Travis Mayer
SCU ID: 00001184297
CSEN 342
Jan 31, 2024
Rank: 4
MCC Score: 0.9063

# Development of a Neural Network for Peptide String Classification

## Abstract:

This report documents the exploratory journey of developing a neural network for peptide string classification, a journey that began with no prior knowledge in machine learning. The project unfolded in two phases: initial exploration using the SKLEARN library and subsequently building a neural network from scratch. The key learning curve revolved around understanding and implementing oversampling techniques to address data imbalances, which significantly influenced the model's error rate and Matthew's Correlation Coefficient (MCC) score. The exploration included experimenting with various epoch ranges, ultimately settling on a specific count, and navigating the nuances of activation functions and hyperparameter tuning.

## Introduction:

Embarking on a project to classify peptide strings using a neural network presented a unique learning opportunity in the field of machine learning. With no prior experience in this domain, the project was initiated using SKLEARN to establish a foundational understanding of neural networks, focusing on the effects of hyperparameters, data formatting, and the role of activation functions like ReLu. This also lent an easier way to play around with tuning the oversampling rate of the positive class. For both the neural network from scratch, and the SKLEARN neural network, the feature extraction implementation was done using the k-mer model that was explored in class.

## Approach Methodology:

The approach to this project was twofold. The first phase involved leveraging the capabilities of the SKLEARN library to grasp the fundamental aspects of neural networks. This stage emphasized understanding the intricacies of hyperparameters, particularly the impact of oversampling in skewed datasets. The realization how tuning the oversampling rate could drastically affect the model's performance was pivotal. Exploring the k-mer model was also very important. Using a k-mer size of 2 left vectors of $x_i \in \mathbb{R}^d, \ where\ d\ = 436$. Utilizing this knowledge was unnecessary in SKLEARN, but was obviously essential in the neural network from scratch.

## Actual Approach:

Transitioning from SKLEARN to developing a custom neural network required intensive research and hands-on experimentation. Key steps in this process included:

- Deliberating on the number of layers and neurons in the model. The final implementation utilizes a total of 4 layers. The number of neurons per layer follows the power of 2 rule, with the first layer utilizing 64 neurons for 436 dimension input, and the final layer utilizing 16 for a singular output. Shifting the k-mer value would require changing the dimension of the input, and SHOULD naturally mean changing the number of neurons respectively. Attempting k-mers of size 3 resulted in 7532 dimension vectors, massively increasing the required computational time of the neural network, without significant improvement in the MCC scores, and in some cases even showing notably deteriorated MCC scores.
- Utilizing the Tanh activation function over ReLu and Sigmoid, as it seemed to result in a dramatically improved error value in the epochs.
- Developing a methodology to translate the neural network's probabilistic outputs into concrete classifications (-1 or 1), utilizing a fairly standard threshold value of 0.5.
- Rigorously experimenting with oversampling rates, noting their critical impact on both error rates and the MCC score. The provided training set was so highly skewed to negative values, that attempting to oversample to positive class to any significant degree led to multiple duplicates of the same positive class values being spread throughout the data. Thus, experimentation with the rate of oversampling was necessary. This experimentation was essential in avoiding model overfitting and ensuring the model learned underlying patterns rather than memorizing the training data. The final oversampling rate utilized is the difference between the positive class and the negative class, divided by 2.
- Extensive tuning of the learning rate and the number of epochs. Epoch ranges from as low as 50 to as high as 10,000 were tested. Ultimately, a balance was struck with an epoch count of 200, which effectively reduced error rates without leading to overfitting.
- Learning to recognize that the error rate of the epochs during the training was not necessarily indicative of the model's performance on the test set, and corresponding MCC score. An error rate too good was usually an indication of overfitting through excessive oversampling, epochs, or from extraneous learning rates that would either skip over so called 'valleys' in the data and thus fail to converge, or get stuck in the 'valley's and lead to the overfitting.

## Conclusion:

This neural network project underscored the complex and iterative nature of machine learning model development. The final achievement, an MCC score of 0.9063, was a direct result of meticulous tuning, especially in terms of oversampling rates and epoch count, as well as what I presume to be luck, in the randomness of weights and biases (as sometimes the exact same hyperparameters could still lead to significant differences in the MCC score). This experience not only fulfilled the objective of accurately classifying peptide strings but also provided profound insights into the dynamic between theoretical understanding and practical application in machine learning. Learning the critical role of experimental tuning and the adaptable nature of neural network development was very important, and highlighted by the contrast of the SKLEARN network, and the neural network from scratch.