form of a Y-score for each position in the sequence:

$$Y_i = \sqrt{C_i \Delta_d S_i}$$

where $\Delta_d S_i$ is the difference between the average S-score of d positions before and d positions after the position i:

$$\Delta_d S_i = \frac{1}{d}\left(\sum_{j=1}^{d} S_{i-j} - \sum_{j=0}^{d-1} S_{i+j}\right)$$

Figure 3.1 shows the S-score and C-score values for two putative sequences. Figure 3.1 (a) shows that there is an abrupt change in S-score curve, while there is one sharp peak in C-score curve. In this case both S-score and C-score correctly predicted the cleavage site position. Figure 3.1 (b) shows that C-score curve have several peaks. Prediction only based on maximum C-score value is incorrect, while predicting based on the combined Y-score identifies the true cleavage site.

Nielsen et al. designed back-propagation ANNs (for both S and C scores) with zero or one hidden layer and 2-10 hidden units, and used the following procedure for training and testing the networks: First, they divided their organism-specific data into five subsets. Then, they trained through cross-validation an S-score ANN and a C-score ANN for each subset of data, independently. Therefore at the end of this phase, they had five C-score ANNs and five S-score ANNs. Finally, they obtained the individual C-score and S-score by averaging over the related five ANNs. During training, they also found that using a symmetric large sliding window for S-score networks (e.g., 39 amino acids long for E.coli), and an asymmetric small one for C-score networks with 15 amino acids upstream and 2-4 amino acids downstream of the cleavage site, gave the best results. They evaluated the performance of their method with two measures: the percentage of correctly identified the cleavage sites, and the amount of signal peptide discrimination in terms of correlation coefficients. They could obtain 67.9%, 79.3% and 70.2% accuracy in cleavage site identification,

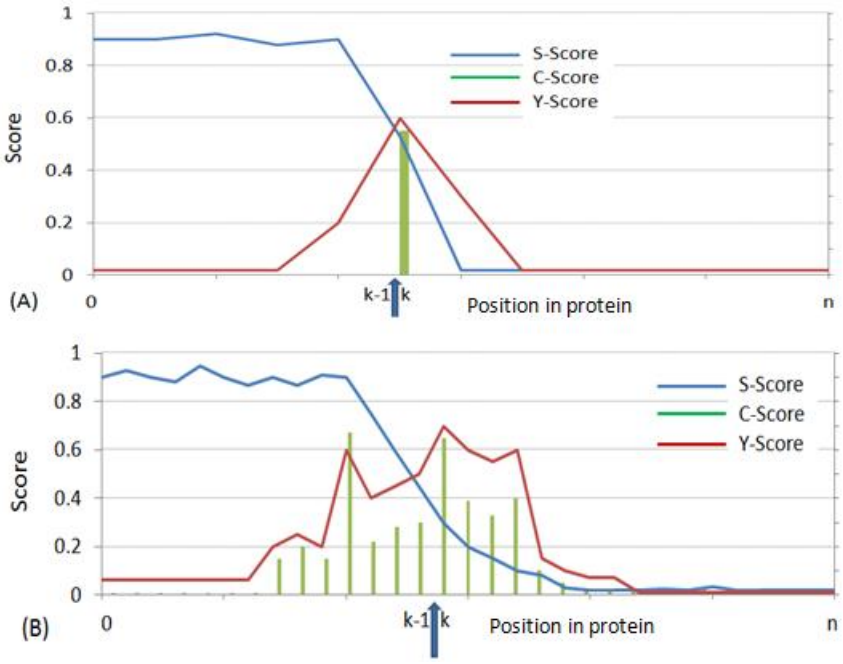

Figure 3.1: Examples of signal petide and cleavage site predictions for two putative sequences. The S-score, C-score, and Y-score values are shown for each position in the sequence. The true cleavage sites are marked with arrows. **A** is a putative sequence with all positions correctly predicted according to both C-score and S-score. **B** is a putative sequence with all positions correctly predicted according to the combined Y-score. *The original idea of this figure obtained from Baldi and Brunak [2001].*

and 96%, 88% and 97% in signal peptide discrimination for gram-positive bacteria, gram-negative bacteria and Eukaryotes, respectively.

This solution is a distinct example among ANN-based applications in biology because of the intelligent interpretation of the problem which has improved the overall performance.

3.1.2 Using SVM

There are also SVM-based solutions for signal peptide prediction problem such as Vert [2002], but none of them are as successful and strong as the ANN-based solution of Nielsen et al. [1997]. As the results of the existing SVM methods are not comparable to the similar ANN method, we refrain from illustrating them here.

3.2 Prediction of Protein Secondary Structure

Prediction of secondary structure is one of the classic problems in bioinformatics. Its goal is to classify a pattern of adjacent amino acid residues as alpha-helix (α), beta-sheet (β) or coil (C).

3.2.1 Using ANN

There are several ANN-based tools for secondary structure prediction problem:

- PHDsec, by Rost and Sander [1993],
- The work done by Riis and Krogh [1996],
- PSPIRED, by Jones [1999],
- SSpro, by Cheng et al. [2005], based on methods from Pollastri et al. [2002], and
- Jpred 3, by Cole et al. [2008], based on Jnet method from Cuff and Barton [1999].

All of these works have their own unique design; however, their levels of accuracy are comparable to each other, with 70–80% accuracy. Each has its own users and may produce different results for a specific organism. There is no benchmark that shows their superiority. As a sample, I explain the PHDsec architecture developed by Rost and Sander.

Rost and Sander developed a multi-level ANN system for predicting secondary structure. They had found that sequence families contains much more structural information for extraction than single sequences, so they used sequence profiles as inputs for their ANN system. First, they extracted similar sequences from a protein database. Second, they generated a multiple sequence alignment for the similar sequences. Third, they used amino acid-residue frequencies derived from the multiple sequence alignment as input.

Figure 3.2 shows the architecture of PHDsec. PHDsec has three distinct ANN-based levels:

- first level: sequence to structure network,
- second level: structure to structure network, and
- third level: jury decision network.

PHDsec looks like a 3-layer neural network where each layer consists of neural network subunits. The first level is the input layer. A network in this level gets a sequence alignment in a window with w residues, and predicts (for the central position residue) real values in $[0, 1]$ for α, β and C. The input layer has $20w$ units, i.e., 20 units, one for the proportion of each possible residue at a position in the window. The second level,

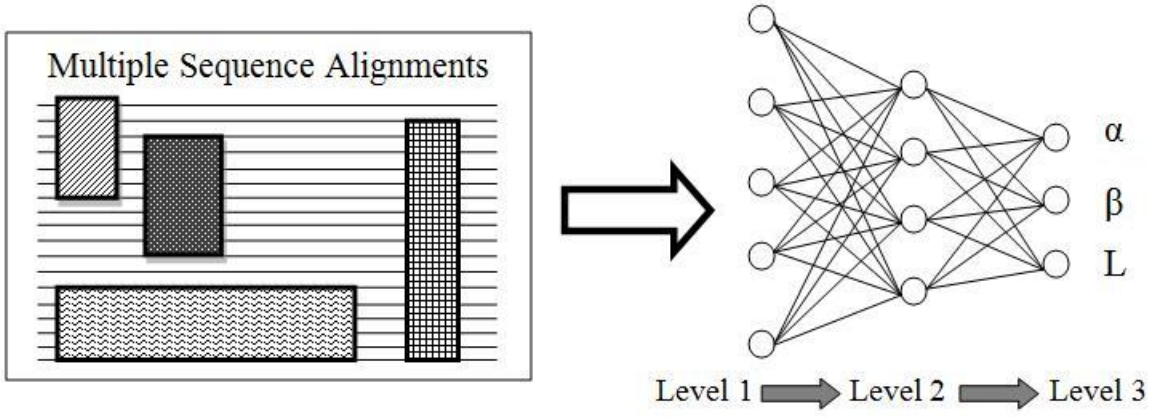

Figure 3.2: The PHDsec architecture for secondary structure prediction. *The original idea of this figure obtained from Rost and Sander [1993].*

or hidden layer, reads the outputs of the first level, and predicts three outputs for alpha-helix, beta-strand and loop(coil). The second level learns structural context, e.g., length of the secondary structures. The third level, or output layer, consists of 12 networks (4 separately trained 3-output networks) for predicting the secondary structure of the residue at position i in the first window. A jury of the 12 networks, by averaging or voting, specifies the secondary structure for the residue. Rost and Sander could achieve to 69.7% accuracy using a 7-fold cross-validation on 130 proteins.

The key contribution of the PHDsec method was the use of multiple sequence alignments that contain more structural information than single sequences. After PHDsec, the PSPIRED and the Jpred methods also used similar neural network architecture as the PHDsec method, but with different alignment approaches for input data that made some improvements over the PHDsec method. PSPIRED uses the PSI-BLAST method, which gives PSSMs, to find more distantly related sequences. The Jpred 3 method uses PSSMs and profile HMMs as input to the neural network,

and it also changed the number of hidden nodes in the sequence-to-structure units. Jpred3 claims 81% accuracy in the prediction of secondary structures.

3.2.2 Using SVM

Hua and Sun [2001] introduced a method based on SVM for protein secondary structure prediction. First, they considered two protein datasets, one used by Rost and Sander [1993] in PHDsec with 126 proteins (RS) and the other used by Cuff and Barton [1999] in Jpred with 513 proteins (CB), so they are able to train and test their SVM method and compare it to the most successful ANN methods. Second, they constructed six SVM-based binary classifiers: $\alpha/\neg\alpha$, $\beta/\neg\beta$, $C/\neg C$, α/β, β/C and C/α. Similar to the neural network approach, they used a coding scheme for protein sequences. The input vectors had length $21w$ where w was the sliding window length. The input vector was a multiple sequence alignment. Hua and Sun chose the RBF kernel, and tried to find the best SVM related parameters and also the optimal window size by training and testing each binary classifier on the RS dataset. Third, they used the best parameters in binary classifiers, ran 7-fold cross-validation tests on both datasets, and measured the accuracy of each binary classifier. Fourth, they assembled the binary classifiers to make a tertiary classifier for the three secondary structure states (α, β and C).

Figure 3.3 shows an example of the decision-tree structure of a tertiary classifier. Hua and Sun constructed several tertiary classifiers with different combinations and arrangements of binary classifiers to find the best tertiary classifier. Then, they used the jury technique to combine all the results of the constructed tertiary classifiers. Hua

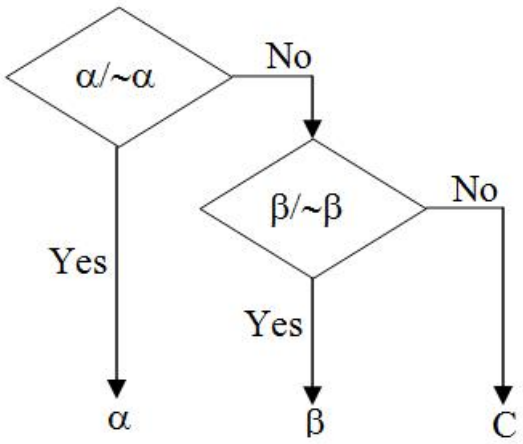

Figure 3.3: An example of a SVM tertiary classifier with two cascaded binary classifiers for secondary structure prediction. *The original idea of this figure obtained from Hua and Sun [2001].*

and Sun evaluated their classifier using a different accuracy measure called Segment OVerlap (SOV) accuracy. The SOV measure, proposed by Zemla et al. [1999], is used for the evaluation of secondary structure prediction methods, and it is based on the average overlap between the observed and the predicted segments instead of the average per-residue accuracy. Hua and Sun's final SVM-jury classifier had 76.2% SOV using a 7-fold cross-validation on the CB dataset. Hua and Sun did not provide accuracy, as defined in Section 2.3.1. Therefore, direct comparison of their method with the ANN-based methods in Section 3.2.1 is not possible.

Later, Kim and Park [2003] made some improvements to the method of Hua and Sun [2001], by using a PSSM framework for input, and constructed SVMpsi with 80.1% SOV on the CB dataset.

3.3 DNA-binding Site Prediction

A DNA-binding protein is a protein that binds to a DNA, typically to pack or modify the DNA, or to regulate gene expressions. Predicting DNA-binding sites in proteins is a demanding task, and its goal is to find the probable binding amino acid residues of proteins.

3.3.1 Using ANN

Ahmad and Sarai [2005] developed an ANN for finding DNA-binding sites. In their ANN, the evolutionary information of a residue and its two neighbors on each side of it, expressed as a PSSM, is used as an input. Then the ANN identifies whether the central residue is part of a DNA-binding site or not, as its output. They trained and tested their ANN with a six-fold cross validation on a dataset with 62 DNA-binding proteins, and achieved 68.2% sensitivity and 66.0% specificity on their proteins dataset.

3.3.2 Using SVM

Ma et al. [2009] developed a model based on SVMs for predicting DNA-binding residues. In their model, each segment of the sequence with 11 residues is an input instance, and the goal is to classify its central residue to either the DNA-binding class or the non-binding class. For an input instance, they extracted five features: evolutionary information in terms of a PSSM plus four other physical-chemical features. Using this SVM on a dataset with 62 DNA-binding proteins (same as protein-DNA complexes used in the ANN case), they obtained 74.19% sensitivity and 79.20% speci-

ficity, using cross-validation.

3.4 Splice Site Prediction

The RNA splicing process removes the introns of a premature mRNA, and joins the remaining parts or exons together. The result is a mature mRNA. Almost all introns have a GT dinucleotide at the upstream end, or donor site, and an AG dinucleotide at the downstream end, or acceptor site, but not all GT and AG dinucleotides correspond to intron boundaries. In a typical splice site prediction problem, the objective is to find the probable splice sites of a DNA sequence.

3.4.1 Using ANN

Ogura et al. [1997] developed two sets of ANNs, one for learning GT dinucleotide splice sites and one for learning AG dinucleotide splice sites, with different topologies in each set. They extracted an asymetric window of bases around all potential splice sites (30 bases in the intron side and 20 bases in the exon side), and made two distinct training and testing datasets with more than 2000 patterns, which included both positive and negative cases, for each set of ANNs. Then, they trained their ANNs with the entire training data (without using cross validation). Ogura et al. tested their ANNs with the testing dataset and achieved at least 80% percent specificity and sensitivity on test data.

3.4.2 Using SVM

Sonnenburg et al. [2007] constructed an SVM-based predictor to recognize the true and false splice sites in worm, fly, cress, fish, and human genomes. Their model input space contained representations of sequences of length N, ranging from 18 to 218, where each sequence is labeled either +1 or -1 correspond to true and false splice sites, respectively. They used three kernel functions on different sequence lengths, and developed several SVMs. Also, they collected millions of training examples. Then, they trained and evaluated their SVMs using five-fold cross-validation for each of the acceptor and donor splice site recognition problems of the model organisms. Their results indicated very accurate prediction (more than 0.95 value for AUC) in all the model genomes.

Chapter 4

Insertion Site Prediction Problem

There are many works on finding preferred insertion sites of transposons, but most of them exclusively focused on the biological aspects. Here, we describe some of those solutions that have noted some computational aspects, in two sections: solutions that have a statistical approach and solutions that have a feature-based approach. Some of the works try to find insertion sites only based on the statistical analysis of the existing experimental data of insertion sites, while some try to predict insertion sites based on the sequential or structural features of DNA.

4.1 Statistical Approaches

de Ridder et al. [2006] introduced a framework, which uses a kernel density estimation technique, to find Common Insertion Sites (CISs) in a genome. As defined by de Ridder et al., a CIS is a region in the genome that has been hit by viral insertions in multiple independent tumors significantly more frequently than expected by

chance. In statistics, kernel density estimation is a non-parametric way to estimate a prediction function. Non-parametric estimators do not have a fixed structure and depend upon all the data points to reach an estimate. In kernel density estimation, a distribution is estimated from sample points by convolution with a kernel, such as a Gaussian kernel. This framework is capable of taking into account both noise and bias. This property is advantageous for insertion site detection, as the insertion process has both a random nature (noise) and a preferential nature (bias). Another property of this framework is scalability, i.e., the data can be evaluated at any biologically relevant scale; however, it is very time-consuming for large datasets, requiring hundreds to thousands of CPU hours, as Bergemann et al. [2012] stated. For additional information on kernel density estimation and convolution method, please see de Smith [2011] and Weisstein [2012].

The research done by Bergemann et al. [2012] is a recent work in computational prediction of insertion sites. Bergemann et al. used a Poisson regression model for finding CISs. In this model, the number of times that insertions appear within a defined interval follows a Poisson distribution. The Poisson distribution density function is:

$$P\left(X = k\right) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

where the parameter λ is the rate of insertion and k is the number of insertions residing within a given window. For individual regions R_i, $i = 1, 2, ..., n_w$, where n_w is the number of windows of size w, the Poisson regression calculates the expected rate of insertion λ_i for region R_i using some information about the region (e.g., the window size and the number of TA sites within the window). They used their regression model

to analyze two insertional mutagenesis datasets, one transposon-based and one virus-based genetic screens, for finding relevant cancer genes. Bergemann et al. found that their method identifies more relevant genes compared to some similar previous methods, but they did not report any specific accuracy level for their method.

Spradling et al. [2011] studied Drosophila P-element transposon preferred insertion sites. They observed that a single copy of this transposon in a genome can multiply rapidly in the next generations of the genome; however, the P-element transposon uses cut-and-paste mechanisms, and it does not inherently increase copy number. They analyzed more than 18000 independent transpositions and found that P-elements selectively integrate into a subset of promoters of a genome. Then they compared the distribution of P-element insertion sites and the promoters that were also replication origins of the cell, and observed a striking correlation between them. Carrying out several other comparisons, Spradling et al. found that P-elements preferentially transpose to replication regions. They concluded that their findings might explain the increasing copy number of P-elements in a genome.

4.2 Feature-based Approaches

In the following, we explain the problem and solution methods of several research works whose goal is to identify insertion sites based on the features in the DNA sequence or structure. We describe these works in two separate sections, based on their solution methodologies.

4.2.1 Statistical Methods

The research done by Berry et al. [2006] is a significant work where a mathematical method has been used to find the target sites of tranposons based on genomic features. Berry et al. studied the target site selection of some transposons in the human genome. First, they considered several genomic features (e.g., gene density and CpG islands) and transposable elements (e.g., SB). Then, they analyzed the influence of each genomic feature in the selection of integration sites by a specific transposon, using ROC curve analysis. In particular, they collected a pool of positive and negative transposon integration sites, used a range of cut-points for each genomic feature to sort the sites into true and false integration sites, and plotted the corresponding ROC curves. Using the area under the ROC curves, they could predict the likelihood of hosting an integration event within a specific genomic interval size for each base pair in the human genome. They also found that their results may be very different depending on the interval size.

For applying a method similar to Berry et al.'s, large-scale genomic datasets are required. Also, this method helps in finding global preferences for integration. It does not determine specific sites of integration.

4.2.2 Rule-based Methods

This section includes the two papers, by Liu et al. [2005] and Geurts et al. [2006], which are the works that this research is extending.

Liu et al. studied multiple factors that have influence on the target-site selection process in a genomic environment. They found that among the studied factors, the

local DNA structure has additional influence in the target-site selection process. They identified two properties in the local DNA structure around insertion sites of the SB transposon: the existence of TA sites, and a specific deformability property or V_{step} profile in the vicinity of the TA dinucleotide. Specifically, Liu et al. found that there is a relationship between the V_{step} profile of target sites and the mechanism of the target-site selection of SB transposon.

Geurts et al., aiming to generalize the work of Liu et al., analyzed integration-site preferences of SB, piggyBac and Drosophila P-element transposons to detect V_{step} patterns for preferred sites. Geurts et al. developed a program, called ProTIS, for generating V_{step} profiles of 12 bp TA insertion sites (i.e., a target TA dinucleotide in the middle plus 5 bp flanking each side of it) using the values in Table 2.1. ProTIS identifies the TA sites in a target genome and scores the sites as a function of their V_{step} profiles. Geurts et al. tried to develop rules based on the profiles of integration sites of each transposon separately. They considered the 7758 bp plasmid pFV/Luc (Fish Vector Luciferase) as the host genome, and the SB transposon. In this case, Geurts et al. extracted all the 12 bp TA sites of the host genome, and obtained the V_{step} profile of each site. Then, they used the rules, and classified the TA sites into three classes - basal, semi-preferred and preferred - based on the graphical pattern of the V_{step} profiles of the sites. Figure 4.1 shows how the rules worked. For example, if a V_{step} profile of a TA site had four peaks in its diagram, it would be categorized to the preferred class. On the other hand, having the experimental data, Geurts et al. calculated the number of transposon insertions per each class of TA site, and obtained three coefficients corresponding to the three classes of TA sites. They used

these coefficients in calculating the total V_{step} score for every bin of a given length of the sequence using the formula:

$$Total\ V_{step} = \sum_{N \to N+(binsize)} [13(\#\text{preferred sites}) + 5(\#\text{semi} - \text{preferred sites}) + 1(\#\text{basal sites})]$$

Afterward, Geurts et al. divided the host sequence into 78 bins of size 100bp, obtained the number of each class of TA site per bin, and calculated the total V_{step} score for each bin using the aforementioned formula. Figure 4.2 (a) and (b) show the related plots. Comparing the distribution of observed insertion sites shown in Figure 4.2 (c) with the distribution of the TA sites and total V_{step} scores in Figure 4.2 (a) and (b), they found that there is a statistically significant overlap between each set of distributions, i.e., between

(i) the experimental data and the total V_{step} scores plot, and

(ii) the experimental data and the number of TA sites plot.

Therefore, they concluded both the number of TA sites and the total V_{step} distribution can be a predictor of insertion sites; however, the total V_{step} is a better predictor due to having the smaller Akaikie coefficient value. Figure 4.2 (c) shows that the V_{step} is a better indicator, as the top three most preferred insertion regions, marked by the asterisks, have the highest V_{step} scores, while there are some regions that are rich in TA sites but less favorable for integration, like the one marked by the arrow.

Geurts et al. repeated the same steps for finding the SB insertion sites in two other host sequences, and observed similar result to the plasmid pFV/Luc. Therefore, they

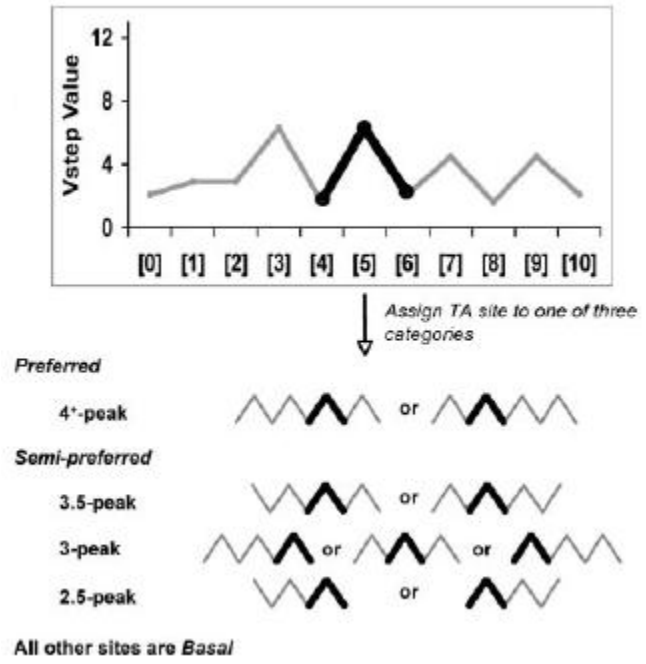

Figure 4.1: Categorizing the Vstep profile of a TA site into one of three categorizes: preferred, semi-preferred, and basal. *Figure taken from Geurts et al. [2006], used under Creative Commons license.*

concluded that there are some V_{step} patterns shared amongst the integration sites of the SB transposon. Next, Geurts et al. used the same procedure to analyze the insertion-site preferences of the piggybac and P-element transposons in the host genomes, but their study did not find any consistent V_{step} pattern.

The technique that Geurts et al. used fails in some cases, as they mentioned, due to not incorporating all the parameters that affects the transposon integration-site selection. There is also another drawback in the way of developing the preference rules based on V_{step} profiles: first, they find the preferred sites based on observations; then, they try to infer the general form of V_{step} patterns of a transposon preferred sites

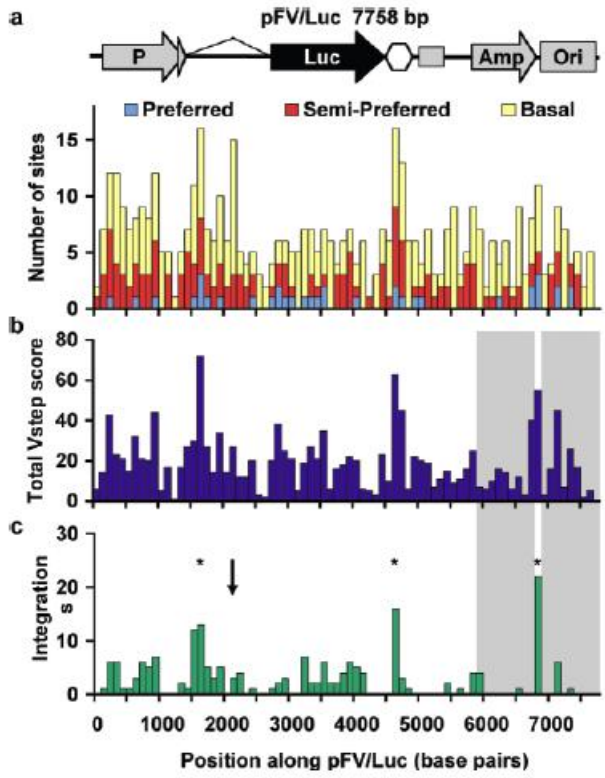

Figure 4.2: Total Vstep profile of the 7758 bp plasmid pFV/Luc. The sequence divided into 78 bins of size 100bp. **(a)** Plot of the number of each type of TA per bin. **(b)** Plot of Total Vstep score per bin. **(c)** Distribution of observed insertion sites. *Figure taken from Geurts et al. [2006], used under Creative Commons license.*

just by visually comparing the V_{step} diagrams of observed integration sites. Hence, their rule extraction method is more ad-hoc than structured. This is the weakness that led us to use machine learning methods for predicting transposon insertion sites.

Chapter 5

Prediction of Insertion Sites using Machine Learning

The problem we address in this thesis is constructing computational predictors for finding most preferred insertion sites of transposons. These predictors are based on two machine learning methods, ANN and SVM, and use V_{step} profiles in the vicinity of every TA integration site. In constructing a predictor, we considered two different purposes, and constructed a predictor specific for that purpose:

(i) Constructing a predictor (ANN- or SVM-based) for preferred individual sites: Having the V_{step} vector for a TA site, the tool predicts if the TA site is a SB transposon preferred insertion site or not; or

(ii) Constructing a predictor (ANN- or SVM-based) for preferred bins: Having the V_{step} scores of a sequence, the tool predicts the most preferred SB insertion bins in the sequence (i.e., it predicts the insertion distribution along the sequence

bins, as seen in Figure 4.2 (c)).

Similar to Geurts et al. [2006], and based on the given V_{step} values for various dinucleotides, we used the V_{step} profile of a window of 12 bp, including 5 bp flanking each side of a target TA dinucleotide, as input. We possessed a 7758 bp plasmid pFV/Luc sequence as the host genome. We also had the actual SB transposon TA integration sites and the number of hits per integration site in the host sequence, from Hackett [2011]. Therefore, our dataset consisted of all TA sites of the 7758 bp plasmid pFV/Luc sequence. In the 7758 bp plasmid pFV/Luc sequence, there were 489 TA sites: 97 positive sites (i.e., the TA sites in which integration happened with the number of hits in the range [1,9] and 193 total number of insertions) and 392 negative sites (i.e., the TA sites in which integration did not happen). It should be noted that the dataset is unbalanced, i.e., the positive and negative classes are not evenly distributed. Therefore, a baseline classifier which simply predicts the majority class (i.e., the negative class) has 0% SN, 100% SP, and 80% ACC. This baseline classifier has no predictability power, but it determines a baseline performance as a benchmark for our predictors in individual sites.

The details of the constructed ANN-based and SVM-based predictors are explained in the following sections.

5.1 ANN-based Predictors

In this section, we illustrate the architecture and the result of the constructed ANNs for finding preferred individual sites and regions, respectively.

5.1.1 Preferred Individual Sites

Data Representation

A V_{step} vector for an insertion site of 12 bp is the input data for the ANN. Since there are 11 consecutive V_{step} values for a 12 bp insertion site, a V_{step} vector has 11 elements. The output of the ANN is the insertion frequency, i.e., the number of integrations per site. We used min-max normalization technique to normalize the input and output data for training and testing. Inputs are V_{step} values of two consecutive nucleotides, which were values in the range $[1.6, 12.1]$, and outputs are integration frequencies whose values were very small and vary in the range $[0, 0.046]$. Using the normalization technique, input or output values D in the range $[D_{min}, D_{max}]$ are scaled to I in the range $[0, 1]$:

$$I = \frac{D - D_{min}}{D_{max} - D_{min}}$$

Network Architecture

We used an RBF neural network for prediction. As described in Section 2.2.1, in a 3-layer RBF network, the first layer is the input layer, the second layer has the hidden neurons with radial basis function as their activation function, and the third layer is the output layer. We constructed a set of RBF networks with different configurations and applied a 5-fold cross validation over each to find the best neural network. Finding the best RBF neural network requires searching for the optimal number of hidden neurons, as well as the parameter σ in the radial basis function and the threshold values. However, there is no specific algorithm for determining the appropriate architecture, so it is said that the design of an ANN is more of an art

than a science. We used a destructive method in design. We started with a maximal network, i.e., network with maximal number of hidden neurons and connections, and gradually deleted hidden neurons to reach the optimal network. Meanwhile, we tried to find the best σ by a random selection for each network. We also had to use a threshold or cut-off to have binary outputs. Generating a ROC curve, we could find the best threshold.

The best network has the following configuration:

- Number of input neurons = 11 (fixed by input format)
- Number of hidden neurons = 262
- Number of output neurons = 1 (fixed)
- $\sigma = 2.75$
- Cut-off = 0.1465

Using a 5-fold cross-validation, the best network revealed 78% SN, 71% SP and 72% ACC (in the case of stability, the estimated variance of the accuracy will be 0.04% based on the formula mentioned on Section 2.3.2). Figure 5.1 shows the generated ROC curve for the best network. The nearest point of ROC curve to the point (0,1) is the cut-off point. The area under the curve (AUC) is 0.71 that shows the indicator variable, i.e., the V_{step} profile, has a fairly good discriminatory power.

5.1.2 Preferred Regions

Now, we study the predictive power of the best neural network in finding the preferred regions. Similar to individual sites, we ran a 5-fold cross validation over

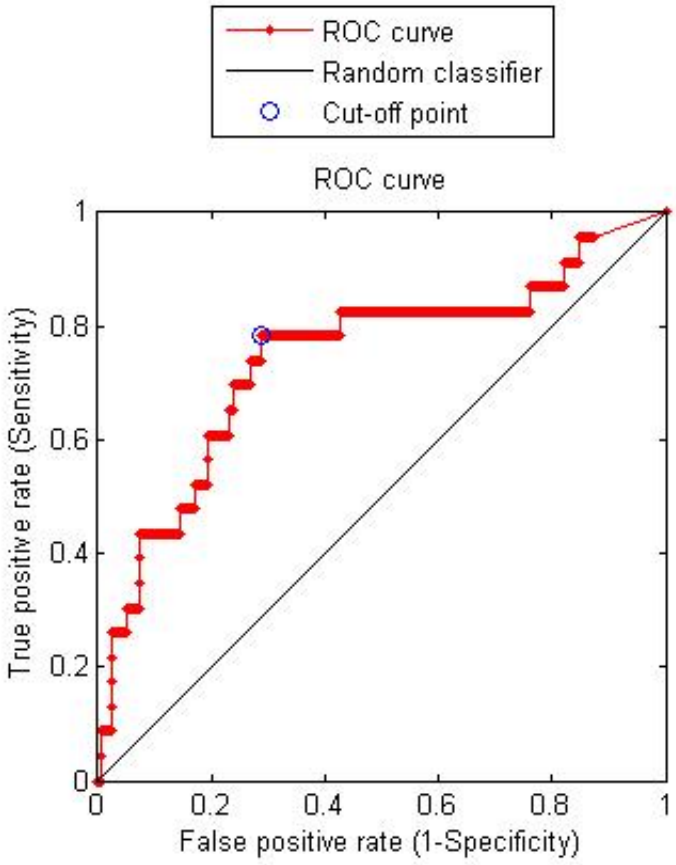

Figure 5.1: The ROC curve for the best RBF networks in individual sites. The AUC is 0.71.

all insertion sites. But this time, we did not apply any threshold on output frequencies. Instead, we computed the summation of frequencies for each bin, scaled them to the range [0,1], and obtained the distribution of predicted integration frequencies. Figure 5.2 has three plots for comparing distribution of observed and predicted insertion sites in the 7758 bp pFV/Luc sequence. The sequence is divided to 77 bins of 100 bp. Chart (a) shows the distribution of observed insertion sites. Chart (b) shows the distribution of integration frequencies predicted by the RBF network; and

Chart (c) shows both distribution of observed and predicted integration frequencies. Chart (c) illustrates an apparent overlap between these two distributions. For example, it shows that the neural network could predict the four most preferred bins (bin #16, #17, #47, and #69) successfully. Therefore, if our concern is to predict the preferred regions in the sequence, then our RBF neural network will produce better results compared to predicting individual sites. Figure 5.3 shows two generated ROC curves in two different cases. In Figure 5.3 (a), our goal was to recognize the most preferred insertion regions. We considered the most preferred insertion regions in our observed data as the bins in which the number of insertions is four or more. (i.e., bins in which the scaled insertion frequency is more than 0.4). In this case, the cut-off point has 100% SN, 97% SP and 96% ACC. In Figure 5.3 (b), our goal was to recognize the preferred insertion regions. We viewed the preferred insertion regions in our observed data as the bins in which the number of insertions is three or more (i.e., bins in which the scaled insertion frequency is more than 0.3). In this case, the cut-off point has 100% SN, 72% SP and 75% ACC. The AUC in both ROC curves is equal or greater than 0.90, which indicates an excellent discriminatory power of the network in finding preferred insertion regions.

Comparing the results in the individual sites with the results in the regions shows that the ANN has a better performance in finding the preferred insertion regions than the individual sites.

The ANNs were constructed in Open Desire, an interactive free software package for modeling and simulation, developed by Korn [2007].

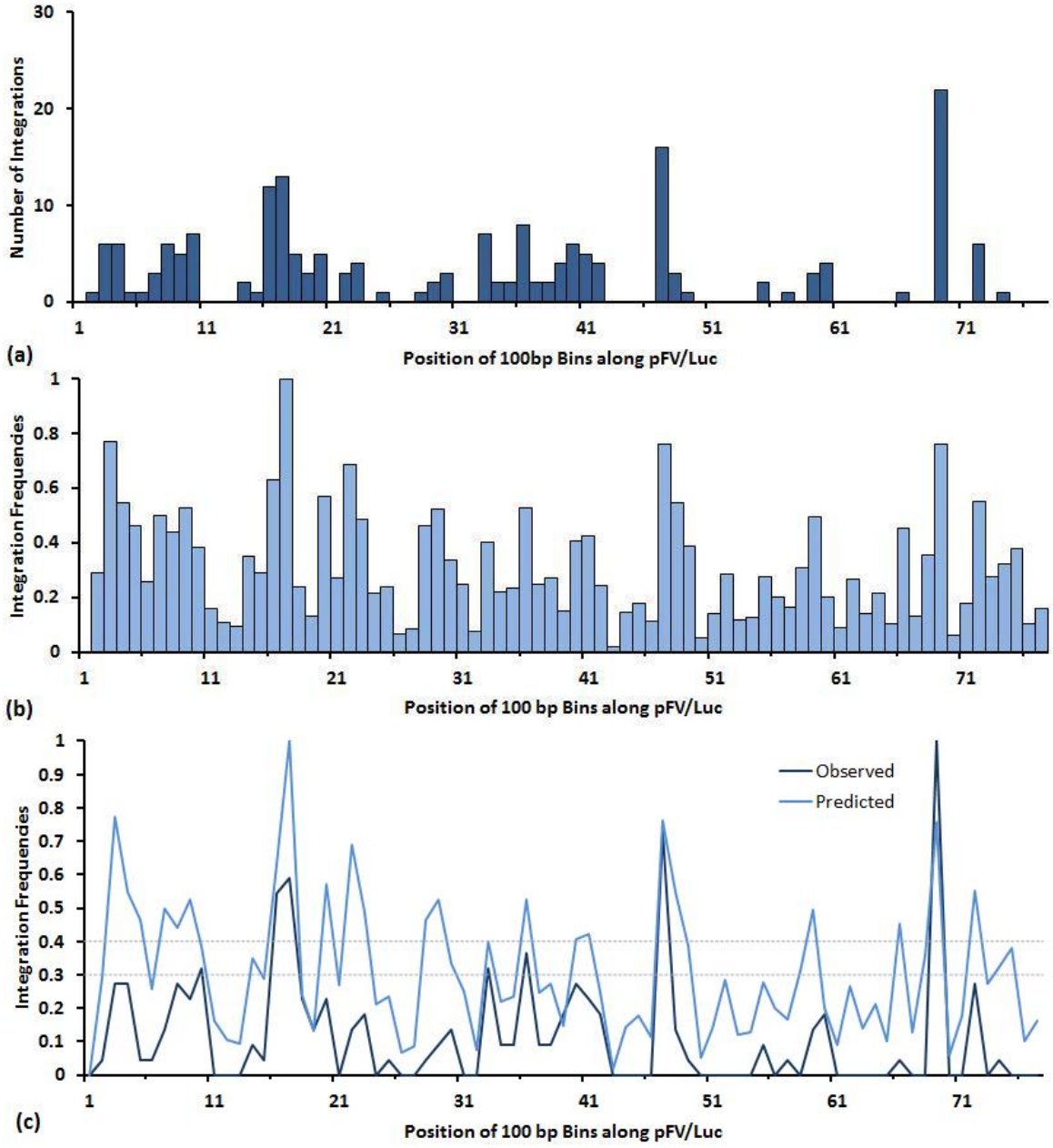

Figure 5.2: Plots of ANN-Predicted and Observed Integrations per bin in the 7758 bp plasmid pFV/Luc. The sequence is divided into 77 bins of size 100 bp. **(a)** Distribution of observed insertion sites. **(b)** Distribution of ANN-Predicted Integration Frequencies. **(c)** Distribution of observed versus Distribution of ANN-predicted insertion frequencies in bins. Dashed lines show threshold values 0.3 and 0.4, used in defining preferred and most preferred insertion bins in observed data, respectively.

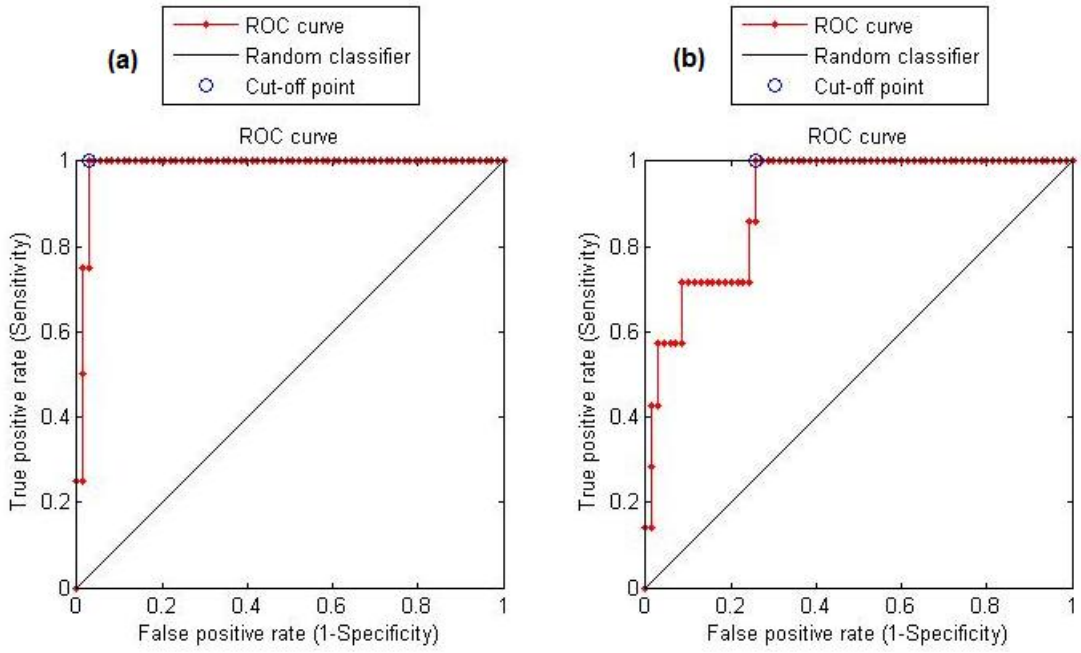

Figure 5.3: The ROC curves for the RBF neural network predictor in regions. **(a)** The ROC curve for finding most preferred bins. The AUC is 0.98. **(b)** The ROC curve for finding preferred bins. The AUC is 0.90.

5.2 SVM-based Predictors

In this section, we illustrate the architecture and the result of the constructed SVMs for finding preferred individual sites and regions, respectively.

5.2.1 Preferred Individual Sites

Data Representation

The V_{step} values are the features in the SVM. Similar to the ANN predictor, a V_{step} vector for an insertion site of 12 bp is the input data for the SVM. Since there are 11 consecutive V_{step} values for a 12 bp insertion site, a V_{step} vector has 11 elements.

The output of the SVM is the label of the class, i.e., +1 or -1 corresponding to a preferred or not-preferred insertion site. We used min-max normalization technique, mentioned in the ANN section, to normalize the input and output data for training and testing.

SVM Architecture

We used a binary SVM with an RBF kernel. In general, when the number of features is not very large, the RBF kernel is a suitable choice in SVMs, as Hsu et al. [2010] stated. As described in Section 2.2.2, an SVM with an RBF kernel has two parameters: the soft margin parameter C and the kernel parameter γ. Therefore, searching for a good (C, γ) is required for a given problem. A method for selecting parameters is a grid-search on C and γ using cross-validation. In this method, various pairs of (C, γ) are tried and the one with the best cross-validation accuracy is chosen (see Hsu et al. [2010] for a detailed explanation). Applying a grid-search on $C \in \{2^1, 2^{1.5}, ..., 2^5\}$ and $\gamma = \{2^{-3}, 2^{-2.5}, ..., 2^1\}$, along with 5-fold cross-validations, we found a (C, γ) which produced satisfiable results. In particular, when we found a set of parameters that produced an AUC greater than 0.8 for ROC curve, we stopped searching. We also set the parameter C for both positive and negative classes by different weights due to having unbalanced data. The best binary SVM has the following configuration:

- Number of features = 11 (fixed by input format)
- $\gamma = 0.25$
- $C = 2.0$

- Weight for positive class = 20
- Weight for negative class = 1

Figure 5.4 shows the ROC curve for the related SVM. The cut-off point recognizes the best SVM which has 83% SN, 72% SP and 95% ACC (in the case of stability, the estimated variance of the accuracy will be 0.01% based on the formula mentioned on Section 2.3.2). The AUC is 0.85, which indicates that the SVM has a good performance in finding preferred individual insertion sites.

5.2.2 Preferred Regions

For predicting the most preferred regions, we took advantage of support vector regression. First, we constructed an epsilon-SVR with the RBF kernel and the same parameters we had found in the binary SVM, to model the relationship between the V_{step} vectors and integration frequencies of insertion sites (both normalized). We also set the tolerance criterion, ϵ, to 0.001. Next, we ran a 5-fold cross validation over all insertion sites, and obtained the predicted frequency for each insertion site. Then, we computed the summation of frequencies for each bin, scaled them to the range [0,1], and obtained the distribution of predicted integration frequencies. Figure 5.5 has three plots for comparing distribution of observed and predicted insertion sites in the 7758 bp pFV/Luc sequence. The sequence is divided to 77 bins of 100 bp. Chart (a) shows the distribution of observed insertion sites. Chart (b) shows the distribution of integration frequencies predicted by the epsilon-SVR; and Chart (c) shows both distribution of observed and predicted integration frequencies. Chart (c) illustrates an apparent overlap between these two distributions. For example, it shows

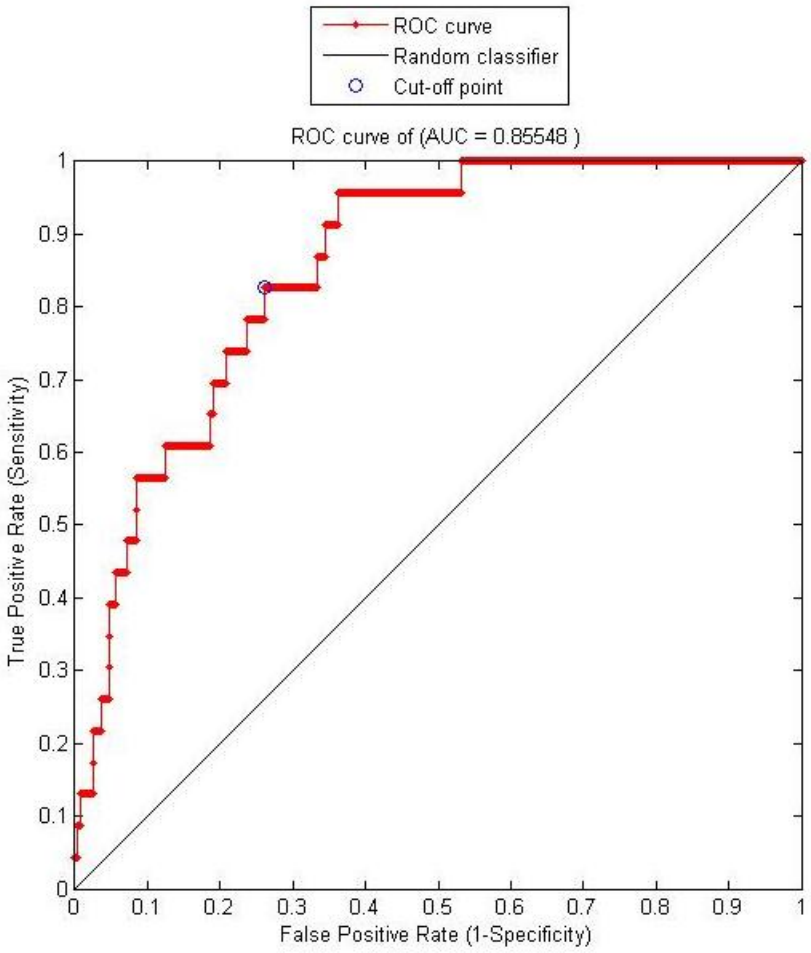

Figure 5.4: The ROC curve for the best Gaussian kernel SVM in individual sites. The AUC is 0.85.

that the SVM could predict the four most preferred bins (bin #16, #17, #47, and #69) successfully. Therefore, if our concern is to predict the preferred regions in the sequence, then the epsilon-SVR will produce better results compared to the binary SVM for individual sites. Figure 5.6 shows two generated ROC curves in two different cases. In Figure 5.6 (a), our goal was to recognize the most preferred insertion regions (e.g., bins in which the scaled insertion frequency is more than 0.4). In this case, the

cut-off point has 100% SN, 94% SP and 94% ACC. Also, the AUC is equal to 0.97. In Figure 5.6 (b), our goal was to recognize the preferred insertion regions (e.g., bins in which the scaled insertion frequency is more than 0.3). In this case, the cut-off point has 85% SN, 90% SP and 89% ACC. The AUC in this ROC curve is equal to 0.89. Both AUC values indicate an excellent discriminatory power of the predictor in finding preferred insertion regions.

Comparing the results in the individual sites with the results in the regions shows that the SVM has a better performance in finding the preferred insertion regions to the individual sites.

The SVMs were constructed in MATLAB environment using LIBSVM, a Library for Support Vector Machines, developed by Chang and Lin [2011].

5.3 Comparing Results

In this section, we intend to compare

- the result of the ANN and SVM predictor to each other; and
- the result of our machine learning methods to Geurts et al. 's rule-based method.

ANN versus SVM

We have summarized the performance measures of the ANN- and SVM-based predictors for finding preferred individuals sites and regions in two different tables. Table 5.1 contains the 5-fold cross-validation result of the predictors in individual sits. According to these results, the SVM-based predictor outperforms the ANN-based

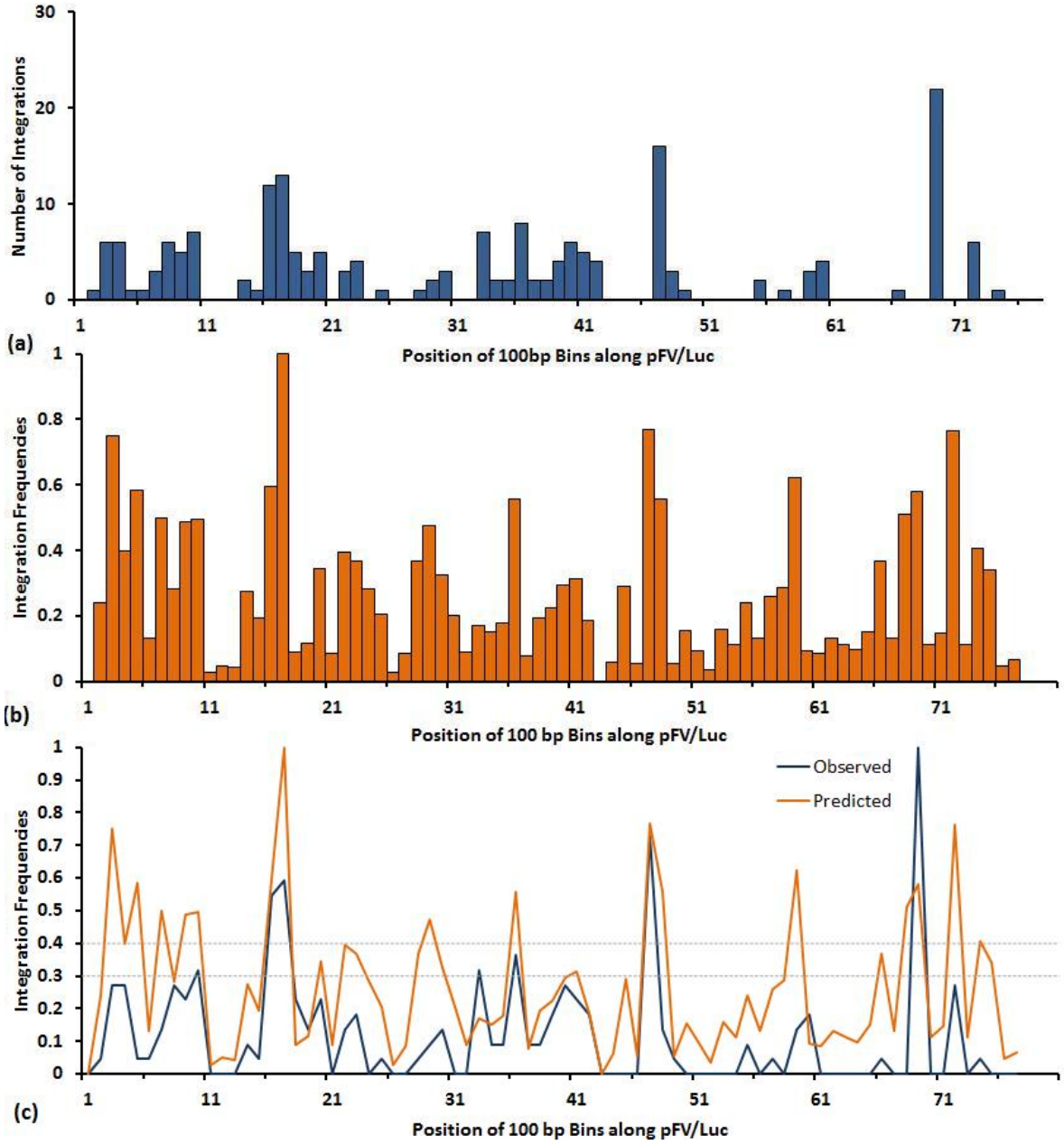

Figure 5.5: Plots of SVM-Predicted and Observed Integrations per bin in the 7758 bp plasmid pFV/Luc. The sequence is divided into 77 bins of size 100 bp. **(a)** Distribution of observed insertion sites. **(b)** Distribution of SVM-Predicted Integration Frequencies. **(c)** Distribution of observed versus Distribution of SVM-predicted insertion frequencies in bins. Dashed lines show threshold values 0.3 and 0.4, used in defining preferred and most preferred insertion bins in observed data, respectively.

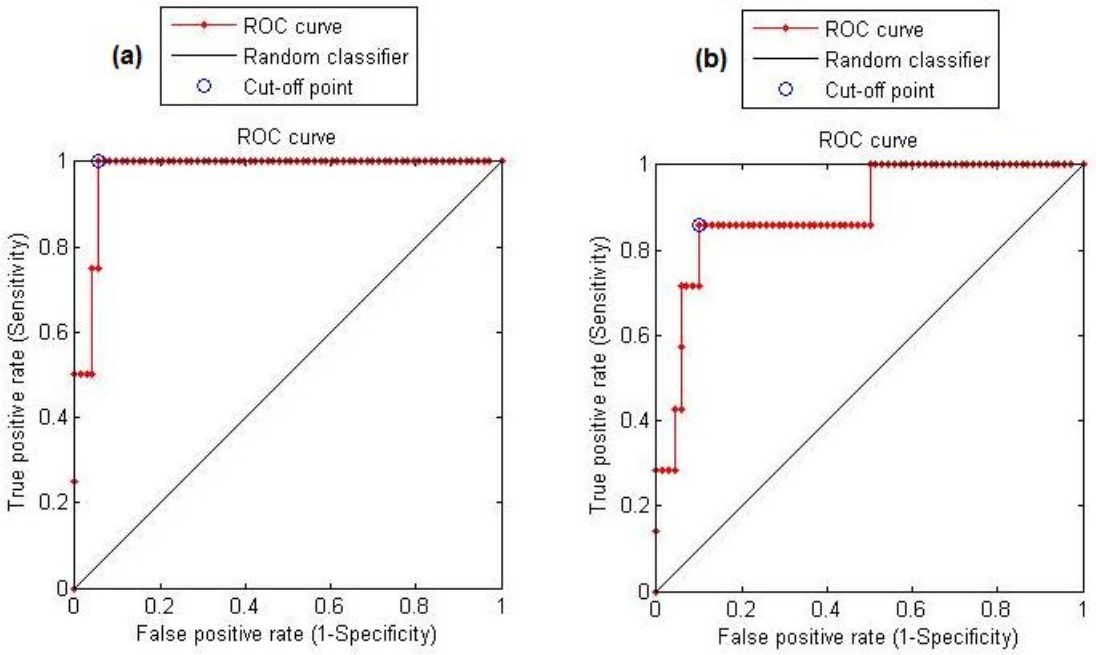

Figure 5.6: The ROC curves for the SVM predictor in regions. **(a)** The ROC curve for finding preferred bins. The AUC is 0.97. **(b)** The ROC curve for finding preferred bins. The AUC is 0.89.

predictor in identifying preferred individual insertion sites due to having higher AUC, SN and SP. Table 5.2 contains the 5-fold cross-validation result of the predictors in identifying preferred 100 bp insertion regions. For this case, we have also considered the cross-validation correlation coefficient between predicted and observed results (r), in addition to SN, SP, and AUC measures, to help us make better decisions. In general, the ANN produces more accurate insertion frequencies in regions than the SVM, as it has a higher r. However, both predictors are excellent in recognizing most preferred insertion regions (similar values for AUC, SN, and SP), and the SVM performs better in identifying preferred regions (similar AUCs, but higher ACC).

Table 5.1: The ANN versus SVM predictor in identifying preferred individual sites.

Predictor	SN(%)	SP(%)	ACC(%)	AUC
ANN	78	71	72	0.71
SVM	83	72	95	0.85

Table 5.2: The ANN versus SVM predictor in identifying preferred 100 bp regions.

Predictor	r	Prediction	SN(%)	SP(%)	ACC(%)	AUC
ANN	0.56	most preferred	100	97	96	0.98
		preferred	100	72	75	0.90
SVM	0.45	most preferred	100	94	94	0.97
		preferred	85	90	89	0.89

Our methods versus Geurts et al. 's method

Geurts et al. developed some rules for describing the insertion-site preferences of the SB transposon. They did not report any SN, SP or ACC, neither for individual sites nor for regions, as they had not made any tool (predictor) based on their rules. However, Geurts et al. demonstrated the successfulness of their rules (for SB transposon) in finding preferred pFV/Luc insertion bins, as explained in Section 4.2.2 and Figure 4.2. We considered the chart in Figure 4.2 (b) as the only resource to compare our methods with Geurts et al. 's, and tried to measure its classification performance. Consequently, we generated two ROC curves displayed in Figure 5.7, based on the total V_{step} scores in 100 bp bins. ROC (a) shows the performance of the rules in finding most preferred bins and ROC (b) shows the performance in finding preferred bins. The cut off point for ROC (a) has 100% SN and 89% SP, add for ROC (b) 85% SN and 91% SP. Based on these results, we can say that

1. Our predictors have similar SN, but better SP in recognizing most preferred

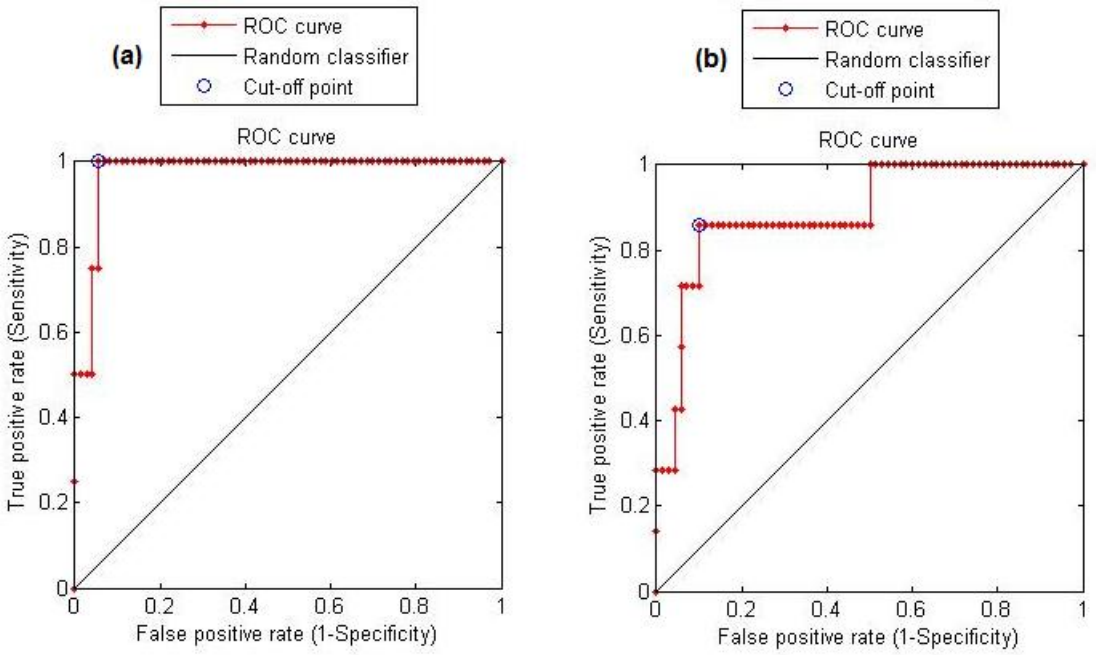

Figure 5.7: The generated ROC curves for Geurts et al. 's method in regions. **(a)** The ROC curve for finding most preferred bins. The AUC is 0.97. **(b)** The ROC curve for finding preferred bins. The AUC is 0.91.

regions, compared to Geurts et al. 's rules; and

2. The SVM predictor performs as good as Geurts et al. 's rules in identifying preferred regions.

Chapter 6

Conclusion

The main goal of this thesis was constructing computational predictors for identifying preferred insertion sites of transposons. Such prediction problems can be considered to be similar to some of fundamental and challenging bioinformatics problems like protein secondary structure prediction problem, or any other problems reviewed in Chapter 3. Therefore, similar to the applied solution methodologies in these problems, we developed insertion site predictors based on two machine learning methods, ANN and SVM. In particular, we developed RBF neural network and RBF kernel SVM predictors. As our available datasets were related to the SB transposon, we designed our predictors specific to the SB. Based on the achievements of Liu et al. [2005] and Geurts et al. [2006] explained in Chapter 4, we extracted local DNA sequences around the target sites (TA sites for the SB) and used their V_{step} profiles as the classification features of insertion sites. Then, we used the predictors for finding preferred individual sites (12 bp sequences) and preferred regions (100 bp sequences). Finally, we evaluated our predictors with the SB insertion data in the 7758 bp plasmid

pFV/Luc. According to the 5-fold cross validation results, summarized in Table 5.1 and Table 5.2, we conclude that

1. The SVM predictor outperforms the ANN predictor in identifying preferred individual insertion sites, having 83% SN, 72% SP, and the value 0.85 for AUC.

2. Both ANN and SVM predictors have excellent performance in recognizing the most preferred insertion regions. The ANN predictor revealed 100% SN, 97% SP, and the value 0.98 for AUC; and the SVM predictor showed 100% SN, 94% SP, and the value 0.97 for AUC.

3. The SVM predictor outperforms the ANN predictor in identifying preferred insertion regions, having 85% SN, 90% SP, and the value 0.89 for AUC.

4. Both ANN and SVM predictors revealed better results in regions than individual sites. An explanation for this fact might be that the amount of preferability of an insertion site not only depends on the local sequence itself, but also depends on the region that encompasses the insertion site. Therefore, it is worth considering larger sequences of insertion sites than 12 bp as inputs for the ANN and SVM predictors, or adding some region-related features to the current models in future.

We also compared the performance of our predictors with the Geurts et al.'s V_{step} rules in Section 5.3, and saw that the SVM predictor is as successful as the Geurts et al.'s V_{step} rules for the SB transposon. However, the main preference of the ANN and SVM-based solutions over the Geurts et al.'s rule-based method is that ANN and SVM are able to extract the rules, or the relations between inputs and

outputs, themselves. That's why the Geurts et al.'s ad-hoc rules were not successful in identifying preferred insertion sites of the other transposons, except the SB. We possessed an SB integration dataset (i.e., the integration information of SB transposon in the pFV/Luc host genome) for evaluating of our SB-specific predictors. With datasets from other transposons such as piggyBac and Drosophila P-element, we could train other transposon-specific predictors, and compare more accurately the ANN and SVM predictors to the Geurts et al.'s rules, in particular. Constructing predictors specific to the other transposons can be done as a future work.

Transposon-specific insertion-site predictors can be helpful in all the applications in which a transposon is used. Transposons are used in gene transfer technologies to transfer genes of interest to animals and plants. They have applications in discovery of new genes and their functions (especially those that cause cancer). Also they have applications in therapy of genetic disorders in humans. The information from a predictor such as the constructed SB predictors in this thesis can help direct experiments by helping researchers focus on potential sites of high likelihood of insertion before beginning experiments. For example, a lab scientist may feed a transposon-specific predictor with a sequence, and receive the distribution of predicted insertion frequencies of the transposon along the sequence. Therefore, she/he can recognize the most probable insertion regions (e.g., regions of 100 bp in length). Furthermore, she/he can feed the predictor with the sequence of an individual insertion site (e.g., sites of 12 bp in length) and check whether it is a preferred insertion sites or not.

Bibliography

L. Aguado. DNA. `http://www.clker.com/cliparts/B/o/c/f/P/7/dna-hi.png`. [Online; accessed 15-December-2012].

S. Ahmad and A. Sarai. PSSM-based prediction of DNA binding sites in protein. *BMC Bioinformatics*, 6(33), 2005.

E. L. Aronovich, R. S. McIvor, and P. B. Hackett. The sleeping beauty transposon system: a non-viral vector for gene therapy. *Human Molecular Genetics*, 20(R1): R14–20, 2011.

P. Baldi and S. Brunak. *Bioinformatics: The machine learning approach*. MIT Press, Cambridge, MA, USA, 2nd edition, 2001.

V. P. Belancio, P. L. Deininger, and A. M. Roy-Engel. LINE dancing in the human genome: transposable elements and disease. *Genome Medicine*, 1(10):97, 2009.

T. L. Bergemann, T. K. Starr, H. Yu, M. Steinbach, J. Erdmann, Y. Chen, R. T. Cormier, D. A. Largaespada, and K. A. Silverstein. New methods for finding common insertion sites and co-occurring common insertion sites in transposon- and virus-based genetic screens. *Nucleic Acids Research*, 40(9):3822–33, 2012.

C. Berry, S. Hannenhalli, J. Leipzig, and F. D. Bushman. Selection of target sites for mobile DNA integration in the human genome. *PLoS Computational Biology*, 2(11):e157, 2006.

Broad Institute of MIT and Harvard. Transposon. `http://www.broadinstitute.org/education/glossary/transposable-elements`, 2012. [Online; accessed 15-January-2012].

C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

J. Cheng, A. Randall, M. Sweredoski, and P. Baldi. SCRATCH: a Protein Structure and Structural Feature Prediction Server. *Nucleic Acids Research*, 33(web server issue):w72–76, 2005.

C. Cole, J. D. Barber, and G. J. Barton. The Jpred 3 secondary structure prediction server. *Nucleic Acids Research*, 36(suppl. 2):w197–w201, 2008.

J. A. Cuff and G. J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, 34(4):508–19, 1999.

J. de Ridder, A. Uren, J. Kool, M. Reinders, and L. Wessels. Detecting statistically significant common insertion sites in retroviral insertional mutagenesis screens. *PLoS Computational Biology*, 2(12):e166, 2006.

M. J. de Smith. STATSREF: Statistical analysis handbook - a web-based statistics

resource. `http://www.statsref.com/HTML/kernel_density_estimation.html`, 2011.

M. Di Matteo, E. Belay, M. K. Chuah, and T. Vandendriessche. Recent developments in transposon-mediated gene therapy. *Expert Opinion on Biological Therapy*, 12 (7):841–58, 2012.

S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.

A. M. Geurts, C. S. Hackett, J. B. Bell, T. L. Bergemann, L. S. Collier, C. M. Carlson, D. A. Largaespada, and P. B. Hackett. Structure-based prediction of insertion-site preferences of transposons into chromosomes. *Nucleic Acid Research*, 34(9):2803–2811, 2006.

C. S. Hackett, A. M. Geurts, and P. B. Hackett. Predicting preferential DNA vector insertion sites: implications for functional genomics and gene therapy. *Genome Biology*, 8(S12):Suppl 1, 2007.

P. B. Hackett. SB transposon insertion data in the 7758 bp plasmid pFV/Luc, 2011. [Personal Communication].

S. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall., 2009.

C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`, 2010.

S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *Journal of Molecular Biology*, 3(1):397–407, 2001.

J. P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

Z. Ivics and Z. Izsvak. The expanding universe of transposon technologies for gene and cell engineering. *Mobile DNA*, 1(25), 2010.

D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, 1999.

H. Kim and H. Park. Protein secondary structure prediction based on an improved support vector machines approach. *Protein Engineering*, 16(8):553–60, 2003.

B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research*, 31(13):3423–8, 2003.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conferences on Artificial Intelligence, IJCAI*, pages 1137–1143, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995.

G. A. Korn. *Advanced Dynamic-System Simulation: Model-Replication Techniques and Monte Carlo Simulation*. John Wiley & Sons, Inc., 2007.

D. A. Largaespada and L. S. Collier. Transposon-mediated mutagenesis in somatic cells: Identification of transposon-genomic DNA junctions. *Methods in Molecular Biology*, 435:95–108, 2008.

M. Lee. RNA. `http://www.clker.com/cliparts/U/g/M/3/X/d/rna-hi.png`. [Online; accessed 15-December-2012].

L. Li, W. Jianga, X. Lia, K. L. Moserd, Z. Guoa, L. Dua, Q. Wange, E. J. Topolf, Q. Wangf, and S. Raoa. A robust hybrid between genetic algorithm and support vector machine for extracting an optimal feature gene subset. *Genomics*, 85(1): 16–23, 2005.

G. Liu, A. M. Geurts, K. Yae, A. R. Srinivasan, S. C. Fahrenkrug, D. A. Largaespada, J. Takeda, K. Horie, W. K. Olson, and P. B. Hackett. Target-site preferences of sleeping beauty transposons. *Journal of Molecular Biology*, 346(1):161–173, 2005.

X. Ma, J.-S. Wu, H.-D. Liu, X.-N. Yang, J.-M. Xie, and X. Sun. SVM-based approach for predicting DNA-binding residues in proteins from amino acid sequences. In *International Joint Conferences on Bioinformatics, Systems Biology and Intelligent Computing, IJCBS*, pages 225–229, 2009.

W. H. Majoros, M. Pertea, C. Antonescu, and S. L. Salzberg. GlimmerM, Exonomy and Unveil: three ab initio eukaryotic genefinders. *Nucleic Acids Research*, 31(13): 36013604, 2003.

H. Nielsen, J. Engelbrecht, S. Brunak, and G. von Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10(1):1–6, 1997.

W. S. Noble. What is a support vector machine? *Nature Biotechnology*, 24(12): 1565–1567, 2006.

C. Notredame and D. G. Higgins. SAGA: Sequence Alignment by Genetic Algorithm. *Nucleic Acids Research*, 24(8):1515–1524, 1996.

H. Ogura, H. Agatab, M. Xiea, T. Odakaa, and H. Furutanic. A study of learning splice sites of DNA sequence by neural networks. *Computers in Biology and Medicine*, 27(1):67–75, 1997.

W. K. Olson, A. A. Gorin, X.-J. Lu, L. M. Hock, and V. B. Zhurkin. DNA sequence-dependent deformability deduced from proteinDNA crystal complexes. *Proceedings of the National Academy of Sciences of the United States of America: PNAS*, 95 (19):1116368, 1998.

G. Pollastri, D. Przybylski, B. Rost, and P. Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins*, 47(2):228–235, 2002.

S. K. Riis and A. Krogh. Improved prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of Computational Biology*, 3(1):163–183, 1996.

B. Rost and C. Sander. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proceedings of the National Academy of Sciences of the United States of America: PNAS*, 90:7558–7562, 1993.

A. J. Smola and B. Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Ratsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(S10):1–16, 2007.

A. C. Spradling, H. J. Bellen, and R. A. Hoskins. Drosophila P elements preferentially transpose to replication origins. *Proceedings of the National Academy of Sciences of the United States of America: PNAS*, 108(38):15948–15953, 2011.

U.S. National Library of Medicine. DNA. `http://ghr.nlm.nih.gov/handbook/basics/dna`, 2012. [Online; accessed 15-January-2012].

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

J.-P. Vert. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, pages 649–660, 2002.

D. E. Volk. Ribbon diagram of the protein. `http://en.citizendium.org/wiki/File:ProteinRibbonByDEVolk.jpg`, 2007. [Online; accessed 15-December-2012].

E. W. Weisstein. Convolution. `http://mathworld.wolfram.com/Convolution.html`, 2012.

A. Zemla, C. Venclovas, K. Fidelis, and B. Rost. A modified definition of SOV, a segment-based measure for protein secondary structure prediction assessment. *PROTEINS: Structure, Function, and Genetics*, 34:220–223, 1999.