Antibiofilm Peptide Classification using Feed Forward Neural Network

Satya Harthik Sonpole
SCU ID: 07700012343
MCC Score: 82.58%, Rank: 10

Abstract

This report presents the implementation of a feedforward neural network for a binary classification problem using peptide sequence data. The focus is on developing a model from scratch using NumPy and optimizing it for performance. Feature extraction was performed using a k-mer based bag-of-words approach to convert peptide sequences into numerical representations. The dataset was imbalanced, requiring oversampling to improve class distribution. The model architecture consists of an input layer corresponding to k-mer features, two hidden layers with ReLU activation, and a final output layer with Sigmoid activation for binary classification. Training was performed using mini-batch gradient descent with a learning rate of 0.001. Regularization techniques were applied to control overfitting. The model was trained for 200 epochs and evaluated using the Matthews Correlation Coefficient (MCC) metric. The results were as expected, and improvements can be made by exploring different feature representations, adjusting hyperparameters, or incorporating alternative architectures.

1 Introduction

The objective of this assignment is to develop a neural network from scratch using NumPy for a classification task. Given a dataset of peptide sequences, the goal is to extract numerical features and train a model to distinguish between two classes. This involves preprocessing data, designing a neural network architecture, implementing forward and backward propagation, and optimizing model performance. Unlike standard machine learning libraries such as TensorFlow or PyTorch, this assignment requires manually handling weight updates and activations, reinforcing core deep learning concepts.

2 Methodology

2.1 Data Preparation

The dataset consists of labeled peptide sequences in the training set and unlabeled sequences in the test set. Each sequence is composed of characters representing amino acids. The labels are provided as -1 (non-antibiofilm) and 1 (antibiofilm). The dataset is highly imbalanced, requiring resampling techniques to improve classification performance.

The data was preprocessed by extracting numerical features from the sequences. The training dataset was split into 80% training and 20% validation. To address class imbalance, oversampling was applied to the minority class using the RandomOverSampler method with a 0.2 sampling ratio.

2.2 Feature Extraction

A k-mer based bag-of-words approach was used to convert peptide sequences into feature vectors:

- Each sequence was decomposed into overlapping k-mers (subsequences of length k). A value of $k = 4$ was used.
- Unique k-mers across the dataset were identified, forming the vocabulary of features.
- Each sequence was transformed into a feature vector based on k-mer frequency.
- The feature matrix was normalized to ensure numerical stability during training.

This resulted in a feature space of 7264 dimensions.

2.3 Neural Network Design

The classification model was implemented as a fully connected feedforward neural network with the following architecture:

- Input layer: 7264 neurons (corresponding to extracted k-mer features).
- Hidden layers:
  - First hidden layer: 625 neurons with ReLU activation.
  - Second hidden layer: 110 neurons with ReLU activation.
- Output layer: 1 neuron with Sigmoid activation for binary classification.

2.4 Training Strategy

- The dataset was split into training and validation sets.
- The network was trained using mini-batch gradient descent.
- A batch size of 128 was used for weight updates.
- The learning rate was initially set to 0.001 and reduced adaptively every 20 epochs.
- L2 regularization was applied to control weight magnitudes and prevent overfitting.

2.5 Loss Function

The Mean Squared Error (MSE) loss function was used:

$$\text{Loss} = \sum((y - y_{pred})^2) + \lambda(||W_1|| + ||W_2|| + ||W_3||)$$

Regularization was included to limit excessive weight updates.

2.6 Weight Initialization

- Xavier initialization was used for layers with Sigmoid activation.
- He initialization was used for layers with ReLU activation.

2.7 Evaluation Metric

Since the dataset is imbalanced, accuracy is not a suitable metric. The Matthews Correlation Coefficient (MCC) was used instead:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC considers all four confusion matrix components and provides a better representation of model performance in imbalanced datasets.

3 Results

The model was trained for 200 epochs and evaluated on the test set. The results were consistent with expectations.

Hidden Layers	Neurons	Activation Function	Epochs	Train MCC	Test MCC
2	625, 110	ReLU, ReLU, Sigmoid	200	0.9216	0.8258

Table 1: Final model performance metrics

4 Conclusion

This project implemented a feedforward neural network from scratch using NumPy to classify peptide sequences. Feature extraction was performed using k-mers, and training was conducted using mini-batch gradient descent with ReLU and Sigmoid activations. The expected results were obtained, demonstrating the effectiveness of the implemented approach. Future improvements could include experimenting with different feature extraction techniques, alternative network architectures, and additional optimization methods.

5 References

1. Imbalanced-learn documentation, "RandomOverSampler," Available: `https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html`
2. ScienceDirect, "Neural Network Optimization," Available: `https://www.sciencedirect.com/science/article/pii/S2001037024001703`
3. TensorFlow Playground, Available: `https://playground.tensorflow.org/`
4. Medium, "Neural Network Feedforward and Backpropagation with NumPy," Available: `https://medium.com/the-bytedoodle-blog/neural-network-feedforward-and-backpropagation-`
5. ChatGPT, OpenAI