UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECB 4313-01

BIOINFORMATICS MODELING AND SIMULATION

ASSIGNMENT 2

Lecturer:

Dr. Azurah bte A Samah

Group Members:

Chong Kah Wei (A20EC0027)

Heong Yi Qing (A20EC0043)

Mek Zhi Qing (A20EC0077)

Zereen Teo Huey Huey (A20EC0173)

Semester 2 2023/2024

1. Identify 4 hyperparameters and propose two values for each hyperparameter. Justify the selection of the hyperparameters and its corresponding values.

The four hyperparameters chosen are loss function, optimizer, learning rate of the optimizer and also the number of epochs during the training process. Table 1 shows the hyperparameter with their corresponding values.

Hyperparameter	Values
Loss function	Mean Squared Error
	Binary Cross Entropy
Optimizer	Adam
	Lion
Learning rate of optimizer	0.01
	0.001
Number of Epochs	500
	1000

Table 1: Summary of Hyperparameters and Values

Loss Function

Loss function helps to evaluate the effectiveness of the algorithm in modeling the featured dataset and it is crucial in training the deep learning model. Generally, there are two main types of loss functions, specifically designed for different types of neural networks. Mean Squared Error (MSE) is normally used for regression problems and it finds the average squared differences between the actual values and the predicted output. It is particularly relevant in this case study because the output layer of the neural network uses a sigmoid activation function, producing continuous probability values between 0 and 1. MSE helps the model minimize the error between these predicted probabilities and the actual values. On the other hand, Binary Cross Entropy is suited for binary classification problems. It measures the performance of a classification model by comparing the predicted probability with the actual binary outcome. By

optimizing binary cross entropy, the model's output probabilities become more accurate representations of the true class probabilities. Performing hyperparameter tuning using these two loss functions tailor the model training to the specific nature of the problem, ensuring the employment of the most appropriate loss function for improving model performance.

Optimizer

An optimizer is a function or an algorithm that adjusts the attributes of the neural network, such as weights and learning rates in order to reduce the overall loss and improve the model accuracy. As one of the most popular optimizers, Adam (Adaptive Moment Estimation) is an iterative optimization algorithm used to minimize the loss function during the training of neural networks. Adam can be considered as a combination of RMSprop and Stochastic Gradient Descent with momentum. This combination allows Adam to dynamically adjust the learning rate for each parameter, making it particularly effective for dealing with sparse gradients and achieving faster convergence. Using Adam as an optimizer ensures that the model benefits from robust training dynamics, lowering the chances of getting stuck in suboptimal solutions. On the other hand, Lion (EvoLved Sign Momentum) is a stochastic-gradient-descent method that utilizes the sign operator to control the magnitude of the update. It is more memory-efficient than Adam as it only focuses on tracking the momentum. Other than that, it is proven more effective than Adam in some of the cases. Therefore, in this case study, Adam and Lion are used for hyperparameter tuning to find a more effective optimizer in producing a more robust classification model.

Learning Rate of Optimizer

The learning rate determines how many steps a model takes to achieve the minimum loss function. Basically, a higher learning rate allows the model to learn more quickly, but it may miss the minimum loss function or merely reach its surroundings. A lower learning rate improves the chances of finding a minimum loss function. As a trade-off, lower learning rates require higher epochs, or more time and memory capacity resources. In this case study, a learning rate of 0.01 and 0.001 is used in hyperparameter tuning. A learning rate of 0.01 is relatively high and can lead to faster convergence, making it suitable for problems where the initial model weights are far from optimal. In contrast, a learning rate of 0.001 is lower, allowing for finer adjustments to

the weights. Hyperparameter tuning in this case helps to identify a more suitable learning rate for the model.

Number of Epochs

Epoch is defined as one entire passing of training data through the algorithm. In this context, number of epochs is one of the important hyperparameters in model training as it is crucial for determining how well the model can learn the patterns in the data. In this case study, 500 and 1000 epochs are used for hyperparameter tuning. Training for 500 epochs can basically provide sufficient opportunities for the model to learn and refine its weights with acceptable computational cost. On the other hand, extending the training to 1000 epochs might be beneficial for complex datasets as the model might need more iterations to fully capture the underlying patterns. However, there are also risks of overfitting if too many epochs are used, causing the model to lose its ability to generalize to new data. Therefore, it is important to perform hyperparameter tuning to make sure the model is trained under a reasonable number of epochs to get a more reliable model.

2. Construct a tree diagram that displays the proposed hyperparameters and its corresponding values.

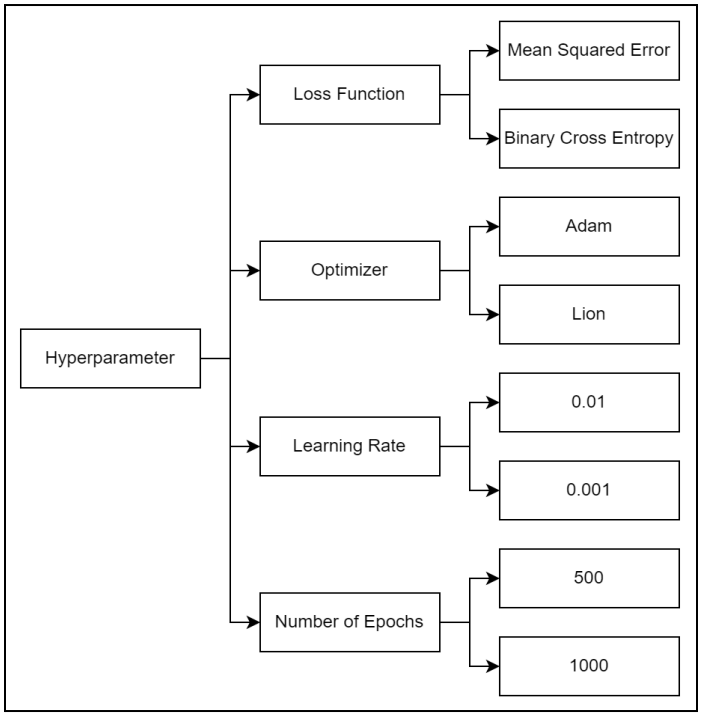

Figure 1: Tree Diagram

3. Tabulate the proposed experimental design based on your tree diagram.

Experiment	Loss Function	Optimizer	Learning Rate	Epochs
1	Mean Squared Error	Adam	0.01	500
2	Mean Squared Error	Adam	0.01	1000
3	Mean Squared Error	Adam	0.001	500
4	Mean Squared Error	Adam	0.001	1000
5	Mean Squared Error	Lion	0.01	500
6	Mean Squared Error	Lion	0.01	1000
7	Mean Squared Error	Lion	0.001	500
8	Mean Squared Error	Lion	0.001	1000
9	Binary Cross Entropy	Adam	0.01	500
10	Binary Cross Entropy	Adam	0.01	1000
11	Binary Cross Entropy	Adam	0.001	500
12	Binary Cross Entropy	Adam	0.001	1000
13	Binary Cross Entropy	Lion	0.01	500
14	Binary Cross Entropy	Lion	0.01	1000
15	Binary Cross Entropy	Lion	0.001	500
16	Binary Cross Entropy	Lion	0.001	1000

Table 2: Experiment with the Corresponding Hyperparameter Set

4. Perform *hyperparameter tuning* to improve current result. (Simulate the model and collect the results)

To achieve optimal results in hyperparameter tuning for the defined sets, we utilize for loops to systematically test each hyperparameter combination one by one. In this deep learning classification task, a total of 16 iterations will be executed, corresponding to the number of

hyperparameter sets being evaluated. Each loop iteration involves training the model with a unique set of hyperparameters and evaluating its performance. The code implementation of this process is illustrated in Figure 2.

```
# Placeholder for the results
results = []

# Hyperparameters
optimizers = {'Adam': tf.keras.optimizers.Adam, 'Lion': tf.keras.optimizers.Lion}
learning_rates = [0.01, 0.001]
loss_functions = ['mse', 'binary_crossentropy']
epochs_list = [500, 1000]

# Iterate over all combinations of hyperparameters
for optimizer_name, optimizer in optimizers.items():
    for lr in learning_rates:
        for loss_function in loss_functions:
            for epochs in epochs_list:
                # Build the model
                model = Sequential()
                model.add(Dense(64, input_dim=21, activation='softmax'))
                model.add(Dense(32, activation='softmax'))
                model.add(Dense(1, activation='sigmoid'))

                # Compile the model
                model.compile(loss=loss_function, optimizer=optimizer(learning_rate=lr), metrics=['accuracy'])

                # Train the model
                model.fit(x_train, y_train, validation_data=(x_val, y_val), epochs=epochs, batch_size=16, verbose=1)

                # Predict on test data
                y_pred = model.predict(x_test)
                y_pred = np.round(y_pred).astype(int)

                # Performance Evaluation
                accuracy = accuracy_score(y_test, y_pred)
                precision = precision_score(y_test, y_pred)
                recall = recall_score(y_test, y_pred)

                # Store the result
                result = {
                    'optimizer': optimizer_name, 'learning_rate': lr,
                    'loss_function': loss_function, 'epochs': epochs,
                    'test_accuracy': accuracy,
                    'precision': precision, 'recall': recall
                }
                results.append(result)
```

Figure 2: Hyperparameter Tuning using For Loop

5. Tabulate the results based on your proposed experimental design.

Experiment	Hyperparameter Values	Accuracy (%)	Precision (%)	Recall (%)
1	Loss function: Mean Squared Error	87.10	93.33	82.35
	Optimizer: Adam			
	Learning Rate: 0.01			
	Epochs: 500			
2	Loss function: Mean Squared Error	80.65	78.95	88.24

	Optimizer: Adam			
	Learning Rate: 0.01			
	Epochs: 1000			
3	Loss function: Mean Squared Error	58.06	56.67	100.00
	Optimizer: Adam			
	Learning Rate: 0.001			
	Epochs: 500			
4	Loss function: Mean Squared Error	80.65	92.31	70.59
	Optimizer: Adam			
	Learning Rate: 0.001			
	Epochs: 1000			
5	Loss function: Mean Squared Error	77.42	77.78	82.35
	Optimizer: Lion			
	Learning Rate: 0.01			
	Epochs: 500			
6	Loss function: Mean Squared Error	74.19	76.47	76.47
	Optimizer: Lion			
	Learning Rate: 0.01			
	Epochs: 1000			
7	Loss function: Mean Squared Error	83.87	92.86	76.47

	Optimizer: Lion			
	Learning Rate: 0.001			
	Epochs: 500			
8	Loss function: Mean Squared Error	83.87	87.50	82.35
	Optimizer: Lion			
	Learning Rate: 0.001			
	Epochs: 1000			
9	Loss function: Binary Cross Entropy	83.87	87.50	82.35
	Optimizer: Adam			
	Learning Rate: 0.01			
	Epochs: 500			
10	Loss function: Binary Cross Entropy	77.42	77.78	82.35
	Optimizer: Adam			
	Learning Rate: 0.01			
	Epochs: 1000			
11	Loss function: Binary Cross Entropy	67.74	62.96	100.00
	Optimizer: Adam			
	Learning Rate: 0.001			
	Epochs: 500			
12	Loss function: Binary Cross Entropy	80.65	92.31	70.59

	Optimizer: Adam			
	Learning Rate: 0.001			
	Epochs: 1000			
13	Loss function: Binary Cross Entropy	77.42	77.78	82.35
	Optimizer: Lion			
	Learning Rate: 0.01			
	Epochs: 500			
14	Loss function: Binary Cross Entropy	74.19	80.00	70.59
	Optimizer: Lion			
	Learning Rate: 0.01			
	Epochs: 1000			
15	Loss function: Binary Cross Entropy	83.87	92.86	76.47
	Optimizer: Lion			
	Learning Rate: 0.001			
	Epochs: 500			
16	Loss function: Binary Cross Entropy	80.65	86.67	76.47
	Optimizer: Lion			
	Learning Rate: 0.001			
	Epochs: 1000			

Table 3: Experimental Result

6. Analyze the results. Which combination of hyperparameters generate the most improved result?

The hyperparameter tuning process explored various combinations of optimizers, learning rates, loss functions, and epochs to determine the most effective configuration for training a neural network. Different combinations of these hyperparameters might lead to different effects and produce models of different performance. Based on the findings, it is observed that the combination of the Adam optimizer with a learning rate of 0.001 and 500 epochs can result in a model with relatively low performance. This might be because a learning rate of 0.001 is quite small, and more epochs are needed for the Adam optimizer to achieve better performance. Additionally, the findings also show that the Lion optimizer generally produces better performance with a learning rate of 0.001 compared to 0.01 with other fixed hyperparameters.

Among the tested combinations of hyperparameters, the best results were obtained in Experiment 1 with the Adam optimizer, a learning rate of 0.01, the Mean Squared Error (MSE) loss function, and 500 epochs. This combination of hyperparameters achieved a testing accuracy of 87.10%, precision of 93.33%, and recall of 82.35%. For a better understanding of these performance metrics, the accuracy of 87.10% shows that the model correctly classifies 87.10% and can be considered as the majority of the test samples, precision of 93.33% means that when the model predicts the positive class, it is 93.33% correctly predicted whereas recall of 82.35% indicates that the model correctly identifies 82.35% of all actual positive cases. The accuracy and precision of the model produced using this particular combination have outperformed the other 15 hyperparameter sets, while also attaining a relatively high recall.

Deep into each hyperparameter of the best combination, the choice of the Mean Squared Error (MSE) as the loss function, proved to be effective in this classification context. This might be due to the specific characteristics of the data and the model architecture, where MSE is able to provide smoother and more consistent gradient updates compared to binary cross-entropy. The Adam optimizer played a crucial role in obtaining this result due to its adaptive learning rate mechanism, which combines the advantages of both RMSprop and Stochastic Gradient Descent with momentum. This adaptability allows Adam to dynamically adjust the learning rates for each parameter, effectively handling the varying gradients encountered during training. Other than

that, a higher learning rate of 0.01 enables the model to make significant updates to the weights in each iteration. This allows the model to capture important patterns in the dataset within a given number of epochs. Furthermore, 500 epochs is adapted in this combination of hyperparameters as it strikes a balance between sufficient learning and overfitting.