Prediction of residue-residue contact matrix for protein-protein interaction with Fisher score features and deep learning

by

Ranojoy Barua

Summer 2017

OBJECTIVE

Our main objective is to implement the paper “Prediction of residue-residue contact matrix for protein-protein interaction with Fisher score features and deep learning” written by Tianchuan Du , Li Liao, Cathy H. Wu, Bilin Sun and for this we will get data from 3DID and Pfam database and use the HMMER software to make pHMM, after this we need to generate fisher score and train the model using an autoencoder which is type of unsupervised deep neural network.

TABLE OF CONTENTS

PART - 1

Deep Neural Network
Working principle of neural networks
Normal neural network
Convolutional neural network
Autoencoders

Neural Network

Introduction

Artificial neural network in a computational model used in machine learning, computer research and other disciplines. Artificial neural network tries to mimic the function of the human brain. ANN is based on a large collection of connected simple units called artificial neurons. Each point in the computation graph of the ANN is termed as Neurons.

Analogy of ANN with human brain

This is the image of the neurons in the human nervous system including brain. The neurons reads signals in chemical or electrical form via the dendrites, processing on this signal is done in the cell body and if the signal is important it is transmitted to another neuron.

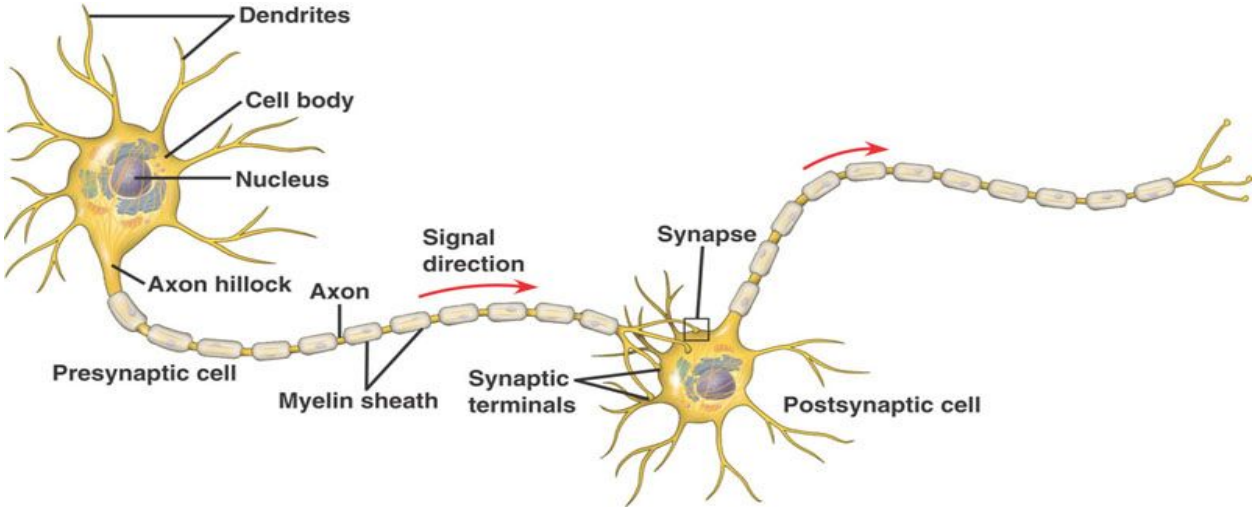

Figure - 1.1

Similarly the structure of the artificial neural network is similar to the working of the human brain. In figure 1.2 the structure of the artificial neural network is explained.

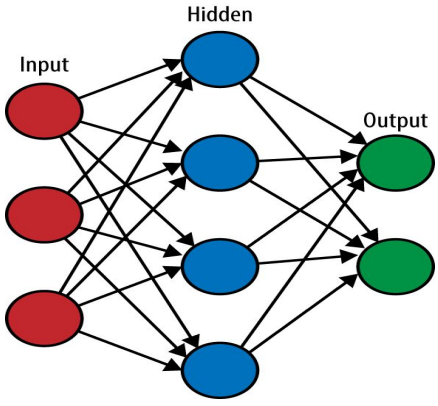

Figure - 1.2

In this network or diagram, each node is said as one independent individual neuron in neural network. Each layer is denoted by one particular color i.e., all nodes having same color belong to same layer e.g., input layer is denoted by red color, hidden layer by blue color and output layer by green color. There can be more than one hidden layer depending upon the requirement of the network.

The input layer is analogous to the dendrites i.e., data is given to the neural network via the input layer the cell body is denoted by the hidden layer/layers where processing on the data takes place and an non-linear function (activation function) is applied and finally the output layer is analogous to the synapse terminals.

Single and Multiple Layer Neural Network

If a neural network have only one layer of hidden state involved the the neural network is termed as single layer neural network. Similarly, if a neural network is having multiple layer of hidden states then it is called multiple layer neural network or even deep neural network.

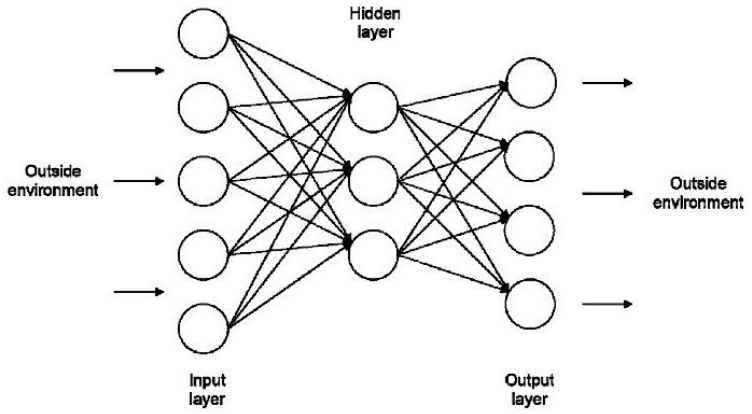

Figure - 1.3 (Single layer neural network)

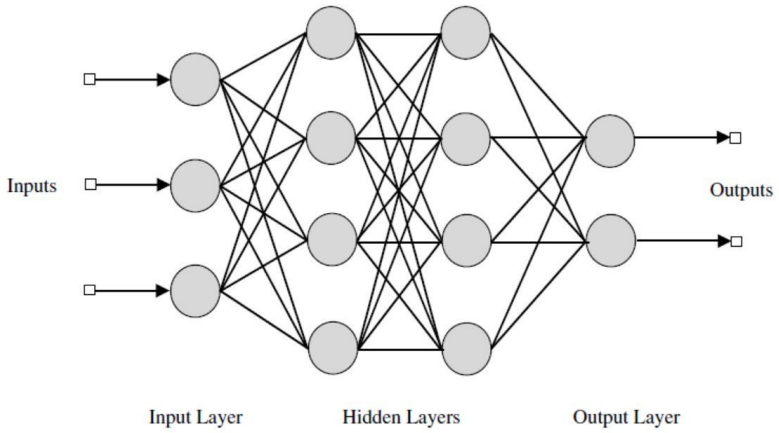

Figure - 1.4 (Deep neural network)

Working of neural network

Let us consider a simple single layer neural network with two inputs in the network and two output states. Let the inputs be i1 and i2 and the weights in the layers are generated randomly i.e., w1, w2, w3 and w4 are generated randomly and multiplied to the input i1 and i2. Similarly the weights of the bias are generated randomly and multiplied to the bias 1.

So input to state

$$h1 = i1*w1 + i2*w2 + 1*b1$$
$$h2 = i1*w3 + i2*w4 + 1*b1$$

and

$$o1 = h1*w5 + h2*w6 + 1*b2$$
$$o2 = h1*w7 + h2*w8 + 1*b2$$

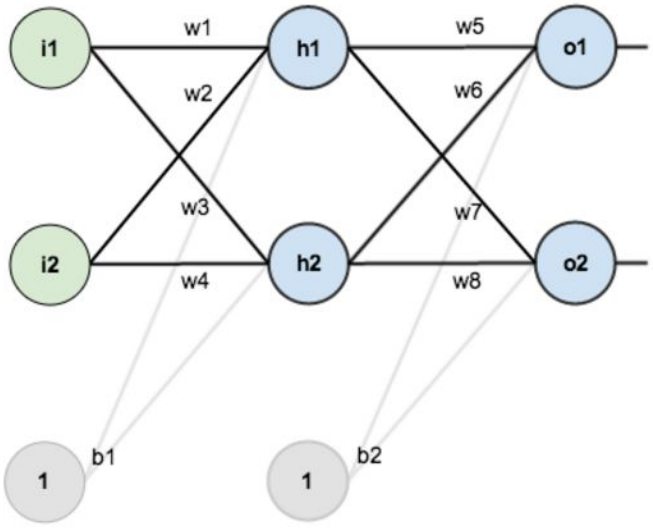

Figure - 1.5

The result obtained in o1 and o2 are some random value. The error can be calculated as we know the original value and we have the calculated value. Now our aim will be to reduce this error percentage or cost of calculation. For this we implement backward propagation algorithm.

Note: *Here after calculation of h1 and h2 we didn't applied the activation function. When implementing neural network we should implement activation function to introduce non-linearity in the equation and we need to introduce bias in the computation tree because if in a case there is no input then none of the neurons will fire to overcome this problem we need to introduce bias. Neural network works best if it is randomised at maximum possible ways.*

Backward propagation algorithm

Our goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer the target output i.e., to minimize the cost of computation, thereby minimizing the error for each output neuron and the network as a whole.

Updating the weights of the neural net

Consider w5, We want to know how much a change in w5 affects the total error or cost of computation i.e., partial derivative of total error with respect to w5. ($\delta Etotal / \delta w5$).

By applying chain rule we can say

$$\delta Etotal / \delta w5 = \delta Etotal / \delta out(o1) * \delta out(o1) / \delta net(o1) * \delta net(o1) / \delta w5$$

Basically this is what is done here,

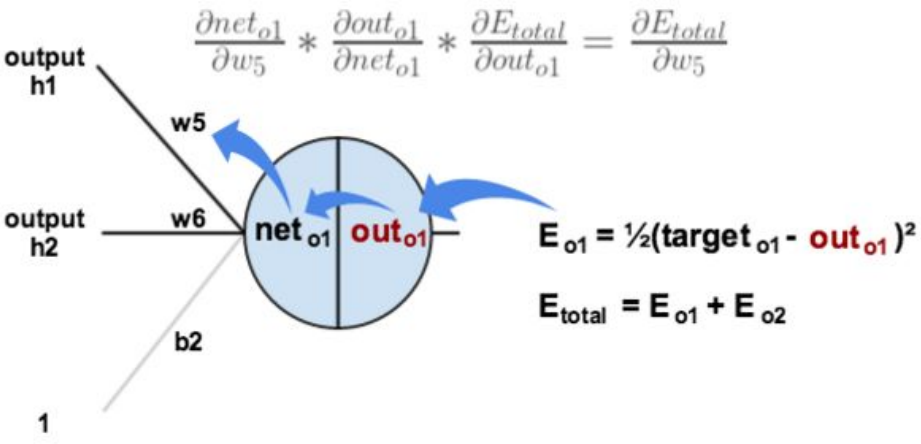

Figure - 1.6

Now to update w5 we use the formula

w5 = w5 - (learning rate) * ($\delta Etotal / \delta w5$)

Now we have updated one of the weights, next we need to update all of these weights. By, updating all of the weights we will reach to first layer or layer-1. Then we again start all over again from layer-1 multiply weights and add biases and when we reach output layer or final layer we again use back propagation to calibrate the weights and continue this process until the error comes under a certain limit or we can also specify the number of epochs for the network. (1 epoch = 1 forward propagation + 1 backward propagation)

Activation Function

Activation functions are all nonlinear function. They are implemented to introduce nonlinearity in the network. There are many activation functions available

1) Identity function
2) Binary step function
3) Sigmoid function
4) Tanh function
5) Arc Tanh function
6) ReLU function
7) Leky ReLU
8) Soft Max function

1) **Identity function:** This is the most basic activation function.

$$f(x) = x$$

2) **Binary step function:** This function is helpful in classification problem as it divides the total function in two parts.

$$f(x) = 0 \quad \text{for } x < 0$$
$$1 \quad \text{for } x >= 0$$

3) **Sigmoid function:** Sigmoid function helps to squash the data in a range between of 0 and 1. But the problem with this function is with time the gradient or non-linearity of the network is compromised and even this function is not 0 centered.

$$f(x) = 1 / (1 + e^{-x}) \qquad 0 < f(x) < 1$$

4) **Tanh function:** This function is 0 centered but with respect to time the gradient is lost.

$$f(x) = [2 / (1 + e^{-2x})] - 1 \qquad -1 < f(x) < +1$$

5) **Arc Tanh function:** In this function gradient is not lost and the function is 0 centered.

$$f(x) = \tan^{-1}(x) \qquad -\pi/2 < f(x) < \pi/2$$

6) **ReLU function:** According to the paper *ImageNet classification with deep neural network by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton from University of Toronto* found that ReLU is 6x times better than TanH activation function.

$$f(x) = 0 \quad \text{for } x < 0$$
$$x \quad \text{for } x >= 0$$

7) **Leky ReLU function:** This function is similar to ReLU but allows negative data by reducing the magnitude of it.

$$f(x) = 0.01x \quad \text{for } x < 0$$
$$x \quad \text{for } x >= 0$$

8) **SoftMax function:** This function gives the output in form of probability among all the class and class having highest score is winner.

$$f(x)_j = e^{xj} / \Sigma e^{xk}$$ summation is over k from 1 to k.

Introduction to Architecture of Deep Neural Networks

The deep neural network is a concept and it is implemented in different ways to solve different problems that we face in machine learning, daily life etc. The main advantage of deep neural network in that we don't need to specify a model for learning the features. The deep neural network can be implemented for following types of learning

a) Supervised learning
b) Unsupervised learning
c) Reinforcement learning

These are the some of the mostly used structures of artificial neural network for these three category.

a) Supervised learning: 1) CNN (Convolutional Neural Network)
   2) RNN (Recurrent Neural Network)
   3) LSTM (Long Short Term Memory networks)

b) Unsupervised learning: 1) Auto-encoders
   2) Boltzmann machine

There are many other types of artificial neural networks available. These are some of the most important neural networks. We need to select a neural network architecture based on the type of work that we want the network to do, for example CNN is best for image analysis and image recognition, RNN and LSTM are best for time dependent sequential work such as speech recognition.

The architecture for normal neural network

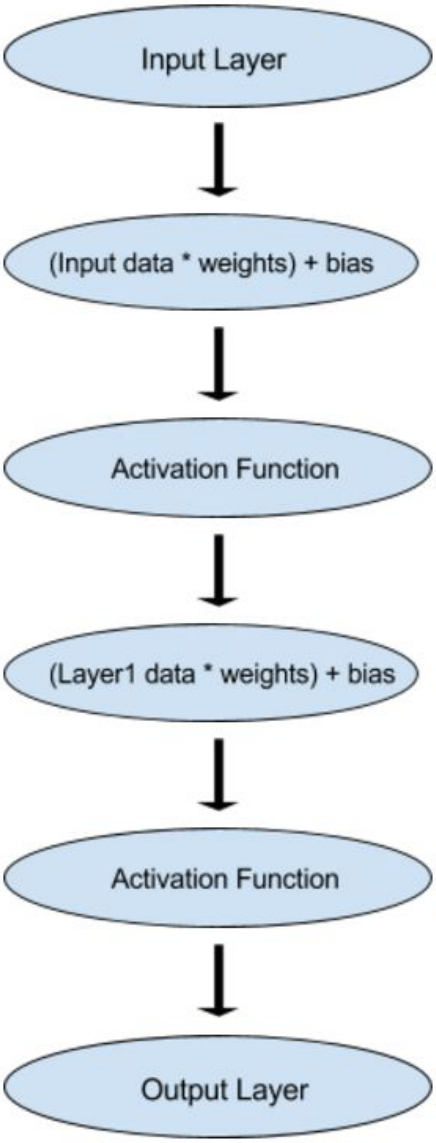

Figure - 1.7
(Architecture of a normal neural network)

Data is given to this neural network via the input layer then in layer-1 the weights are multiplied to the input data and biases are added. Then this data is passed through a activation function to introduce nonlinearity in the computation graph. The output of layer-1 is then passed on to the input of layer-2 where the weights are multiplied and biases are added and then the output is passed on to input of layer-3 this continues for n number of hidden layers in the network then in the output layer same steps are followed and softmax activation function is applied for

classification problem or different activation function is applied based on the type of work that we want to accomplish.

Architecture for convolutional neural network

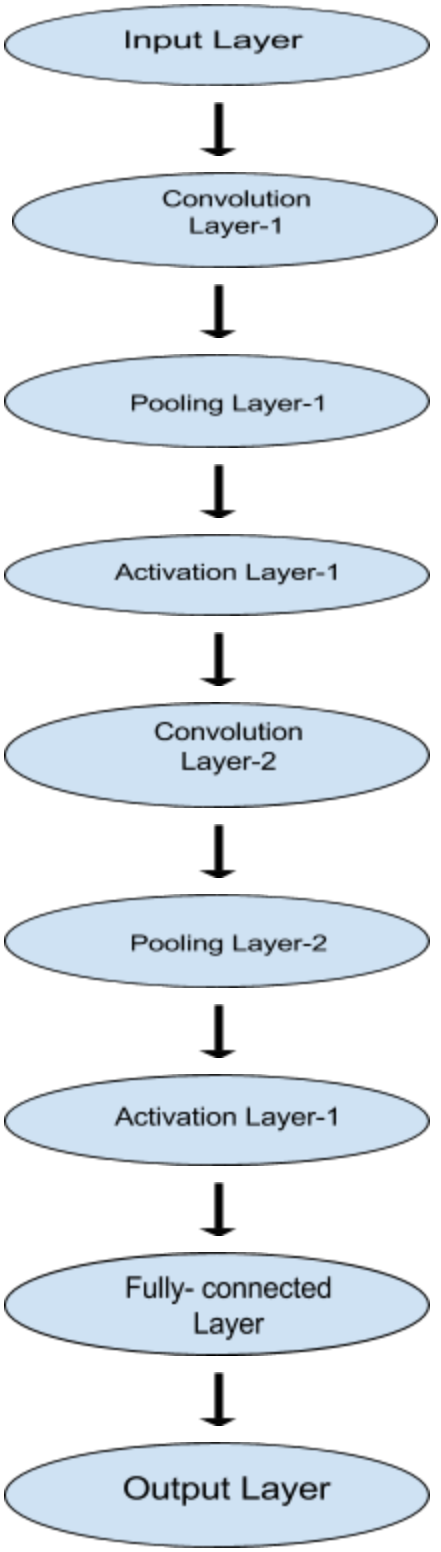

Figure 1.8

Architecture for convolutional neural network

Data is given to the neural network by the input layer then in layer-1 convolution is done on the input data, after convolution pooling is done then the activation function is applied then the output of this layer is treated as the input of the second convolutional layer and this continues for n number of convolution layer after this generally a fully connected layer is applied and on this layer activation function is applied and output of this layer is the input of the output layer.

Convolution Layer

Note: For simplicity of understanding let us take the input data as an image as CNN works best for image.

The image is represented as single dimensional image in a 2D matrix. The size of the matrix is 4X4 and each cell in the matrix holds the data of the image.

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

Figure - 1.9

Now we apply convolution on this image (matrix) with a 2X2 filter as shown in the image with a stride of 1 i.e., we will shift 1 pixel each time after convolution.

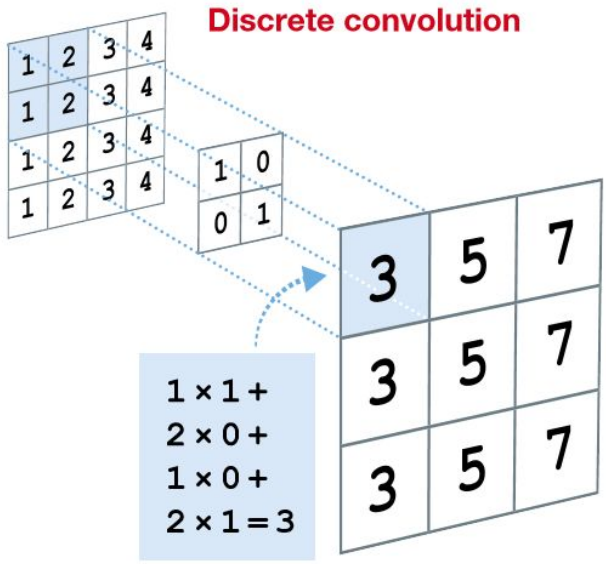

Figure - 1.10

By convolution we can extract unique features form the image such as curves, corners etc. Example of feature extraction using convolutional layer.

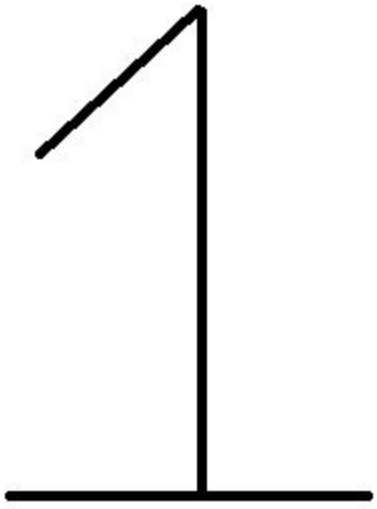

Figure - 1.11
(Let our interest of image is this and by convolution we want to identify if there is any acute angled slope is there in the image or not)

Now represent this image in matrix format for computation. We assume that the size of the matrix is 6X6.

0	0	0	**1**	0	0
0	0	**1**	**1**	0	0
0	0	0	**1**	0	0
0	0	0	**1**	0	0
0	0	0	**1**	0	0
0	0	**1**	**1**	**1**	0

Figure - 1.12
(Representation of the image of interest in binary form i.e., black region is represented by 1 and the white region is represented by 0)

Now we will apply a 3X3 filter on this image to find that does this image have a acute angled line or not. For our convenience we take a filter like this,

0	0	**1**
0	**1**	0
1	0	0

Figure - 1.13
(Filter used on the image)

Now after we apply convolution the result obtained is a matrix of order 4X4

0	2	1	1
1	1	1	1
0	1	1	1
0	1	2	1

Figure - 1.14
(Output obtained after applying convolution with the above mentioned filter on the image of interest, Convolution operation applied is multiplication of the value in image with the filter and adding all of the values as shown in Figure - 1.10)

Now let us assume that the threshold number value of convolution for correctness of the example is 2 and as we see that 2 appeared the matrix so we can say that the image really contained a line with acute angle.

[Note: For more see the video (*Convolution_1.mp4*) in videos folder. Convolution layer works best in the data where the locally present data points are mutually dependent and affect each other and there should be a smooth change of data among the data points, best example of this type of data is images as there is no abrupt change among locally connected data. Convolution layer helps to reduce the dimension of the input without compromising the data. **]**

Pooling Layer

After convolution layer generally pooling layer is applied. Pooling layer helps to further reduce the size of the layer without significantly compromising the data in the layer. There can be various types of pooling operations, such as max-pool, min-pool, average pool etc.

Now we apply max-pool on the **Figure - 1.10**.

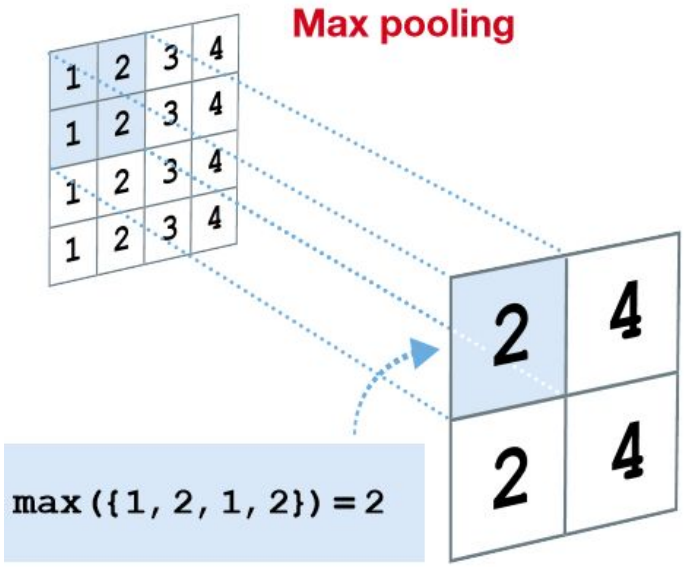

Figure - 1.15
(Max-pool operation on Figure - 1.10)

Similarly if we apply max-pool operation on **Figure - 1.14** with a kernel of 3X3 the obtained result will be

2	2
2	2

Figure - 1.16
(Result obtained if we apply max-pool on Figure - 1.14)

Different types of pooling

1) *Max-pool:* Takes the maximum number in the kernel.
2) *Min-pool:* Takes the minimum number in the kernel.
3) *Average-pool:* Takes the average of all the data in the kernel.
4) We can apply any other type kernels also as required by the network and the input data.

Auto-Encoders

Introduction

In machine learning problem there are mainly two problems Classification and Learning. Now in the above section we have discussed the classification problem, now we will discuss the learning problem briefly. There are many mechanism employed for the learning problem in deep learning but here we will focus on the autoencoders as it is mentioned in the paper.

Autoencoder

This is an mechanism to learn features from an unlabeled data set and it employs unsupervised learning. The basic concept of autoencoder is that in the part of error calculation the obtained value at output layer is compared with the input layer. The architecture of an autoencoder is similar to that of an normal neural network. The number of neurons in the output layer is equal to the number of neurons in the input layer.

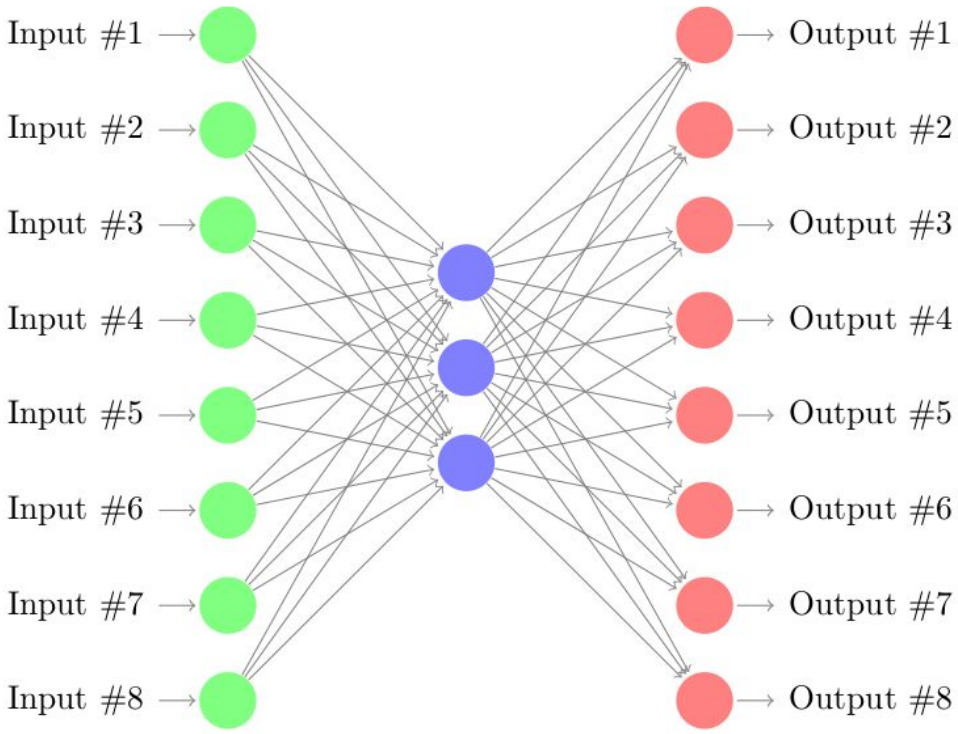

Figure - 1.17
(Architecture of autoencoder)

Now if we employ a large number of autoencoders one in top of another in a format of stack data-type then it is called stacked autoencoder.

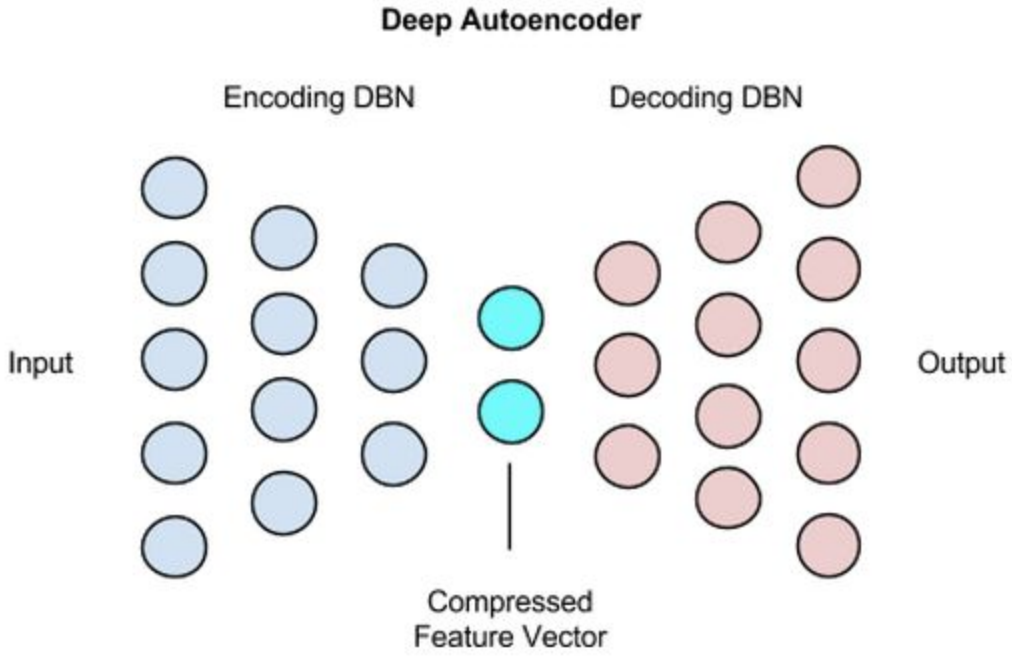

Figure - 1.18
(Architecture of stacked autoencoder)

PART - 2

Hidden Markov Model
Forward Algorithm
Evaluation Problem
Decoding Problem
Learning Problem
Profile Hidden Markov Model
Interacting Profile Hidden Markov Model

HMM (Hidden Markov Model)

Definition

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. HMM is used for temporal pattern recognition such as speech, handwriting, human behaviour, gene sequence etc.

Markov Property

The Current state of the system depends only on the previous state of the system e.g., the present of a human or being is always influenced by the past of that being or entity, or more specifically we can say future of an entity is determined by the past in general case.(**Figure - 2.1**)

There are many states known as hidden states which are not visible or tangible from the outside world but they are present in the computation model they are known as hidden states and each hidden state transmits one symbol known as visible symbol to show the presence of the hidden state. (**Figure - 2.2**)

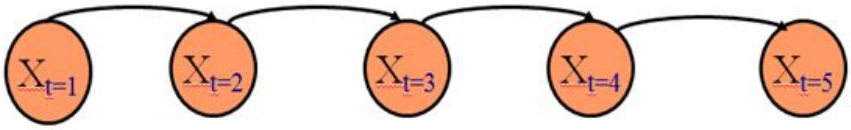

Figure - 2.1
(Value at time step t = 5 depends on data at time step t = 4
i.e., the data at time-step [T] depends on time-step [T-1])

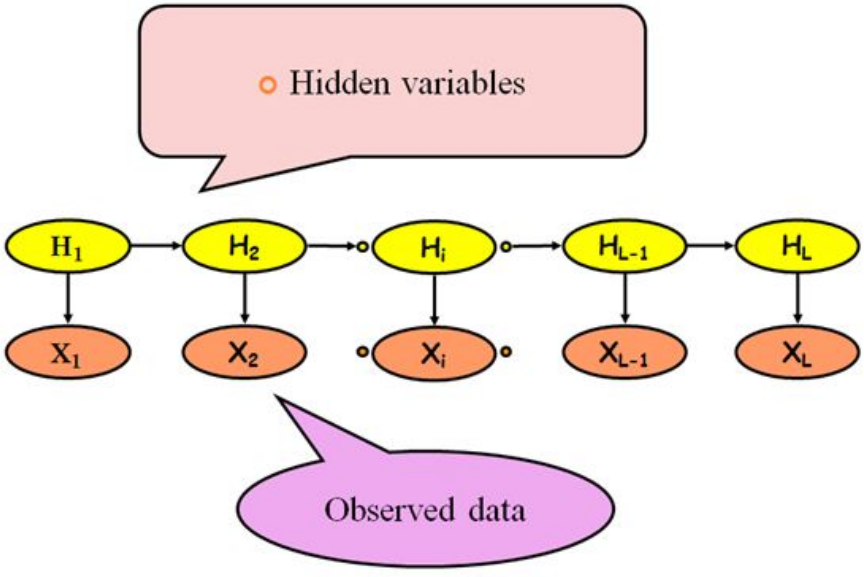

Figure - 2.2

(H_1 , H_2 , … , H_n are the hidden states and each hidden state emits a visible symbol denoted by X_1, X_2, ... , X_n respectively)

Requirement and application on HMM

HMMs are one of the most important tool for many of the complex problems that computer science engineers and scientists face. HMM is the working backbone of neural networks and deep neural networks. HMMs are used for temporal pattern recognition i.e., the patterns that changes with respect to time e.g., human behaviour, DNA sequence, Protein sequence, human walking pattern etc.

Mathematical representation of HMM

HMM is represented mathematically by a set of total four tuples

1) Hidden states
2) Visible states
3) Emission probability
4) Transition probability
5) Starting probability

1) Hidden states: These are the states in the HMM that are not visible to the outside world but are there in the computation model.

2) Visible states: The states emitted by the hidden state in the HMM are known as the visible states and each hidden state can emit all the visible states with same or different probabilities. It is important to note that each hidden state will emit all the visible states.

3) Emission probability: The probability of emission of a visible symbol from a hidden state is known as emission probability.

4) Transition probability: The probability of transition from one hidden state to another hidden state in next time step is known as transition probability.

5) Starting probability: The probability for starting the HMM computation graph is known as starting probability.

Example, now we will define a HMM having three hidden states and three visible states.

Hidden state = {w1, w2, w3}

Visible state = {v1, v2, v3}

Transition probability = a_{jk} (j → starting state at time T, k → destination state at time T+1)

Emission probability = b_{jk} (j → present state, k → emitted visible symbol)

Starting probability = c_{1k} (1 → starting time stamp, k → total number of hidden states)

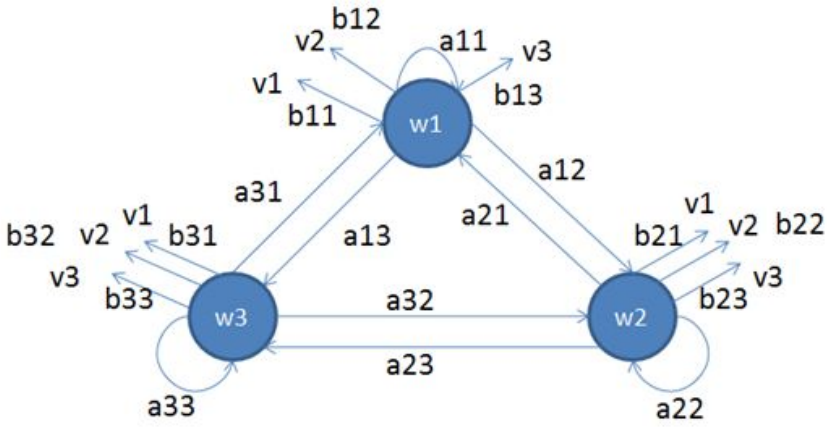

Figure - 2.3
(Diagrammatic representation of the HMM defined above)

Operation on the HMM

There are basically three operations that can be done on the HMM are

1) Evaluation problem
2) Decoding problem
3) Learning problem

1) Evaluation problem: Under this problem we generally are interested to calculate the probability of generation of a given sequence by a given model.

2) Decoding problem: Interested to find the most probable path of hidden states for generation of a given sequence of visible state.

3) Learning problem: Training the HMM model based on known samples to predict the emission and transition probability of the unknown samples.

Evaluation problem

In this problem a full HMM model is given including all transition and emission probabilities and let us denote the model with the symbol θ. Where $\theta = \{ w , v , a_{ij} , b_{jk} \}$ and a sequence V^T is given and we are interested to find the probability of occurrence of V^T given the model θ i.e., $P(V^T | \theta) = ?$

Let us assume that there are i hidden states denoted by w_1 to w_i and 't' time-steps and we try to find the probability of a sequence being generated by the given model.

Symbols and notations

1) Hidden state w_1 to w_i and we assume that w_0 is the halting state for the HMM
2) t = 0 to t = T

3) $\alpha_i(t)$ denotes the probability of the HMM in hidden state i at time-step t
4) a_{ij} the probability of transition from hidden state i to hidden state j
5) b_{jk} the probability of state j emitting k^{th} visible state

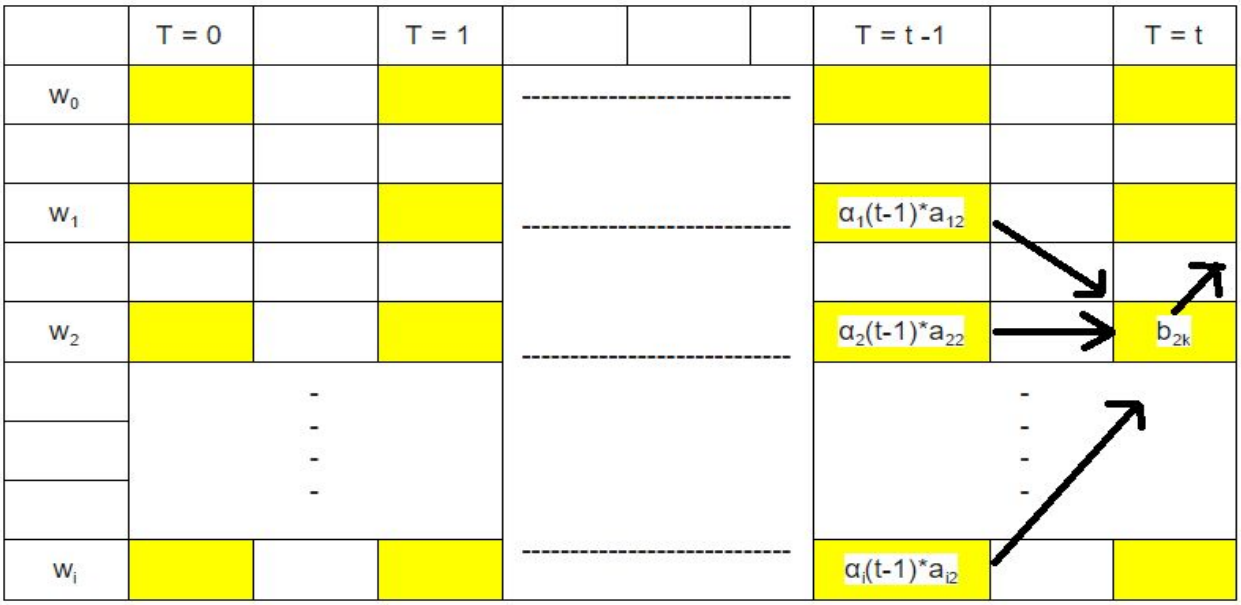

Figure - 2.4
(Representation of the total HMM in tabular format)

Note: As we have assumed that w_0 is a the final state so if the HMM visits the state w_0 then it can't come out of the state and means there are no transition from the state w_0 , So in the diagram no transitions are shown from the state w_0 to any other state.

Here we are interested to find the probability of a sequence to be generated by the given model, to find it we will find probability of occurrence each residue or element in the sequence at given positions and multiply them to find the probability of the sequence.
e.g., let the sequence of interest is 'AABDCCBCAB' so our sequence have 10 elements to find the occurrence of this sequence in hypothetical model θ we will find the probability of occurrence of A in 1^{st} place, A in 2^{nd} place, B in 3^{rd} place,....., C in 8^{th} place, A in 9^{th} place, B in 10^{th} place according to the sequence and then multiply all these probabilities together.

To find the probability of a residue or element at a particular instance of time (T) we need to add the product of occurrence of the HMM at that state in time instance (T-1) and transition probability from state at time (T-1) to state at time (T) and then multiply the emission probability of visible symbol from state at time (T).
e.g., let form **Figure - 2.4** we want to find the occurrence of a symbol at state w_2 at time step (T). So, we will add $\alpha_1(t-1)*a_{12}$, $\alpha_2(t-1)*a_{22}$, …….. , $\alpha_i(t-1)*a_{i2}$ and multiply it with b_{2k}
Where, (i) $\alpha_i(t-1)$ is the probability of occurrence of the HMM at state i at time (t-1)
(ii) a_{i2} is the probability of transition from state i at time (t-1) to state 2^{nd} at time (t)
(iii) b_{2k} is the probability of emission of visible state from state w_2 at time step (t)

So, probability of that there will be a transition to hidden state w_2 at time (t) from time step (t-1) and it will emit a visible state b_k at time (t) is = $b_{2k}(\alpha_1(t-1)*a_{12} + \alpha_2(t-1)*a_{22} + \ldots\ldots\ldots + \alpha_i(t-1)*a_{i2})$
Let us denote the probability of HM in state w_j in time step (t) after emitting first t number of visible symbols in the sequence of given visible symbols V^T by $\alpha_j(t)$. The definition of α_{jt} is

$\alpha_j(t) = \{ \alpha_{jt} = 0$ if t = 0 and j != initial state

$\alpha_{k0} = 1$ if k = initial state and t = 0 and there is only 1 starting state

$\alpha_{jt} = [\Sigma \alpha_i(t-1).a_{ij}] * [b_{jk\,(v(t))}]$ where $b_{jk\,(v(t))}$ is emission probability of state in (t) from j^{th} state in (t-1) }

Algorithm for evaluation problem (Forward Algorithm)

Initialization: $t \leftarrow 0$, a_{ij} , b_{jk} , V^T , $\alpha_j(0)$

For: $t \leftarrow t + 1$

$\alpha_j(t) = b_{jk(v(t))} \Sigma \alpha_i(t-1) * a_{ij}$

Until t = T

Return: $P(V^T | \theta) \leftarrow \alpha_0(T)$ i.e., probability of final state

Halt

Example for evaluation problem

Let us take an example where we are provided with a model Θ and we have to find the occurrence of a given sequence V^T.

Let, $V^T = (V_3 , V_1 , V_2 , V_1 , V_0)$

And the parameters in the model Θ are:

1) Hidden state = (w_1 , w_2 , w_3 , w_0)

2) Visible state = $(V_1 , V_2 , V_3 , V_4 , V_0)$

3) Transition probability =
$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.7 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$

4) Emission probability =
$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0.0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0.0 & 0.5 & 0.2 & 0.1 & 0.2 \end{bmatrix}$$

We are interested to find the probability $P(V^T | \Theta) = ?$

	0	1	2	3	4
w_0	0.0	0.0	0.0	0.0	0.00182
w_1	1.0	0.09	0.0052	0.005192	0.0
w_2	0.0	0.01	0.0217	0.000543	0.0
w_3	0.0	0.2	0.0057	0.000964	0.0

Figure - 2.5
(Total computation in the HMM)

At time-step t = 0
We assumed that there is only one starting state w_1 so starting probability of that state is 1.0 and of other states are 0.0.

At time-step t = 1
We assumed that w_0 is accepting state so this state can be only present only as last time stamp so the probability at other time-steps are 0.0.
By following the above mentioned algorithm,

1) For w_1 state at time-step 1 = 1.0 * 0.3 * 0.3 + 0.0 + 0.0 = 0.09
   In 1.0 * 0.3 * 0.3 this 1.0 is the probability of the machine in w_1 at time t = 0, 0.3 is the transition probability from state w_1 in time t= 0 to state w_1 at time t = 1 and 0.3 is the emission probability of symbol V_1 from state w_1
   In second tuple we did not consider the emission and transition probability because the occurrence of the HMM in states w_2 and w_3, is 0.0 so the product will be 0.0
2) For w_2, state at time-step 1 = 1.0 * 0.1 * 0.1 + 0.0 + 0.0 = 0.01
3) For w_3 state at time-step 1 = 1.0 * 0.4 * 0.5 + 0.0 + 0.0 = 0.20

Similarly following these steps we will get the computation table as shown in **Figure - 2.5**.

Decoding Problem
Interested to find the most probable path of hidden states for generation of a given sequence of visible state. For the above mentioned example the most probable path will be (w_1,w_3,w_2,w_1,w_0) Because at time t = 0 the highest probability is 1.0 corresponding to w_1 then at time instance at t=1 the highest probability is 0.2 corresponding to w_3 similarly we find the above mentioned most probable path for the given sequence.

PHMM (Profile Hidden Markov Model)

Definition

This is a probabilistic markovian model for representation of protein or gene sequences from multiple sequence alignment. This model is basically designed to represent gene or protein sequence in a probabilistic model.

Sequence Alignment

The process of aligning sequence (protein, DNA or RNA) to a stable form. Let us we assume that there are two sequences given to us and tasked to align them in such a way that they are in most stable form, the sequences are (S1 = A-T-G-C) and (S2 = T-G-C). So we align them and label them among the three possible operations (mismatch, match, gap) as

A - T → Mismatch
T - G → Mismatch
G - C → Mismatch
C - _ → Gap

Figure - 2.6
(One of the possible sequence alignment)

Now let we attach an numeric value to these three possible operations, for match we apply +2 score, for mismatch -1 and for gap -2 is awarded to the sequence and then the total score is added for the sequence. In this sequence there are 3 mismatch, 0 match and 1 gap so the total score for this alignment is 3 * (-1) + 0 * (+2) + 1 * (-2) = (-3) + (+0) + (-2) = -5 i.e., the sequence gets a score of -2 for this alignment. Now if we make the alignment as

A - _ → Gap
T - T → Match
G - G → Match
C - C → Match

Figure - 2.7
(Sequence alignment)

So here we see there is a total of 3 match, 0 mismatch and 1 gap. So according to the previous scoring system as mentioned above we get a score of 3 * (+2) + 0 * (-1) + 1 * (-2) = (6) + (-2) = 4 Here we see that the second alignment produces an higher score with respect to the first alignment and hence we conclude that the second alignment is more stable and suitable than

the first alignment. Now if the length of the given sequences are very large i.e., in order of thousands or millions of entity then the computation becomes very costly and time consuming.

Multiple Sequence Alignment

When a huge number of sequences in order of millions or more are aligned instead of only two sequences in known as multiple sequence alignment. Some of the standard algorithms are FASTA, BLAST etc. Multiple sequence alignment is implemented to classify new protein into some classes, this is important because each individual class holds its own specific property and by classification we can group protein of same nature together.

[Note: Here discussion on how to create multiple sequence alignment is not discussed because we are interested in finding pHMM not multiple sequence alignment. **]**

Example of multiple sequence alignment,

A	**C**	**D**	**E**	**F**	**A**	**D**	**F**
A	**F**	**D**	**A**	**-**	**C**	**C**	**F**
A	**-**	**-**	**E**	**F**	**F**	**D**	**C**
A	**C**	**A**	**E**	**F**	**A**	**-**	**C**
A	**D**	**D**	**E**	**F**	**A**	**D**	**F**

Figure - 2.8
(Example of multiple sequence alignment having 5 different sequences)

Now we will create profile of this multiple sequence alignment means a table that will say the probability of occurrence of each individual element at each position of the multiple sequence alignment.

For example at first position only a occurs so the probability of occurrence of element A is 1.0 and for all other i.e., C, D, E and F is 0.0 then at second position we see that C, D and F occurs so the probability of occurrence of A is 0.0, C → 2/4 = 0.5 here numerator is 2 as C occurs twice and denominator is 4 because there are total of 4 elements along with one gap state at second position and we do not include gap position, D → ¼ = 0.25, E → 0.0, F → ¼ = 0.25. Similarly occurrence of each each element at each state is calculated and noted down to create the

profile of the multiple sequence alignment and then this profile is used classify new multiple sequence alignment into different class so that we don't have to individually test or check all the properties of the protein as we can predict the common feature to each class will also belong to the newly classified protein.

A	1	0	1/4	1/5	0	3/5	0	0
C	0	2/4	0	0	0	1/5	1/4	2/5
D	0	1/4	3/4	0	0	0	3/4	0
E	0	0	0	4/5	0	0	0	0
F	0	1/4	0	0	1	1/5	0	3/5

Figure - 2.9
(Profile of the above mention multiple sequence alignment
In Figure - 2.8 using the same procedure as mentioned above)

Now suppose a new sequence is introduced to us (A-F-D-E-F-C-D-F) and we want to find the probability of this sequence belonging to this pHMM, so we multiply the probability of occurrence of each element at given location and multiply them. The obtained probability is
(1 * 0.25 * 0.75 * 0.8 * 1 * 0.2 * 0.75 * 0.6 = 0.0135).

Now in reality there are many pHMM built form a large number of proteins. When a new protein comes then its probability of belonging to these pHMMs are calculated and the protein is assigned to the pHMM having maximum probability.

Presently the pHMM will accept sequences having 8 elements only but will not be able to accept larger or smaller sequences

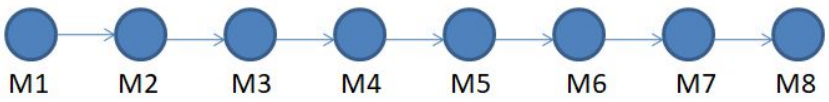

Figure - 2.10

The basic problem with this system is that no sequence of length more or less than the specified length can be taken into consideration.

For example earlier we saw that we assumed an example sequence of length 8 but what will happen if a sequence of greater length comes or a sequence with elements that were not seen before, so we need to modify our current pHMM.

So now we introduce a new state other than transition state i.e., insertion states to include sequences of greater length than the specified length of the pHMM

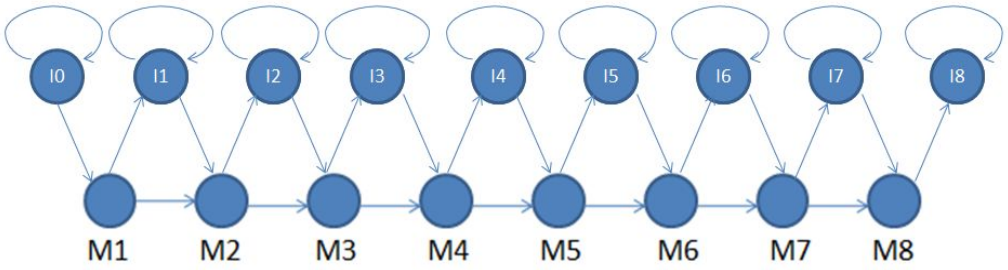

Figure - 2.11
(profile of multiple sequence alignment with insertion states)

Now we can clearly see that sequences of any length greater than or equal to the specified length can be introduced into the pHMM. E.g., let we assume an example where the total length of the sequence is 16 (A-F-D-***X-Y-Z***-E-F-***L-M-N-O***-C-***P***-D-F) so now to this sequence can be fitted in the given pHMM with the transitions from the following states i.e., A → M1, F → M2, D → M3, X → I3, Y → I3, Z → I3, E → M4, F → M5, L → I5, M → I5, N → I5, O → I5, C → M6, P → I6, D → M7 and F → M8.

Now we will focus on the problem of inability of this pHMM to include smaller sequences. So we introduce deletion states which will be used to remove or jump states in the pHMM

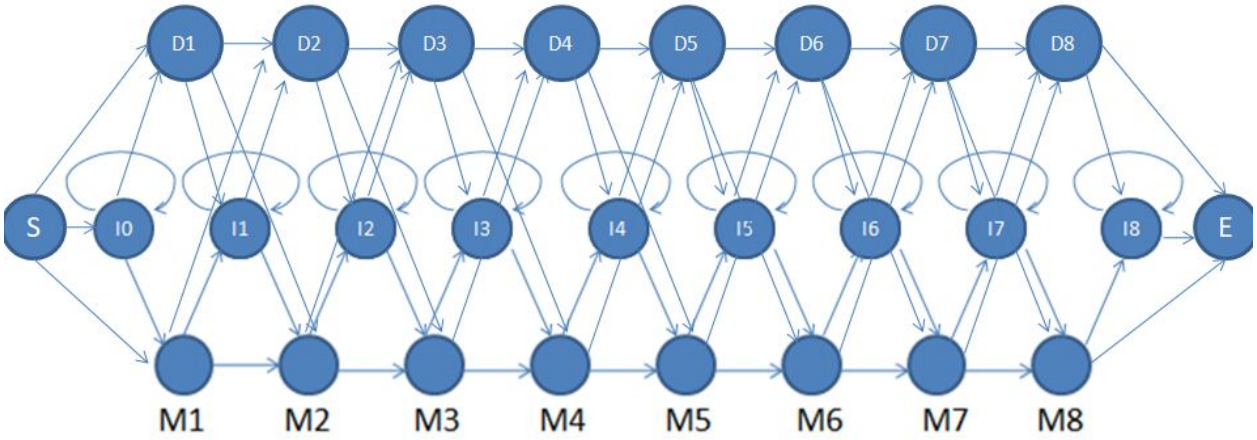

Figure - 2.12
(pHMM created from the profile of multiple sequence alignment)

Now let we assume a new sequence i.e., (A-F-D-__-__-__-__-F) so to fit this sequence in the above mentioned model we will follow these transitions A → M1, F → M2, D → M3, __ → D4, __ → D5, __ → D6, __ → D7 and F → M8. So now we can say that any sequence independent of its length can be inserted into this pHMM.

So a pHMM is defined by the following probability for each of its states

1) Transition probability
    a) $M_k \rightarrow M_{k+1}$ (transition from one state at time t to match state at time t+1)
    b) $M_k \rightarrow I_k$ (transition from a state to insertion state)
    c) $M_k \rightarrow D_{k+1}$ (transition from one state at time t to deletion state at time t+1)
2) Insertion probability
    a) $I_k \rightarrow M_{k+1}$ (transition from insertion state to match state at time t+1)
    b) $I_k \rightarrow I_k$ (transition from insertion state to insertion state at same time stamp)
3) Deletion probability
    a) $D_k \rightarrow M_{k+1}$ (transition from deletion state to match state at next time stamp)
    b) $D_k \rightarrow D_{k+1}$ (transition from deletion state at time t to next deletion state at t+1)

Example of pHMM,

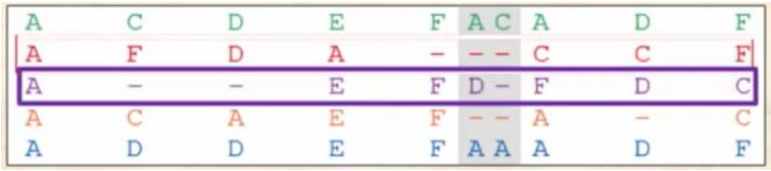

Figure - 2.13
(an example of multiple sequence alignment with 5 sequences)

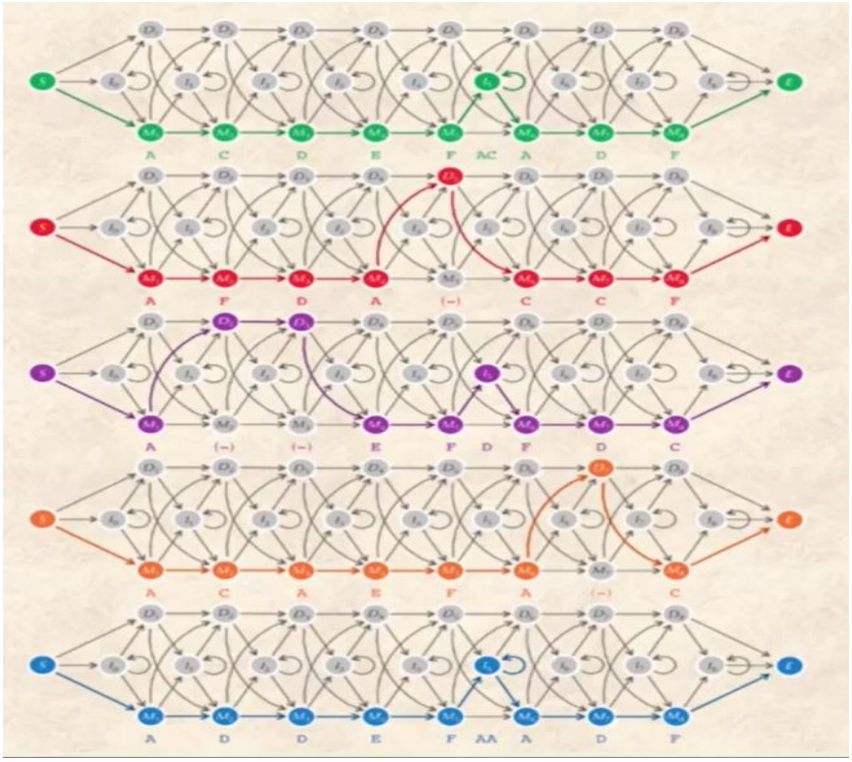

Figure - 2.14
(pHMM for each sequence mentioned in Figure - 2.13)

ipHMM (Interaction Profile Hidden Markov Model)

Definition

Interaction profile hidden markov model shows discriminative power and plays an important role in prediction of contact matrix.

Contact matrix

Suppose you are given two multiple sequence alignments MSA1 and MSA2 each having 1000 states each, out of this 1000 states you are given a set of states that are said to be interacting states i.e., these states will interact when these two multiple sequences will come in contact. Now the matrix representation of this phenomenon is called contact matrix. The contact matrix plays an important role in making of generic drugs and knowing the specific properties of the protein sequence.

[Note: The elements of contact matrix can be single sequences or even multiple sequence alignments.**]**

E.g., let SEQN1 is "A-T-D-S-K-L-M-C-B-F-K-P-S-C-J-C-H-A-F-J" and SEQN2 is the sequence "R-O-P-A-S-K-S-M-C-Z-A-G-V-N-E-Q-F-P-X-Z" and it is said that the interacting positions are 1,3,5,8,9,13,17 i.e.,

A -- R	→	Interacting
T -- O	→	Non-Interacting
D -- P	→	Interacting
S -- A	→	Non-Interacting
K -- S	→	Interacting
L -- K	→	Non-Interacting
M -- S	→	Non-Interacting
C -- M	→	Interacting
B -- C	→	Non-Interacting
F -- Z	→	Non-Interacting
K -- A	→	Non-Interacting
P -- G	→	Non-Interacting
S -- V	→	Interacting
C -- N	→	Non-Interacting
J -- E	→	Non-Interacting
C -- Q	→	Non-Interacting
H -- F	→	Interacting
A -- P	→	Non-Interacting
F -- X	→	Non-Interacting
J -- Z	→	Non-Interacting

Figure - 2.15
(Interacting and noninteracting sites of SEQN1 and SEQN2 as mentioned above)

The contact matrix of this mentioned example is,

	A	T	D	S	K	L	M	C	B	F	K	P	S	C	J	C	H	A	F	J
R	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure - 2.16
(Contact matrix for SEQN1 and SEQN2)

Contact matrix using the multiple sequence alignment

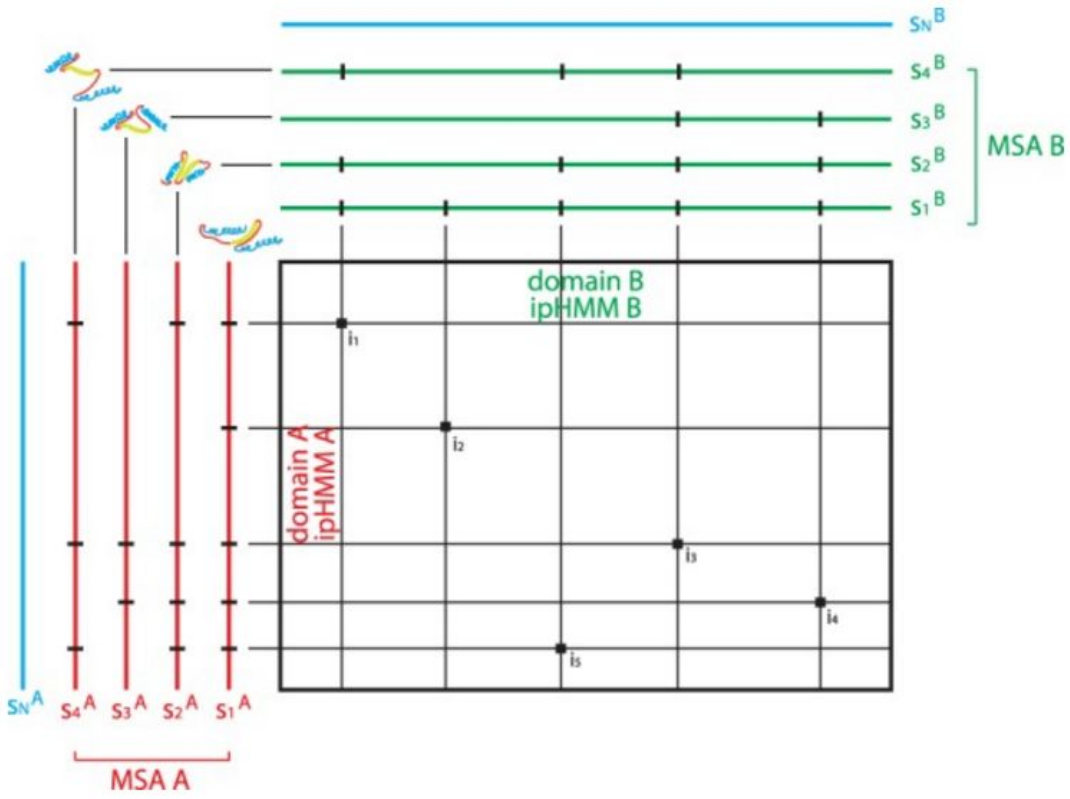

Figure - 2.17
(Contact matrix prepared from two multiple sequence alignments named 'MSA A' and 'MSA B')

Converting the pHMM to ipHMM

Initially we have created pHMM form the multiple sequence alignment. Now, we are interested to make the ipHMM from the pHMM. At this point of time we assume that we are given the pHMM and the locations of the interacting sequences. So, we annotate the interacting sequences in the pHMM with 1 and the noninteracting sequences with 0 and all the probability and other parameters of the ipHMM is exactly same as the pHMM.

Now we will show an small example of pHMM and an ipHMM and each part of this example will be explained in details in future.

```
HMMER3/f [3.1b2 | February 2015]
NAME  Rhodopsin_N
LENG  36
ALPH  amino
RF    no
MM    no
CONS  yes
CS    yes
MAP   yes
DATE  Thu Jun 15 21:28:02 2017
NSEQ  177
EFFN  1.812241
CKSUM 2590383827
STATS LOCAL MSV       -7.4742  0.71964
STATS LOCAL VITERBI   -7.6501  0.71964
STATS LOCAL FORWARD   -4.3034  0.71964
HMM          A        C        D        E        F        G        H
            m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO   2.88226  4.96252  3.36503  2.72259  2.69094  2.49662  4.18287
          2.68659  4.42266  2.77487  2.73164  3.46395  2.40453  3.72535
          0.21746  1.69889  4.37756  1.97248  0.14979  0.00000        *
      1   3.16310  5.34704  3.24441  2.87534  4.78519  3.72907  3.99361
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494
          0.20998  4.54883  1.72139  0.61958  0.77255  0.49917  0.93404
      2   3.38120  5.08033  4.15097  4.12311  5.20349  0.26220  5.18433
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494
          0.01920  4.35805  5.08040  0.61958  0.77255  0.76748  0.62396
      3   2.71664  4.49352  3.89433  3.51184  3.19923  3.52143  4.36373
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494
          0.01920  4.35805  5.08040  0.61958  0.77255  0.76748  0.62396
      4   3.48127  5.60690  2.82171  0.49203  5.03183  3.67028  4.33778
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494
          0.01920  4.35805  5.08040  0.61958  0.77255  0.76748  0.62396
```

Figure - 2.18
(pHMM of domain named Rhodopsin_N)

```
HMMER3/f [3.1b2 | February 2015]
NAME  Rhodopsin_N
LENG  36
ALPH  amino
RF    no
MM    no
CONS  yes
CS    yes
MAP   yes
DATE  Thu Jun 15 21:28:02 2017
NSEQ  177
EFFN  1.812241
CKSUM 2590383827
STATS LOCAL MSV       -7.4742  0.71964
STATS LOCAL VITERBI   -7.6501  0.71964
STATS LOCAL FORWARD   -4.3034  0.71964
HMM          A        C        D        E        F        G        H        I        K        L
            m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO   2.88226  4.96252  3.36503  2.72259  2.69094  2.49662  4.18287  3.18054  3.13022  2.84573
          2.68659  4.42266  2.77487  2.73164  3.46395  2.40453  3.72535  3.29395  2.67782  2.69328
          0.21746  1.69889  4.37756  1.97248  0.14979  0.00000        *
      1   3.16310  5.34704  3.24441  2.87534  4.78519  3.72907  3.99361  4.30224  2.48789  3.78484
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355
          0.20998  4.54883  1.72139  0.61958  0.77255  0.49917  0.93404
          0
      2   3.38120  5.08033  4.15097  4.12311  5.20349  0.26220  5.18433  4.94990  4.36817  4.52962
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355
          0.01920  4.35805  5.08040  0.61958  0.77255  0.76748  0.62396
          1
      3   2.71664  4.49352  3.89433  3.51184  3.19923  3.52143  4.36373  3.03454  3.44205  2.81489
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355
          0.01920  4.35805  5.08040  0.61958  0.77255  0.76748  0.62396
          1
```

Figure - 2.19
(ipHMM of domain named Rhodopsin_N, see the red box in this picture they are the annotation for the interacting and noninteracting locations)

PART - 3

3DID database

3DID database

Introduction

3DID basically stands for 3-dimensional domain domain interaction, this 3DID database contains the the 3 dimensional data of domain domain interaction.

Steps to get the data

Visit the website of 3DID (http://3did.irbbarcelona.org/)

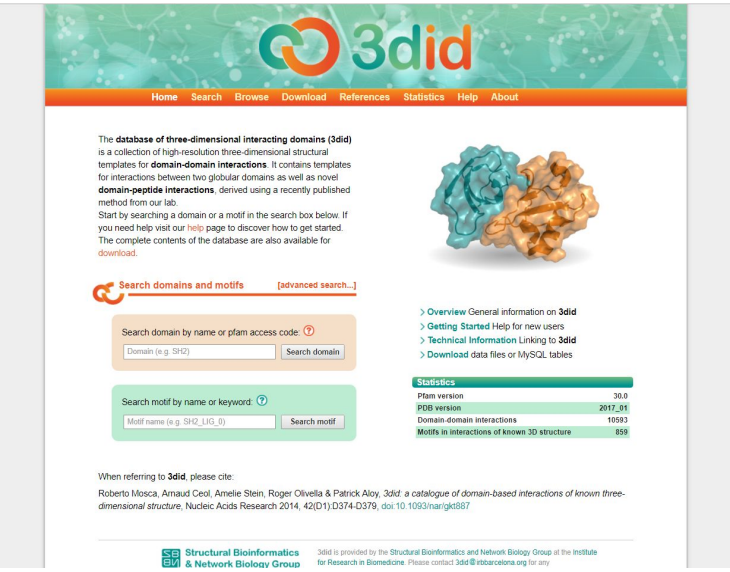

Figure - 3.1

Then go to the downloads page of the website (http://3did.irbbarcelona.org/download.php)

Figure - 3.2

From this page download the latest flat file named as “3did_flat.gz” and extract it. This file contains the domain domain interaction data available at the time of download.

Extracted data i.e., data obtained after extraction of the downloaded file,

```
#=ID    1-cysPrx_C  1-cysPrx_C   (PF10417.7@Pfam    PF10417.7@Pfam)
#=3D    5jcg    A:157-192   H:157-192   1.1 1.60435 0:0
Q   E   159     162     sm
Q   T   159     163     sm
E   Q   162     159     ms
T   Q   163     159     sm
T   T   163     163     ss
#=3D    1n8j    E:153-185   O:153-185   0.99    1.28334 0:0
A   A   158     159     mm
A   H   158     160     mm
A   A   159     159     ss
A   H   159     160     ss
H   A   160     159     mm
//
#=ID    1-cysPrx_C  AhpC-TSA     (PF10417.7@Pfam    PF00578.19@Pfam)
#=3D    4llr    A:162-197   A:8-142 5.39    3.66636 8:6
F   W   163     38  ss
F   I   163     132     ss
F   R   163     140     sm
F   Q   163     141     sm
Q   W   164     38  ss
Q   I   164     69  sm
Q   G   164     70  sm
E   K   167     136     ss
E   R   167     140     ss
#=3D    4llr    A:162-197   B:8-142 5.9 3.43601 0:10
V   F   172     50  sm
V   V   172     51  ss
C   V   173     51  ms
P   V   174     51  ms
P   T   174     54  ms
A   V   175     51  mm
```

Figure - 3.3

Rules to read the file:

1) **#=ID:** this tag denotes the two interacting domains, plus their Pfam IDs

2) **#=3D:** one structural instance of this interaction, with the following fields:

PDB chain1:(domain1Start-domain1End) chain2:(domain2Start-domain2End) score Zscore topology

Below each **#=3D** entry, the residue contacts are listed (residue 1 and 2, position 1 and 2, and contact type -- m = main chain, s = side chain). Please note that all positions given here refer to the numbering of residues in the original PDB file. Two slashes (**//**) indicate the end of a domain pair.

Explanation of above mentioned rules

As above this file contains all known domain domain interaction at the time of download so we conclude that this file contains more than 1 domain-domain interaction and they are separated by this '**//**' symbol.

Now in first line of all domain domain interaction we will see a tag '**#=ID**' this line contains the names of interacting domains with their Pfam IDs, we will consider the first line from the **Figure-3.3** i.e (#=ID 1-cysPrx_C 1-cysPrx_C (PF10417.7@Pfam PF10417.7@Pfam)) the 1st position #=ID is the tag, 2nd contains the name of 1st interacting domain at 3rd place we have the 2nd interacting domain in 4th we have Pfam ID of 1st domain at 5th place contains the Pfam ID of 2nd domain.

Now below this line we can see multiple times '**#=3D**' tag appears because in a domain-domain interaction there are multiple interaction sites. The line containing this tag are the interaction site of the domain domain interaction. E.g., if there are 5 '**#=3D**' tags under one '**#=ID**' tag then it means that there are 5 interaction sites in this domain domain interaction.
Now there are many tuples in the line containing the tag '**#=3D**' the first one is a PDB (protein data bank) ID, at second place there is chain name of the 1st interacting domain followed by '**:**' and then the interacting range i.e., two integers denoting the starting and ending of interacting site separated by '**-**', next we have the details of the interacting site of 2nd domain followed by a empirical potential score i.e., denotes the possibility of interaction of the domains, then Zscore is given and then the topology is given.
Now as an example we ill choose the second line from **Figure - 3.3**
(#=3D 5jcg A:157-192 H:157-192 1.1 1.60435 0:0)
5jcg → PDB ID
A:157-192 → A is the name of the chain belonging to this PDB ID, followed by '**:**' and 157-192 is the interacting range of the chain of this PDB ID and all of this is for the first domain mentioned in the first line.
H:157-192 → Contains the same data as mentioned above but it contains for the second domain.
1.1 → Empirical potential score
1.606435 → Zscore
0:0 → Is the topology of interaction

After the line starting with '**#=3D**' tag we see each line have 5 tuples, in this first one is the interacting residue of first domain followed by the interacting residue of second interacting domain then followed by their exact positions respectively and then one of these comes (ss, sm, mm, ms) where m ⇒ main chain and s ⇒ side chain.

E.g., (Q E 159 162 sm)

Q → Q of the first domain is interacting

E → E is interacting residue of the second domain

159 → Specific location of Q in the above mentioned chain

162 → Specific location of E as mentioned above

sm → Q belongs to side chain and E belongs to main chain

Extracting data from the downloaded file

According to the paper that we are trying to implement we have to select some of the domain domain interaction from all of the DDI (domain domain interaction) in the file. Rule for the selection of the specific DDI are

1) Interacting domains must be different i.e., the data in second and third tuples in the line having tag '**#=ID**' must be different.
2) The domain must have more than one topology i.e., in the whole DDI the last tuple in the line having tag '**#=3D**' must be more than one in number like one DDI should not have same topology, there should be more topology.
3) Each topology should have more than 10 example i.e., each topology should have more than 10 interaction sites.
4) Each DDI should have more than or equal to 40 interactions and less than or equal to 60 interactions (>=40 and <=60). The range is more than 40 because for training purpose the number of examples will be very low of the number is less than 40 and upper limit is 60 because if it is greater than 60 then the training will become very costly and is not feasible.

After applying all these rules on the data set we get a total of 123 domain domain interactions and this number is exactly equal to the number mentioned in the paper.

[Note : The code for this is given in the "extract_3DID.py" file **]**

After this we get the name of all the domains involved in DDI from those 123 domain domain interactions. Now we will create pHMM for each of these domains so we need multiple sequences alignments, procedure to get multiple sequence alignment and for creating pHMM is explained below.

PART - 4

Pfam database

Pfam

Introduction

This is a database that holds multiple sequence alignment of different domains. We will download these multiple sequences and make pHMM out of them.

Procedure

Visit the homepage of the Pfam website (http://pfam.xfam.org/)

Figure - 4.1

(Home page of the Pfam website)

Then visit this link to get the MSA (http://pfam.xfam.org/search#tabview=tab2)

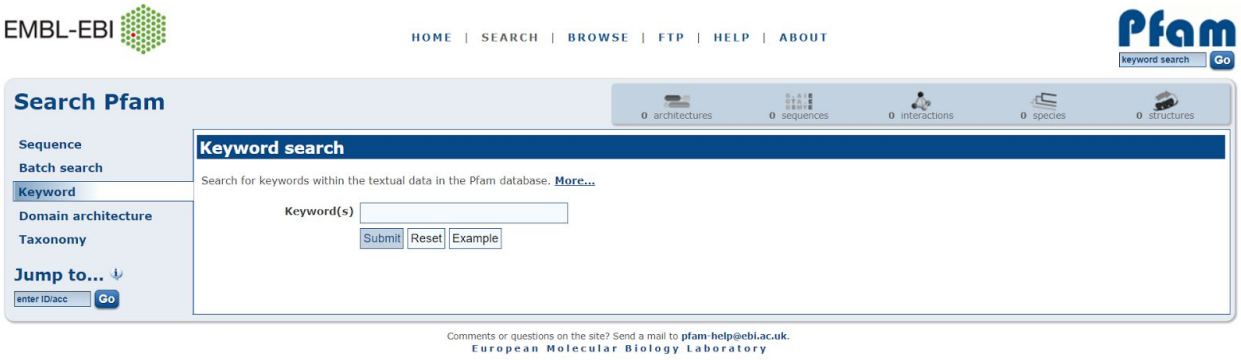

Figure - 4.2

In the '**Keyword**' entry the name of the domain for which you want to get the multiple sequence alignment and then press the submit button, for example we will take the domain '**Rhodopsin_N**'.

Figure - 4.3

(Webpage viewed when we search for the domain 'Rhodopsin_N' in the above mentioned link)

Now in the top right corner we see that there are five options they are **Architecture**, **Sequences**, **Interactions**, **Species** and **Structures** click on the **Sequences** tab.

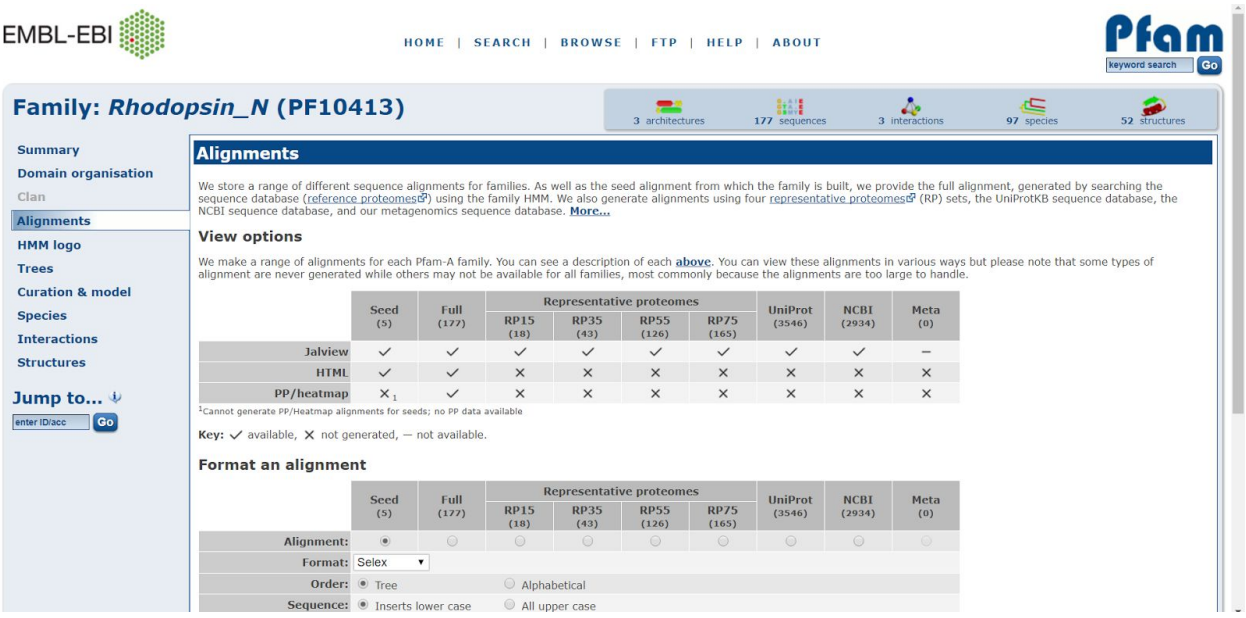

Figure - 4.4

In this page you can see basically three main tabs **View Options**, **Format an Alignment** and **Download Options**. We are basically interested in the second option Format an Alignment.

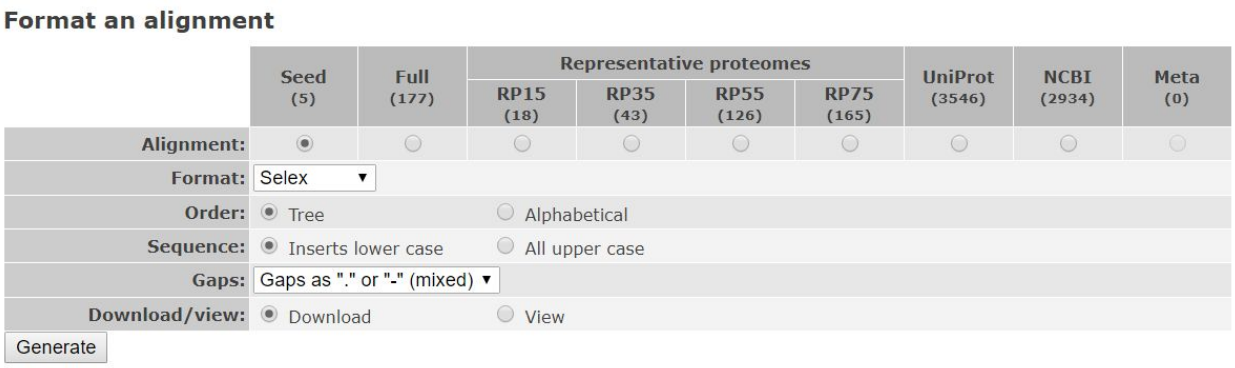

Figure - 4.5

Now in this part we can see that there are many tabs we will explain each individually

1) Alignment : In this 'Seed' means a small part of the total sequence alignment whereas selecting 'Full' will download the total multiple sequence alignment.
2) Formal : There are many formats in which the multiple sequence alignment can be available to us, some of them are FASTA, Stockholm, MSF etc. We will download in Stockholm format because the software we will be using it make pHMM will accept this format only.
3) Gaps : Select the '**-**' because this significantly reduce the file size.

[Note : You cannot download large files directly from the website as dated July 2017. You need to use their API, some of the links you require for the FTP are

(1) ftp://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam31.0/

(2) ftp://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam31.0/userman.txt

(3) ftp://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam31.0/Pfam-A.full.uniprot.gz **]**

After doing all these settings click on the '**Generate**' button. Now you have downloaded the multiple sequence alignment for a particular domain.

No we will make pHMM using the HMMER software using the multiple sequence alignment that we just downloaded.

PART - 5

Cygwin

HMMER Software installing

Making pHMM from multiple sequence alignment

Fisher Score

HMMER

Introduction

We have used this software create pHMM though it have more other options. In the following section procedure for downloading, installing and using it is explained in details.

Download

The HMMER software is given in the attachments in the folder named ***hmmer*** in .zip format. This software needs ***unix*** environment, so for the people working in windows you need to install the ***Cygwin*** software.

Installing Cygwin

Visit this website (https://cygwin.com/install.html) to download the Cygwin software.

Figure - 5.1

(Home page of the above mentioned link)

Then download Cygwin for windows, please select the version (32 bit or 64 bit) carefully. After the download completes run the installer by the double clicking the file.

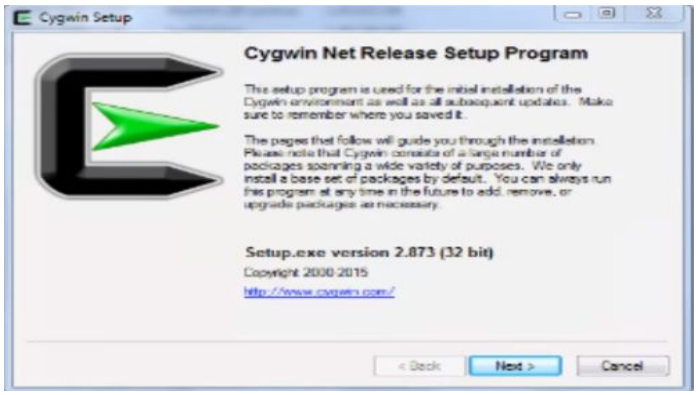

Figure - 5.2

After clicking a couple of 'Next' and selecting the installation directory the install of the software will take place then after that a window will pop up which will ask for the installation of the required software packages, and you need to install these packages

1) Gcc-g++ GNU Compiler collection (c++) under g++
2) Openssh: the open SSH server and client programs under SSH
3) Make: The GNU version of the make utility under MAKE
4) Ncurses: terminal display utilities under NCURSES
5) Clisp: An ANSI Common lisp implementation CLISP
6) Vim-minimal: Minimal VI text editor under VIM

After this click 'Next' and finish the installation and run the Cygwin programme.

Installing HMMER

After running the Cygwin programme now it's time for installing the HMMER so that we can create pHMM. First navigate to your root directory in the Cygwin environment then to the user account now at this place copy the downloaded HMMER software in .zip format from the hmmer folder. Now execute these commands step by step on this .zip folder in Cygwin environment

1) To unzip the zip file execute the command '**tar -xvf hmmer-3.1b2-cygwin32.tar.gz**'
2) Now change directory to the extracted files '**cd hmmer-3.1b2-cygwin32**'
3) Execute '**./configure**'
4) Execute '**make**'
5) Execute '**make check**'

[Note : For more detailed explanation check **YouTube** or check this link if the video is still available (https://www.youtube.com/watch?v=hh-V6el8Oxk) **]**

Creating pHMM out of multiple sequence alignment

Open Cygwin

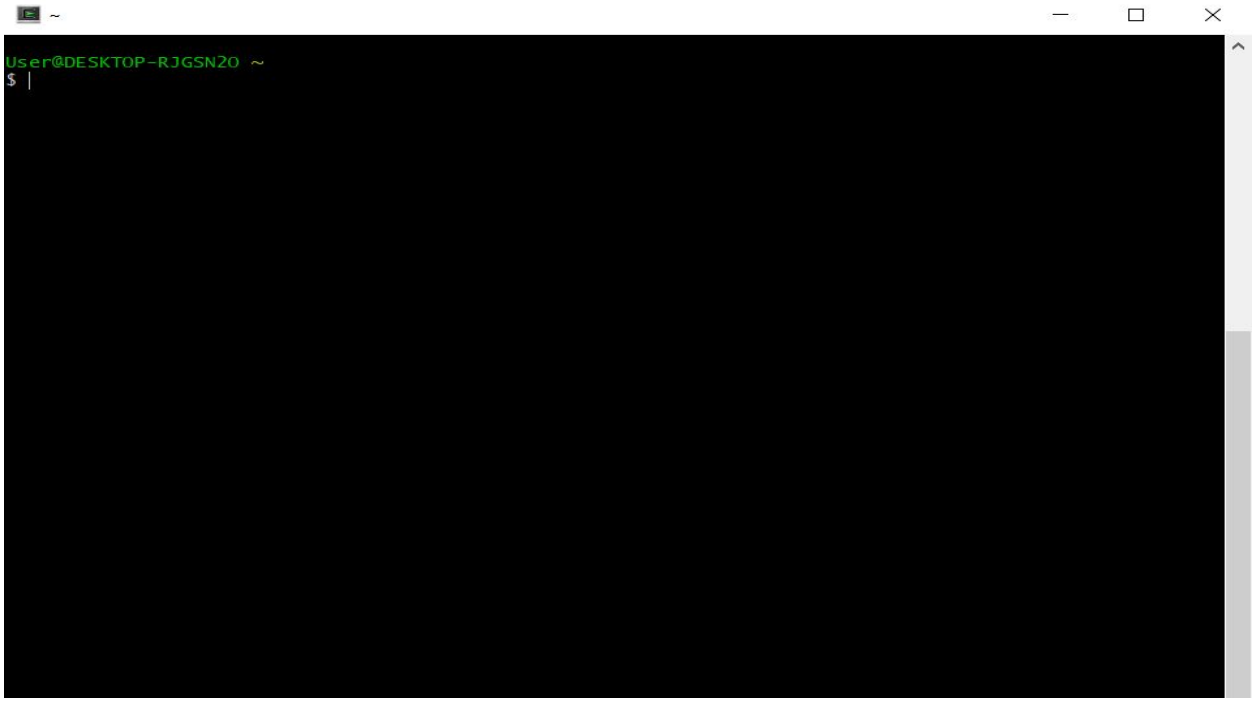

Figure - 5.3

Change directory to the HMMER software

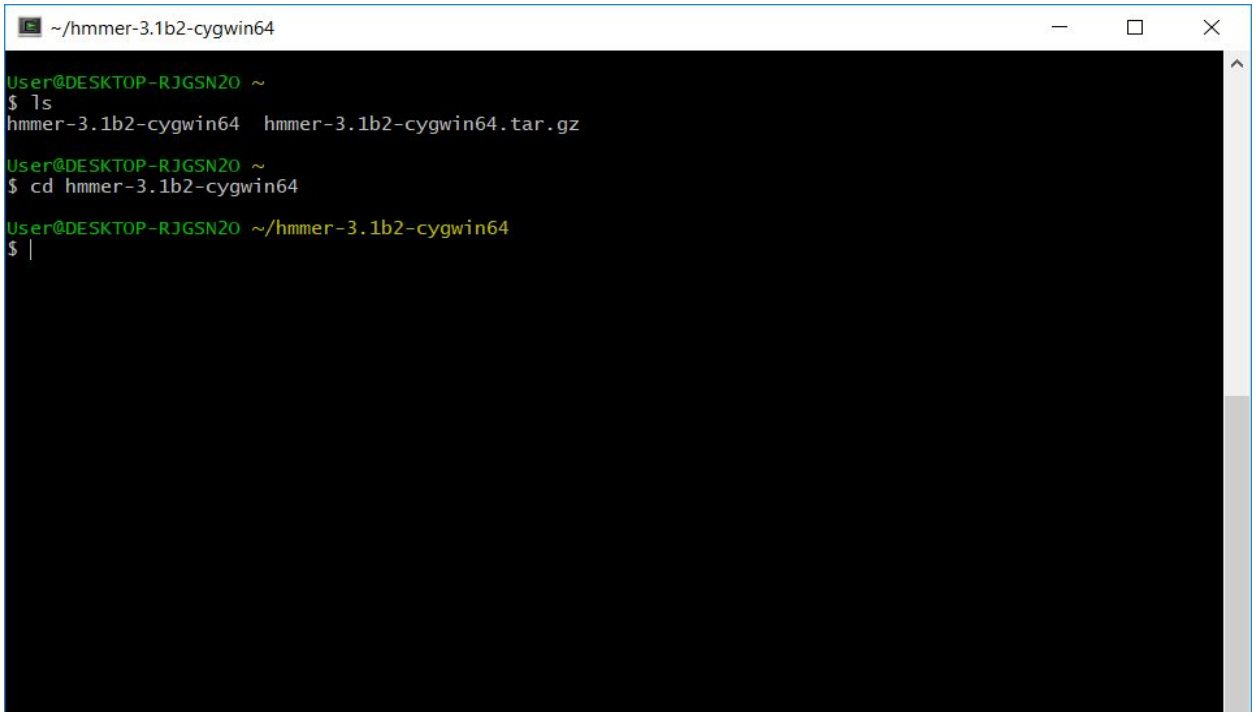

Figure - 5.4

Now change to ***tutorial*** directory

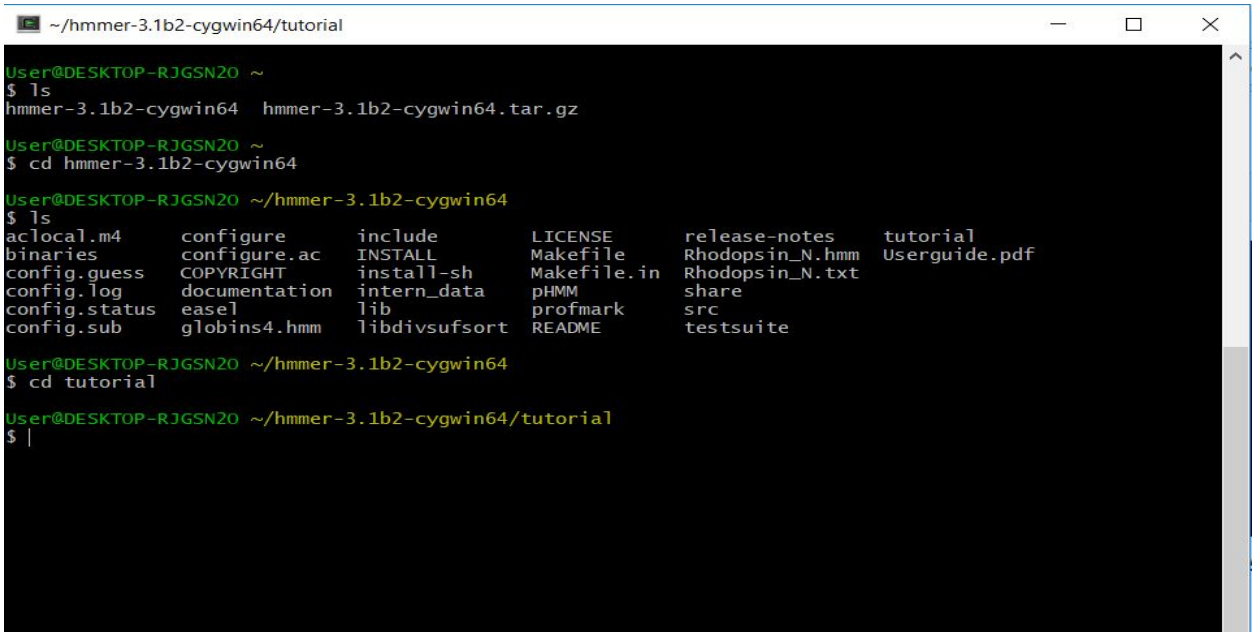

Figure - 5.5

Now we will create pHMM of a domain named '***globins4***' having MSA in '***.sto***' format.

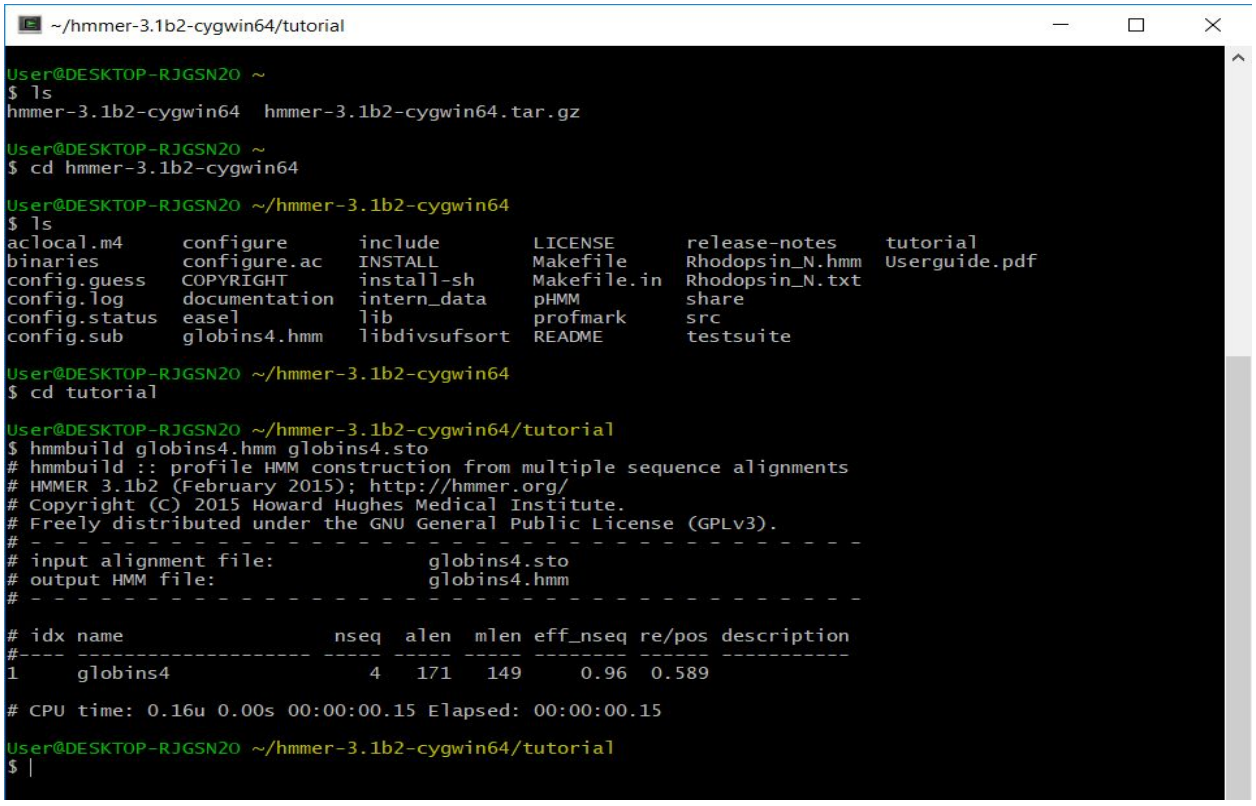

Figure - 5.6

Notice that the command for the creation of pHMM is ***'hmmbuild globins4.hmm globins4.sto'*** here the '**hmmbuild**' is the command to create the the pHMM from the file mention in second tuple i.e., '**globins4.sto**' to the file named '**globins4.hmm**'. Now you can check the file in the same directory named '**globins4.hmm**' to get your pHMM.

Now we will explain how to read this file that is generated by the programme

```
HMMER3/f [3.1b2 | February 2015]
NAME  globins4
LENG  149
ALPH  amino
RF    no
MM    no
CONS  yes
CS    no
MAP   yes
DATE  Mon Jul  3 17:43:15 2017
NSEQ  4
EFFN  0.964844
CKSUM 2027839109
STATS LOCAL MSV       -9.9014  0.70957
STATS LOCAL VITERBI  -10.7224  0.70957
STATS LOCAL FORWARD   -4.1637  0.70957
HMM          A        C        D        E        F        G        H        I        K        L        M        N        P
             m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO   2.36553  4.52577  2.96709  2.70473  3.20818  3.02239  3.41069  2.90041  2.55332  2.35210  3.67329  3.19812  3.455
          2.68640  4.42247  2.77497  2.73145  3.46376  2.40504  3.72516  3.29302  2.67763  2.69377  4.24712  2.90369  2.737
          0.57544  1.78073  1.31293  1.75577  0.18968  0.00000        *
      1   1.70038  4.17733  3.76164  3.36686  3.72281  3.29583  4.27570  2.40482  3.29230  2.54324  3.63799  3.55099  3.931
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355  4.24690  2.90347  2.737
          0.03156  3.86736  4.58970  0.61958  0.77255  0.34406  1.23405
      2   2.62748  4.47174  3.31917  2.82619  3.63815  3.49607  2.75382  3.03401  2.75280  2.74783  3.65114  3.24714  2.623
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355  4.24690  2.90347  2.737
          0.02321  4.17053  4.89288  0.61958  0.77255  0.48576  0.95510
      3   3.50771  4.88753  4.66754  4.31907  3.27776  4.35743  4.88268  2.50779  4.08449  0.57907  3.22569  4.56607  4.748
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355  4.24690  2.90347  2.737
          0.02321  4.17053  4.89288  0.61958  0.77255  0.48576  0.95510
      4   2.34080  4.28719  3.51550  3.22063  4.37406  3.06195  4.29366  3.74891  3.24370  3.47337  4.31943  3.39310  3.802
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355  4.24690  2.90347  2.737
```

Figure - 5.7

HMMER3/f : denotes the version of HMMER used to create this pHMM.

MANE : is the name of the domain

LENG : Length of the total pHMM

RF, MM, CONS, CS, MAP : Are some of the boolean parameters

DATE : Date of the creation of the pHMM.

NSEQ, EFFN, CKSUM, STATS LOCAL MSV : Parameters

STATS LOCAL VITERBI, STATS LOCAL FORWARD : Parameters

HMM : This line contains all of the possible 20 residues

[NOTE : Read these chapters from the documentation of HMMER chapter-3 and chapter-8 for better understanding of how HMMER works and to understand the data in the file **]**

Fisher Score

Introduction

This parameter says the amount of information hold by an unknown variable. We will use fisher score to know the amount of information in the sequence and represent it as a vector for the training of the stacked auto encoder. Fisher score is a good strategy to get to know the amount of information hold by the sequence as it takes both structural data and sequential structure into account.

Theory

Fisher score is defined as the derivative of the log-likelihood score for the query sequence x with respect to a particular parameter of the model. In this work we will focus on the emission probability of the ipHMM. If the probability of emitting amino acid *x* from state *s* is named $\theta_{x,s}$ the Fisher score of the model with respect to $\theta_{x,s}$ is therefore defined as

$$\frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \log P\left(x|\theta\right) = \frac{\varepsilon\left(\tilde{x},\tilde{s}\right)}{\theta_{\tilde{x},\tilde{s}}} - \varepsilon\left(\tilde{s}\right)$$

Where $\varepsilon(\tilde{s}) = \Sigma_x \varepsilon(x, \tilde{s})$ and the summation runs over the 20 different amino acids.

[**Note :** Read paper named Fisher Score for more details]

References

1) Constrained Fisher Scores Derived from Interaction Profile Hidden Markov Models Improve Protein to Protein Interaction Prediction
2) Prediction of residue-residue contact matrix for protein-protein interaction with Fisher score features and deep learning
3) Predicting Protein-protein Interactions, Interaction Sites And Residue-residue Contact Matrices With Machine Learning Techniques
4) Convolutional neural network architectures for predicting DNA–protein binding
5) Deep learning for computational biology
6) Clustering with the Fisher Score
7) HMMER User's Guide
8) Modelling interaction sites in protein domains with interaction profile hidden Markov models
9) Predicting domain-domain interaction based on domain profiles with feature selection and support vector machines
10) Prediction of contact matrix for protein–protein interaction