Antibiofilm Peptide Classification using Feed Forward Neural Network

Samyuktha Venkatesan, SCU ID: 07700006461, MCC Score: 87.93%, Rank: 13

Abstract

With the exponential growth of medical data, Machine Learning algorithms have been used to analyse, diagnose, and extract valuable insights from them. Biofilms are microorganisms that cause majority of human infections. Due to their robust multicellular matrix structure, they are resistant to both human immune systems and to antimicrobial compounds. This report discusses an approach to identify antibiofilm peptides that have complex mechanisms of action, directly targeting bacterial membranes, causing loss of cellular integrity thus treating biofilm infections. The approach involves using a feed forward neural network for antibiofilm peptide classification and k-mers for feature extraction.

1. Introduction

Proteins are macromolecules, composed of at least one long chain of amino acid residues, called polypeptides. They perform wide array of functions in the human body and differ from one another in their sequence of amino acids. Short polypeptides, often called peptides, typically contain fewer than 20-30 residues and are not considered proteins. Encoded in the genetic code, the sequence of gene defines the sequence of amino acid residues in a protein

Biofilms, tightly connected microorganisms are a major cause of diseases in human beings contributing to 75% of human infections. They are resistant to both host defense mechanisms and to traditional antibiotics. Therefore, it is necessary to identify peptide sequences that are not only antimicrobial but also antibiofilm. This experiment is intended to determine whether a given peptide sequence is an antibiofilm or not. The below methodology aims to handle this classification problem using a Feed Forward Neural Network.

2. Approach

In this experiment, the classification problem was addressed by creating an Artificial Neural Network with 2 hidden layers. Feature extraction was performed on the training dataset which consisted of peptide strings using k-mers. The feature vectors were then normalised and split into 80-20 ratio for training and validation. The network architecture was then designed to have 2 hidden layers, ReLU as their intermediate activation functions and sigmoid activation was used for the final classification. Several other techniques like oversampling, regularization and adaptive learning rate were implemented to get improved results. Based on the performance on the validation set, the loss function, activation function along with the hyperparameters of the ANN model such as number of neurons, learning rate, were tuned to achieve best results.

3. Methodology

3.1. Data Preparation:

The training data consisted out of 1566 records out of which 1424 were negative samples and 142 were positive. The data was then split uniformly across the classes into 80% training dataset (1138 negative sample and 114 positive samples) and 20% validation dataset (286 negative samples and 28 positive samples). To avoid the model's bias, the observable imbalance in the training dataset was fixed by oversampling the positive sample using SMOTE (Synthetic Minority Oversampling

Technique) method making it 227 samples after training and testing data split. The index for oversampling was set after several experimentations with different values (1138 negative samples and 227 positive samples).

3.2. Feature Extraction

Once the data was analysed and processed, feature extraction was performed using k-mers. Explored with different values of k-mers and attained optimal results when combining features generated from k-mer values of 1, 2 and 3. Consequently, generated a dense feature matrix with each row representing a training instance, each column representing a feature generated from k-mer and each value in the matrix representing the frequency of every feature in all samples. The features matrix was then normalised using min-max normalisation. Started with an initial feature matrix of 406 features and attained best results with a feature matrix of 7264 features.

3.3. Network Architecture

The experiment began with a small network architecture and employed a small subset of training samples (100) to strategically identify a model that overfits. Subsequently, increased the number of instances, features and fine-tuned the network configurations. Mini batch gradient descent was used for weight updates.

3.3.1. Hidden Layers and Hidden Neurons

Started with a single hidden layer incorporating 50 neurons and gradually increased the neuron count. Introduced another hidden layer, employing 400 hidden neurons for the first layer and 50 for the second. The decision was made to maintain the number of hidden neurons somewhere between input (7264) and output neurons. Optimal results were achieved with 2 hidden layers, that included 600 hidden neurons in the first layer and 100 in the second layer.

3.3.2. Activation Functions

ReLU was chosen as the activation function for the hidden layers as it reduces the likelihood of vanishing gradient and is computationally efficient. For the output layer, experimented with Tanh and Sigmoid activation functions. Tanh was chosen at the beginning as it provided an output range of -1 to 1. Transitioned to Sigmoid and achieved better results after tuning the threshold to 0.6.

3.3.3. Loss Function

Evaluated model's performance with several loss functions such as Hinge loss, BCE loss and MSE loss. When using Tanh activation function, employed Hinge Loss but observed unsatisfactory results. The Binary Cross Entropy loss function was used when using Sigmoid activation function. Following a thorough assessment, the Mean Squared Error (MSE) loss function was selected as it consistently yielded best results.

3.3.4. Weight Initialization

The weight initialization was performed using standard He initialization for layers with ReLU activation function and Xavier initialization for the layer using Sigmoid.

3.3.5. Learning Rate

A range of learning rate from 0.01 to 0.001 were applied to assess the model's performance. Experimented with adaptive learning rate that was reduced to 0.9 of its value every 20 epochs. Finally, a constant learning rate of 0.001 was selected.

3.3.6. Number of epochs

The number of epochs was determined by identifying the onset of overfitting. For the final model, settled with an epoch value of 200.

3.3.7. Regularization

L2 regularization was utilised to overcome the issues of overfitting. An alpha value of 0.0001 was chosen after several assessments.

4. Results

The Matthew's Correlation Coefficient (MCC) was used to examine the performance of the network as it is preferred when using imbalanced dataset. Table below shows the results of the model for different network configurations.

Number of input features	Number of hidden layers	Activation function for hidden layers	Number of hidden neurons	Activation function for output layer	Number of epochs	ALPHA (Regularization)	MCC on validation dataset	MCC on testing dataset
406	1	ReLU	50	Sigmoid	70	0	0.8259	0.6977
427	1	ReLU	150	Sigmoid	300	0	0.9036	0.7889
7264	1	ReLU	150	Sigmoid	300	0	0.9404	0.8181
7264	2	ReLU, ReLU	400, 50	Sigmoid	120	0	0.9215	0.8310
7264	2	ReLU, ReLU	500, 50	Sigmoid	200	0	0.9404	0.8436
7264	2	ReLU, ReLU	600,100	Tanh	200	0	0.9215	0.8436
7264	2	ReLU, ReLU	600,100	Sigmoid	600	0	0.9215	0.7944
7264	2	ReLU, ReLU	600,100	Sigmoid	200	0	0.9404	0.8712
7264	2	ReLU, ReLU	600,100	Sigmoid	200	0.0001	0.9404	0.8793

Figure 1: Training and Validation loss over epoch, Training and Validation MCC over epoch

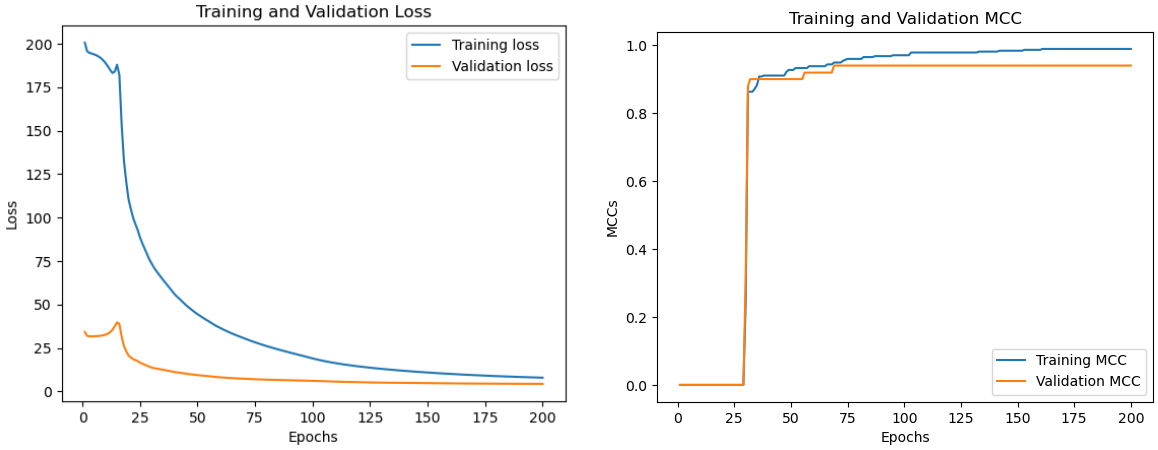
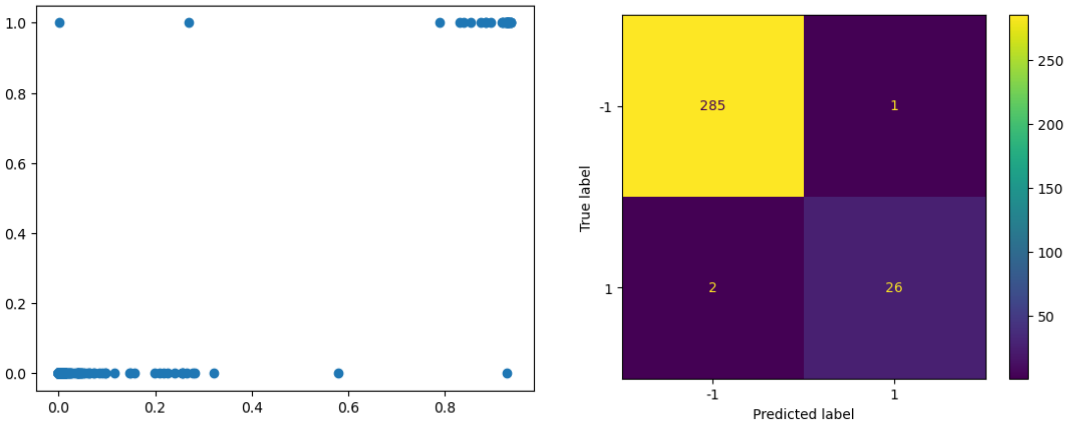


Figure 2: Scatter plot of Validation data, Confusion matrix for Validation data



5. Conclusion

The proposed method uses Feed Forward Neural Network for antibiofilm peptide classification and achieves a test accuracy of 87.93%. This can be further improved by performing different types of feature engineering, using larger dataset, and performing better regularization.

References

1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5019066/#:~:text=ANTIBIOFILM%20PEPTIDES%20TARGET%20A%20CELLULAR%20STRESS%20RESPONSE&text=Many%20events%20in%20biofilm%20development,%2Dsensing%20mechanisms%2C%20and%20dispersal.
2. https://en.wikipedia.org/wiki/Protein
3. https://byeungchun.medium.com/machine-learning-for-biological-sequence-data-using-python-573d82f6f17a
4. https://www.kaggle.com/code/singhakash/dna-sequencing-with-machine-learning
5. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
6. https://discourse.pymc.io/t/dealing-with-unbalanced-data-with-minibatches/1309/4
7. https://imbalanced-learn.org/stable/over_sampling.html
8. https://machinelearningmastery.com/develop-your-first-neural-network-with-pytorch-step-by-step/
9. https://www.frontiersin.org/articles/10.3389/fmicb.2021.783284/full
10. https://chat.openai.com/