TIMPRED - A WEBSERVER for the Prediction of TIM-barrel Protein Fold

Ajanma Singh

Delhi Technological University, Delhi, India

ABSTRACT

TIM-barrel is highly conserved and an ubiquitously occurring protein fold in nature and is the most common protein fold in the Protein Data Bank. TIM barrel scaffold is a highly diverse fold with at least known 33 distinct enzyme super families catalyzing activity completely unrelated enzymatic reactions. Moreover, some proteins with TIM barrel architecture, particularly belonging to GH-k (Glycosyl hydrolases) clan have been reported without any enzymatic reactions. The pair wise amino acid sequence similarity of Tim-barrel folds from non-homologous enzyme families is generally below the detectable level. Tim-Barrel prediction server has been developed.

A machine learning algorithm using SVM-based method have been used as the prediction method. The prediction model has been developed consisting of training and test set of about 7600 non redundant sequences. Three different features using amino acid count, dipeptide composition and position specific scoring matrix have been used to train the SVM. The overall accuracy and maximum MCC for all the features are 0.75 & 84.93% , 0.75.78 & 75.78%, 0.9434 & 94.34% has been achieved respectively. A five-fold cross validation was used to evaluate the performance of these methods.

Currently there is no prediction server that is devoted for TIM-BARREL fold recognition; therefore TIMPRED will prove itself a very important tool in the area of protein fold prediction and identification.

INTRODUCTION

With the growth of genome projects the number of sequences information have shot up in an astounding rate and lot of data has been amassed. In order to limit the vast space among the large number of new sequence data and experimental enactment of the matching proteins, hence scientists have to find the computational ways to efficiently analyze these data.

Analyzing functions of a protein is considered as one of the most disputing jobs of the post-genomic era. Previously various techniques have been evolved to predict the functions of proteins with the help of information which was devised from sequence similarity or clumping patterns of protein - protein interaction (PPI), co-regulated genes etc. It is very significant to understand interaction of one protein with other protein or ligands to deduce its function.

The triosephosphate isomerase (TIM)- barrel motif forms frequently(Albery *et al.*,1976) in the proteasome of various organisms and the TIM-barrel proteins known to play different and various functional jobs. TIM barrel domain is widely studied since it is one of the most common structure and mediates diverse function maintain overall structure.

TIM barrels are looked at α/β (Deirdre Reardon *et al.*, Gregory, K.Farber) protein folds since they involved an alternating figure of α-helices and β-strands in an individual domain. In TIM-barrel the alpha helices and beta strands commonly 8 of each form a solenoidal shape pattern that wind to close on its own an annulus shape, which is topologically recognized as a toroid.

The parallel β-strands forms the inner side of the annulus (hence, a β-barrel) and the α-helices constitute the outer wall of the toroid. Each β-strand are connect to the very next neighboring strand of the barrel with the help of a right-handed loop which includes one helices, in turn ribbon of N-to-C coloring at the top view proceeds in rainbow order all over the barrel. The fold can consider as solenoidal shape since it is made up of eight imbricate, right-handed super secondary structure (Xiaoyan Yang *et a*l., 2009) protein structures β-α-β.

More than 80% of TIM barrel in clusters share exactly the same catalytic function. To quicken the expansion of the sequence-structure protein in TIM-barrel protein motif, computationally devised tools which permit tender spotting of TIM-barrel proteins.

In order to provide the detailed information about the protein sequence, TIMPRED is designed in such a way that it predicts the fold in the structure and in future it can also predicts super family containing sequence. Super family will give user an idea about the evolutionary relationship with the other protein sequences.

The eightfold (beta-alpha) toroid barrel structure, first abided in enzyme named triosephosphateisomerase, happens ubiquitously in characteristics. It is actually an enzyme and that basically occurs in molecular or energy metabolism process within the cell.

In this project we tried to attempt the prediction of TIM-barrel fold in a protein sequences data by the help of binary pattern and PSSM profiles of 2024 TIM barrel and non-TIM barrel of 2,232 non-redundant protein sequence chains. At the first step TIM-barrel fold was analyzed, and then SVM model files were developed using binary pattern of TIM-barrel.

It has been found earlier that evolutionary data provide much more relevant information; hence SVM model will develop with the help of PSSM profiles which will be generated from PSI-BLAST. We tried to develop three models and evaluate using five-fold cross validation technique. TIMPRED can directly predict the TIM-barrel fold using binary pattern of sequence and its evolutionary information. Our server can be utile in experimental biologist working on different enzyme function and protein engineering.

LITERATURE REVIEW

Proteins have complex three dimensional shapes, a fact well explained by above 60,000 experiments which were done to study protein structures and submitted in the one of the most known database Protein Data Bank (PDB). The total numbers of singular protein folds in terms of their construction types are much less as compare to the number of protein families which are outlined by sequence similarity (Gregory K.Farber *et al.*,1994). As people are studying more and more structures and getting submit in PDB it clears that the division of proteins in their different folds are not even.

Although various folds studied upto now and so far very few have been observed and reported only in few proteins, out of them some protein folds are known as super folds which occur very frequently. As reported at (Salem *et al.,* 1999) the top ten super folds are responsible for about one third of all proteins which are present in the PDB (Hegyi, H. *et al.*,1999).

The eight fold (b/a) barrel structure (Deirdre Reardon *et al,*. 1995), first abided in triose-phosphate isomerase, occurs ubiquitously type. It is approximately invariably an enzyme and most frequently occur in molecular or energy metabolism process that occurs inside the cell. In this work we tried get together the data of the protein sequences, function to determine and study the fold.

We high light the sequences and functional diversity in 21 homologous super families, which include 76 different sequence families (Dayhoff, *et al.,* 1975). In many structures fold and barrels are mixed together and coupled with the other domains which in turn lead to additional variety.

Local and global structure-based alignments help to understand the division of the associated functional residues on the common structural arrangements. Various co-factors or substrates consist phosphate moiety, generally which is bind towards the C-terminal end of the sequence. Some of these structures exhibit a conserved region such as "phosphate binding motif".

And the metal-ligating residues and catalytic residues are distributed, on the sequences. We also got the some spectacular structural superposition of these residues. Another than these we also look at the possibility of evolutionary relationships in these proteins whereas sequences are at so distantly related that most potent approaches are able to get few relationships, yet their active sites get cluster at their one of the barrel.

This is one the extreme example of the "one fold- many functions" (Nagano N *et al.*, 2002) epitope which exemplify the trouble of allotting functions by a structural genomics method for some folds.

Nearly 10% of enzyme known so far have TIM barrel domain. However, their sequence similarity is not high enough to detect their evolutional relationship and structural similarity (Vijayabaskar MS *et al*. 2012). TIM barrel fold is also one of the most functionally versatile folds. The active sites of TIM barrel always come at the end of C-terminal barrel composed of inside beta-sheets.

This suggests TIM barrel have evolved from ancestral TIM barrel with divergent evolution (R.R Copley *et al.,* 2000). The functional diversity of TIM barrel comes from the fact that the TIM (Albery*et al*., 1976) barrel function can be changed by slight variation around from ancestral TIM barrel with divergent evolution.

The functional diversity of TIM barrel comes from the fact that the TIM barrel function can be changed by slight variation around the active site, loops connecting C-terminal inside beta –sheets and outside alpha helices.

Function variation of the TIM barrel is divided into two categories, substrate specificity and catalytic activity. Variation at inside barrel changes substrate specificity of the TIM- barrels. On the other hand, variation around active sites changes substrate catalytic activity.

These TIM barrel's two ways of function creation made it an efficient fold to manipulate new enzyme activity. One fold many function character of the TIM barrel makes it difficult to assign function based on structure. Just recognizing overall structure of TIM barrel is not enough to understand relationship between structures of TIM barrel domains.

Many proteins structural databases such as SCOP and CATH classified TIM barrels according to their structures. However, TIM barrel classification by these databases does not detect specific differences among TIM barrel domains with different functions since TIM barrel domains in each family from the database have diverse function, not unique function.

One of the top ten super folds is the triosephosphate isomerase (TIM)-0 barrel fold. It was first conserved in triosephosphate isomerase and consists of eight alpha- helices on the outside and eight parallel beta-strands inside the alternate structure with the peptide backbone. Earlier, numbers of protein structures are reported which consist the TIM- barrel fold, this allow complete interpretation of the fold space of the TIM-barrel.

In the SCOP database, the TIM-barrel fold contains 33 super families and 101 families. As a common fold with the multiple functions, a TIM-barrel protein often performs enzymatic reactions. These folds with different arrangements can perform five out of the six categories of chemical reactions. On evolution of the fold (Alexey G.*et al.,* 1995) it also got the significant attention, and it is found and explained that the TIM-barrel fold is one of the most ancestral folds.

3.1 Composition

The TIM-barrel domain by itself has typically about 250 residues, but it can be as small as 200 residues (Traut T. *et al.,* 2000) . TIM-barrel enzymes can be small (only one domain), such as hevamine (monomeric, 273 residues), or very large, such as β-galactosidase (tetrameric, 1023 residues per polypeptide chain (D.H.J Juers *et al.,*1999), consisting of five domains). In β-galactosidase the catalytic TIM-barrel domain is in the middle (domain 3, residues 335–624), whereas domains 1 and 2 are located at the N-terminus and domains 4 and 5 are at the C-terminus.

TIM barrels fold are considered of α/β protein folds since they involves an alternating design of alpha-helices and beta-strands which forms a single domain (Xiaoyan Yang *et al.*, 2009). Alpha helices and beta strands usually are in eight numbers together forms coil like structure which forms doughnut like structure by curves around coil and named as toroid topologically. The parallel beta-strand forms the inner wall of the doughnut (hence beta sheet) whereas the alpha-helices form the exterior side of the structure.

In structure beta-strands connect to the it's very next strand in the barrel, with the help of long right-handed loop which includes it's one of the helices, in turn ribbon N-to-C, colored in the top view runs to rainbow arrangement around the barrel. Figure-1 shows the TIM-barrel fold.

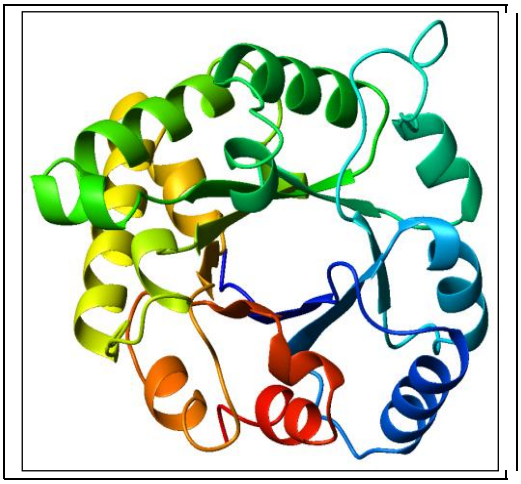


Figure 1.1: Triosephosphateisomerase (TIM) barrel (PDB accession code 8TIM), colored from blue (N-terminus) to red (C-terminus).

In the structure its core is tightly packed by the large residues which are hydrophobic in nature but few glycine amino acid still permit to joggle to form compact core. While packing of the core other hydrophobic interaction which is dominating other residues such as branched aliphatic residues leucine, isoleucine and valine also exist and forms 40% in the beta strands.

The TIM-barrel fold has intrigued by various scientists for last few years basically for two reasons:

1. One is α/β barrel takes part in catalysis of large number of reactions, these fold carrying enzymes or reactions generally majorly act as target of various drug designing and protein engineering methods.

2. Secondly don in order to untangle its evolutionary history. Due to lack of significant knowledge about its sequence homology with other protein members of this family cause difficult to study and perform evolutionary analysis.

   This fact, combines with the geometric arguments concerning the barrel structure (Saha, *et al.*, 2006), has lead to suggestions that the proteins are related by convergent evolution to stable fold. Meanwhile, large number of researches has been done and still going on in this field still solid concrete are lacking.

   Answers to these questions can have confederative benefits in protein designing field. As well as protein designer can understand the causes for different sequences acquiring the same fold and thus help in designing a de-novo sequence that can fold into α/β barrel.

3.2 Loop Regions

Approximately 200 residues are needed to form a TIM fold. Nearly 160 residues are believed to be structurally divided equally at different proteins dealing this fold (Ochoa-Leyva, A. *et al.*. 2009). The left over residues are placed on loop site which linked the alpha loops and beta sheets, C-terminal loops end of the sheets incorporate the active site, that's why this fold is quite common. These amino acid residues demanded to keep the structures and the residues which involve in the catalysis for the mostly for distinct subsets.

It has been recently shown that catalytic loops can also be substituted at different TIM barrel enzymes as semiautonomous units of functional groups.

3.3 Triose-phosphatase isomerase

Triose-phosphatase isomerase (TPI or TIM), act as an enzyme which perform catalysis of reversible inter conversion of the triose phosphatase isomers dihydroxyacetone phosphate to its isomers dihydroxyacetone phosphate and D-glyceraldehyde phosphate.

TPI perform significant function in glycolysis pathway (Zaffagnini M *et al.*,January 2014) and it is necessary for release of effective production of energy. TPI occurs almost in all organisms which include animals such as mammals, insects, fungi, plants and bacteria. However, those bacteria do not perform glycolysis, like urea plasmas, lack TPI.

3.4 Families and super families

A class contains a set which shares common properties. The term class for protein is used to refer a group of protein sequences (Dayhoff M.O *et al.*, 1974) which possess common attributes. A set gets zoned when it is distinguished into subsets where each element belongs to a member of some subset and no element belongs to more than one subset.

In protein family represents a protein class which is marked to demonstrate extent or threshold of any relationship. The relationship permits the set of all protein sequences or subsequences to be

partitioned and each family must be closed under transitivity. A family may contain single member. A protein super family is a protein class composed of one or more protein families; a superfamily is the union of its constituent families.

The set of all super families must constitute a partitioning of the set of all protein sequences or subsequences under the relationship that defines the protein families and superfamily must be closed under transitivity.

Proteins in a family derive from their common ascendant and possess alike three dimensional structures, functions and significant sequences similarity. Although evaluation of the functional significance and/or similarity in sequence is difficult, there is a reasonable and well developed model to form for evaluating the significance of similarity, there is quite well developed framework for evaluation of the significance of similarity between a group of sequences using sequences alignment methods.

Proteins that do not share a common ancestor are unlikely to show statistically significant sequences similarity, making sequences alignment a powerful tool for identifying the members of protein families.

Tough 60,000 protein families have been defined the ambiguity in the definition of a protein family continues.7

3.5 Evolutions of protein families

The pairwise amino acid sequence similarity of TIM-barrel folds in non-homologous enzyme families is generally below the detectable level. Nevertheless, structure-based sequence alignments reveal the presence of physicochemically similar clusters of residues, which are observed to exist at equivalent topological positions and which therefore could direct, stabilise and determine the common TIM-barrel folding pattern (Selvraj. S. *et al.*,1998). Indeed, circular permutated sequence variants of a TIM-barrel enzyme fold as the wild-type; in these variants the wild-type N- and C-termini have been joined but discontinuities were made in βα loop 6 and αβ loop 6 respectively.

According to current dogma, protein families arise (Mishra, N.K. *et al.,* 2010) in two ways. Firstly, the separations of parent taxon genetically into disjunct related species, which allow gene/protein independently get cumulate and bring variance in the descents. As a outcome these variations or mutations orthologous proteins families form which contain conserved or same sequence motifs.

Another way is gene duplication which may create a second copy (called paralogs). In this ancestral gene retain its own functionality. In this duplicated gene get deviates freely and can take new functions as result of mutations.

Some protein families, eukaryotes particularly, experience utmost diversion and compressions during the evolution sometimes the complete genome gets duplicate (Ochoa-Leyva, A. *et al.* 2009).This is one of the prominent characteristics of evolution,

As given in the SCOP database, 33 super families exist of TIM-barrel protein fold.

Table 1.1: TIM-Barrel protein Super families

Family	**Family name (abbreviation)**	**Number of sequence families**	**No. of sequences (UNIPROT)**
1.	Triosephosphate isomerase (TIM)	1	6,743
2.	Ribulose-phosphate binding barrel	6	34,543
3.	Thiamine phosphate synthase	1	5,374
4.	Pyridoxine 5'-phosphate synthase	1	1,818
5.	FMN-linked oxidoreductases	1	34,066
6.	Inosine monophosphate Dehydrogenase (IMPDH)	1	12,781
7.	PLP-binding barrel	2	21,394
8.	NAD (P)-linked oxidoreductase	1	25,512
9.	(Trans) glycosidases	14	105,361
10.	Metallo-dependent hydrolases	18	53,888
11.	Aldolase	8	62,168
12.	Enolase C-terminal domain-like	2	15,949
13.	Phosphoenolpyruvate/ pyruvate domain	7	31,757
14.	Malate synthase G	1	2,513
15.	ruBisco, C-terminal domain	1	49,525
16.	Xylose isomerase-like	7	19,393
17.	Bacterial luciferase-like	4	13,510

18.	Nicotinate/QuinolinatePRTase C-termianl domain-like	2	7,534
19.	PLC-like phophodiesterases	3	12,719
20.	Cobalamin (vitamin B12) -dependent enzymes	4	3,347
21.	tRNA-guanine transglycolyase	1	4,622
22.	Dihydropteroatesynthetase-like	2	9,435
23.	UROD/MetE-like	2	10,919
24.	FAD-linked oxidoredutase	2	6,901
25.	Monomethylaminemethyltranseferase (MtmB)	1	22
26.	Homocysteine S-methyltransferase	1	4,098
27.	(2r)-phospho-3-sulfolactate synthase ComA	1	215
28.	Radical SAM enzymes	3	63,647
29.	GlpP-like	1	902
30.	CutC-like	1	2,012
31.	ThiG-like	1	2,403
32.	TM1631-like	2	2,183
33.	EAL domain like	1	24,002

3.6 SVM (Support Vector Machine)

SVM's (Support Vector Machines) were first acquainted by Vladimir Vapnik and colleagues. In the beginning it was first mention by Vapnik, 1979, but the first recognized paper was published by Vapnik in, 1995.

SVM (Support Vector Machine) are comparatively newly learning method practiced for binary classification (Chih-Wei Hsu *et al.*, 2003 and Chih-Jen *et al.*, 2010). The canonic idea to get a hyper plane, is to distinguish the d-dimensional data absolutely into two classes on the basis of similar characteristics.

SVM's introduce the impression of "kernel get feature space" (Byvatov E *et al.*, 2004) which draws the data in higher dimensional space in which data is distinguishable. Usually, drawing into higher dimension space would create problem of overfitting. The only way to deal with this problem is used in SVM's is to use dot product in that space and do not deal directly the higher dimensional space.

SVM's can also be used to calculate denotatively, unlike other learning ways, such as neural networks, for which there is no measures. In general SVM's are non-rational, theoretically tenable and proved successful (Debasish Basak *et al.*,2007). SVM's have broadened to use and solve and workout regression problem where system or machine is trained to give result a numerical value unlike the classification problem where solutions are in terms of yes or no.

Support vector machine is the supervised method of learning models with associated learning algorithms (Ramana, J.*et al.*, 2010) that examine the information and distinguish on the basis of patterns and used in classifying data as well as analysis by regression. The SVM takes a set of information or data as input, on the basis of predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. A set of training as examples data has given in which data is labelled that indicates the categories of data out of two. On basis of SVM algorithm model get builds this assigns new category and predicts the test set.

Classification job (Furey T *et al.*,2000) generally requires separation of the data into 2 sets namely training and testing sets. Each instance of the training set consists of one "target value" called class labels and various "attributes" which is features or remarked variables. The main aim of SVM's is to generate a model which is constructed on basis of training data, to predict the target values of the test data which contains only test data attributes.

A SVM model symbolizes the training data in form of dots in blank space, these dots get mapped and divide the data and are categorized by drawing a clear gap as wide as possible. Now test or example data get map in same space and forecast the class to which they belong. SVM In addition to performing linear classification (Garg, A. *et al.*, 2008), SVM using kernel effectively performs a non-linear classification and implicitly maps their inputs data into high-dimensional characteristics spaces.

For classification, regression support vector machine builds a hyperplane or set of hyperplanes in infinite-dimensional space. Non-rationally these hyperplanes attains a good separation. This separation lies between the dot at the largest distance and nearest training data point on the hyperplane of class (so-called functional margin), generally the larger the margin the lesser the error of the classifier. The original trouble occur when these hyperplanes fails to separate the data by using linear separation method the space.

For this kind of jobs it is suggested to use multilayer classification in finite dimensional layer (Kaur H *et al.*,2003). Where space get map often at higher-dimensional space, surmise makes separation easier in space.

In order to decrease the load of computer, the marking done by SVM, for this outline is design in such a way that it assure computationally dot products easily get computed in their original finite space by using kernel function. These kernel function get selected according to the problem

The hyperplanes can be defined as set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters of images of feature vectors that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation:

Note that if K(x,y) becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point xi. In this way, the sum of kernels (Jeong E *et al.*, 2006) above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space.

3.7 Support Vector Classification

Classification is basically confined to the two-class problem where data can be separate into yes and no, without losing its generality. In classification kind of jobs the main aim is to sort out the data in two classes (Kumar M. *et al.,* 2008) by a function which is derived from the model of example data. This classifier model act efficiently for unseen examples, i.e. it generalizes well

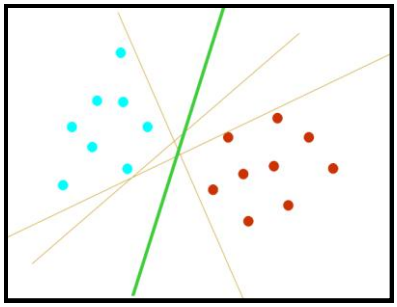


Figure 1.2 SVM classifications of data. Here data is blue and red color dots which is separated by linear

In figure there are numerous potential linear classifiers to distinguish the data, but there is one hyperplane out of these all hyperplanes which classifies the data at its best and maximizes the margin

(maximizes the distance between it and the nearest data point of each class). This linear classifier is known as the optimal separating hyperplane. As a result it is assumed that boundary which generalize as well as opposed to the other possible boundaries.

Binary classifiers SVM are those which can separate data points in to two categories. Each data point is made up by a n-dimensional vector. All the data points belong to one of the class out of two and linear classifier distinguishes them with the help of hyperplane.

In fig 1.2 two group of data which is divided by hyperplanes in two-dimensional space. Like this many linear classifiers accurately classify the two data into two groups like as L1, L2 and L3 given in figure. To attain maximum division in two classes, SVM choose the hyperplane which possess the largest margin (Stephen Winters-Hilt et al., 2006). The margin is basically add-on of the shortest distance from dividing hyperplane to data point which lying to its nearest distance in both groups. These hyperplanes represent better generalization i.e. these hyperplane accurately classifies the unobserved or testing data.

SVM perform mapping of input space to feature space to affirm nonlinear classification jobs.one of its feature i.e. kernel trick is which used to permit absence of specific formulation of mapping function which cause the consequence of dimensionality. This made the new space for linear classification which is equivalent to the original space of nonlinear classification.

SVMs perform these by mapping input vectors to its higher dimensional space where hyperplane is constructed.

SVM also allows nonlinear classification. Non-linear classifier permits error ξi. This is depend upon outturn result of the function **wTx+b**. the error symbolizes the number those samples which are not sorted or classified.

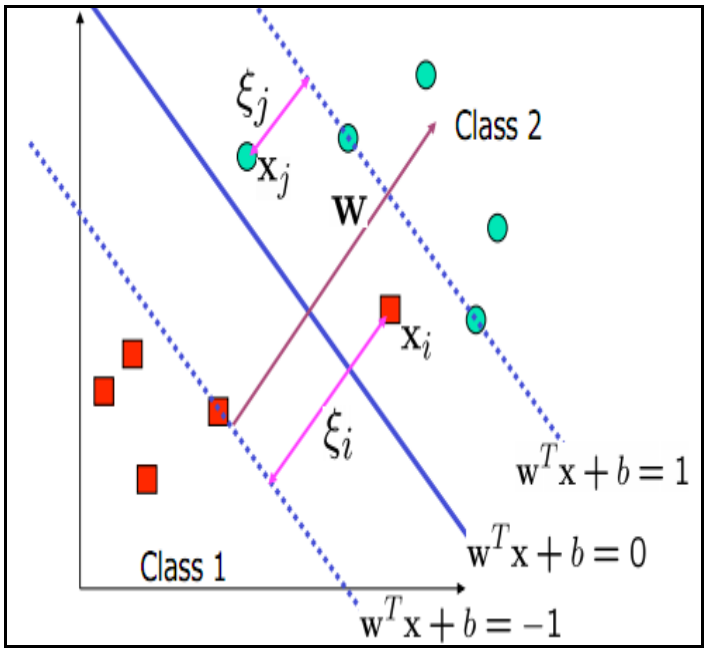


Figure 1.3 : SVM non-linear classifier, in this ξi represent the error which causes due to misclassification of data.

In this kind of job SVM perform transformation form non-linear to linear with help of XOR discrimination and Kernel tricks. Kernel tricks refer to the use of kernel functions which involve the usage of eigen values but not explicitly. Kernel function (A. Smola *et al.*, 1996) is inner product of the quadratic equation and basically measures similarity between the objects.

SVM also possess flexibility while classification of data in terms of parameters and this work is validated by VC dimension (Christopher J.C. Burger *et al.*,1998). To achieve the complete classification on training data VC-dimension of closest neighbour samples. More the vc dimension value higher flexibility will occur.

METHODOLOGY

4.1 Software and hardware requirements

- Operating system: Windows or Linux (Ubuntu version 11.10 or higher)
- Web browser: Microsoft Internet Explorer 7.0 or higher or Firefox 3.0 higher

4.2 System Configuration

The configuration on system on which the TIMPRED will develop:

- Operating System : Linux (Ubuntu version 11.10)
- Web Servlet (Servlet Conatiner) : Apache2

4.3 Server

- GHz Quad core
- 2 GB RAM
- 25 GB root partition for the system
- 1 GB standard swap partition
- 4GB additional swap partition
- 100 GB data storage partition for documents and indexing

4.4 Web Browser

- Microsoft Internet Explorer 7.0 or higher, with following settings required.
- Scripting must be activated
- Firefox 3.0 or higher.

4.5 Dataset selection

In the first step all the sequences were extracted from the PDB with 'Tim-Barrel' keyword and SCOP classification as TIM beta/alpha barrel. Total 7632 Tim barrel protein sequences are available on Protein Data Bank (www.rcsb.org/pdb).

For the negative training set, all representative sequences with the SCOP classification as all alpha and all beta proteins (a/b) were chosen from PDB. For this, around 2442 sequences except TIM beta/alpha- barrel fold have been downloaded from PDB.

4.6 Removing redundancy

Second step is to remove the redundancy found in the dataset obtained from PDB. For this we used CD-HIT suite program which is online program.

CD-HIT stands for cluster database at High Identity with Tolerance, the program (CD-HIT) it takes a fasta format sequences database as input and produces a set of non-redundant (nr) representative sequences as output.

In addition CD-HIT outputs a cluster file (Terribilini M *et al.,* 2006), documenting the sequence group is for each nr sequence representative. The idea is to reduce the overall size of the database without removing the sequence representative and to reduce the overall size of the database without removing the sequence information by only removing redundant sequences. This is the resulting dataset called non-redundant (nr).

In this open the CD-HIT suite upload fasta format sequences file and set the redundancy level as 30%. This 30% implies that at max only 30% percent sequences can redundant in the result.

From our data initially total positive set number is 7632 and after CD-Hit number is 2032.

Likewise, repeat the same process for negative set initially number of sequences are 9643 and after the CD-HIT with 30% similarity number is 2442.

4.6 Feature Extraction:

After this third step is feature selection process.

The downloaded sequences were converted into appropriate format using perl scripts and three main features: Amino acid composition, Dipeptide composition and PSSM profiles (Kumar M*et al.* 2008, Bhasin M *et.a.,* 2005) were selected for machine learning to predict TIM barrel fold. Features extraction was again done by using perl script (Figure 1.3, 1.5).

4.7 Five Fold cross Validation:

Fourth step is the five-fold cross validation (Sachdeva G*et al.*, 2005) to evaluate the performance of all the models which are develop from this study. The five fold cross validation was done by using following command

svmtrain –s 0 –c 500 –g 0.1 –v 5 data_file_name

This command perform five-fold cross validation for classifier using the parameters c = 500 and gamma = 0.1.

4.8 SVM and SVM^{light}

First pioneered by Vapnik in 1995, SVM is a supervised machine learning method which delivers state of the heart performance in recognition and discrimination of cryptic patterns in complex datasets (C.J.C. Burges *et al.*,1998). SVM is used in conjunction with kernel (Kumar M *et al.*, 2008) functions which will implicitly map input data to high dimensional non-linear features space (T. Joachims *et al.*, 1999).

4.9 Construction of TIMPRED learning model by using SVM light by using all the three features.

TIMPRED webserver will be constructed using PHP and HTML and these SVM models will be used for prediction.

4.10 **Installation:**

i. To install SVM^{light} first it is required to download the file svm_light.tar.gz from http://svmlight.joachims.org/.(Joachims,T. *et al.*, 1999 website)

ii. Now open a command prompt create a new directory named svm_light by using the command

mkdir svm_light

and move the downloaded folder into the new directory.

iii. Now move into the new directory and unzip downloaded svm_light folder by using command

gunzip –c svm_light.tar.gz | tar xvf.

iv. Once the file get unpack and all the folders get extracted execute the command

make or make all

v. Now this will command compile all the files and create two exe files executable files named :

svm_learn (learning module)

svm_classify (classification module)

4.11 **How to Run/Use:**

SVM^{light} consist of two executables files named learning module (svm_learn) and second classification module (svm_classify).

Svm_learn:

Svm_learn command is one of the very important commands. For generation of the model files and make the machine to learn the data in particular format.

- Svm_learn command only execute if the data is presented in the required format, which is:
- Input format +1 1:0.5 3:1 5:0.4,

  -1 2:0.9 3:0.1 4:2
- Which is <label> +1/-1 <index1>:<value1> <index2>:<value2>…..
- Format involves each line contains an instance which is ended by '\n' character. For classification <label> is an integer to indicate the class label. In the project classification is done. For positive set +1 was used as a label and for negative set -1 was used.
- Pair of <index>:<value> refers to feature (attribute) value. Whereas <index> represents integer starting from 1 and <value> is a real number.
- Feature selections in the project were amino acid count, dipeptide count and pssm profiles. Since data was too large specific perl scripts (Randal Schwartz *et al.*,1997, Norman Matloff *et al.*, 2007) were used to convert the data into the prescribed format of svmlight. File name amino_acid_count.pl is used to count the total 20 different type of amino acids

```
#!/usr/bin/perl -w
$proteinfilename = 'sequence1';
#!/usr/bin/perl -w
$proteinfilename = 'sequence1';
#!/usr/bin/perl -w
$proteinfilename = 'sequence1';
open (FH, "positive_set.fasta");
open (FA, ">>test");
while ($seq=<FH>)

{$seq1=$seq1.$seq;}

@seq=split (/>/,"$seq1");
$size= @seq;
for ($i=1;$i<$size;$i++)
{
@test=grep{$_!~/^>/}@seq[$i];
$protein = join( '', @test);

@protein1 = split('',$protein);
```

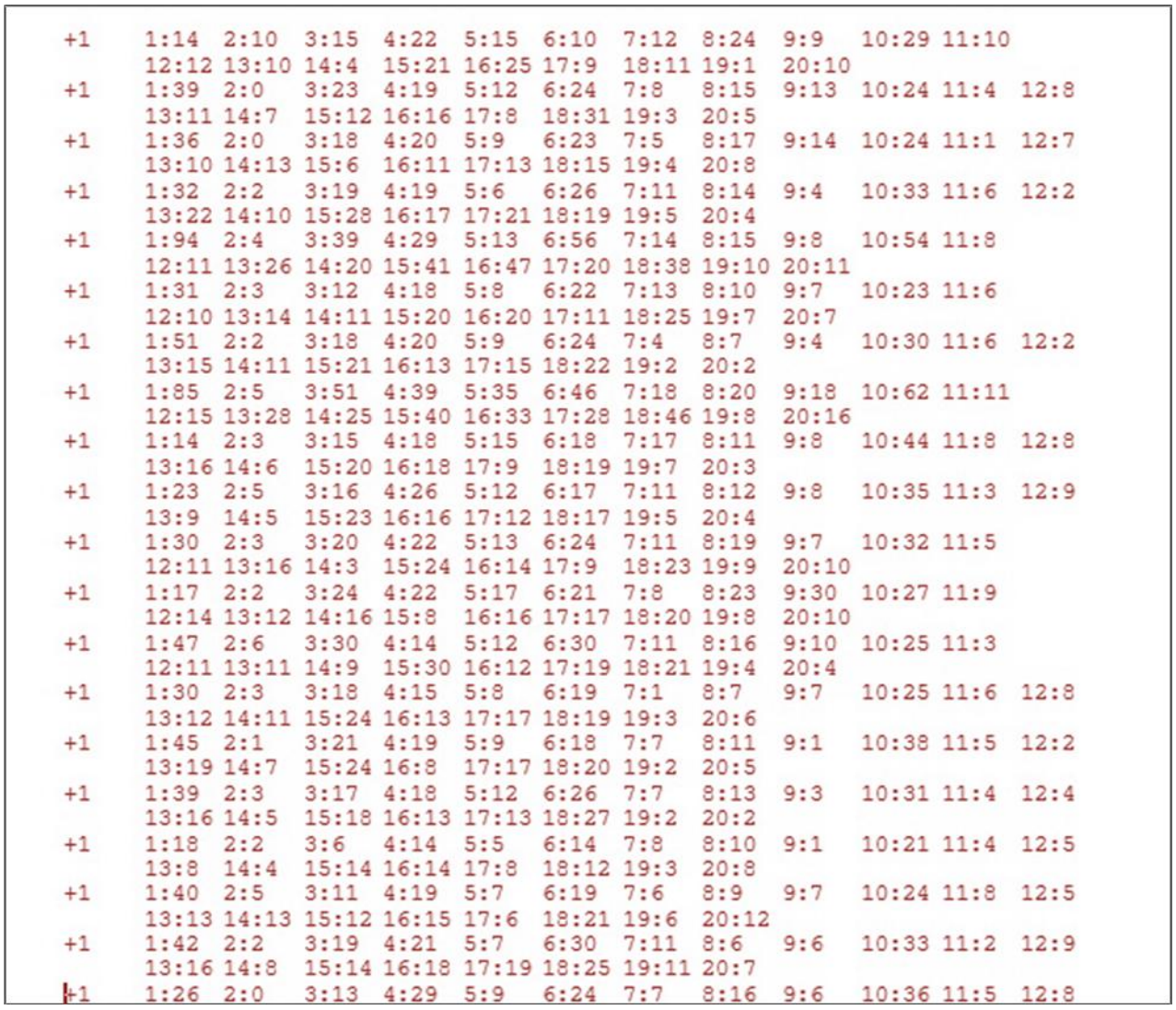
Figure 1.4: Amino_acid_count.pl perl script

```
$A = 0;
$C = 0;
$D = 0;
$E = 0;
$F = 0;
$G = 0;
$H = 0;
$I = 0;
$K = 0;
$L = 0;
$M = 0;
$N = 0;
$P = 0;
$Q = 0;
$R = 0;
$S = 0;
$T = 0;
$V = 0;
$W = 0;
$Y = 0;
$B = 0;
$J = 0;
$O = 0;
$U= 0;
$X = 0;
$Z = 0;
foreach $aa (@protein1)
{
if ( $aa eq 'A')
                {
                  ++$A;
                }
        elsif ( $aa eq'C')
                {
                  ++$C;
                }

        elsif ( $aa eq'D')
                {
                  ++$D;
                }

        elsif ( $aa eq'E')
                {
                  ++$E;
                }
        elsif ( $aa eq'F')
```

```
elsif ( $aa eq'G')
        {
          ++$G;
        }


elsif ( $aa eq'H')
        {
          ++$H;
        }

elsif ( $aa eq'I')
        {
          ++$I;
        }

elsif ( $aa eq'K')
        {
          ++$K;
        }

elsif ( $aa eq'L')
        {
          ++$L;
        }
elsif ( $aa eq'M')
        {
          ++$M;
        }
elsif ( $aa eq'N')
        {
          ++$N;
        }

elsif ( $aa eq'P')
        {
          ++$P;
        }
elsif ( $aa eq'Q')
        {
          ++$Q;
        }
```

```
        elsif ( $aa eq'S')
                {
                 ++$S;
                }

        elsif ( $aa eq'T')
                {
                 ++$T;
                }

        elsif ( $aa eq'V')
                {
                 ++$V;
                }

        elsif ( $aa eq'W')
                {
                 ++$W;
                }

elsif ( $aa eq'B')
                {
                 ++$B;
                }
        elsif ( $aa eq'J')
                {
                 ++$J;
                }
        elsif ( $aa eq'O')
                {
                 ++$O;
                }
        elsif ( $aa eq'U')
                {
                 ++$U;
                }
```

```
        elsif ( $aa eq'X')
                {
                  ++$X;
                }
        elsif ( $aa eq'Z')
                {
                  ++$Z;
                }
        else
        {
          #print "!!!!!!!! Error - I don\'t recognize this aminoacid: $aa\n";
                #++$errors;
        }
}
print FA "-1"."\t"."1:". $A."\t". "2:".$C."\t". "3:".$D."\t"."4:".$E."\t"."5:".$F."\t". "6:".$G."\t".
"7:".$H."\t"."8:".$I."\t"."9:".$K."\t"."10:".$L."\t"."11:".$M."\t"."12:".$N."\t"."13:".$P."\t"."14:".$Q."\t
"."15:".$R."\t"."16:".$S."\t"."17:".$T."\t"."18:".$V."\t". "19:".$W."\t"."20:".$Y."\n";
}
```



```
+1    1:14  2:10  3:15  4:22  5:15  6:10  7:12  8:24  9:9   10:29 11:10
      12:12 13:10 14:4  15:21 16:25 17:9  18:11 19:1  20:10
+1    1:39  2:0   3:23  4:19  5:12  6:24  7:8   8:15  9:13  10:24 11:4  12:8
      13:11 14:7  15:12 16:16 17:8  18:31 19:3  20:5
+1    1:36  2:0   3:18  4:20  5:9   6:23  7:5   8:17  9:14  10:24 11:1  12:7
      13:10 14:13 15:6  16:11 17:13 18:15 19:4  20:8
+1    1:32  2:2   3:19  4:19  5:6   6:26  7:11  8:14  9:4   10:33 11:6  12:2
      13:22 14:10 15:28 16:17 17:21 18:19 19:5  20:4
+1    1:94  2:4   3:39  4:29  5:13  6:56  7:14  8:15  9:8   10:54 11:8
      12:11 13:26 14:20 15:41 16:47 17:20 18:38 19:10 20:11
+1    1:31  2:3   3:12  4:18  5:8   6:22  7:13  8:10  9:7   10:23 11:6
      12:10 13:14 14:11 15:20 16:20 17:11 18:25 19:7  20:7
+1    1:51  2:2   3:18  4:20  5:9   6:24  7:4   8:7   9:4   10:30 11:6  12:2
      13:15 14:11 15:21 16:13 17:15 18:22 19:2  20:2
+1    1:85  2:5   3:51  4:39  5:35  6:46  7:18  8:20  9:18  10:62 11:11
      12:15 13:28 14:25 15:40 16:33 17:28 18:46 19:8  20:16
+1    1:14  2:3   3:15  4:18  5:15  6:18  7:17  8:11  9:8   10:44 11:8  12:8
      13:16 14:6  15:20 16:18 17:9  18:19 19:7  20:3
+1    1:23  2:5   3:16  4:26  5:12  6:17  7:11  8:12  9:8   10:35 11:3  12:9
      13:9  14:5  15:23 16:16 17:12 18:17 19:5  20:4
+1    1:30  2:3   3:20  4:22  5:13  6:24  7:11  8:19  9:7   10:32 11:5
      12:11 13:16 14:3  15:24 16:14 17:9  18:23 19:9  20:10
+1    1:17  2:2   3:24  4:22  5:17  6:21  7:8   8:23  9:30  10:27 11:9
      12:14 13:12 14:16 15:8  16:16 17:17 18:20 19:8  20:10
+1    1:47  2:6   3:30  4:14  5:12  6:30  7:11  8:16  9:10  10:25 11:3
      12:11 13:11 14:9  15:30 16:12 17:19 18:21 19:4  20:4
+1    1:30  2:3   3:18  4:15  5:8   6:19  7:1   8:7   9:7   10:25 11:6  12:8
      13:12 14:11 15:24 16:13 17:17 18:19 19:3  20:6
+1    1:45  2:1   3:21  4:19  5:9   6:18  7:7   8:11  9:1   10:38 11:5  12:2
      13:19 14:7  15:24 16:8  17:17 18:20 19:2  20:5
+1    1:39  2:3   3:17  4:18  5:12  6:26  7:7   8:13  9:3   10:31 11:4  12:4
      13:16 14:5  15:18 16:13 17:13 18:27 19:2  20:2
+1    1:18  2:2   3:6   4:14  5:5   6:14  7:8   8:10  9:1   10:21 11:4  12:5
      13:8  14:4  15:14 16:14 17:8  18:12 19:3  20:8
+1    1:40  2:5   3:11  4:19  5:7   6:19  7:6   8:9   9:7   10:24 11:8  12:5
      13:13 14:13 15:12 16:15 17:6  18:21 19:6  20:12
+1    1:42  2:2   3:19  4:21  5:7   6:30  7:11  8:6   9:6   10:33 11:2  12:9
      13:16 14:8  15:14 16:18 17:19 18:25 19:11 20:7
+1    1:26  2:0   3:13  4:29  5:9   6:24  7:7   8:16  9:6   10:36 11:5  12:8
```

Figure 1.5 : Training file generated by amino acid count feature extraction perl script.

```
#!/usr/bin/perl

use strict;
use warnings;

my $file = $ARGV[0];
open(FILE,$file);

open OUT, ">outfiledipep";


while (my $sequence = <FILE>){
chomp $sequence;

if($sequence=~m/[A-Z]/)
{
my$AA=($sequence=~tr/AA//);my$AC=($sequence=~tr/AC//);my$AD=($sequence=~tr/AD//);my$AE=($
sequence=~tr/AE//);my$AF=($sequence=~tr/AF//);my$AG=($sequence=~tr/AG//);my$AH=($sequence=~
tr/AH//);my$AI=($sequence=~tr/AI//);my$AK=($sequence=~tr/AK//);my$AL=($sequence=~tr/AL//);my$
AM=($sequence=~tr/AM//);my$AN=($sequence=~tr/AN//);my$AP=($sequence=~tr/AP//);my$AQ=($seq
uence=~tr/AQ//);my$AR=($sequence=~tr/AR//);my$AS=($sequence=~tr/AS//);my$AT=($sequence=~tr/A
T//);my$AV=($sequence=~tr/AV//);my$AW=($sequence=~tr/AW//);my$AY=($sequence=~tr/AY//);my$
CA=($sequence=~tr/CA//);my$CC=($sequence=~tr/CC//);my$CD=($sequence=~tr/CD//);my$CE=($seque
nce=~tr/CE//);my$CF=($sequence=~tr/CF//);my$CG=($sequence=~tr/CG//);my$CH=($sequence=~tr/CH/
/);my$CI=($sequence=~tr/CI//);my$CK=($sequence=~tr/CK//);my$CL=($sequence=~tr/CL//);my$CM=($
sequence=~tr/CM//);my$CN=($sequence=~tr/CN//);my$CP=($sequence=~tr/CP//);my$CQ=($sequence=~
tr/CQ//);my$CR=($sequence=~tr/CR//);my$CS=($sequence=~tr/CS//);my$CT=($sequence=~tr/CT//);my$
CV=($sequence=~tr/CV//);my$CW=($sequence=~tr/CW//);my$CY=($sequence=~tr/CY//);my$DA=($seq
uence=~tr/DA//);my$DC=($sequence=~tr/DC//);my$DD=($sequence=~tr/DD//);my$DE=($sequence=~tr/
DE//);my$DF=($sequence=~tr/DF//);my$DG=($sequence=~tr/DG//);my$DH=($sequence=~tr/DH//);my$
DI=($sequence=~tr/DI//);my$DK=($sequence=~tr/DK//);my$DL=($sequence=~tr/DL//);my$DM=($seque
nce=~tr/DM//);my$DN=($sequence=~tr/DN//);my$DP=($sequence=~tr/DP//);my$DQ=($sequence=~tr/D
Q//);my$DR=($sequence=~tr/DR//);my$DS=($sequence=~tr/DS//);my$DT=($sequence=~tr/DT//);my$DV
=($sequence=~tr/DV//);my$DW=($sequence=~tr/DW//);my$DY=($sequence=~tr/DY//);my$EA=($sequen
ce=~tr/EA//);my$EC=($sequence=~tr/EC//);my$ED=($sequence=~tr/ED//);my$EE=($sequence=~tr/EE//);
my$EF=($sequence=~tr/EF//);my$EG=($sequence=~tr/EG//);my$EH=($sequence=~tr/EH//);my$EI=($seq
uence=~tr/EI//);my$EK=($sequence=~tr/EK//);my$EL=($sequence=~tr/EL//);my$EM=($sequence=~tr/E
M//);my$EN=($sequence=~tr/EN//);my$EP=($sequence=~tr/EP//);my$EQ=($sequence=~tr/EQ//);my$ER
=($sequence=~tr/ER//);my$ES=($sequence=~tr/ES//);my$ET=($sequence=~tr/ET//);my$EV=($sequence=
~tr/EV//);my$EW=($sequence=~tr/EW//);my$EY=($sequence=~tr/EY//);my$FA=($sequence=~tr/FA//);
```

Figure 1.6: Perl script for dipeptide count feature extraction

```
my$FC=($sequence=~tr/FC//);my$FD=($sequence=~tr/FD//);my$FE=($sequence=~tr/FE//);my$FF=($sequence=~tr/FF//);my$FG=($sequence=~tr/FG//);my$FH=($sequence=~tr/FH//);my$FI=($sequence=~tr/FI//);my$FK=($sequence=~tr/FK//);my$FL=($sequence=~tr/FL//);my$FM=($sequence=~tr/FM//);my$FN=($sequence=~tr/FN//);my$FP=($sequence=~tr/FP//);my$FQ=($sequence=~tr/FQ//);my$FR=($sequence=~tr/FR//);my$FS=($sequence=~tr/FS//);my$FT=($sequence=~tr/FT//);my$FV=($sequence=~tr/FV//);my$FW=($sequence=~tr/FW//);my$FY=($sequence=~tr/FY//);my$GA=($sequence=~tr/GA//);my$GC=($sequence=~tr/GC//);my$GD=($sequence=~tr/GD//);my$GE=($sequence=~tr/GE//);my$GF=($sequence=~tr/GF//);my$GG=($sequence=~tr/GG//);my$GH=($sequence=~tr/GH//);my$GI=($sequence=~tr/GI//);my$GK=($sequence=~tr/GK//);my$GL=($sequence=~tr/GL//);my$GM=($sequence=~tr/GM//);my$GN=($sequence=~tr/GN//);my$GP=($sequence=~tr/GP//);my$GQ=($sequence=~tr/GQ//);my$GR=($sequence=~tr/GR//);my$GS=($sequence=~tr/GS//);my$GT=($sequence=~tr/GT//);my$GV=($sequence=~tr/GV//);my$GW=($sequence=~tr/GW//);my$GY=($sequence=~tr/GY//);my$HA=($sequence=~tr/HA//);my$HC=($sequence=~tr/HC//);my$HD=($sequence=~tr/HD//);my$HE=($sequence=~tr/HE//);my$HF=($sequence=~tr/HF//);my$HG=($sequence=~tr/HG//);my$HH=($sequence=~tr/HH//);my$HI=($sequence=~tr/HI//);my$HK=($sequence=~tr/HK//);my$HL=($sequence=~tr/HL//);my$HM=($sequence=~tr/HM//);my$HN=($sequence=~tr/HN//);my$HP=($sequence=~tr/HP//);my$HQ=($sequence=~tr/HQ//);my$HR=($sequence=~tr/HR//);my$HS=($sequence=~tr/HS//);my$HT=($sequence=~tr/HT//);my$HV=($sequence=~tr/HV//);my$HW=($sequence=~tr/HW//);my$HY=($sequence=~tr/HY//);my$IA=($sequence=~tr/IA//);my$IC=($sequence=~tr/IC//);my$ID=($sequence=~tr/ID//);my$IE=($sequence=~tr/IE//);my$IF=($sequence=~tr/IF//);my$IG=($sequence=~tr/IG//);my$IH=($sequence=~tr/IH//);my$II=($sequence=~tr/II//);my$IK=($sequence=~tr/IK//);my$IL=($sequence=~tr/IL//);my$IM=($sequence=~tr/IM//);my$IN=($sequence=~tr/IN//);my$IP=($sequence=~tr/IP//);my$IQ=($sequence=~tr/IQ//);my$IR=($sequence=~tr/IR//);my$IS=($sequence=~tr/IS//);my$IT=($sequence=~tr/IT//);my$IV=($sequence=~tr/IV//);my$IW=($sequence=~tr/IW//);my$IY=($sequence=~tr/IY//);my$KA=($sequence=~tr/KA//);my$KC=($sequence=~tr/KC//);my$KD=($sequence=~tr/KD//);my$KE=($sequence=~tr/KE//);my$KF=($sequence=~tr/KF//);my$KG=($sequence=~tr/KG//);my$KH=($sequence=~tr/KH//);my$KI=($sequence=~tr/KI//);my$KK=($sequence=~tr/KK//);my$KL=($sequence=~tr/KL//);my$KM=($sequence=~tr/KM//);my$KN=($sequence=~tr/KN//);my$KP=($sequence=~tr/KP//);my$KQ=($sequence=~tr/KQ//);my$KR=($sequence=~tr/KR//);my$KS=($sequence=~tr/KS//);my$KT=($sequence=~tr/KT//);my$KV=($sequence=~tr/KV//);my$KW=($sequence=~tr/KW//);my$KY=($sequence=~tr/KY//);my$LA=($sequence=~tr/LA//);my$LC=($sequence=~tr/LC//);my$LD=($sequence=~tr/LD//);my$LE=($sequence=~tr/LE//);my$LF=($sequence=~tr/LF//);my$LG=($sequence=~tr/LG//);my$LH=($sequence=~tr/LH//);my$LI=($sequence=~tr/LI//);my$LK=($sequence=~tr/LK//);my$LL=($sequence=~tr/LL//);my$LM=($sequence=~tr/LM//);my$LN=($sequence=~tr/LN//);my$LP=($sequence=~tr/LP//);my$LQ=($sequence=~tr/LQ//);my$LR=($sequence=~tr/LR//);my$LS=($sequence=~tr/LS//);my$LT=($sequence=~tr/LT//);my$LV=($sequence=~tr/LV//);my$LW=($sequence=~tr/LW//);my$LY=($sequence=~tr/LY//);my$MA=($sequence=~tr/MA//);my$MC=($sequence=~tr/MC//);my$MD=($sequence=~tr/MD//);my$ME=($sequence=~tr/ME//);my$MF=($sequence=~tr/MF//);my$MG=($sequence=~tr/MG//);my$MH=($sequence=~tr/MH//);my$MI=($sequence=~tr/MI//);my$MK=($sequence=~tr/MK//);my$ML=($sequence=~tr/ML//);my$MM=($sequence=~tr/MM//);my$MN=($sequence=~tr/MN//);my$MP=($sequence=~tr/MP//);my$MQ=($sequence=~tr/MQ//);my$MR=($sequence=~tr/MR//);my$MS=($sequence=~tr/MS//);my$MT=($sequence=~tr/MT//);my$MV=($sequence=~tr/MV//);my$MW=($sequence=~tr/MW//);my$MY=($sequence=~tr/MY//);my$NA=($sequence=~tr/NA//);my$NC=($sequence=~tr/NC//);//);my$ND=($sequence=~tr/ND//);my$NE=($sequence=~tr/NE//);my$NF=($sequence=~tr/NF//);my$NG=($sequence=~tr/NG//);
```

```
my$NH=($sequence=~tr/NH//);my$NI=($sequence=~tr/NI//);my$NK=($sequence=~tr/NK//);my$NL=($se
quence=~tr/NL//);my$NM=($sequence=~tr/NM//);my$NN=($sequence=~tr/NN//);my$NP=($sequence=~tr
/NP//);my$NQ=($sequence=~tr/NQ//);my$NR=($sequence=~tr/NR//);my$NS=($sequence=~tr/NS//);my$N
T=($sequence=~tr/NT//);my$NV=($sequence=~tr/NV//);my$NW=($sequence=~tr/NW//);my$NY=($seque
nce=~tr/NY//);my$PA=($sequence=~tr/PA//);my$PC=($sequence=~tr/PC//);my$PD=($sequence=~tr/PD//)
;my$PE=($sequence=~tr/PE//);my$PF=($sequence=~tr/PF//);my$PG=($sequence=~tr/PG//);my$PH=($seq
uence=~tr/PH//);my$PI=($sequence=~tr/PI//);my$PK=($sequence=~tr/PK//);my$PL=($sequence=~tr/PL//);
my$PM=($sequence=~tr/PM//);my$PN=($sequence=~tr/PN//);my$PP=($sequence=~tr/PP//);my$PQ=($seq
uence=~tr/PQ//);my$PR=($sequence=~tr/PR//);my$PS=($sequence=~tr/PS//);my$PT=($sequence=~tr/PT//)
;my$PV=($sequence=~tr/PV//);my$PW=($sequence=~tr/PW//);my$PY=($sequence=~tr/PY//);my$QA=($s
equence=~tr/QA//);my$QC=($sequence=~tr/QC//);my$QD=($sequence=~tr/QD//);my$QE=($sequence=~tr
/QE//);my$QF=($sequence=~tr/QF//);my$QG=($sequence=~tr/QG//);my$QH=($sequence=~tr/QH//);my$
QI=($sequence=~tr/QI//);my$QK=($sequence=~tr/QK//);my$QL=($sequence=~tr/QL//);my$QM=($sequen
ce=~tr/QM//);my$QN=($sequence=~tr/QN//);my$QP=($sequence=~tr/QP//);my$QQ=($sequence=~tr/QQ//
);my$QR=($sequence=~tr/QR//);my$QS=($sequence=~tr/QS//);my$QT=($sequence=~tr/QT//);my$QV=($
sequence=~tr/QV//);my$QW=($sequence=~tr/QW//);my$QY=($sequence=~tr/QY//);my$RA=($sequence=
~tr/RA//);my$RC=($sequence=~tr/RC//);my$RD=($sequence=~tr/RD//);my$RE=($sequence=~tr/RE//);my
$RF=($sequence=~tr/RF//);my$RG=($sequence=~tr/RG//);my$RH=($sequence=~tr/RH//);my$RI=($sequ
ence=~tr/RI//);my$RK=($sequence=~tr/RK//);my$RL=($sequence=~tr/RL//);my$RM=($sequence=~tr/RM//
);my$RN=($sequence=~tr/RN//);my$RP=($sequence=~tr/RP//);my$RQ=($sequence=~tr/RQ//);my$RR=($s
equence=~tr/RR//);my$RS=($sequence=~tr/RS//);my$RT=($sequence=~tr/RT//);my$RV=($sequence=~tr/
RV//);my$RW=($sequence=~tr/RW//);my$RY=($sequence=~tr/RY//);my$SA=($sequence=~tr/SA//);my$S
C=($sequence=~tr/SC//);my$SD=($sequence=~tr/SD//);my$SE=($sequence=~tr/SE//);my$SF=($sequence=
~tr/SF//);my$SG=($sequence=~tr/SG//);my$SH=($sequence=~tr/SH//);my$SI=($sequence=~tr/SI//);my$S
K=($sequence=~tr/SK//);my$SL=($sequence=~tr/SL//);my$SM=($sequence=~tr/SM//);my$SN=($sequenc
e=~tr/SN//);my$SP=($sequence=~tr/SP//);my$SQ=($sequence=~tr/SQ//);my$SR=($sequence=~tr/SR//);my
$SS=($sequence=~tr/SS//);my$ST=($sequence=~tr/ST//);my$SV=($sequence=~tr/SV//);my$SW=($sequen
ce=~tr/SW//);my$SY=($sequence=~tr/SY//);my$TA=($sequence=~tr/TA//);my$TC=($sequence=~tr/TC//);
my$TD=($sequence=~tr/TD//);my$TE=($sequence=~tr/TE//);my$TF=($sequence=~tr/TF//);my$TG=($seq
uence=~tr/TG//);my$TH=($sequence=~tr/TH//);my$TI=($sequence=~tr/TI//);my$TK=($sequence=~tr/TK//
);my$TL=($sequence=~tr/TL//);my$TM=($sequence=~tr/TM//);my$TN=($sequence=~tr/TN//);my$TP=($s
equence=~tr/TP//);my$TQ=($sequence=~tr/TQ//);my$TR=($sequence=~tr/TR//);my$TS=($sequence=~tr/T
S//);my$TT=($sequence=~tr/TT//);my$TV=($sequence=~tr/TV//);my$TW=($sequence=~tr/TW//);my$TY
=($sequence=~tr/TY//);my$VA=($sequence=~tr/VA//);my$VC=($sequence=~tr/VC//);my$VD=($sequence
=~tr/VD//);my$VE=($sequence=~tr/VE//);my$VF=($sequence=~tr/VF//);my$VG=($sequence=~tr/VG//);m
y$VH=($sequence=~tr/VH//);my$VI=($sequence=~tr/VI//);my$VK=($sequence=~tr/VK//);my$VL=($sequ
ence=~tr/VL//);my$VM=($sequence=~tr/VM//);my$VN=($sequence=~tr/VN//);my$VP=($sequence=~tr/V
P//);my$VQ=($sequence=~tr/VQ//);my$VR=($sequence=~tr/VR//);my$VS=($sequence=~tr/VS//);my$VT
=($sequence=~tr/VT//);my$VV=($sequence=~tr/VV//);my$VW=($sequence=~tr/VW//);my$VY=($sequen
ce=~tr/VY//);my$WA=($sequence=~tr/WA//);my$WC=($sequence=~tr/WC//);my$WD=($sequence=~tr/W
D//);my$WE=($sequence=~tr/WE//);my$WF=($sequence=~tr/WF//);my$WG=($sequence=~tr/WG//);my$
WH=($sequence=~tr/WH//);my$WI=($sequence=~tr/WI//);my$WK=($sequence=~tr/WK//);my$WL=($seq
uence=~tr/WL//);my$WM=($sequence=~tr/WM//);my$WN=($sequence=~tr/WN//);my$WP=($sequence=~
tr/WP//);my$WQ=($sequence=~tr/WQ//);my$WR=($sequence=~tr/WR//);my$WS=($sequence=~tr/WS//);
```

```
my$WT=($sequence=~tr/WT//);my$WV=($sequence=~tr/WV//);my$WW=($sequence=~tr/WW//);my$
WY=($sequence=~tr/WY//);my$YA=($sequence=~tr/YA//);my$YC=($sequence=~tr/YC//);my$YD=($s
equence=~tr/YD//);my$YE=($sequence=~tr/YE//);my$YF=($sequence=~tr/YF//);my$YG=($sequence=
~tr/YG//);my$YH=($sequence=~tr/YH//);my$YI=($sequence=~tr/YI//);my$YK=($sequence=~tr/YK//);
my$YL=($sequence=~tr/YL//);my$YM=($sequence=~tr/YM//);my$YN=($sequence=~tr/YN//);my$YP
=($sequence=~tr/YP//);my$YQ=($sequence=~tr/YQ//);my$YR=($sequence=~tr/YR//);my$YS=($seque
nce=~tr/YS//);my$YT=($sequence=~tr/YT//);my$YV=($sequence=~tr/YV//);my$YW=($sequence=~tr/
YW//);my$YY=($sequence=~tr/YY//);

print OUT
"1"."\t1:".$AA."\t2:".$AC."\t3:".$AD."\t4:".$AE."\t5:".$AF."\t6:".$AG."\t7:".$AH."\t8:".$AI."\t9:".$AK.
"\t10:".$AL."\t11:".$AM."\t12:".$AN."\t13:".$AP."\t14:".$AQ."\t15:".$AR."\t16:".$AS."\t17:".$AT."\t18
:".$AV."\t19:".$AW."\t20:".$AY."\t21:".$CA."\t22:".$CC."\t23:".$CD."\t24:".$CE."\t25:".$CF."\t26:".$
CG."\t27:".$CH."\t28:".$CI."\t29:".$CK."\t30:".$CL."\t31:".$CM."\t32:".$CN."\t33:".$CP."\t34:".$CQ."\
t35:".$CR."\t36:".$CS."\t37:".$CT."\t38:".$CV."\t39:".$CW."\t40:".$CY."\t41:".$DA."\t42:".$DC."\t43:"
.$DD."\t44:".$DE."\t45:".$DF."\t46:".$DG."\t47:".$DH."\t48:".$DI."\t49:".$DK."\t50:".$DL."\t51:".$D
M."\t52:".$DN."\t53:".$DP."\t54:".$DQ."\t55:".$DR."\t56:".$DS."\t57:".$DT."\t58:".$DV."\t59:".$DW."
\t60:".$DY."\t61:".$EA."\t62:".$EC."\t63:".$ED."\t64:".$EE."\t65:".$EF."\t66:".$EG."\t67:".$EH."\t68:".
$EI."\t69:".$EK."\t70:".$EL."\t71:".$EM."\t72:".$EN."\t73:".$EP."\t74:".$EQ."\t75:".$ER."\t76:".$ES."\t
77:".$ET."\t78:".$EV."\t79:".$EW."\t80:".$EY."\t81:".$FA."\t82:".$FC."\t83:".$FD."\t84:".$FE."\t85:".$
FF."\t86:".$FG."\t87:".$FH."\t88:".$FI."\t89:".$FK."\t90:".$FL."\t91:".$FM."\t92:".$FN."\t93:".$FP."\t94
:".$FQ."\t95:".$FR."\t96:".$FS."\t97:".$FT."\t98:".$FV."\t99:".$FW."\t100:".$FY."\t101:".$GA."\t102:".
$GC."\t103:".$GD."\t104:".$GE."\t105:".$GF."\t106:".$GG."\t107:".$GH."\t108:".$GI."\t109:".$GK."\t1
10:".$GL."\t111:".$GM."\t112:".$GN."\t113:".$GP."\t114:".$GQ."\t115:".$GR."\t116:".$GS."\t117:".$G
T."\t118:".$GV."\t119:".$GW."\t120:".$GY."\t121:".$HA."\t122:".$HC."\t123:".$HD."\t124:".$HE."\t12
5:".$HF."\t126:".$HG."\t127:".$HH."\t128:".$HI."\t129:".$HK."\t130:".$HL."\t131:".$HM."\t132:".$HN.
"\t133:".$HP."\t134:".$HQ."\t135:".$HR."\t136:".$HS."\t137:".$HT."\t138:".$HV."\t139:".$HW."\t140:".
$HY."\t141:".$IA."\t142:".$IC."\t143:".$ID."\t144:".$IE."\t145:".$IF."\t146:".$IG."\t147:".$IH."\t148:".$
II."\t149:".$IK."\t150:".$IL."\t151:".$IM."\t152:".$IN."\t153:".$IP."\t154:".$IQ."\t155:".$IR."\t156:".$IS.
"\t157:".$IT."\t158:".$IV."\t159:".$IW."\t160:".$IY."\t161:".$KA."\t162:".$KC."\t163:".$KD."\t164:".$K
E."\t165:".$KF."\t166:".$KG."\t167:".$KH."\t168:".$KI."\t169:".$KK."\t170:".$KL."\t171:".$KM."\t172:
".$KN."\t173:".$KP."\t174:".$KQ."\t175:".$KR."\t176:".$KS."\t177:".$KT."\t178:".$KV."\t179:".$KW."
\t180:".$KY."\t181:".$LA."\t182:".$LC."\t183:".$LD."\t184:".$LE."\t185:".$LF."\t186:".$LG."\t187:".$L
H."\t188:".$LI."\t189:".$LK."\t190:".$LL."\t191:".$LM."\t192:".$LN."\t193:".$LP."\t194:".$LQ."\t195:".
$LR."\t196:".$LS."\t197:".$LT."\t198:".$LV."\t199:".$LW."\t200:".$LY."\t201:".$MA."\t202:".$MC."\t
203:".$MD."\t204:".$ME."\t205:".$MF."\t206:".$MG."\t207:".$MH."\t208:".$MI."\t209:".$MK."\t210:".
$ML."\t211:".$MM."\t212:".$MN."\t213:".$MP."\t214:".$MQ."\t215:".$MR."\t216:".$MS."\t217:".$MT.
"\t218:".$MV."\t219:".$MW."\t220:".$MY."\t221:".$NA."\t222:".$NC."\t223:".$ND."\t224:".$NE."\t225
:".$NF."\t226:".$NG."\t227:".$NH."\t228:".$NI."\t229:".$NK."\t230:".$NL."\t231:".$NM."\t232:".$NN."
\t233:".$NP."\t234:".$NQ."\t235:".$NR."\t236:".$NS."\t237:".$NT."\t238:".$NV."\t239:".$NW."\t240:".
$NY."\t241:".$PA."\t242:".$PC."\t243:".$PD."\t244:".$PE."\t245:".$PF."\t246:".$PG."\t247:".$PH."\t24
8:".$PI."\t249:".$PK."\t250:".$PL."\t251:".$PM."\t252:".$PN."\t253:".$PP."\t254:".$PQ."\t255:".$PR."\t
256:".$PS."\t257:".$PT."\t258:".$PV."\t259:".$PW."\t260:".$PY."\t261:".$QA."\t262:".$QC."\t263:".$Q
D."\t264:".$QE."\t265:".$QF."\t266:".$QG."\t267:".$QH."\t268:".$QI."\t269:".$QK."\t270:".$QL."\t271:
```

```
."\t276:".$QS."\t277:".$QT."\t278:".$QV."\t279:".$QW."\t280:".$QY."\t281:".$RA."\t282:".$RC."\t28
3:".$RD."\t284:".$RE."\t285:".$RF."\t286:".$RG."\t287:".$RH."\t288:".$RI."\t289:".$RK."\t290:".$RL.
"\t291:".$RM."\t292:".$RN."\t293:".$RP."\t294:".$RQ."\t295:".$RR."\t296:".$RS."\t297:".$RT."\t298:
".$RV."\t299:".$RW."\t300:".$RY."\t301:".$SA."\t302:".$SC."\t303:".$SD."\t304:".$SE."\t305:".$SF."
\t306:".$SG."\t307:".$SH."\t308:".$SI."\t309:".$SK."\t310:".$SL."\t311:".$SM."\t312:".$SN."\t313:".$
SP."\t314:".$SQ."\t315:".$SR."\t316:".$SS."\t317:".$ST."\t318:".$SV."\t319:".$SW."\t320:".$SY."\t32
1:".$TA."\t322:".$TC."\t323:".$TD."\t324:".$TE."\t325:".$TF."\t326:".$TG."\t327:".$TH."\t328:".$TI."
\t329:".$TK."\t330:".$TL."\t331:".$TM."\t332:".$TN."\t333:".$TP."\t334:".$TQ."\t335:".$TR."\t336:".
$TS."\t337:".$TT."\t338:".$TV."\t339:".$TW."\t340:".$TY."\t341:".$VA."\t342:".$VC."\t343:".$VD."\
t344:".$VE."\t345:".$VF."\t346:".$VG."\t347:".$VH."\t348:".$VI."\t349:".$VK."\t350:".$VL."\t351:".
$VM."\t352:".$VN."\t353:".$VP."\t354:".$VQ."\t355:".$VR."\t356:".$VS."\t357:".$VT."\t358:".$VV."
\t359:".$VW."\t360:".$VY."\t361:".$WA."\t362:".$WC."\t363:".$WD."\t364:".$WE."\t365:".$WF."\t3
66:".$WG."\t367:".$WH."\t368:".$WI."\t369:".$WK."\t370:".$WL."\t371:".$WM."\t372:".$WN."\t373
:".$WP."\t374:".$WQ."\t375:".$WR."\t376:".$WS."\t377:".$WT."\t378:".$WV."\t379:".$WW."\t380:".
$WY."\t381:".$YA."\t382:".$YC."\t383:".$YD."\t384:".$YE."\t385:".$YF."\t386:".$YG."\t387:".$YH.
"\t388:".$YI."\t389:".$YK."\t390:".$YL."\t391:".$YM."\t392:".$YN."\t393:".$YP."\t394:".$YQ."\t395:
".$YR."\t396:".$YS."\t397:".$YT."\t398:".$YV."\t399:".$YW."\t400:".$YY."\n";

}
}
```

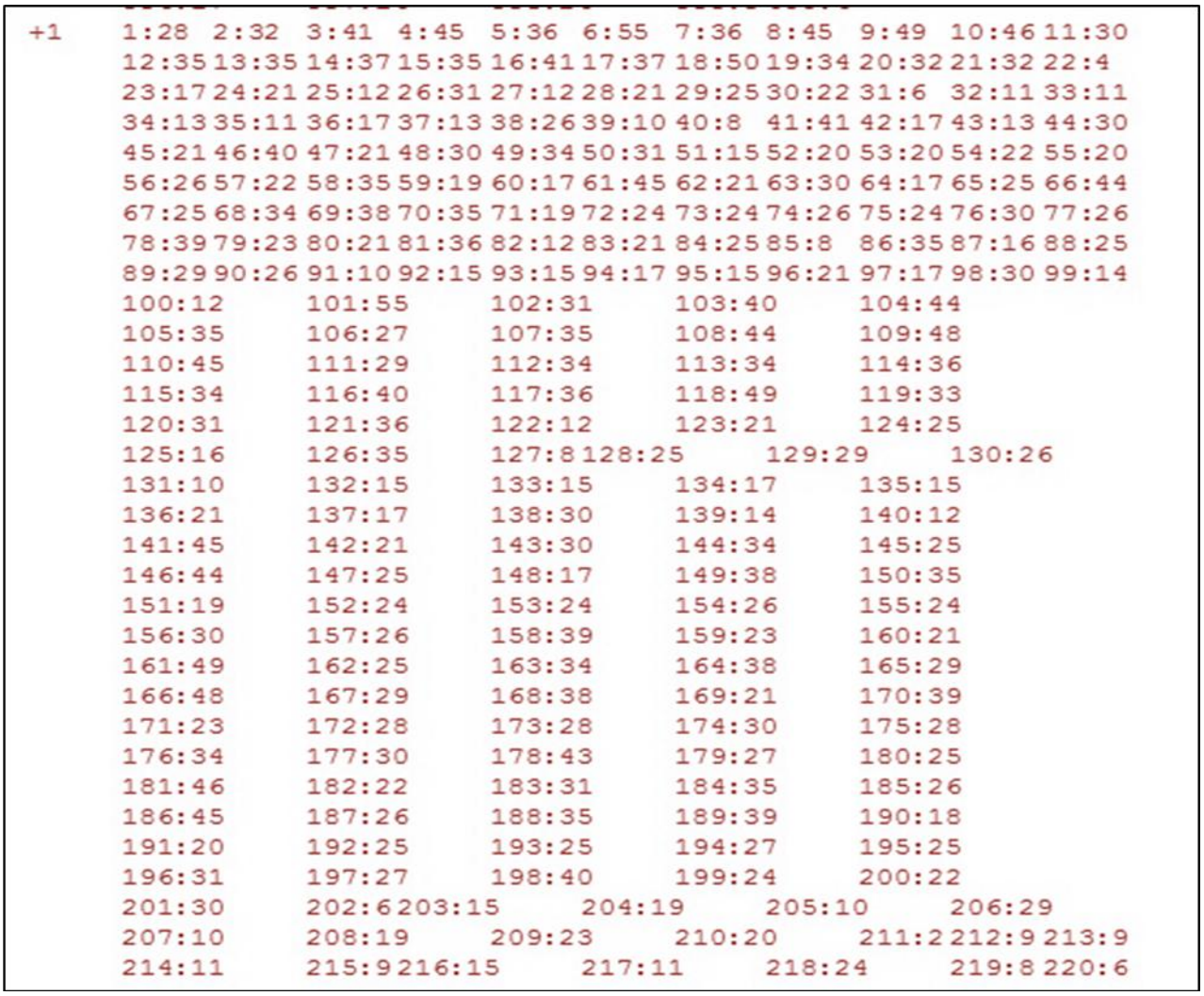


Figure 1.7: Feature selection dipeptide count training file format for svm_learn.

```
#!/usr/bin/perl -w
$proteinfilename = 'tim_barrel';
open(PROTEINFILE, $proteinfilename);
open (FA, ">>pssm_tim");
@protein1 = <PROTEINFILE> ;
close PROTEINFILE ;
$protein = join('', @protein1) ;
while ($protein =~ />\n(.*?)>/g)
{
$prot = $1 ;
@new_protein = split('', $prot);
$c =0;
$length = scalar@new_protein ;
for($i=0 ; $i < $length ; ++$i)
{
$p = ($i+1). ":" ;
splice ( @new_protein, $c, 0, $p) ;
$c = $c + 2 ;
#print @new_protein ;
}
```

Figure 1.8: Perl script for PSSM profiles

```
$proteinn .= "\n\n" . join(", @new_protein) ;
}
#print $proteinn ;
$proteinn =~s/Sequence// ;
$proteinn =~s/A/00000000000000000001\t/g;
$proteinn =~s/R/00000000000000000010\t/g;
$proteinn =~s/N/00000000000000000100\t/g;
$proteinn =~s/D/00000000000000001000\t/g;
$proteinn =~s/C/00000000000000010000\t/g;
$proteinn =~s/E/00000000000000100000\t/g;
$proteinn =~s/Q/00000000000001000000\t/g;
$proteinn =~s/G/00000000000010000000\t/g;
$proteinn =~s/H/00000000000100000000\t/g;
$proteinn =~s/I/00000000001000000000\t/g;
$proteinn =~s/L/00000000010000000000\t/g;
$proteinn =~s/K/00000000100000000000\t/g;
$proteinn =~s/M/00000001000000000000\t/g;
$proteinn =~s/F/00000010000000000000\t/g;
$proteinn =~s/P/00000100000000000000\t/g;
$proteinn =~s/S/00001000000000000000\t/g;
$proteinn =~s/T/00010000000000000000\t/g;
$proteinn =~s/W/00100000000000000000\t/g;
$proteinn =~s/Y/01000000000000000000\t/g;
$proteinn =~s/V/10000000000000000000\t/g;
print FA $proteinn ;

exit;
```

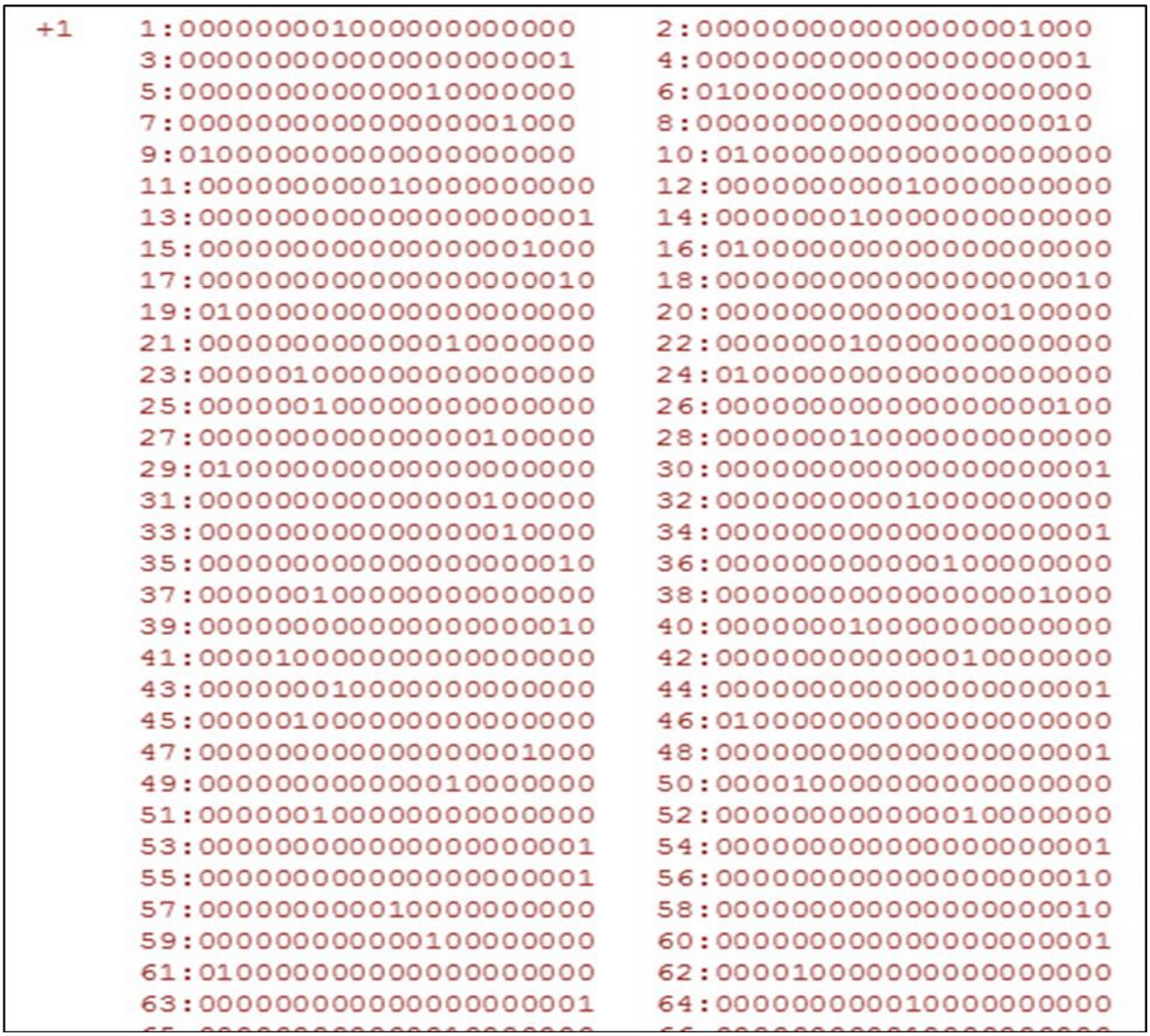


Figure 1.9: Feature selection PSSM profiles training file input file format for svm_light.

- Once all the perl script convert the data into prescribed format it named training files, they used for generating model files.for generating model files command was given by a command prompt:

  svm_learn [options] training_file model_file.

- Various options are available in SVM^{light} to train the machine which are given below:

  -? - this help
  -v [0..3] - verbosity level (default 1)

Learning options:

- -z {c,r,p} - select between classification (c), regression (r), and preference ranking (p)
- -c float - C: trade-off between training error and margin (default [avg. x*x]^-1)
- -w [0..] - epsilon width of tube for regression (default 0.1)
- -j float - Cost: cost-factor, by which training errors on positive examples out weight errors examples (default 1)
- -b [0,1] - use biased hyperplane (i.e. x*w+b0) instead of unbiased hyperplane (i.e. x*w0) (default 1)
- -i [0,1] - remove inconsistent training examples and retrain (default 0)

Performance estimation options:

- -x [0,1] - compute leave-one-out estimates (default 0)
- -o]0..2] - value of rho for Xi Alpha-estimator and for pruning leave-one-out computation (default 1.0)
- -k [0..100] - search depth for extended XiAlpha-estimator (default 0)

Transduction options:

- -p [0..1] - fraction of unlabeled examples to be classified into the positive class (default is the ratio of positive and negative examples in the training data)

Kernel options:

- -t int - type of kernel function:
  - 0: linear (default)
  - 1: polynomial (s a*b+c)^d
  - 2: radial basis function exp(-gamma ||a-b||^2)
  - 3: sigmoid tanh(s a*b + c)

- 4: user defined kernel from kernel.h

- -d int - parameter d in polynomial kernel
- -g float - parameter gamma in rbf kernel
- -s float - parameter s in sigmoid/poly kernel

-r float - parameter c in sigmoid/poly kernel

-u string - parameter of user defined kernel

Optimization options:

- -q [2..] - maximum size of QP-subproblems (default 10)
- -n [2..q] - number of new variables entering the working set in each iteration (default n = q). Set n<q to prevent zig-zagging.
- -m [5..] - size of cache for kernel evaluations in MB (default 40) The larger the faster...
- -e float - eps: Allow that error for termination criterion [y [w*x+b] - 1] = eps (default 0.001)
- -h [5..] - number of iterations a variable needs to be optimal before considered for shrinking (default 100)
- -f [0,1] - do final optimality check for variables removed by shrinking. Although this test is usually positive, there is no guarantee that the optimum was found if the test is omitted. (default 1)
- -y string -> if option is given, reads alphas from file with given and uses them as starting point. (default 'disabled')
- -# int -> terminate optimization, if no progress after this number of iterations. (default 100000)

Output options:

- -l char - file to write predicted labels of unlabeled examples into after transductive learning
- -a char - write all alphas to this file after learning (in the same order as in the training set)

Various combinations were done to deduce the best learning model. One which best worked for the model is Leave on out learning method; it's time taking learning method.

Svm_classify :

- After svm_learn svm_classify is a executable file. In our work we have done binary classification. Once model file was written svm_classify were used to determine the sample problem or test_file contains tim-barrel protein fold or not.

- In classification the target values were denoted the class of example i.e. whether it fall under the category or not. +1 as the target value marks the positive example, -1 a negative example respectively.
- `-1 1:0.43 3:0.12 9284:0.2 # abcdef`
- specifies a negative example for which feature number 1 has the value 0.43, feature number 3 has the value 0.12, feature number 9284 has the value 0.2, and all the other features have value 0.
- In addition, the string `abcdef` is stored with the vector, which can serve as a way of providing additional information for user defined kernels. A class label of 0 indicates that this example should be classified using transduction.
- The predictions for the examples classified by transduction are written to the file specified through the -l option. The order of the predictions is the same as in the training data:

svm_classify [options] test_file model_file output_file

- Whereas, test_file contains the data to be tested model_file generated through the svm_learn command and output file will have the score which tells whether test data was the tim-barel protein fold or not.
- Like svm_learn svm_classify also have options given below
  - h Help.
  - -v [0.3] : Verbosity level (default 2).
  - -f [0,1] 0 : old output format of V1.0
  - 1 : output the value of decision function (default)
- The test examples in `test_file` are given in the same format as the training examples (possibly with 0 as class label). For all test examples in `test_file` the predicted values are written to `output_file`. There is one line per test example in `output_file` containing the value of the decision function on that example.
- For classification, the sign of this value determines the predicted class.
- For regression, it is the predicted value itself, and for ranking the value can be used to order the test examples.
- The test example file has the same format as the one for `svm_learn`. Again, `<class>` can have the value zero indicating unknown.
-

4.12 .1 Construction of TIMPRED

In this project three feature selections or descriptors were combined into a prediction system called TIMPRED with the help of the SVM algorithm. As a machine learning method for two classes of classification SVM focus to find a path that best maps each and every member of training sets to the correct classification.

In this SVM was used to train to distinguish two different protein pairs related to TIM-barrel protein. First, type of protein pair (i.e., positive sample), both proteins are TIM barrel proteins. Protein database bank contains total 13,203 positive samples i.e. proteins with the TIM-barrel fold.

In the second type of protein pairs (i.e.) negative samples belongs to the non- proteins parts or non-TIM barrel proteins folds, it contains total 137,490 proteins samples.

Thus, total parameters were used in SVM learning.

The PDB dataset was compiled into 150612 protein pairs, which was further divided into 5 roughly equal subsets. An evaluation similar to 5 fold cross validation were performed. To predict whether a given set named or labeled as "test" set, whereas as other remaining 4 set considered as training sets. SVM model were developed for each of the training set. The class label for positive and negative samples were set to be +1 and -1, respectively. The ratio of positive to negative samples was 1:10 in the training set.

Using set at such a ratio would inevitably cause the SVM model to predict every pair as negative case. The optimized ratio in the training set was 1:2.5. Each training set was modified by discarding a random selection of the negative samples prior to training. The training resulted in four separate SVM models.

The implemented SVM algorithm was SVM-light. The applied kernel function is the radial basis function (RBF). The corresponding parameter settings of the SVM learning were automatically optimized by the SVM-light (Aarti Garg et al., 2005).

It is worth mentioning here that the predicted score of each protein pair can be regarded as a combination of the corresponding seven parameters with the assistance of SVM. Based on the predicted scores, the performance of TIMPRED was assessed in the same way as we evaluated the individual descriptors.

4.12.2 Webserver: TIMPRED:

To facilitate the community's research, a webserver of TIMPRED was constructed and will be freely available. To sufficiently represent the known structural TIM-barrel proteins as well as

allow a reasonable computational time. Generally, the predicted score reflects the query sequence's probability of adopting TIM-barrel fold.

Finally, the predicted scores for the all protein pairs are ranked and the top 10 hits are reported. In the resulting page provided by TIMPRED, the accession id, kernel used prediction score, and confidence level for the query sequence. The whole process for each query normally takes some time with a single processor on our Ubuntu 11.0 Linux work station.

To provide confidence levels for different prediction scores resulting from TIMPRED, a stringent negative dataset was complied. First, in the initial dataset only the non-TIM-barrel protein that belongs to α/β class (i.e., the same structural class as the TIM-barrel fold) was kept.

Second the proteins with a sequence length < 100 or > 1000 were removed. Third, the proteins that had been used in training TIMPRED (i.e. the SVM model) were further discarded. Finally, the 500 non-TIM-barrel retained proteins retained. We processed all 500 proteins on TIMPRED. Compared with other structural classes, query proteins belonging to the α/β class should have a higher probability of being predicted as TIM-barrel proteins.

We only selected the α/β proteins as negative controls, which should guarantee a reliable estimate of threshold for different confidence levels.

The typical web application (and probably many other server side applications), naturally divides into a frontend, and a backend. The backend talks to the DB and really only should provide crud functionality: Create, Read, Update, Delete.

Then there is frontend, which implements the browsed part: it navigates the miscellaneous steps one has to go through to create an order, and so on. All of this is fully server side code. Quite separate from all this is the actual view- layer, which handles input validation, output formatting, page creation, etc. In particular there should be no code in the presentation templates.

Front-End Design: The front end analyzes the source code to build an internal representation of the program, called the intermediate representation or IR.

It also manages the symbol table, a data structure mapping each symbol in the source code to associated information such as location type and scope. The front end designing involves computer languages HTML and PHP.

Back-End Design: The back end is sometimes thought as a code generator because odd the overlapped functionality of generating assembly code. Sometimes literature uses middle end to distinguish the generic analysis and optimization phases in the back end from the machine dependent code generators. The backend designing of the TIMPRED was done with the help og PERL, PHP, and different PHP shell commands.

Architecture of TIMPRED:

TIMPRED is a bi-layer model given in the figure:

First Layer: This layer the query sequence provided by the user is processed according to the chosen strategy of prediction or feature selection amino acid count (AAC), dipeptide count (DPC) and position specific scoring matrix (PSSM) and an SVM score is generated.

Second Layer: In this layer, the SVM score generated will be evaluated, and a prediction output is generated.

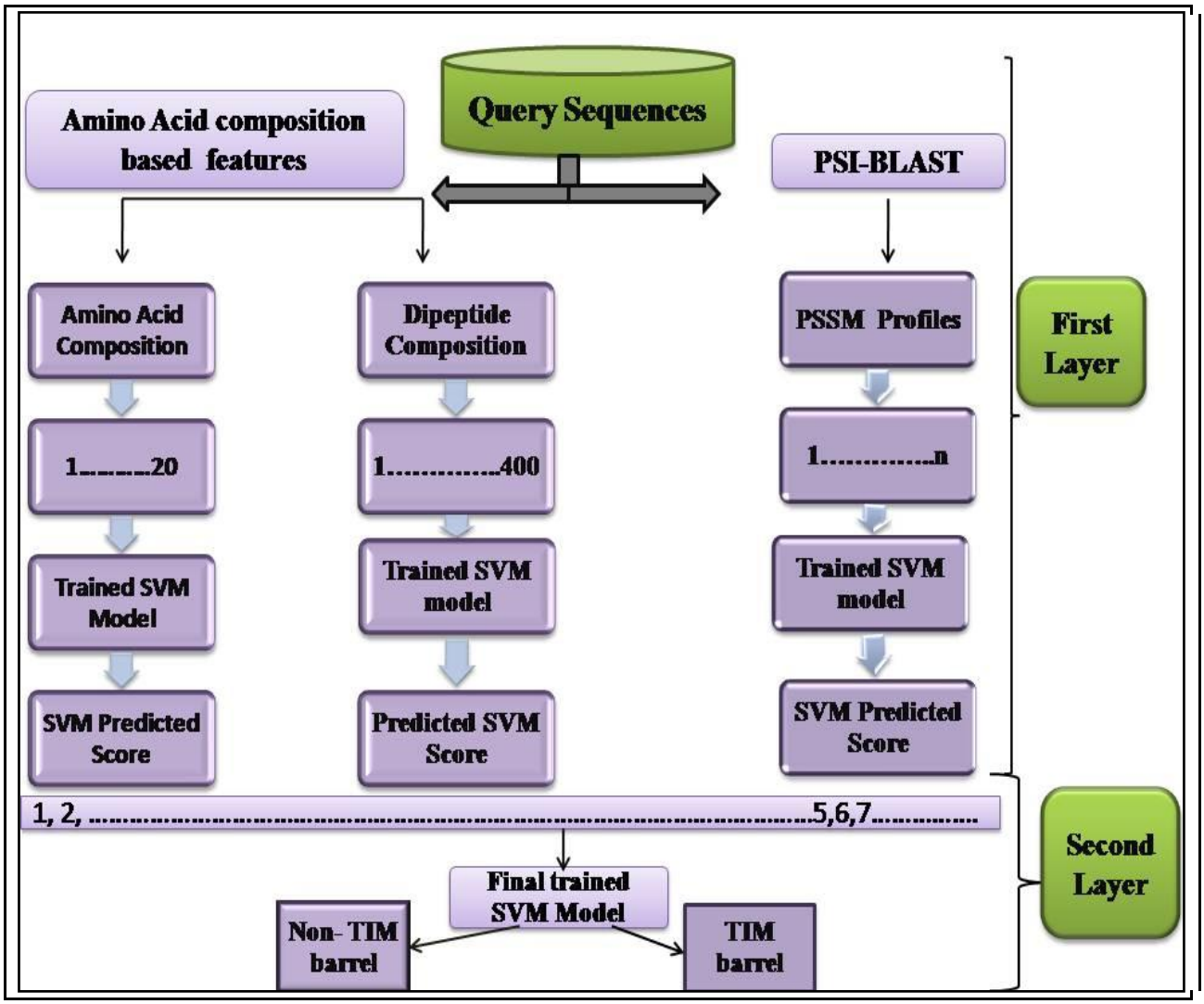


Figure 1.10: TIMPRED webserver architecture.

4.14 Construction of amino acid composition and dipeptide count based SVM model:

The AAC base SVM model was trained to distinguish TIM-barrel and non-TIM-barrel proteins. Briefly, the 1952 TIM-barrel proteins in the dataset were considered positive instances and their labels were set to +1, while 2032 were labeled as negative instances and their label were set as -1.

The AAC for each protein was used as the input feature vector. A fivefold-cross validation was performed and for learning method was Leave one out. We divided the subset into roughly 5 equal subsets. In each evaluation step, one subset was selected for testing, while rest 4 were merged into a training dataset.

SVM-light with the RBF kernel was employed to train the models and other SVM parameters were automatically optimized and best parameters were used to get models.

4.15 Performance Measure:

To assess the performance of various modules developed in this study were computed following threshold dependent parameters (Garde A et al., 2013):

Sensitivity (Sn): or percent coverage of TIM-barrel protein folds.

Specificity (Sp): or percent coverage of non-TIM-barrel protein.

Overall Accuracy (Ac): percent probability of correct prediction of TIM-barrel protein, also called as accuracy of interacting residues,

And Matthew's correlation coefficient (MCC) using following equations:

$$Sn = \frac{tp}{tp + fn} \times 100$$

$$Sp = \frac{tn}{tn + tp} \times 100$$

$$Ac = \frac{tp + tn}{tp + fn + tn + fp} \times 100$$

$$MCC = \frac{(tp)(tn) - (fp)(fn)}{\sqrt{(tp + fp)}\sqrt{(tp + fn)}\sqrt{(tn + fp)}\sqrt{(tn + fn)}} \times 100$$

Where tp and tn are correctly predicted positive and negative examples, respectively. Similarly, fp and fn are wrong predicted as positive and negative patterns respectively. These equations ultimately give the SVM score of the test set with the help of model files.

ROC (Receiver Operating Curve) :

Overall performance measurements also involves one very important aspects i.e. is ROC (Receiver Operating Curve). ROC is a method of projecting the performance of the model in graphical form. ROC represents the binary classification of the given data (Swets *et al.*, 2000). It is basically used to represent the hit rates of true and false rates in data classification done by classifier.

As it is already mentioned that data contains positive as well as negative set, a classifier mapped these instances and generate the model after classification. This model contains four possible results those are t_p (true positive) instances which are sorted as positive and in actual they are positive, f_n (false negative) due to error if any positives are accounted as negative, t_n (true negative) these are negative cases which are accounted as negatives and f_p (false positive) these cases are negative but accounted as positives.

These outcomes generated a confusion matrix. On basis of these outcomes ROC has generated. ROC graph is a two dimensional graph where X-axis represents 1-Specificty that is false positive and Y – axis represents sensitivity i.e. true positive.

AUC (Area Under Curve) this term represent the area covered by curve in the plot (Hanley and Mneil *et al.*,, 1982)The area of the plot ranges from 0 to 1.0 and divided at 45 degree, at this diagonal area is considered as 0.5,0.5(Bradley *et al.*,,1997). Hence in graph more the area covered better the performance of the classifier model.

5.1 Analysis

On analyzing the results of amino acid composition and di-peptide it was found that certain amino acids content is more as compare to others. TIM- barrel protein fold is rich in amino acid like Aspartic acid and Leucine (Leu) as compare to non-TIM- barrel protein folds.

The dominance of these amino acid residues confers the major content of protein fold. In order to confirm this thing we counted the amino acids and dipeptide composition with help of perl script which shows the abundance of Asp and leucine in TIM-barrel protein fold.

Sequences containing TIM barrel fold were selected and assigned positive, and sequences having fold other than TIM-barrel, assigned negative.

The SVM based model was able to achieve maximum MCC 1.679 and accuracy 84.93% for amino acid composition and maximum MCC 1.489 and accuracy 75.78% for dipeptide count by using leave one out method.

Previously, it was observed that evolutionary information obtained from multiple sequences alignment provides more comprehensive information about protein as compare to single sequence.

In current study evolutionary information about protein deduced from position specific substitution matrix (PSSM) generated PSI-blast profiles was also used for predicting TIM-barrel protein fold. Performance increased significantly when PSSM was used as input instead of single sequence. For PSSM maximum MCC was 1.72 and 94.34% of accuracy.

Table 1.2: Performance of SVM classifiers for various combination of training features , kernels and values

Feature	Kernel	Sn	Sp	Accuracy	MCC
AAC	R	80.14	87.34	84.93	1.679
DPC	R	77.29	77.10	75.78	1.489
PSSM	R	89.70	89.15	94.34	1.720

AUC (Zweig *et al.*, 1993) for the above used selection features i.e. for Amino Acid composition, Dipeptide, PSSM are .8644, .8166, .93.21 respectively. Curve near to 1 show the more accurate the model. By comparing roc plot (DeLong *et al.*, 1988) of all three we can estimate that PSSM profiles classification give the best result.

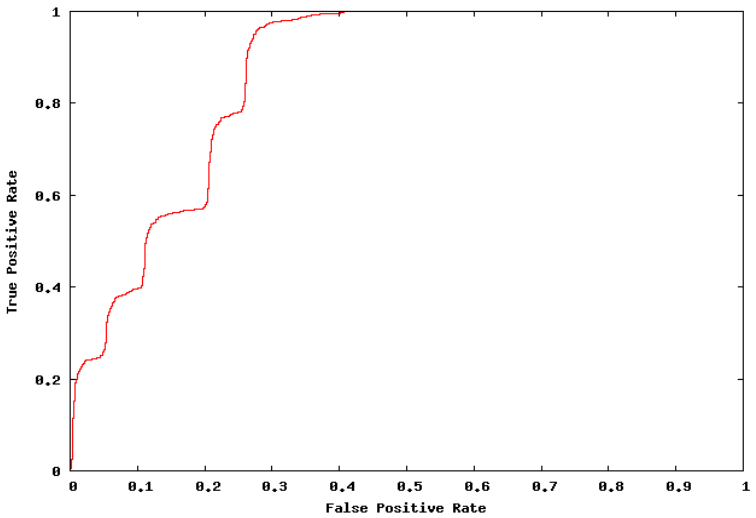


Figure 1.11: Roc plot for AAC feature area under curve in the graph.

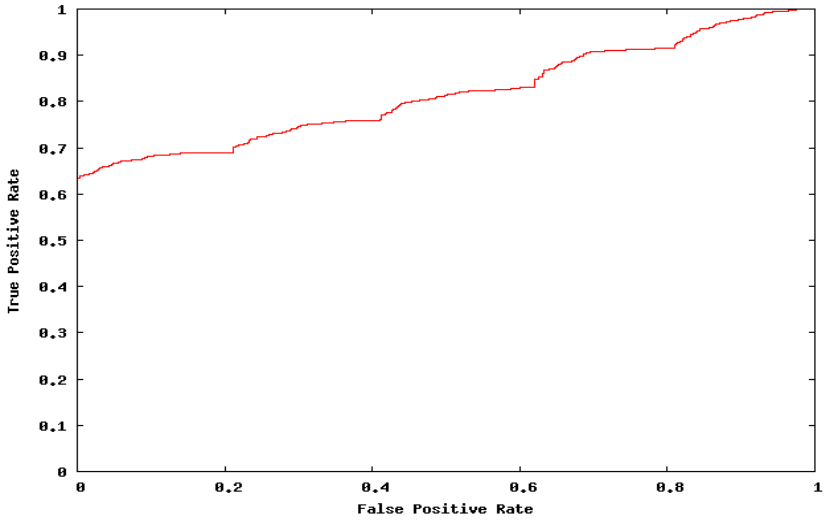


Figure 1.12: ROC plot for dipeptide composition, area under curve is 0.8166

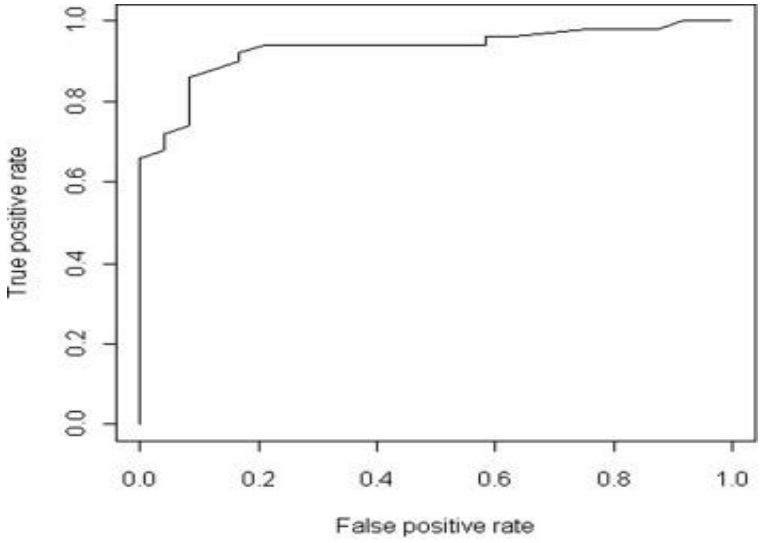


Figure 1.13 : Roc plot for PSSM profiles. Area under curve is 0.932

Functionality of TIMPRED webserver:

1) Open the webserver by typing the url which is currently working on localhost. The home page appears. Snapshot of the webserver with its working given below.

2) Homepage describe about the TIM barrel. It also consist various links such as Submit, Tim-Barrel proteins, FAQs, Help, Team.

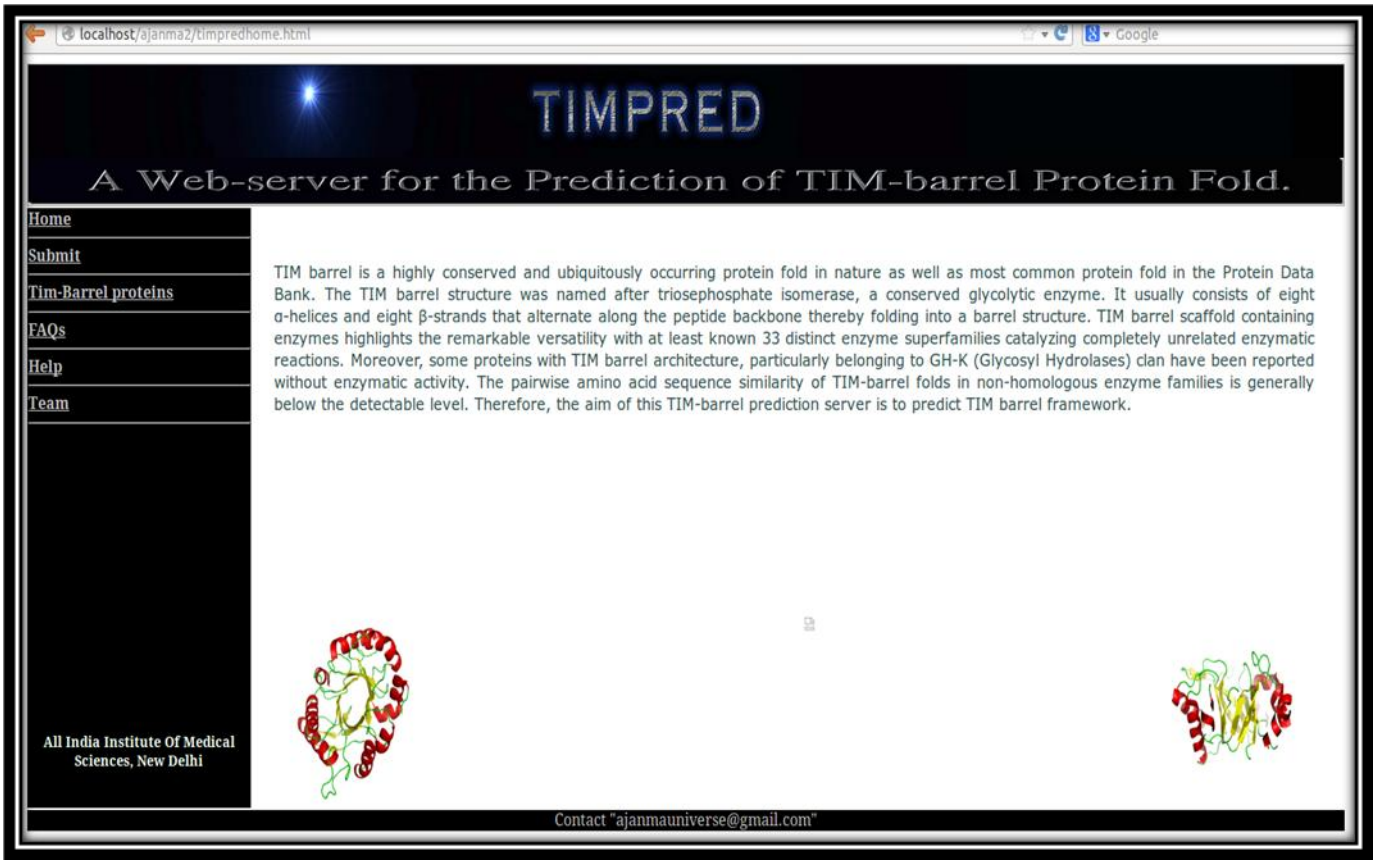


Figure 1.14: TIMPRED home page

3) By clicking on option Tim-Barrel proteins another linked webpage will open. This page descibes about the Tim barrel proteins. The snapshot is given below.

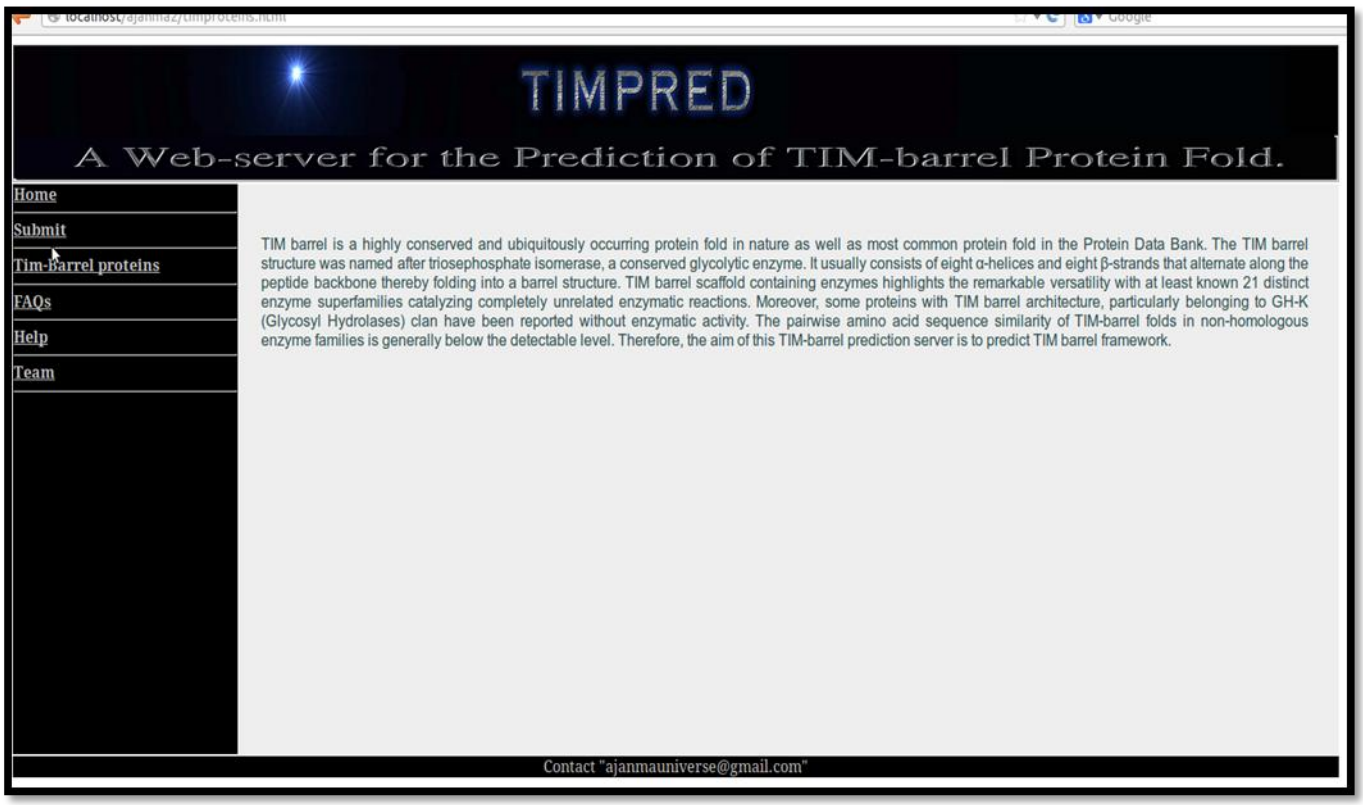


Figure 1.15: Tim barrel webserver next page describing about Tim barrel proteins

4) To check whether query sequences contain tim-barrel fold or not click on Submit button.

5) Now paste or upload the fasta format of the protein.

6) Choose the feature which you want to select by clicking on the radio button , lets say we select Amino Acid Composition and click on submit button.

7) The snapshot is given below.

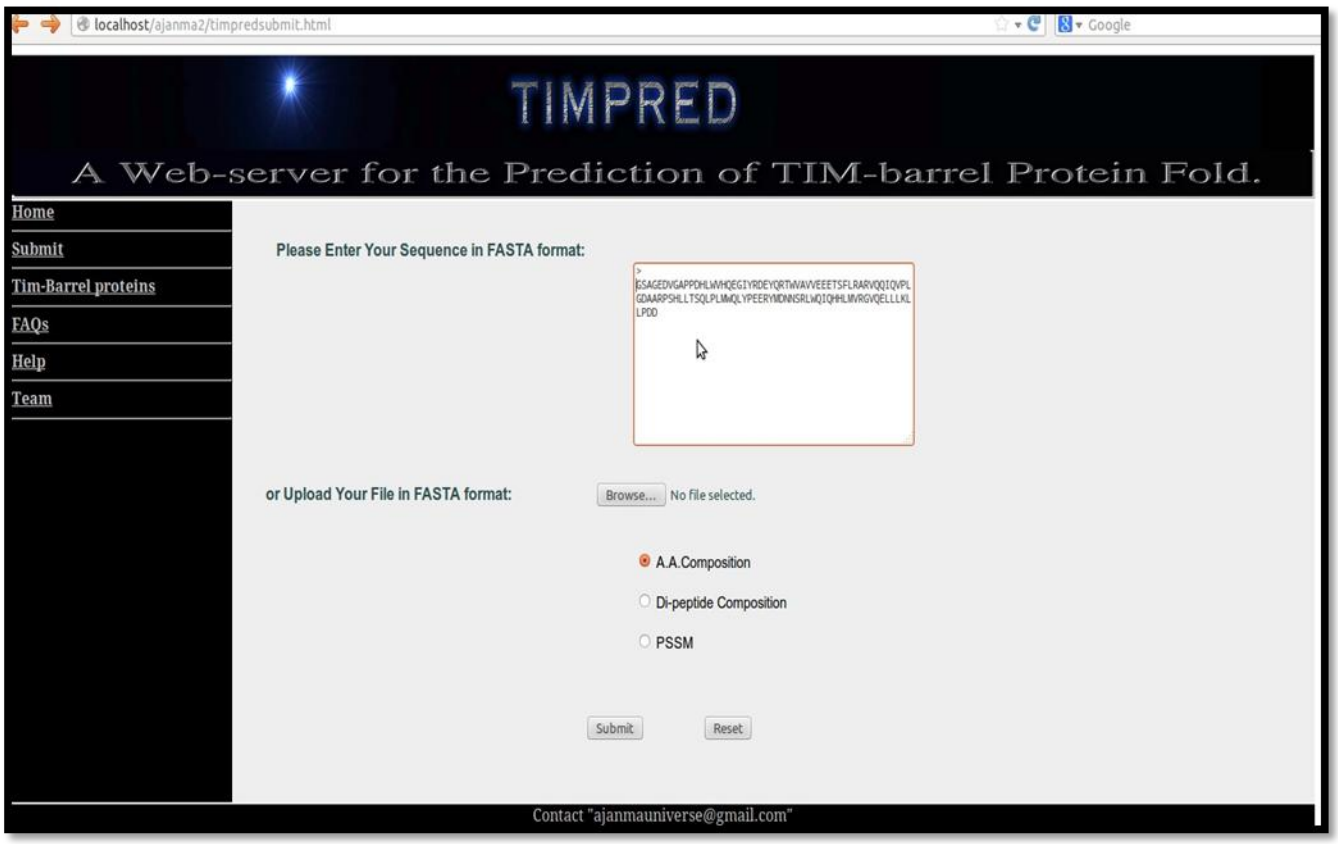


Figure 1.16: Submission page: Protein sequence can be submitted in FASTA format.

8) On submission of the query sequences the very next page comes which generates the job id. User can perform the follow up by the help of jobid. Snapshot is given below:

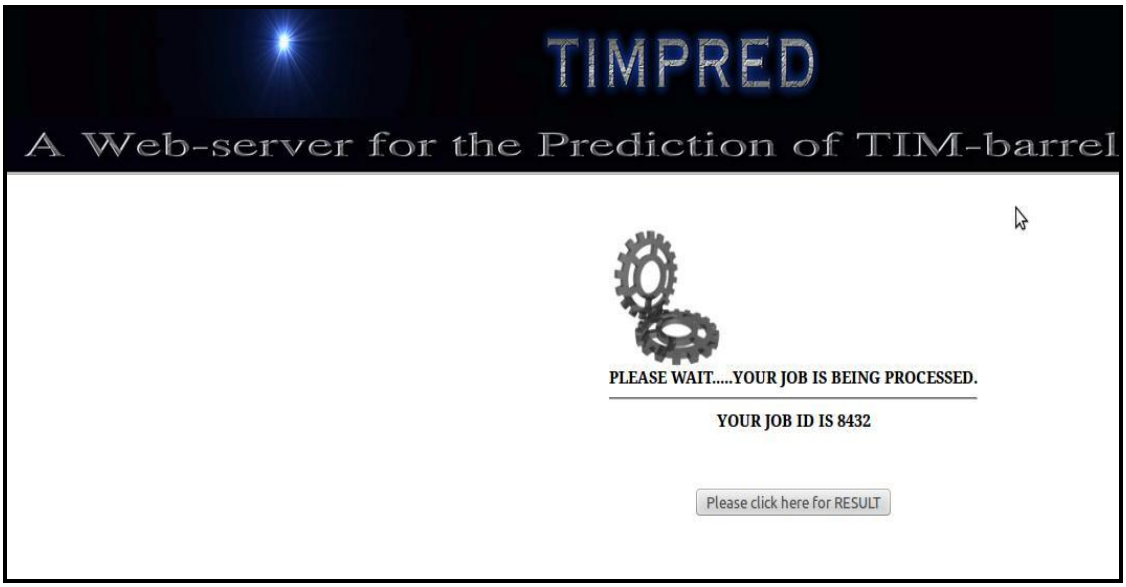


Figure 1.17: This snapshot gives the detail of next page after submission. This pages generates the JOB Id.

9) Now the result page wil display the SVM-score. For the given example it is negative, which indicates the example sequence does not conatin Tim- barrel fold.

10) Result can be downloaded by clicking on the link mention above Download result.

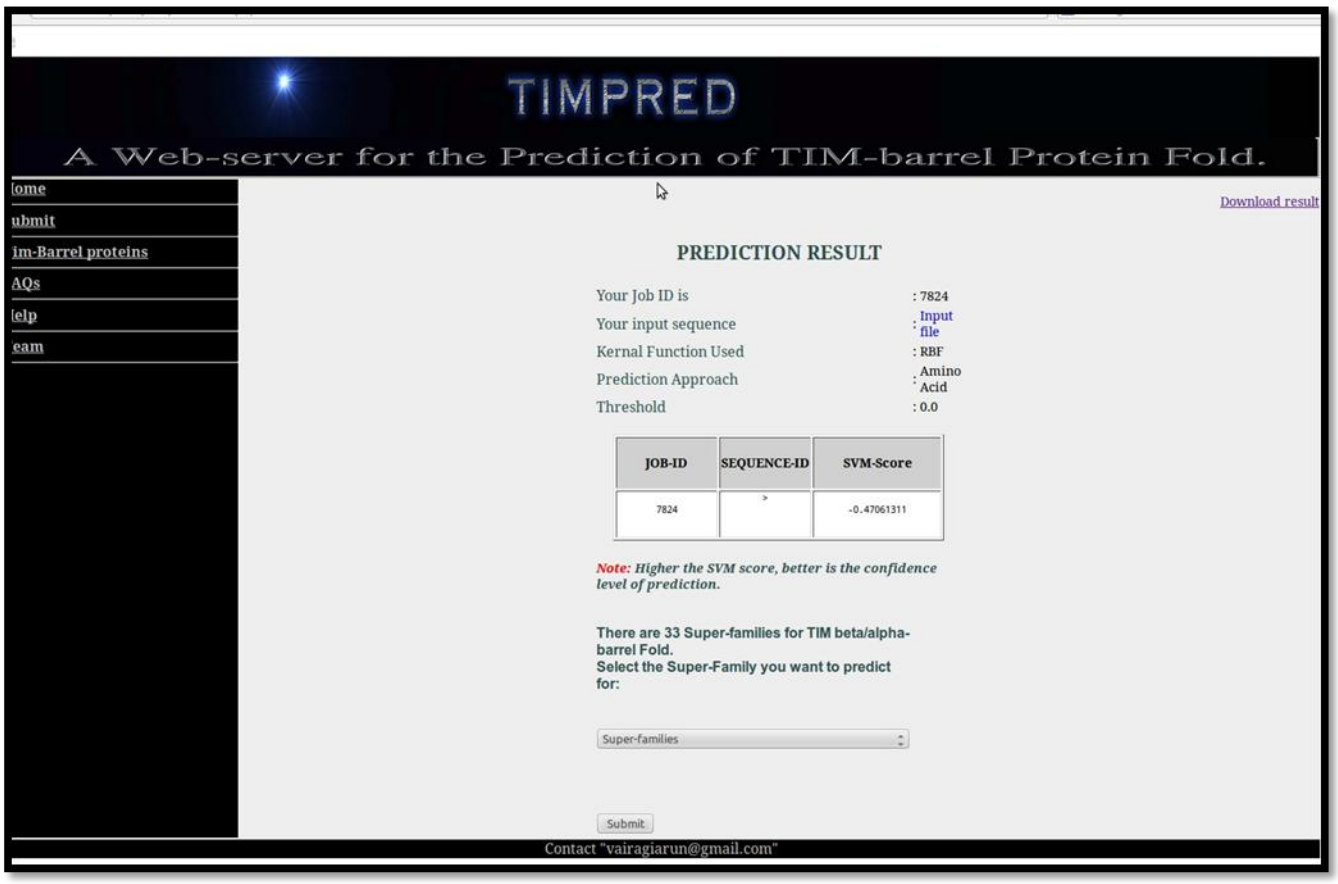


Figure 1.18: The snapshot of result page displaying the SVM score.

10) If any kind of queries related to TIMPRED webserver user has then user can click on TIM-FAQ and can go through their doubts and clear them.

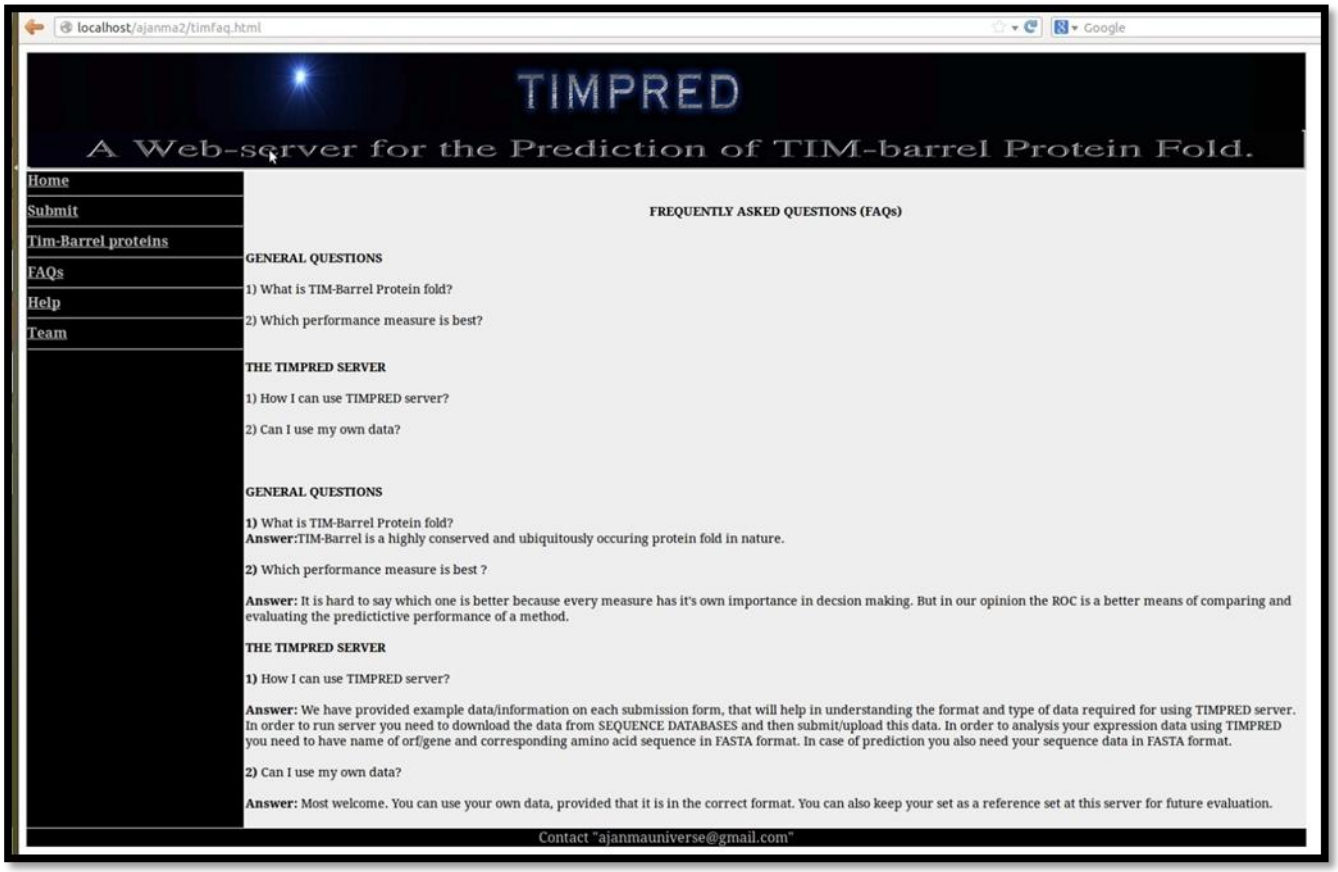


Figure 1.19: FAQs related to TIMPRED webserver.

11) Webserver also have Help option, user can click on help option and go through solutions or procedure to operate the webserver.

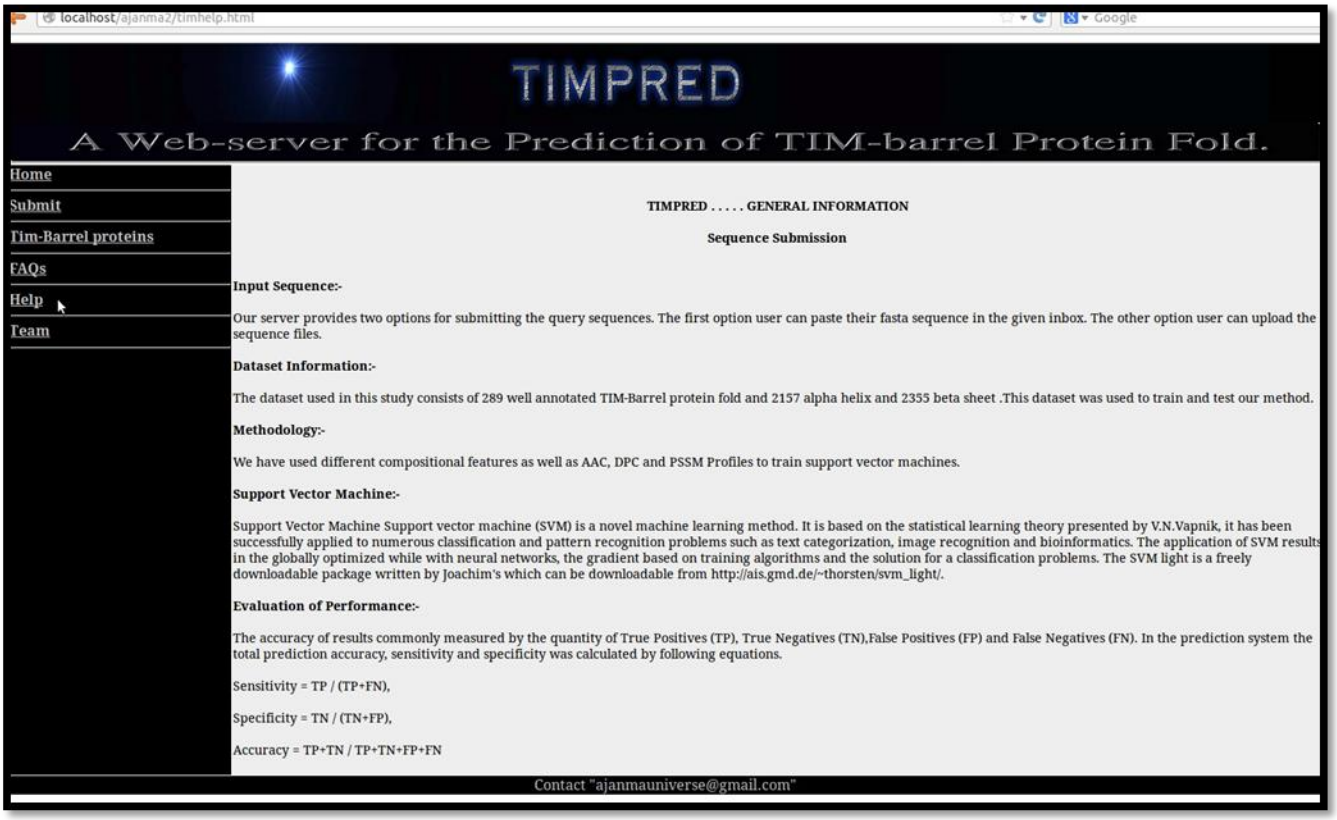


Figure 1.20 : TIMPRED webserver help page for users.

DISCUSSSION and FUTURE PRESPECTIVES

To facilitate the community's research, a web server of TIMPRED was constructed and will be freely available. Due to world wide effort of structural genomics projects, the number of known three-dimensional protein structures rapidly increases.

It is now even frequent that structures are determined prior to any knowledge of their biological function. The ability to predict the details of the protein function and their biological role from structure becomes thus of great importance.

To date several prediction (Ramana, J *et al*., 2009) tools are available but none is devoted solely to TIM-barrel fold and its super family prediction. To sufficiently represent the known structural TIM-barrel proteins as well as a reasonable computational time.

TIM-PRED has been developed to predict whether a protein belongs to the TIM-barrel fold. With the assistance of Support Vector Machine (SVM) TIM-pred combined three different descriptors which are the amino acid composition based, dipeptide composition based, position specific weight matrix based sequence alignment. TIM-Finder scans a database of TIM-barrel proteins, calculates the scores based on the established SVM models, output results as SVM score.

This study will be helpful for biologist in proteome annotation (Saha, S.*et. al.*, 2008). One of the major advantages of this study is that we developed free web server; TIMPRED. Our web server allows users to identify TIM-barrel fold containing proteins in given sequences using the model trained on our dataset.

The input is the query protein sequence with FASTA format. A session ID will be generated when you submit a sequence, and you can query the result through this ID when the processing is ready. The result page displays the output in the form of svm scores.

The proposed TIM-barrel protein identification system, gives highly accurate results. It has been intensively benchmarked to have good performance, suggesting that it can serve as a powerful predictor to be practically applied in proteome-wide TIM-barrel protein detection.

Concerning future development, the following aspects should be taken into account to obtain a more comprehensive prediction system. 1) From the viewpoint of structural biologists, it may be more interesting to target new TIM-barrel superfamily proteins.

Therefore, in the future version of TIM-pred, we may consider including a prediction option to indicate whether a query sequence belongs to a new TIM-barrel superfamily.

REFERENCES

1. Aarti Garg, Manoj Bhasin, and Gajendra P. S. Raghava (2005). SVM-based method for subcellular localization of human proteins using amino acid compositions, their order and similarity search. J. Biol. Chem. 280:14427-32.
2. Aho, Alfred V.; Sethi, Ravi, and Ulman, Jeffrey D., Compliers: Principles, Techniques and Tools (ISBN 0-201-10088-6) link to publisher. Also known as "The Drago Book."
3. Albery, W.J.; Knowles, J.R.(1976), "Free Energy Profile for the Reaction Catalyzed by Triosephaphosphate Isomerase." Biochemistry 15 (25): 5627-5631.
4. Allen, Frances E. (1981), "A history of Languages Processor Technology in IBM," IBM Journal of Research and Development", v.25, no.5.
5. Alexey G. Murzin , S. E. B., Tim Hubbard and Cyrus Cthothia,(1995) "SCOP: A structural classification of proteins and database for the investigation of sequences and structures". J MolBiol 247: 536-540.
6. A. Smola, B. Scholkopf, and K.-R. Muller (1998), "The connection between regularization operators and support vector kernels.Neural Networks", 11:637–649.
7. Bradley, A.P., (1997). "The use of area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition" 30(7), 1145-1159.
8. Bhasin M, Raghava GPS , (2005). "GPCRsclass: a web tool for the classification of amine type of G-protein-coupled receptors". Nucleic Acids Res. 33(Web Server issue):W143-7.
9. Byvatov E, Schneider G., (2004) "SVM-based feature selection for characterization of focused compound collections". J Chem Inf Comput Sci. ;44(3):993-9.PubMed PMID: 15154767.
10. C.J.C. Burges, (1998), "A tutorial on support vector machines for pattern recognition", *Knowledge Discovery and Data Mining, 2(2)*.
11. Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen L,(updated April 15, 2010) , "A Practical Guide to Support Vector Classication, Department of Computer Science".

12. CHRISTOPHER J.C. BURGE (1998). "A Tutorial on Support Vector Machines for Pattern Recognition" Bell Laboratories, Lucent Technologies, Data Mining and Knowledge Discovery 2, 121-167.
13. Dayhoff, M.O., McLaughlin, P.J. Barker, W.C., and Hunt, L.T.(1975)."Evolution of sequences within protein and superfamilies", Naturwisenchaften 62, 154-161, 1975.
14. Dayhoff, M.O. (1974). "Computer analysis of protein sequences", Fed. Proc. 33, 2314-2316,
15. Debashish Basak[1], Srimanta Pal[2], and Dipak Chandra Patranabis[3] (2007 Oct). "Support Vector Regression" ; Neural Information Processing – Letters and Reviews, Vol 11., No. 10.
16. DeLong, ER, DeLong, DM, Clarke-Pearson, DL (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics 44, 837-845
17. Furey TS, Cristianini N, Duffy N, Bednarski DW, Schummer M, Haussler D (Oct, 2000). "Support vector machine classification and validation of cancer tissue samples using microarray expression data." Bioinformatics. 16(10):906-14. PubMed PMID: 11120680.
18. Garde A, Voss A, Caminal P, Benito S, Giraldo BF(2013 Jun). "SVM based feature selection to optimize sensitivity- specificity balance applied to weaning". Comput Biol Med.:43(5):533-40.doi:10.1016/jcompbiomed. 2013.01.014. Epub 2013 Mar 20. PubMed PMID: 23566399.
19. Garg, A., Gupta, D (2008)."VirulatenPred : "A SVM based prediction method for virulent proteins in bacterila pathogens". BMC Bioinformatics 9, 62.
20. Hanley, J.A., McNeil, B.J. (1982). "The meaning and use of the area under a receiver operating characteristic (ROC) curve". Radiology 143, 29–36.
21. Hegyi, H. and Gerstein, M (1999) . "The relationship between protein structure and function: a comprehensive survey with application to the yeast genome " J. Mol. Biol. 288, 147-164.
22. "Jeong E, Miyano S (2006)." A Weighted profile based method for protein-RNA interacting residues prediction. In: Corrado P, Luca C, Stephen E, editors. Lecture notes in computer sciences, Vol.3939. Berlin/Heidelberg: Springer.

23. Joachims,T. (1999). “Making large-scale SVM learning practical”. In Scholkopf,B., Burges,C. and Smola,A. (eds), Advances in Kernel Methods Support Vector Learning. MIT Press, Cambridge, MA and London, pp. 42–56.

24. Juers, D.H.J., Huber, R. and Matthews, B.W. (1999). “Structural comparisons of TIM barrel proteins suggest functional and evolutionary relationships between β-galactosidase and other glycohydrolases ”, Protein Sci.8, 122-136

25. Kaur H, Raghava GPS (2003). “Prediction of b-turns in proteins from multiple alignment using neural network.” Protein Sci; 12:627-634.

26. Klotz SA, Gaur NK, De Armond R, Dheppard D, Khardori N(2007). Candida albicans. Als proteins mediate aggregration with bacteria and yeasts. Madeical Mycology 45: 363-370.

27. Kumar M, Grohima MM, Raghava GPS (2008). “Prediction of RNA binding sites in the protein using SVM and PSSM profile”. Proteins, BMC Bioinformatics 71:189-194, 2008.

28. “Mishra, N.K. and Raghava, G.P.S. (2010). “Prediction of FAD interacting residues in a protein from its primary sequence using evolutionary information”. BMC Bioinformatics 11:S48.

29. Murzin AG, Brenner SE, Hubbard T, Chothia C (1995). SCOP: A structural classification of proteins and database for the investigation of sequences and structures”. J MolBiol 247: 536-540.

30. Nagano N, Orengo CA, Thornton JM (Aug, 2002). “One fold with many functions: the evolutionary relationships between TIM barrel families based on their sequences, structures and functions”. J Mol Biol ;321(5):741-65. Review. PubMedPMID: 12206759.

31. Nobile CJ, Schenider HA, Nett JE, Sheppard DC, Filler SG (2008). Complementary Adhesion Functions in C.albicans Biofilm formation. Current Biology 18: 1017-1024, 2008.

32. Norman Matloff (May, 2007), “A Quick, Painless Introduction to the Perl Scripting Language”.

33. “Ochoa-Leyva, A. *et al.*”. “Protein design through systematic catalytic loop exchange in the (beta/alpha) 8 fold”. *J Mol. Biol*387 (10): 949-964, 2009.

34. "Ochoa-Levya, A.*et al.*" "Exploring the Structure-Function Loop Adaptability of a (beta/alpha) 8-Barrel Enzyme through Loop Swapping and Hinge Variability." *J Mol. Biol.* 411(1): 143-147, 2011.
35. Ramana, J. Gupat, D. FaaPred" A SVM-Based Prediction Method for fungal Adhesions and Adhesin-Like Proteins. PLos One 5, 29695, 2010.
36. Ramana, J., Gupta D.". Machine Learning Methods for Prediction of CDK-Inhibitors. PLoS One5, e13357Ramana, J., Gupta, D. 2009. LipocalinPred: A SVM-based method for prediction of lipocalins. BMC Bioinformatics 10, 445, 2010.
37. Randal Schwartz, Tom Christiansen & Larry Wall (July 1997); "Oreilly Learning Perl" ISBN 1-56592-284-0, 302 pages. Second Edition.
38. Ravindra Kumar, Sohni Jain, Bandana Kumari, Manish Kumar(June, 2014). PLoS One, 9(6): e98345. Protein Sub-Nuclear Localization Prediction Using SVM and Pfam Domain Information doi: 10.1371/journal.pone.0098345 PMCID: PMC4045734.
39. Richard R Copley (2000). "Homology among $(\beta\alpha)_8$ barrels: implications for the evolution of metabolic pathways" P Bork J. Mol. Biol., 303 pp. 627–641.
40. Saha, S. and Raghava, G.P.S.(2006). AlgPred: "Prediction of allergenic proteins and mapping of IgE epitopes". Nucleic Acids Research 34: W202-9.
41. Sachdeva G, Kumar K, Jain P, Ramachandran S(2005). "SPAAN: A software program for prediction of adhesions and adhesion like proteins using neural networks". Bioinformatics 21: 483-491, 2005.
42. T. Joachims (1998), "Text Categorization with Support Vector Machines: Learning with Many Relevant Features". Proceedings of the European Conference on Machine Learning, Springer.
43. Terribilini M, Lee JH, Yan C, Jernigan RL, Honavar V, Dobbs D (2006). "Prediction of RNA binding sites in proteins from amino acid sequences". RNA; 12:1450-1462.
44. Vijayabaskar MS, Vishveshwara S (2012). "Insights into the fold organization of TIM-barrel from interaction energy based structure networks". PLoS Comput Biol.;8(5):e1002505. doi: 10.1371/journal.pcbi.1002505. Epub PubMedPMID: 22615547; PubMed Central PMCID: PMC3355060.

45. S Selvaraj, M.M Gromiha J. Protein Chem., 17 (1998)," An Analysis of the Amino Acid Clustering Pattern in $(\alpha/\beta)_8$ Barrel Proteins" pp. 407–415.
46. Stephen Winters-Hilt, Anil Yelundur, Charlie McChesney, Matthew Landry (2006)." Support Vector Machine Implementations for Classification & Clustering" BMC Bioinformatics. 2006; 7(Suppl 2): S4. 1471-2105-7-S2-S4 PMCID: PMC1683575.
47. Swets, J.A., Dawes, R.M., Monahan, J., (2000). "Better decisions through science". Scientific American 283, 82–87.
48. Traut, T. and Temple, B.R.S. (2000). "The Chemistry of the Reaction Determines the Invariant Amino Acids during the Evolution and Divergence of Orotidine 5′-Monophosphate Decarboxylase". J. Biol ,Chem. 275, 28675-28681.
49. Xiaoyan Yang, Sagar V. Kathuria, Ramakrishna Vadrevu, C. Robert Matthews(2009)): "βα-Hairpin Clamps Brace βαβ Modules and Can Make Substantive Contributions to the Stability of TIM Barrel" PLoS One e7179. doi: 10.1371/journal.pone.0007179 ; PMCID: PMC2747017.
50. Zaffagnini M, Michelet L, Sciabolini C, Di Giacinto N, Morisse S, Marchand CH,Trost P, Fermani S, Lemaire SD (2014). High-resolution crystal structure and redox properties of chloroplastic triosephosphate isomerase from Chlamydomonas reinhardtii. Mol Plant. 2014 Jan;7(1):101-20. doi: 10.1093/mp/sst139. Epub 2013 Oct 24. PubMed PMID: 24157611.
51. Zweig, MH, Campbell G (1993). Receiver-operating characteristic (ROC) plots: A fundamental evaluation tool in clinical medicine. Clin Chem 39/4, 56-577.