NBS-LRRpred: NBS-LRR GENE PREDICTION ALGORITHM

By

UPENDRA CHAUDHARY-031504

VIKAS RAJPUT-031517

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF
BIOTECHNOLOGY & BIOINFORMATICS
JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY-WAKNAGHAT

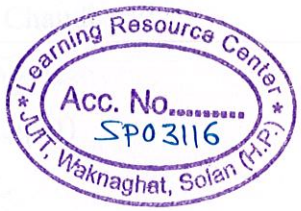
MAY-2007

NBS-LRRpred: NBS-LRR GENE PREDICTION ALGORITHM

By

UPENDRA CHAUDHARY-031504

VIKAS RAJPUT-031517



MAY-2007

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF

BIOTECHNOLOGY & BIOINFORMATICS

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, "**NBS-LRRpred: NBS-LRR gene prediction algorithm**" submitted by Upendra chaudhary and Vikas Rajput in partial fulfillment for the award of degree of Bachelor of Technology in Bioinformatics of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

(Upendra Chaudhary)
Upendra Chaudhary

(Vikas Rajput)
VRajput

24/05/07

Dr Rajinder Singh Chauhan
(Project Supervisor and Head of Department)
Department of Bioinformatics
Jaypee University of Iinformation Technology Waknaghat

ACKNOWLEDGMENT

Many people have contributed to this project in a variety of ways over the past few months. To the individuals who have helped us, we again express our appreciation. We also acknowledge the many helpful comments Received from our teachers of the bioinformatics department. We are indebted to all those who provided reviews & suggestions for improving the results and the topics covered in our project, and we extend our apologies to anyone we may have failed to mention

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN: Artificial Neural Network.

NBS: Nucleotide binding site.

LRR: Leucine-rich repeat.

TIR: Toll/Interleukin-1 receptor.

HR: Hypersensitive response.

R: Resistance.

CC: Coiled-coil.

ARC: Apoptosis, *R* gene products and CED-4

.

.

.

ABSTRACT

In this study a systematic attempt has been develops to integrate various pre-existing approaches in order to predict NBS-LRR gene with high accuracy. This approach is a first kind of approach to predict a NBS-LRR gene because there is not any tool at present which can predict NBS-LRR gene directly. Datasets used for training and testing purpose consist both nucleotide and protein sequences. In case of nucleotide dataset we took 173 NBS-LRR and 173 non -NBS-LRR and in case of protein datasets we took 97 NBS-LRR and 97 non-NBS-LRR obtained from NCBI entrez protein and nucleotide database (http://ncbi.nlm.nih.gov/enterz). In this approach we use artificial neural network using amino acid characteristics, dipeptide and tripeptide composition of nucleotide sequences and achieve a accuracy of 87 %,78.85% and 85.75% , respectively and hybrid result of this training tells us that given gene is NBS or not. Thereafter we train TIR (Toll/Interleukin-1 receptor) domain present in NBS to classify between TIR and non-TIR containing sequences. The performance of this algorithm has been evaluated by performing BLAST (http://ncbi.nlm.nih.gov/blast) search against each predicted gene.

CHAPTER 1

INTRODUCTION

Leucine-rich repeats (LRRs), a nucleotide-binding site (NBS), and a putative amino-terminal signaling domain are present in most disease resistance proteins. They are called NBS-LRR proteins. Wide variety of NBS-LRR proteins from different organism serve as platforms for interaction, and as regulatory modules of protein activation. The LRRs of plant R proteins are determinants of response specificity, and their action can lead to plant cell death in the form of the familiar hypersensitive response (HR). A total of 149 resistant genes are reported in Arabidopsis thaliana genome and plant cell deal with the task of assembling many of the proteins encoded by NBS-LRR signaling complexes.

First, proteins will interact with their proper partners because they restricted to their sub-cellular site of function. Second, these interactions are architecturally organized to avoid inappropriate signaling events and to maintain the fidelity and efficiency of the response when it is initiated. Recent results provide new insights into how the signaling potential of R proteins might be created, managed and held in check until specific stimulation following infection. Nevertheless, the roles of the R protein partners in these regulatory events that have been defined to date are unclear. Disease resistance genes (R genes) are crucial components of the hypersensitive response (HR), a plant defense mechanism that results in localized cell death. The HR is triggered when pathogen molecules, possibly virulence factors, are detected by plant receptors; genetic analysis of the HR has lead to the cloning of R genes, many of which encode receptor-like proteins. Based on their predicted domain structure, R proteins encoded by the R genes have been classified into four groups: intracellular kinases, extra cellular receptors, extra cellular receptors coupled to kinases, and intracellular receptors.

Most characterized R genes encode putative intracellular receptors, which contain either a coiled-coil (CC) or a Toll/Interleukin-1 receptor (TIR) domain at their N-terminal end, followed by a nucleotide binding site (NBS). At the C-terminal end, these proteins consist of a series of leucine rich repeats (LRRs). The functions of the

CC, TIR, and NBS domains are not known fully, but all similar proteins identified in animal systems play roles in protein-protein interactions and signal transduction. The function of LRR domains is clearer because recent data suggest that LRRs in R proteins mediate direct or indirect interaction with pathogen molecules. The tertiary structure of LRRs has been experimentally determined for a diverse group of proteins, most notably porcine ribonuclease inhibitor. Generally, individual LRRs form repeats of β-strand-loop and α-helix-loop units with nonleucine residues exposed and compose a binding surface predicted involved in protein recognition. In R proteins, putatively solvent-exposed residues in β-sheets may interact with pathogen ligands and hence determine specificity for pathogen elicitors.

The candidate recognition domain of NBS–LRR proteins is the C-terminal LRR. It is the most variable region in closely related NBS–LRR proteins and is under selection to diverge. Functional analysis of recombinant R proteins also indicates that recognition specificity resides in the LRR. However, there is indirect evidence that the LRR may contribute to signalling as well as recognition.

Most *R* genes are predicted to encode intracellular proteins with nucleotide-binding site and leucine-rich repeat (NBS–LRR) domains. The NBS of these proteins has an N-terminal sub-domain that contains consensus kinase 1a (P-loop), kinase 2 and kinase 3a motifs common to a large variety of nucleotide-binding proteins; we refer to this as the NB subdomain. The C-terminal part of the NBS, referred to as the ARC (apoptosis, *R* gene products and CED-4) subdomain, is conserved in plant NBS–LRR proteins as well as in several NBS-containing proteins involved in animal innate immunity and apoptosis. In certain animal proteins, the NBS domain mediates oligomerization that ultimately results in activation of the N-terminal signalling domains.

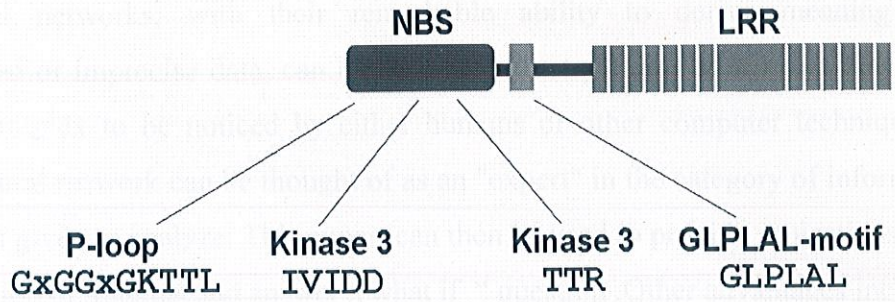


Fig.1.
Structure of NBS-LRR (Nucleotide Binding Site- Luecine Rich Repeat) plant disease resistance genes (modified from Hammond-Kosack ,1997). NBS conserved motifs are shown with amino acid consensus sequences.

MACHINE LEARNING TECHNIQUES

NEURAL NETWORKS:

What is a Neural Network?

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.

Why use neural networks?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situation of interest and answer " what if " question .Other advantages include:

1. 1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Basic Working:

As explained earlier, a neuron is basically a cell which accumulates electrical signals with different strengths. What it does more is that it compares the accumulated signal with one predefined value unique to every neuron. This value is

called bias. Well, it can be explained with the help of following image.

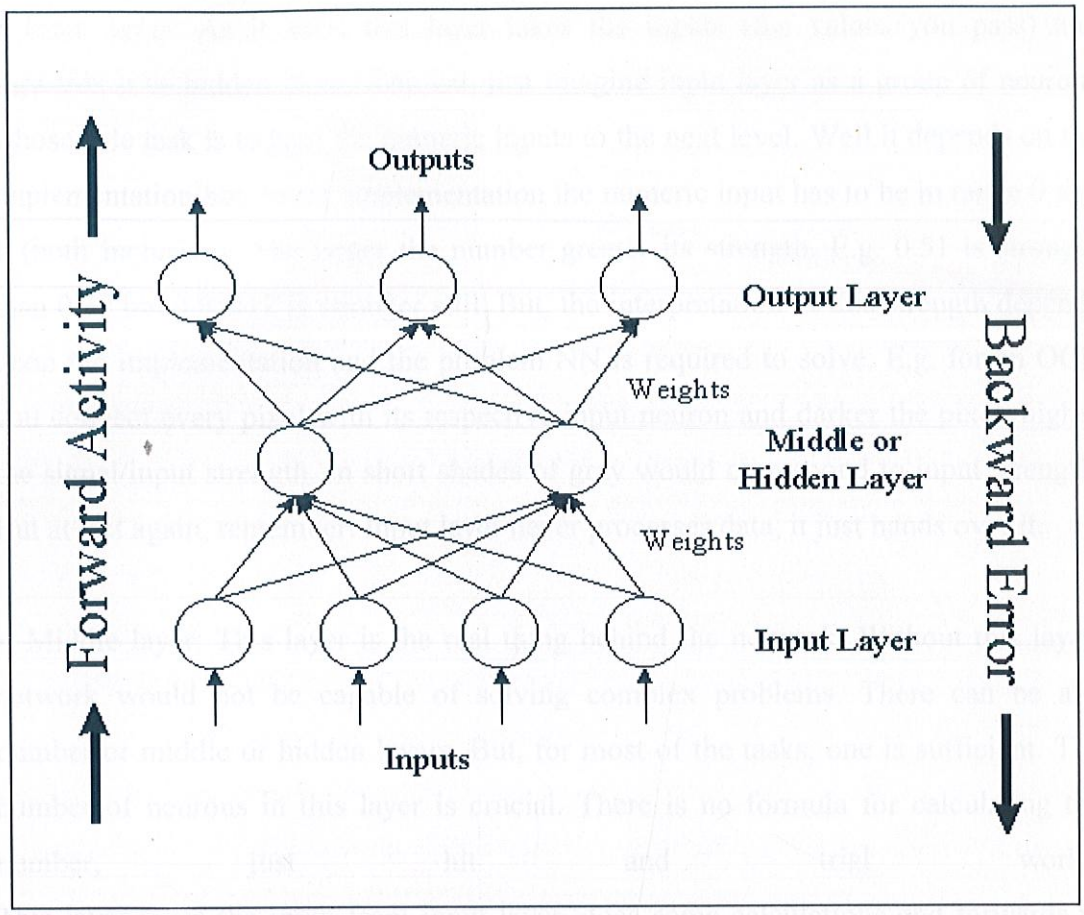


FIGURE 2. Typical Artificial Neural Network Setup (Caudill and Butler, 1992a).

The circles in the image represent neurons. This network or more appropriately this network topology is called feed-forward multi layered neural network. It is the most basic and most widely used network.
The network is called multi layered because it consists of more than two layers. The neurons are arranged in a number of layers, generally three. They are input, hidden/middle and output layers. The names signify the function of the layer. This network is feed-forward, means the values are propagated in one direction only. There are many other topologies in which values can be looped or move in both forward and backward direction. But, this network allows the movement of values only from input layer to output layer. The functions of various layers are explained below:

- Input layer: As it says, this layer takes the inputs (the values you pass) and forwards it to hidden layer. You can just imagine input layer as a group of neurons whose sole task is to pass the numeric inputs to the next level. Well it depends on the implementation but, in my implementation the numeric input has to be in range 0 and 1 (both inclusive). The larger the number greater its strength. E.g. 0.51 is stronger than 0.39 but 0.93412 is stronger still. But, the interpretation of this strength depends upon the implementation and the problem NN is required to solve. E.g. for an OCR you connect every pixel with its respective input neuron and darker the pixel, higher the signal/input strength. In short shades of gray would correspond to input strength. But at last again, remember: Input layer never processes data, it just hands over it.

- Middle layer: This layer is the real thing behind the network. Without this layer, network would not be capable of solving complex problems. There can be any number or middle or hidden layers. But, for most of the tasks, one is sufficient. The number of neurons in this layer is crucial. There is no formula for calculating the number, just hit and trial works.
This layer takes the input from input layer, does some calculations and forwards to the next layer, in most cases it is the output layer.

- Output layer: This layer consists of neurons which output the result to you. This layer takes the value from the previous layer, does calculations and gives the final result. Basically, this layer is just like hidden layer but instead of passing values to the next layer, the values are treated as output.

- Dendrites: No! it is not some creature from X-Files, but a name given to straight lines joining two neurons of consecutive layers, which you can see in the image. They are just a passage (or method) through which values are passed from one layer to the next. There is a value attached with dendrite called weight. The weight associated with dendrites basically determines the importance of incoming value. A weight with

larger value determines that the value from that particular neuron is of higher significance. To achieve this we do is multiply the incoming value with weight. So no matter how high the value is, if the weight is low the multiplication yields the final low value.

Training:

Training is the most important part of a neural network and the one consisting of the most mathematics. We'd used Back propagation method for training the NN. Here is the basic idea how it is done .The best example illustrating this principle is Charles Darwin (what?). Yes, at the time when he wrote '*On the Origin of Species*', DNA was not known. So, he propounded the evolution without even knowing the method of how it is done i.e. how traits are passed on from parents to offspring.

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation.

Most of the algorithms used in training artificial neural networks are employing some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods simulated annealing, and Expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks.

This training procedure must be repeated for larger number of samples so that our NN can produce accurate results for untrained input samples.

CHAPTER 2

MATERIALS AND METHODS:

Modules and layers

In this study along with the machine learning techniques, we used one kind of protein sequence encoding methods i.e. "pepstats", and another kind of nucleotide encoding method used for calculating dinucleotide and trinucleotide properties i.e. "compseq". Using a combination of these methods, and the techniques three modules were created namely; ANN*pepstats*, ANN*dinucleotide*, ANNtrinucleotide. The nomenclature of these modules is according to the rule that first word in the name indicates the machine learning technique say ANN and the second subscripted word signifies the encoding method; "pepstats", "dinucleotide" or "trinucleotide encoding". In first layer of NBS-LRRpred tool is GENEID which gives a total number of genes present in the input sequence. The modules associated with "pepstats", "dinucleotide", "trinucleotide" encoding method constitute the second layer of NBS-LRRpred tool. The "pepstats" and" compseq" is an application from EMBOSS suite. "Pepstats" calculates physicochemical properties of a protein sequence which used in "ANNpepstats" and "compseq" calculates properties of nucleotide sequences which is used in "dinucleotide" and "trinucleotide encoding" as a whole and thus it is used to predicts the nature of a nucleotide sequence whether it's a NBS-LRR or non- NBS-LRR. All the sequences which pass through this layer come to next layer which predict on basis of conserved motifs present in NBS and LRR domains We trained four conserved motifs of NBS through ANN (P-Loop, Kinase-2, GLPL motif) obtained from BLOCKS database. The resulting NBS-LRR sequence which comes through this layer is predicted as NBS-LRR. In third layer we trained ANN for classification of predicted NBS-LRR sequences in to TIR and non-TIR classes.

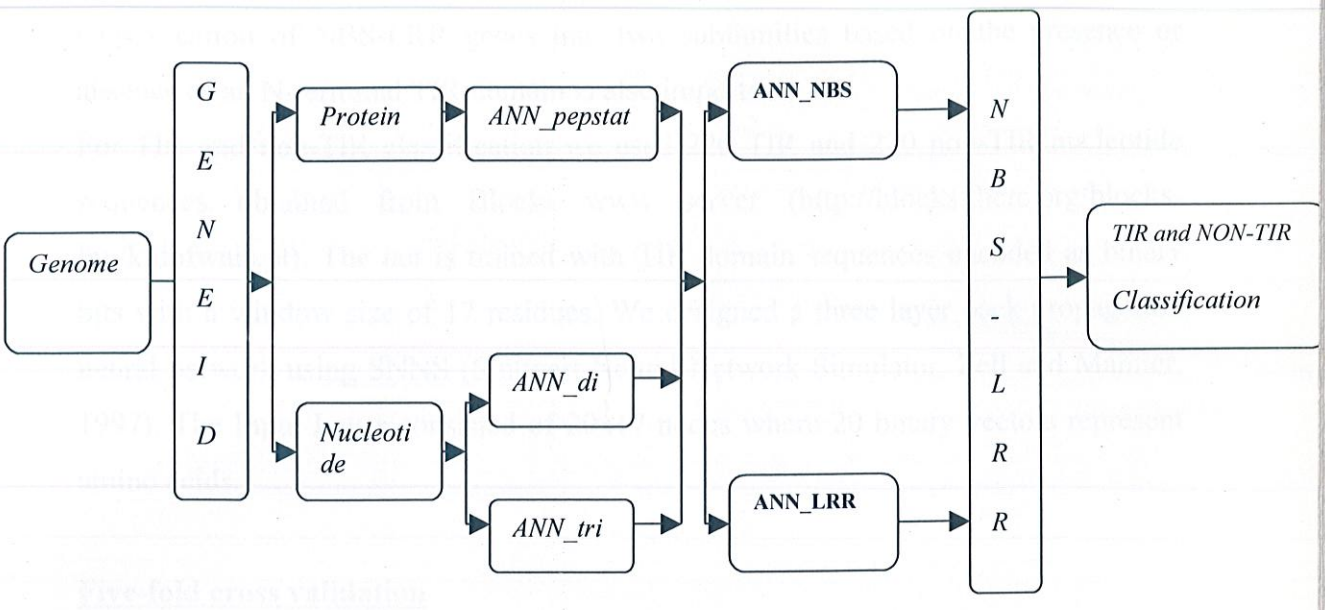


Figure 3: Flow chart of NBS-LRR gene prediction algorithm..

Datasets

1. The initial dataset for proteins consisted of 173 NBS-LRR and 173 non- NBS-LRR sequences were obtained from Entrez database (http://ncbi.nlm.nih.gov). The protein datasets used in the training-testing cycles of the "pepstats" modules were checked for sequence similarity to remove redundancy. Representative sequence from each datasets was taken to form the final datasets. We prepared a five datasets by mixing data from each datasets in equal number.

2. This procedure is repeated for nucleotide sequences in which we took 96 NBS-LRR and 96 non-NBS-LRR sequences were obtained from Entrez database (http://ncbi.nlm.nih.gov). The nucleotide datasets used in the training-testing cycles of the "compseq" modules were checked for sequence similarity to remove redundancy. Representative sequence from each datasets was taken to form the final datasets.

3. We used a two hidden unit's neural network trained to distinguish between NBS-LRR and non NBS-LRR regions. Protein statistics were used as the input units of a neural network to discriminate between the potential NBS-LRR and non-NBS-LRR.

Classification of NBS-LRR genes into two subfamilies based on the presence or absence of an N-terminal TIR domain is also important.

For TIR and non-TIR classification we used 220 TIR and 220 non-TIR nucleotide sequences obtained from Blocks www server (http://blocks.fhcrc.org/blocks-bin/kidofwais.pl). The net is trained with TIR domain sequences encoded as binary bits with a window size of 17 residues. We designed a three layer back propagation neural network using SNNS (Stuttgart Neural Network Simulator, Zell and Mamier, 1997). The Input Layer consisted of 20x17 nodes where 20 binary vectors represent amino acids. .

Five-fold cross validation

A newly developed statistical procedure must be checked for its validity. We have used five-fold cross validation technique to check the validity of all the modules that have been developed. For this purpose a dataset partitioning method was used to create the five sub-datasets. This partitioning method is similar to the previously used methods (Bentsen *et al.*, 2004). Here five sub datasets of sequences were created by randomly assigning a sequence to a sub-dataset such that each sub-dataset had approximately equal number of NBS-LRR and non- NBS-LRR and all five sub datasets had approximately equal number of sequences. Each of the three methods is trained and tested five times where, in each instance of training - testing cycle; four sub-datasets are used for training and the remaining one for testing purpose. The performance measures given have been averaged over the five testing sub-datasets.

Artificial Neural Networks

This neural network module was implemented using the Stuttgart neural network simulator (Zell and Mamier, 1997). A feed-forward neural network with standard back-propagation algorithm is utilized in all cases but the architecture differs as their function differs

- In ANNpepstats module the neural network used had an architecture as 51-2-1 i.e. it had 51 nodes in input layer representing the values of physicochemical properties from the pepstats encoding method, 2 nodes in hidden layer and 1 node in output

layer showing whether a given protein is NBS-LRR or non- NBS-LRR. The cut-off value used for prediction in this module is 0.75, i.e. a query protein is regarded as belonging to NBS-LRR family if its score is greater than or equal to 0.75.

- This process is repeated for ANNdinucleotide modul.Neural network used in nucleotide had an architecture as 16-2-1 i.e. it had 16 nodes in input layer representing the values of physicochemical properties from the dinucleotide and trinucleotide encoding method, 2 nodes in hidden layer and 1 node in output layer showing whether a given protein is NBS-LRR or non- NBS-LRR. The cut-off value used for prediction in this module is 0.8, i.e. a query protein is regarded as belonging to NBS-LRR family if its score is greater than or equal to 0.8.
- Neural network used in ANNtrinucleotide had an architecture as 64-2-1 i.e. it had 64 nodes in input layer representing the values of physicochemical properties from the dinucleotide and trinucleotide encoding method, 2 nodes in hidden layer and 1 node in output layer showing whether a given protein is NBS-LRR or non- NBS-LRR. The cut-off value used for prediction in this module is 0.8, i.e. a query protein is regarded as belonging to NBS-LRR family if its score is greater than or equal to 0.8.

Encoding methods

In this study of encoding methods are used, these are "pepstats", "dinucleotide" and "trinucleotide encoding" methods.

- Using the "pepstats", "dinucleotide" and"trinucleotide encoding" method, the protein sequence and nucleotide sequences are encoded into its physicochemical properties using pepstats and compseq application of EMBOSS suite of programs. In case of, protein out of all the properties, 51 properties are then used to create an input vector for training and testing of the machine learning modules and in case of, nucleotide we use "dinucleotide" and "trinucleotide" frequency . These properties are normalized, prior to creation of the input vectors. The list of properties used for training and testing the modules and their associated normalization factors that are used in this study; are given in the supplementary material.

Performance Measures

The performance measures used to evaluate the nine modules are listed below. These measures have been calculated for each module using five test sub-datasets; and the final measures (Table 1) given have been averaged over these five sub datasets.

- *Accuracy*: The accuracy of the modules have been calculated as:

$$Q_{ACC} = \frac{P + N}{P + N + O + U}$$

Where P and N refer to true positives and true negatives say correctly predicted NBS-LRR and non NBS-LRR proteins respectively; and O and U refer to false positives and false negatives i.e. incorrectly predicted NBS-LRR and non NBS-LRR proteins.

- *Specificity* (Qspec) and *sensitivity* (Qsens) of the modules are defined as:

$$Q_{spec} = \frac{N}{N + O} \qquad Q_{sens} = \frac{P}{P + U}$$

- The *Matthews correlation coefficient* (MCC) is defined as:

$$MCC = \frac{(P \times N) - (O \times U)}{\sqrt{(P+U) \times (P+O) \times (N+U) \times (N+O)}}$$

- QPred (*Probability of correct prediction*) is defined as:

$$Q_{pred} = \frac{P}{P + O} \times 100$$

CHAPTER 3

RESULTS AND DISCUSSION

Most of disease resistance genes encode proteins that have N-terminal nucleotide-binding site (NBS) and C-terminal leucine-rich repeat (LRR) domains interaction between avirulence (avr) gene product and NBS-LRR gene product give Pathogen recognition. NBS region of NBS-LRR proteins has many conserved motifs in which many of them depend upon subclasses of NBS-LRR gene. In contrast LRR region is highly variable. NBS-LRR genes were divided mainly into two subgroups which is based on presence of TIR (toll/interleukin-1 receptor) domain. Most of non-TIR NBS–LRR proteins have a coiled-coil (CC) domain in the N-terminal region. Prediction of NBS-LRR genes is important because it is crucial components of the hypersensitive response, despite their importance; there is not a single method till date for the prediction of NBS-LRR genes in a large segment of DNA sequences. The only possible method is analysis of individual gene, predicted by a gene prediction tool. Here, we have attempted to develop a method for predicting NBS-LRR gene and their classification into subfamilies using ANN. At present, no other method is available for predicting NBS-LRR genes and their classification, so we cannot compare the performance level. ANN is adaptive models and very useful for nonlinear function In addition it can handle large number of variables even there are nonlinear relationship among variables.

Performance over test dataset

We demonstrated whether this approach can identify NBS-LRR genes with significant accuracy. We used protein statistics as an input to discriminate between NBS and non-NBS proteins. Similarly dinucleotide and trinucleotide frequencies were used to discriminate NBS and non-NBS at DNA level. All three methods identified NBS-LRR genes on the basis of whole DNA sequence. Results showed that ANN model trained by protein statistics performed better than ANN model trained by

dinucleotide and trinucleotide frequencies. In all three cases, separation between NBS-LRR and non-NBS-LRR is very clear. But learning ability of artificial neural network is directly proportional to the size of the dataset used for training and in our case due to limited size of dataset performance of prediction affected when we use three models for genome or very large sequences; even we got a high accuracy level for training dataset. Another limitation in this approach is due to GENEID which is predicting potential genes and their protein product before ANN models. Prediction accuracy of genes directly affects overall performance of our model. The performance of the modules over the test dataset is given in Table 1.

Name of the module	*Accuracy*	*Specificity*	*Sensitivity*	*Probability of correct prediction (QPred)*	*Matthews correlation coefficient (MCC)*
ANN*pepstats*	87.00	0.8660	0.8880	85.41	0.7522
ANNdinucleotide	78.85	0.8231	0.7645	84.25	0.5835
ANNtrinucleotide	85.75	0.6925	0.8727	91.26	0.7452

Table 1. Summary of the prediction results of the three modules used in first and second layers on the test dataset.

Validation on rice genome sequence

A complete set of candidate disease resistance (*R*) genes encoding nucleotide-binding sites (NBSs) was identified in the genome sequence of *japonica* rice (*Oryza sativa* L. var. Nipponbare). These putative *R* genes were characterized with respect to structural diversity, phylogenetic relationships and chromosomal distribution, and compared with those in *Arabidopsis thaliana*. The number of NBS in rice genome is 535, including 480 non-TIR (Toll/IL-1 receptor) NBS-LRR genes through rice genome project (Molecular Genetics and Genomics, Volume 271, Number 4, May 2004, pp. 402-415(14)).

In analysis through our tool we tested whole genome (12 chromosomes) and obtained 362 NBS coding sequences. The number of coding sequences obtained in each chromosome given in Table 2.

Chromosome no	*Total prediction*	*No. of NBS-LRR*
1	63	35
2	44	18
3	44	16
4	53	19
5	36	16
6	40	20
7	35	19
8	55	42
9	29	18
10	37	20
11	116	101
12	55	38

Table 2: Summary of the prediction of rice genome.

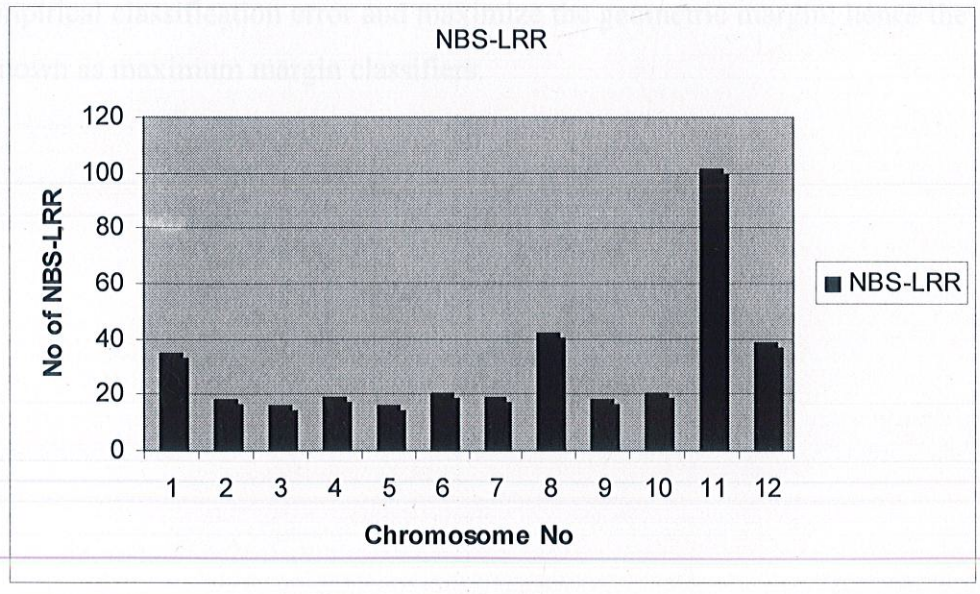


Figure 4 Distribution of output values of NBS-LRR gene over of chromosome of rice genome.

CHAPTER 4

CONCLUSIONS

Prediction of NBS-LRR genes from genome sequence data is important in plants as well as in animals. This approach and tool developed here is vital because at present not such tool available is which can predict NBS-LRR genes. Intelligent systems like the ones used in this study, utilize global information of protein sequence data as well as nucleotide sequence data instead of using simple pattern matching techniques. Such systems can significantly increase the accuracy of NBS-LRR gene related studies. This study gives an insight into development of similar tools and techniques for other important genes.

We can further improve this tool by using SVM (support vector machines) because support vector machines are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

APPENDIX-I

PROGRAMS:

1. Batch_processing.pl

```
$/=undef;

$glob="*.seq";
@files=glob($glob);
open(REMAIN,">g.txt");
foreach $file(@files)
{
open(HAND,"index.txt");
@conindex=<HAND>;
close HAND;

$sequence='';
open(SEQU,$file);
$sequence=<SEQU>;
close SEQU;

$sequence=~ s/^>.*\n//gi;
$sequence=~ s/^\s+//mg;
$sequence=~ s/\s+$//mg;
$sequence=~ s/\s+//mg;

foreach(@conindex)
{
my @array=();
@array=split(' ',$_);
if($array[0] eq "temp_files/".$file)
{
$org_name=$array[1];
}
}
```

```
my $param="geneid -P ".$org_name." ".$file." > ".$file."_res.txt";
print "Running $param....\n";
system($param);

open(HAN,$file."_res.txt");
$result=<HAN>;
close HAN;
@ele=split(/#\sGene\s\d+/,$result);
shift(@ele);

foreach $temp(@ele)
{
$gene="";
@genes=();

open(TEMP,">temp");
print TEMP $temp;
close TEMP;
open(TEM,"temp");
$scalar=<TEM>;
close TEM;
@genes=split(/First|Internal|Terminal|Single/,$scalar);
#$scalar=~ s/\n//gi;

if($scalar =~ />.*\n([\w\n]*)/gi)
{
push(@proteins,$1);
}
    if($genes[0] =~ /(Forward|Reverse)/)
    {
    $sense=$1;
    }
shift @genes;
$genename='';
foreach $gene(@genes)
{
$gene=~ s/\s+/s/gi;
```

```
$exon="";
@array=split('s',$gene);
    if($array[1] =~ /\d/)
    {
    $offset=$array[1]-1;
    $count=$array[2]-$array[1];
    $exon=substr($sequence, $offset, $count);
    $genename=$genename.$exon;
    }
            if($sense eq "Reverse")
            {
            $genename=~ tr /ATGCatgc/TACGtacg/;
            }
}
push(@allgene,$genename);
}

$y=0;
open(OUT11,'>>allgeneshere.txt') or die "file can't be accessable";
foreach $gr(@allgene)
{
print OUT11 ">psdm$y\n";
print OUT11 "$gr\n\n";
$y++
}

open(SAD,">file_proteins.txt");
foreach(0..(scalar(@allgene)-1))
{
$count=$_+1;
$proteins[$count-1]=~ s/\n//gi;
print SAD ">test".$count."\n$proteins[$count-1]\n";
print "Printing gene and protein for $count\n";
open(FH,">".$file.".gene");
print FH ">test".$count."\n".$allgene[$count-1]."\n";
close FH;
```

```
$proteins[$count-1]=~ s/[XBZU*]//gi;
open(FH,">".$file.".protein");
print FH ">test".$count."\n".$proteins[$count-1]."\n";
close FH;
ditripat($file,$count);
propat($file,$count);
}
close SAD;
system("dinucleotide.exe");
system("trinucleotide.exe");
system("protein.exe");

open(HAN,"Seq_di_res.txt");
$di=<HAN>;
close HAN;

open(HAND,"Seq_tri_res.txt");
$tri=<HAND>;
close HAND;

open(HANDL,"Seq_pro_res.txt");
$pro=<HANDL>;
close HANDL;

@din=split(' ',$di);
@trin=split(' ',$tri);
@pron=split(' ',$pro);
open (BAD,">".$file.".result");

open (BAD11,">re1.txt");
open (BAD12,">re2.txt");

$counter=0;
foreach $in(0..(scalar(@din)-1))
{
if(   (($pron[$in] >= 0.75) && ($din[$in] >= 0.8)) || (($pron[$in] >= 0.75) && ($trin[$in] >= 0.8)))
{
```

```
$ge=$in+1;
print                                                                    BAD
"\n\n*****************************************************************************
*************************\n";
print                                                                    BAD
"*****************************************************************************
**********************\n";
print BAD "The gene no. $ge is predicted to be an NBS-LRR coding gene. Its details are as
follows:\n";
print BAD $ele[$in]."\n";
print BAD11 "NOTE The gene no. $ge is predicted to be an NBS-LRR coding gene. Its details are as
follows:\n";
print BAD11 $ele[$in]."\n";
$counter++;
}
elsif(($pron[$in] >= 0.5) || ($din[$in] >= 0.6) || ($trin[$in] >= 0.6))
{
$ge=$in+1;
print BAD12 "NOTE The gene no. $ge is predicted to be an NBS-LRR coding gene. Its details are as
follows:\n";
print BAD12 $ele[$in]."\n";
$counter++;
}
else
{
$ge=$in+1;
print REMAIN $ele[$in]."\n";
}

}
print BAD "\nTotal are $counter\n";
system("profilematch.pl");
$counter=0;

open(EE,"final_res.txt");
open(PAD,">finally.txt");
$s=<EE>;
```

```
close EE;
@nos=split('\n',$s);
foreach(@nos)
{
print                                                                    PAD
"\n\n*************************************************************************
************************\n";
print                                                                    PAD
"*************************************************************************
**********************\n";
print PAD "This gene is predicted to be an NBS-LRR coding gene. Its details are as follows:\n";
print PAD $ele[$_-1]."\n";
$counter++;

}
print PAD "\nTotal are $counter\n";

close PAD;
}

sub ditripat
{
$file=@_[0];
$count=@_[1];
system("compseq -sequence ".$file.".gene -word 2 -outfile out_di.txt");
system("compseq -sequence ".$file.".gene -word 3 -outfile out_tri.txt");

open(HANDLE,"out_di.txt");
my $content=<HANDLE>;
close HANDLE;

open(GOOD,">>Seq_di_out.txt");
print GOOD "#Input_pattern_$count:\n";

@freq=();
my @freq=split(' ',$content);
foreach(0..(scalar(@freq)-1))
```

```
{
if($freq[$_]=~ /^[ATGC][ATGC]/)
{
print GOOD $freq[$_+4]."\t";
$flag=1;
}
}
if($flag == 1)
{
print GOOD "\n";
$flag=0;
}
close GOOD;

$content='';
open(HANDLE,"out_tri.txt");
my $content=<HANDLE>;
close HANDLE;

open(GOOD,">>Seq_tri_out.txt");

print GOOD "#Input_pattern_$count:\n";

@freq=();
my @freq=split(' ',$content);
foreach(0..(scalar(@freq)-1))
{
    if($freq[$_]=~ /^[ATGC][ATGC][ATGC]/)
    {
    print GOOD $freq[$_+4]."\t";
    $flag=1;
    }
}
    if($flag == 1)
    {
    print GOOD "\n";
    $flag=0;
```

```
    }
close GOOD;
}

sub propat{

$filename=@_[0];
$count=@_[1];

$filename=$filename.".protein";
system("pepstats $filename pepstats_outfile.txt -auto 1");
open(HAN,"pepstats_outfile.txt");
$co=<HAN>;
close HAN;
@con=split(' ',$co);
chomp(@con);
open(GOOD,">>pepstats_infile.txt");
print GOOD "#Input_pattern_$count:\n";
foreach(0..(scalar(@con)-1))
{
if($con[$_] eq "Molecular")
{
$value=$con[$_+3]/100000;
print GOOD "$value ";
}

if($con[$_]eq "Average")
{
$value=$con[$_+4]/1000;
print GOOD "$value ";
}

if($con[$_]eq "Isoelectric")
{
print GOOD ($con[$_+3]/10)." ";
}
```

```
if($con[$_] eq "Molar")
{
print GOOD ($con[$_+4]/100000)." ";
}

if( ($con[$_] eq "Ala") || ($con[$_] eq "Cys") || ($con[$_] eq "Asp")|| ($con[$_] eq "Glu") || ($con[$_]
eq "Phe") || ($con[$_] eq "Gly") || ($con[$_] eq "His") || ($con[$_] eq "Ile") || ($con[$_] eq "Lys") ||
($con[$_] eq "Leu") || ($con[$_] eq "Met") || ($con[$_] eq "Asn") || ($con[$_] eq "Pro") || ($con[$_] eq
"Gln") || ($con[$_] eq "Arg") || ($con[$_] eq "Ser") || ($con[$_] eq "Thr") || ($con[$_] eq "Val") ||
($con[$_] eq "Tyr") )
{
if($con[$_] ne "Property")
{
print GOOD ($con[$_+2]/10)." ".$con[$_+3]." ";
}
}

if($con[$_] eq "Tiny")
{
print GOOD ($con[$_+3]/100)." ";
}

if($con[$_] eq "Small")
{
print GOOD ($con[$_+3]/100)." ";
}

if($con[$_] eq "Aliphatic")
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_] eq "Aromatic")
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_] eq "Non-polar")
{
```

```
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_] eq "Polar")
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_] eq "Charged")
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_] eq "Basic")
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_] eq "Acidic")
{
print GOOD ($con[$_+3]/100)." ";
$flag=1;
}
if($flag==1)
{
print GOOD "\n";
$flag=0;
}
}
close GOOD;
}

<>;
```

2. 1. pl

```
$/=undef;
open(HAN,"pepstats_outfile.txt");
$co=<HAN>;
close HAN;
@con=split(' ',$co);
chomp(@con);
open(GOOD,">>pepstats_infile.txt");
print GOOD "Input_pattern_$count:\n";
foreach(0..(scalar(@con)-1))
{
if($con[$_]=~ /Molecular/)
{
$value=$con[$_+3]/100000;
print GOOD "$value ";
}

if($con[$_]=~ /Average/)
{
$value=$con[$_+4]/1000;
print GOOD "$value ";
}

if($con[$_]=~ /Isoelectric/)
{
print GOOD ($con[$_+3]/10)." ";
}

if($con[$_]=~ /Molar/)
{
print GOOD ($con[$_+4]/100000)." ";
}

if( ($con[$_]=~ /Ala/) || ($con[$_]=~ /Cys/) || ($con[$_]=~ /Asp/)||
($con[$_]=~ /Glu/) || ($con[$_]=~ /Phe/) || ($con[$_]=~ /Gly/) ||
($con[$_]=~ /His/) || ($con[$_]=~ /Ile/) || ($con[$_]=~ /Lys/) ||
($con[$_]=~ /Leu/) || ($con[$_]=~ /Met/) || ($con[$_]=~ /Asn/) ||
```

```
($con[$_]=~ /Pro/) || ($con[$_]=~ /Gln/) || ($con[$_]=~ /Arg/) ||
($con[$_]=~ /Ser/) || ($con[$_]=~ /Thr/) || ($con[$_]=~ /Val/) ||
($con[$_]=~ /Tyr/) )
{
if($con[$_] ne "Property")
{
print GOOD ($con[$_+2]/10)." ".$con[$_+3]." ";
}
}

if($con[$_]=~ /Tiny/)
{
print GOOD ($con[$_+3]/100)." ";
}

if($con[$_]=~ /Small/)
{
print GOOD ($con[$_+3]/100)." ";
}

if($con[$_]=~ /Aliphatic/)
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_]=~ /Aromatic/)
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_]=~ /Non-polar/)
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_]=~ /Polar/)
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_]=~ /Charged/)
{
```

```
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_]=~ /Basic/)
{
print GOOD ($con[$_+3]/100)." ";
}
if($con[$_]=~ /Acidic/)
{
print GOOD ($con[$_+3]/100)." ";
$flag=1;
}
if($flag==1)
{
print GOOD "\n";
$flag=0;
}
}
close GOOD;
```

3. PROG1.PL

```
#!/usr/bin/perl
$/=undef;
open(IN,"re1.txt") or die "file can't be accessable";
open(SE,"h1.txt") or die "file can't be accessable";
open(OUUT,'>COUNT.txt') or die "file can't be accessable";
open(MAD,">>RESULT_OF_PREDICTION.txt");

$count=0;

$var=<IN>;
@ele=split(/_AA/,$var);

$ff=<SE>;
@arr=split(/_AA/,$ff);

$count=0;
#print OUT @ele;
$i=0;

$y=0;
$yk=0;

foreach $temp1(@ele)
{
$temp1=~ s/^\s+//mg;
$temp1=~ s/\s+$//mg;
$temp1=~ s/\s+//mg;

 if( $temp1=~ /(\w+)[*]/gi )
  {
   $pregene[$y]=$1;
   $y++;
  }
```

```
}

foreach $ll(@arr)
{
$ll=~ s/^\s+//mg;
$ll=~ s/\s+$//mg;
$ll=~ s/\s+//mg;

 if( $ll=~ /(\w+)[*]/gi )
  {
    $pr[$yk]=$1;
    $yk++;
  }

}

foreach $p(@pr)
{
print MAD "\n>Predicted Protein:\n$p\n";
}

$geneno=0;
$ik=0;

foreach $var(@pregene)
{
$flag=0;
$h=0;
$geneno++;
$ik++;
      open(OUT,'>UPENDRA.txt') or die "file can't be accessable";
      open(OUT4,'>infile.txt') or die "file can't be accessable";
      open(OUT44,'>infile1.txt') or die "file can't be accessable";
      open(OUT555,'>infile_lrr1.txt') or die "file can't be accessable";
      open(OUT5552,'>infile_lrr.txt') or die "file can't be accessable";
```

```
    $i=0;
    @len=split(//,$var);
    $leng=$#len;
    $leng=$leng-16;
    print OUT ">\n";
    print OUT "$len\n";

    for($po=0;$i<=$leng;$po++)
    {
    $var1=$var;
    $var4=$var;
    $var555=$var;

    $tri[$i]=substr($var1,$po,15);
    $tri4[$i]=substr($var4,$po,16);
    $tri555[$i]=substr($var555,$po,14);

    print OUT ">$i\n";
    print OUT "$tri[$i]\n";

@element=split(//,$tri[$i]);
@element4=split(//,$tri4[$i]);
@element555=split(//,$tri555[$i]);


#$j++;
$i++;
$h++;
print OUT4 "\n#Input_pattern_".$h.":";
print OUT4 "\n";

print OUT44 "\n#Input_pattern_".$h.":";
print OUT44 "\n";
```

```
print OUT555 "\n#Input_pattern_".$h.":";
print OUT555 "\n";

print OUT5552 "\n#Input_pattern_".$h.":";
print OUT5552 "\n";

for($z=0;$z<=$#element;$z++)
{
@line=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

if($element[$z] eq 'A')
{
$line[0]=1;
}
elsif($element[$z] eq 'Y')
{
$line[1]=1;
}
elsif($element[$z] eq 'C')
{
$line[2]=1;
}
elsif($element[$z] eq 'D')
{
$line[3]=1;
}
elsif($element[$z] eq 'E')
{
$line[4]=1;
}
elsif($element[$z] eq 'F')
{
$line[5]=1;
}
elsif($element[$z] eq 'G')
{
$line[6]=1;
```

```
}
elsif($element[$z] eq 'H')
{
$line[7]=1;
}
elsif($element[$z] eq 'I')
{
$line[8]=1;
}
elsif($element[$z] eq 'K')
{
$line[9]=1;
}
elsif($element[$z] eq 'L')
{
$line[10]=1;
}
elsif($element[$z] eq 'M')
{
$line[11]=1;
}
elsif($element[$z] eq 'N')
{
$line[12]=1;
}
elsif($element[$z] eq 'P')
{
$line[13]=1;
}
elsif($element[$z] eq 'Q')
{
$line[14]=1;
}
elsif($element[$z] eq 'R')
{
$line[15]=1;
}
```

```
elsif($element[$z] eq 'S')
{
$line[16]=1;
}
elsif($element[$z] eq 'T')
{
$line[17]=1;
}
elsif($element[$z] eq 'V')
{
$line[18]=1;
}
elsif($element[$z] eq 'W')
{
$line[19]=1;
}
else
{
}
print OUT4 "@line ";
}

for($z=0;$z<=$#element555;$z++)
{
@line=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

if($element[$z] eq 'A')
{
$line[0]=1;
}
elsif($element[$z] eq 'Y')
{
$line[1]=1;
}
elsif($element[$z] eq 'C')
{
```

```
$line[2]=1;
}
elsif($element[$z] eq 'D')
{
$line[3]=1;
}
elsif($element[$z] eq 'E')
{
$line[4]=1;
}
elsif($element[$z] eq 'F')
{
$line[5]=1;
}
elsif($element[$z] eq 'G')
{
$line[6]=1;
}
elsif($element[$z] eq 'H')
{
$line[7]=1;
}
elsif($element[$z] eq 'I')
{
$line[8]=1;
}
elsif($element[$z] eq 'K')
{
$line[9]=1;
}
elsif($element[$z] eq 'L')
{
$line[10]=1;
}
elsif($element[$z] eq 'M')
{
$line[11]=1;
```

```
}
elsif($element[$z] eq 'N')
{
$line[12]=1;
}
elsif($element[$z] eq 'P')
{
$line[13]=1;
}
elsif($element[$z] eq 'Q')
{
$line[14]=1;
}
elsif($element[$z] eq 'R')
{
$line[15]=1;
}
elsif($element[$z] eq 'S')
{
$line[16]=1;
}
elsif($element[$z] eq 'T')
{
$line[17]=1;
}
elsif($element[$z] eq 'V')
{
$line[18]=1;
}
elsif($element[$z] eq 'W')
{
$line[19]=1;
}
else
{
}
print OUT555 "@line ";
```

```
print OUT5552 "@line ";
}

for($z=0;$z<=$#element4;$z++)
{
@line=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

if($element4[$z] eq 'A')
{
$line[0]=1;
}
elsif($element4[$z] eq 'Y')
{
$line[1]=1;
}
elsif($element4[$z] eq 'C')
{
$line[2]=1;
}
elsif($element4[$z] eq 'D')
{
$line[3]=1;
}
elsif($element4[$z] eq 'E')
{
$line[4]=1;
}
elsif($element4[$z] eq 'F')
{
$line[5]=1;
}
elsif($element4[$z] eq 'G')
{
$line[6]=1;
}
elsif($element4[$z] eq 'H')
{
```

```
$line[7]=1;
}
elsif($element4[$z] eq 'I')
{
$line[8]=1;
}
elsif($element4[$z] eq 'K')
{
$line[9]=1;
}
elsif($element4[$z] eq 'L')
{
$line[10]=1;
}
elsif($element4[$z] eq 'M')
{
$line[11]=1;
}
elsif($element4[$z] eq 'N')
{
$line[12]=1;
}
elsif($element4[$z] eq 'P')
{
$line[13]=1;
}
elsif($element4[$z] eq 'Q')
{
$line[14]=1;
}
elsif($element4[$z] eq 'R')
{
$line[15]=1;
}
elsif($element4[$z] eq 'S')
{
$line[16]=1;
```

```
}
elsif($element4[$z] eq 'T')
{
$line[17]=1;
}
elsif($element4[$z] eq 'V')
{
$line[18]=1;
}
elsif($element4[$z] eq 'W')
{
$line[19]=1;
}
else
{
}
print OUT44 "@line ";
}
}


system("a.exe");
system("domain3.exe");
system("domain1.exe");
system("lrr1.exe");
system("lrr2.exe");

open(INI,"outfile2.txt") or die "file can't be accessable";
$varu=<INI>;
@d2=split(/\s+/,$varu);


open(INI3,"outfile3.txt") or die "file can't be accessable";
$var3=<INI3>;
@d3=split(/\s+/,$var3);

open(INI4,"outfile1.txt") or die "file can't be accessable";
```

```
$var4=<INI4>;
@d1=split(/\s+/,$var4);


open(LRR2,"outfile_lrr.txt") or die "file can't be accessable";
$lr2=<LRR2>;
@l2=split(/\s+/,$lr2);

open(LRR1,"outfile_lrr1.txt") or die "file can't be accessable";
$lr1=<LRR1>;
@l1=split(/\s+/,$lr1);


$jk=1;

for($x11=0;$x11<=$#d2;$x11++)
{

if(($d2[$x11] > 0.99) || ($d3[$x11] > 0.99) || ($d1[$x11] > 0.995) || ($l1[$x11] > 0.997) || ($l2[$x11] >
0.9973))
{
$index[$ik]=$geneno;
$flag=1;
}

$jk++;
}

if($flag==1)
{
print MAD "\n>Predicted Protein:\n$var\n";
}


}
#print OUUT "\n@index";
system("program2.pl");
system("class.pl");
```

close MAD;

4. Class.pl

```
#!/usr/bin/perl
$/=undef;
open(IN,"RESULT_OF_PREDICTION.txt") or die "file can't be accessable";
open(OUT1,'>TIR_RESULT.txt') or die "file can't be accessable";
open(OUT2,'>NON_TIR_RESULT.txt') or die "file can't be accessable";


$var=<IN>;
@ele=split(/>Predicted Protein:/,$var);

$count=0;

$i=0;

$y=0;

foreach $temp1(@ele)
{
$temp1=~ s/^\s+//mg;
$temp1=~ s/\s+$//mg;
$temp1=~ s/\s+//mg;

 if( $temp1=~ /(\w+)/gi )
  {
   $pregene[$y]=$1;
   $y++;
  }

}

$geneno=0;
$ik=0;

foreach $var(@pregene)
```

```
{
$flag=0;
$h=0;
$geneno++;
$ik++;
    open(OUT4,'>tirinfile.txt') or die "file can't be accessable";

    $i=0;
    @len=split(//,$var);
    $leng=$#len;
    $leng=$leng-25;

    for($po=0;$i<=$leng;$po++)
    {
    $var1=$var;

    $tri[$i]=substr($var1,$po,18);

    print OUT ">$i\n";
    print OUT "$tri[$i]\n";
@element=split(//,$tri[$i]);

#$j++;
$i++;
$h++;
print OUT4 "\n#Input_pattern_".$h.":";
print OUT4 "\n";

for($z=0;$z<=$#element;$z++)
{
@line=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

if($element[$z] eq 'A')
{
$line[0]=1;
}
```

```
elsif($element[$z] eq 'Y')
{
$line[1]=1;
}
elsif($element[$z] eq 'C')
{
$line[2]=1;
}
elsif($element[$z] eq 'D')
{
$line[3]=1;
}
elsif($element[$z] eq 'E')
{
$line[4]=1;
}
elsif($element[$z] eq 'F')
{
$line[5]=1;
}
elsif($element[$z] eq 'G')
{
$line[6]=1;
}
elsif($element[$z] eq 'H')
{
$line[7]=1;
}
elsif($element[$z] eq 'I')
{
$line[8]=1;
}
elsif($element[$z] eq 'K')
{
$line[9]=1;
}
elsif($element[$z] eq 'L')
```

```
{
$line[10]=1;
}
elsif($element[$z] eq 'M')
{
$line[11]=1;
}
elsif($element[$z] eq 'N')
{
$line[12]=1;
}
elsif($element[$z] eq 'P')
{
$line[13]=1;
}
elsif($element[$z] eq 'Q')
{
$line[14]=1;
}
elsif($element[$z] eq 'R')
{
$line[15]=1;
}
elsif($element[$z] eq 'S')
{
$line[16]=1;
}
elsif($element[$z] eq 'T')
{
$line[17]=1;
}
elsif($element[$z] eq 'V')
{
$line[18]=1;
}
elsif($element[$z] eq 'W')
{
```

```
$line[19]=1;
}
else
{
}
print OUT4 "@line ";
}

}
system("tirclass.exe");
open(INI,"tiroutfile.txt") or die "file can't be accessable";
$varu=<INI>;
@d2=split(/\s+/,$varu);

$jk=1;

for($x11=0;$x11<=$#d2;$x11++)
{

if($d2[$x11] > 0.993)
{
$index[$ik]=$geneno;
$flag=1;
}

$jk++;
}


if($flag==1)
{
print OUT1 "\n>Predicted Protein:\n$var\n";
}
else
{
print OUT2 "\n>Predicted Protein:\n$var\n";
}
```

}

APPENDIX-II

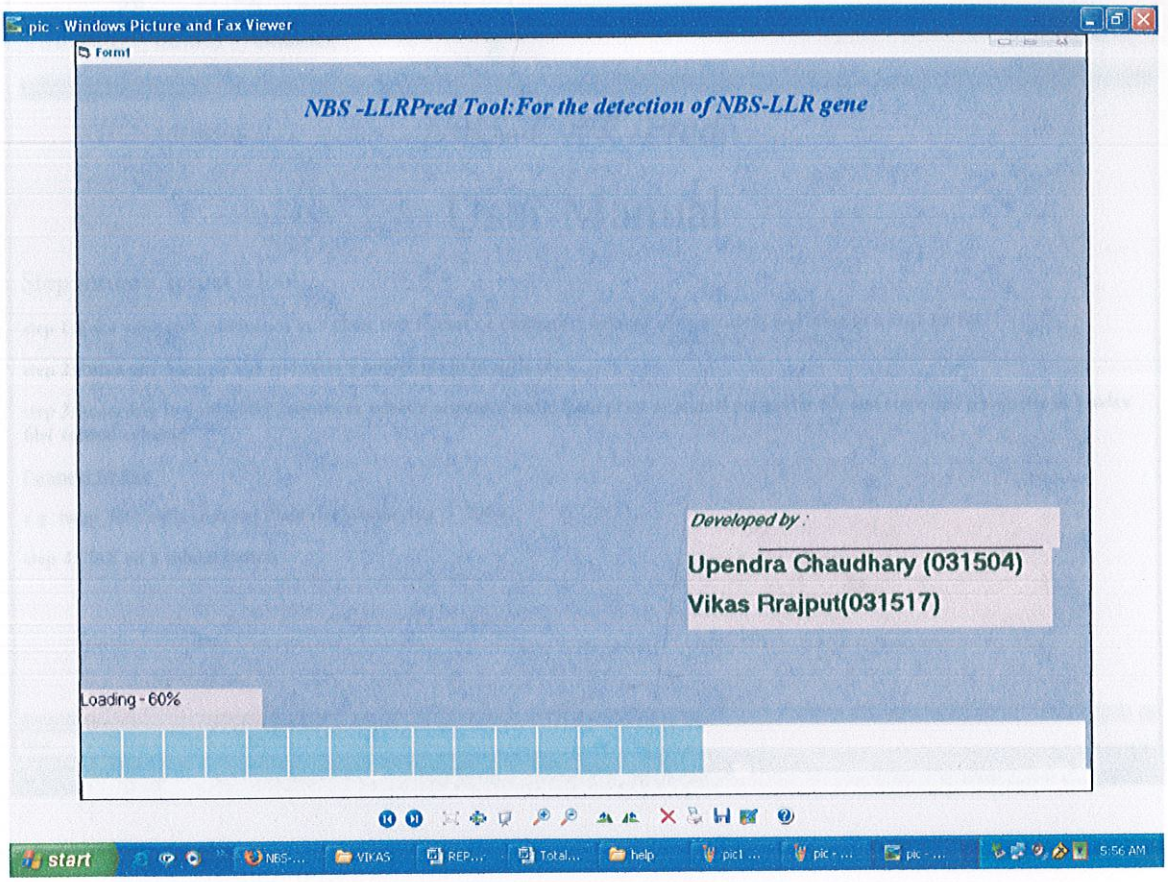
Table for protein properties calculated using PEPSTATS and their associated normalization factors that were used in the study.

Property name	***Normalization factor***
Molecular weight.	10^5
Average Residue Weight.	10^3
Isoelectric Point.	10
A280 Extinction Coefficient 1mg/ml.	10^5
Mole % of 19 standard amino acids: Alanine, Cysteine, Aspartic Acid, Glutamic Acid, Phenylalanine, Glycine, Histidine, Isoleucine, Lysine, Leucine, Methionine, Asparagine, Proline, Glutamine, Arginine, Serine, Threonine, Valine and Tryptophan.	10
DayhoffStat of 19 standard amino acids: Alanine, Cysteine, Aspartic Acid, Glutamic Acid, Phenylalanine, Glycine, Histidine, Isoleucine, Lysine, Leucine, Methionine, Asparagine, Proline, Glutamine, Arginine, Serine, Threonine, Valine and Tryptophan.	1
Mole % of Tiny residues.	10^2
Mole % of Small residues.	10^2
Mole % of Aliphatic residues.	10^2
Mole % of Aromatic residues.	10^2
Mole % of Non-polar residues.	10^2
Mole % of Polar residues.	10^2
Mole % of Charged residues.	10^2
Mole % of Basic residues.	10^2
Mole % of Acidic residues.	10^2

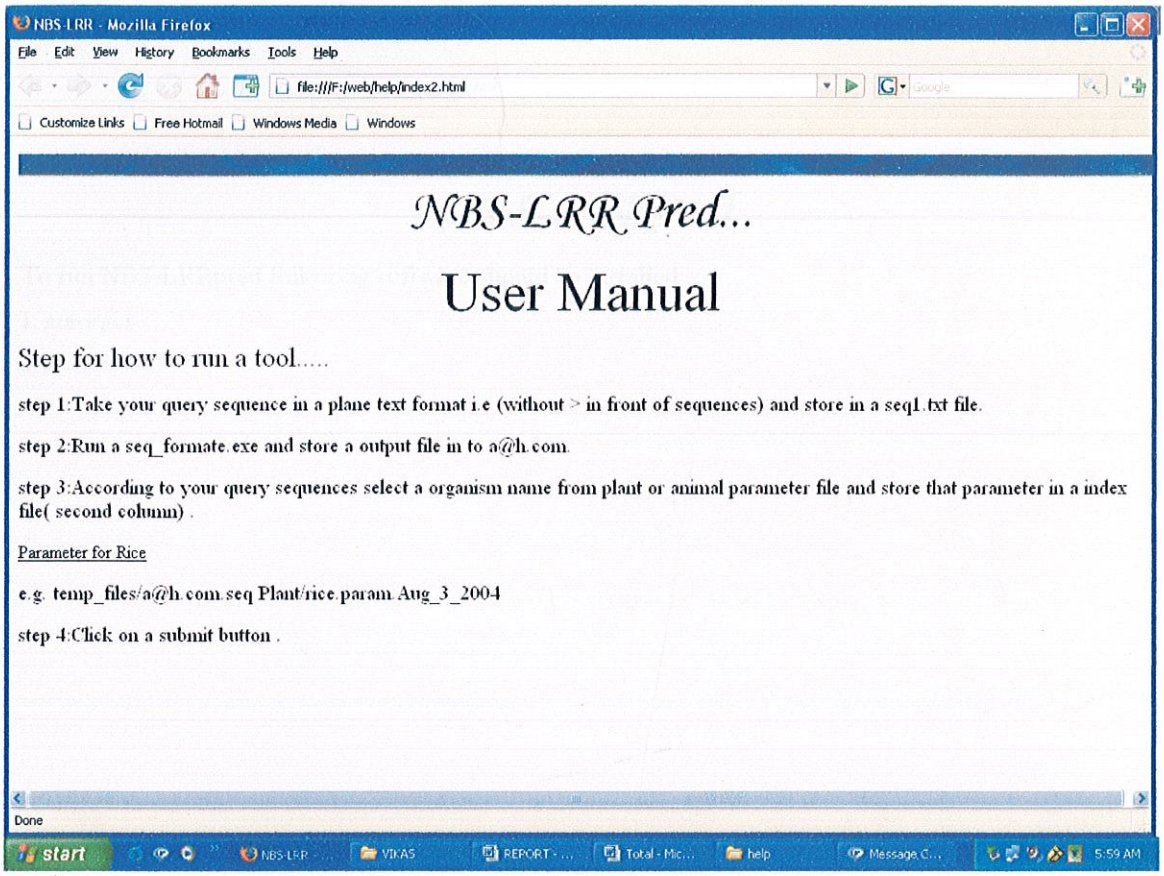
APPENDIX-III

SCREENSHOTS

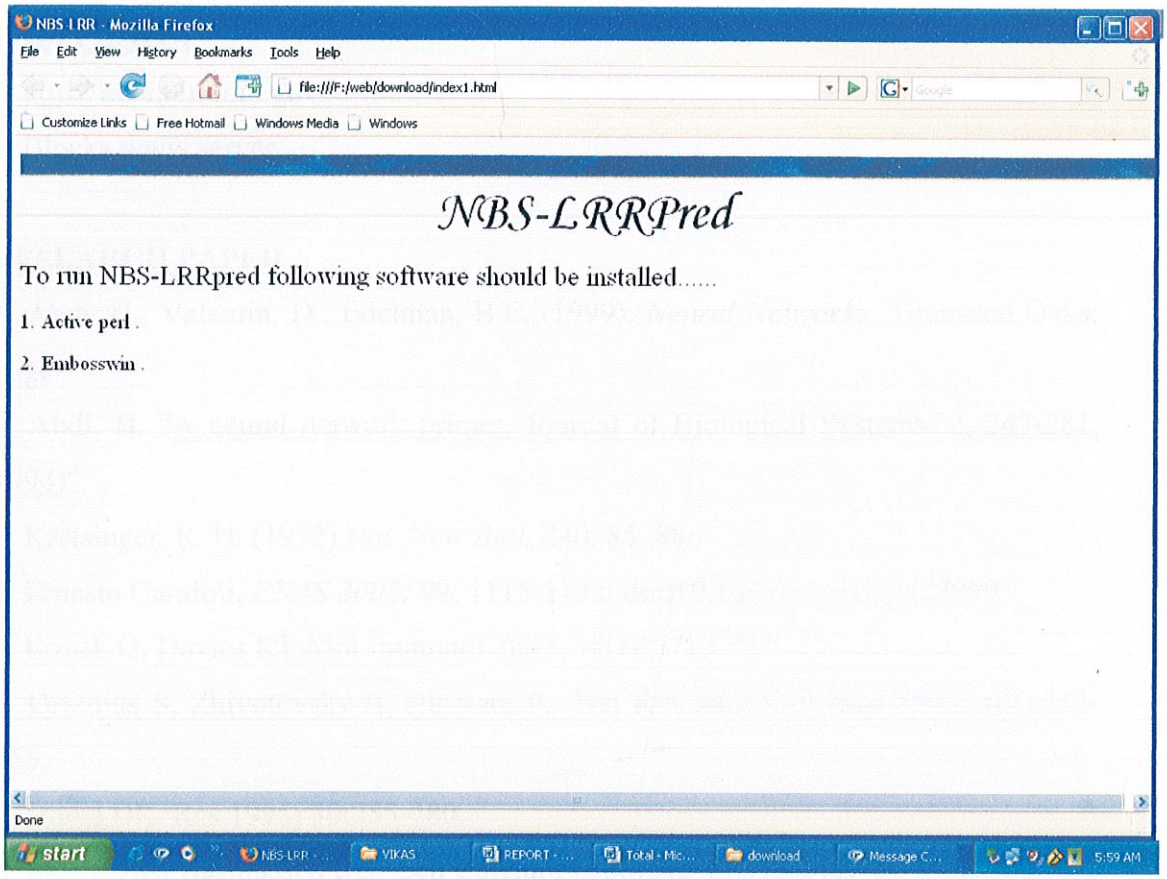
SCREENSHOT 1(HOMEPAGE)



SCREENSHOT 2(HELPAGE)



SCREENSHOT 3(REQUIREMENTPAGE)



BIBLIOGRAPHY

WEB PAGES:

1. http://ncbi.nlm.nih.gov

2. Blocks www server.

RESEARCH PAPER

1. Abdi, H., Valentin, D., Edelman, B.E. (1999). *Neural Networks.* Thousand Oaks: Sage.

2. Abdi, H. "A neural network primer. Journal of Biological Systems, 2, 247-281, 1994)".

3. Kretsinger, R. H. (1972) *Nat. New Biol.* 240, 85–88.

4. Ernesto Carafoli, *PNAS 2002;* 99; 1115-1122; doi:10.1073/pnas.032427999.

5. Ermak G, Davies KJ. Mol Immunol 2002; 38(10):713-721.

6. Orrenius S, Zhivotovsky B, Nicotera P. Nat Rev Mol Cell Biol 2003; 4(7):552-565.

7. Dowd DR. Res 1995; 30:255-280.

8. Means AR, Rasmussen CD. Cell Calcium 1988; 9(5-6):313-319.

9. Greenberg JT, Vinatzer BA: Identifying type III effectors of plant pathogens and analyzing their interaction with plant cells.

10. Curr Opin Microbioly 2003, 6:20-28.

11. Nimchuk Z, Eulgem T, Holt BF III, Dangl JL: Recognition and response in the plant immune system. Annu Rev Genet 2003, 37:579-609.

12. Glaze brook J, Rogers EE, Ausubel FM: Use of Arabidopsis for genetic dissection of plant defense responses. Annu Rev.

13. Altschul, S.F., Madden, T.L., Alejandro, A.S., Zhang, J., Zheng, Z., Miller, W., and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.,* 25, 3389-3402.

14. Bendtsen, J.D., Jensen, L.J., Blom, N., Von, H.G. and Brunak, S. (2004) Feature-based.

15. Prediction of non-classical and leaderless protein secretion. *Protein Eng. Des Sel*, 349-356.

16. Ding, C.H.Q. and Dubchak, I. (2001) Multi-class protein fold recognition using support.

17. Vector machines and neural networks. *Bioinformatics,* 17(4), 349-358.

18. Kesmir, C., Nussbaum, A.K., Schild, H., Detours, V., and Brunak, S. (2002) Prediction Of proteasome cleavage motifs by neural networks. *Protein Engineering.* 15(4), 287-296.

19. Hua, S. and Sun,Z. (2001) Support vector machine approach for protein subcellular Localization prediction. *Bioinformatics*, 17(8), 721-728

BOOKS

1. An introduction to neural computing. Aleksander, I. and Morton, H. 2nd edition.