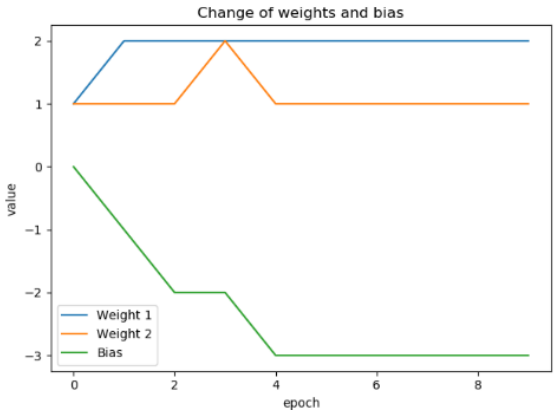
Report AND XOR Gate

Zemin XU

Exercise 1: Determine the minimum number of epochs necessary to train the model in order to obtain only valid predictions.

Answer:

The minimum number of epochs should be the epochs when the weights and bias remain the same after updating. The minimum num is 4 + 1 =5, because we start with 0.



Exercise 2: Modify the program in order to perform the prediction for an OR gate.

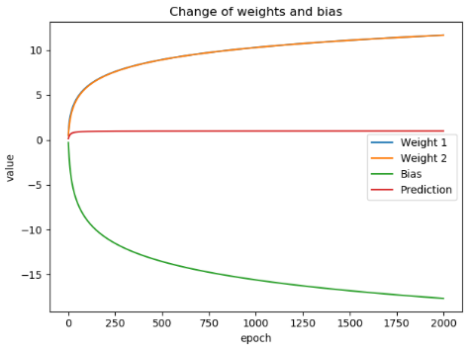
Answer:

In order to implement an OR gate, change the output label to the following: t = [0, 1, 1, 1].

Exercise 3: Change the neuron activation function from step unit to sigmoid. Are there any differences when performing the prediction?

Answer:

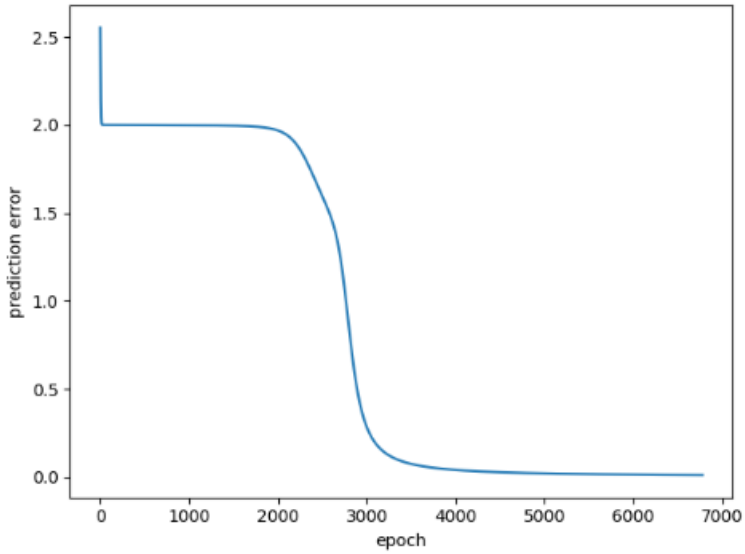
The difference is that the return value of sigmoid function is from 0 to 1, but it hardly achieve 0 or 1(infinite). As an activation function here, the prediction will never equals but near to 1 which will update the weights and bias. The weights and bias will change but the changing rate will converge to 0.



Exercise 4: Determine the minimum number of epochs necessary to train the model in order to obtain a prediction error inferior to 0.01.

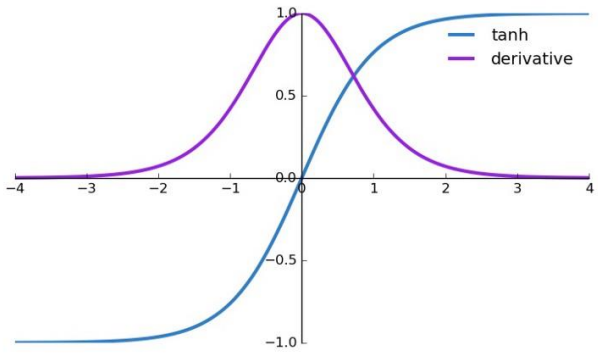
Answer:

Min epoch num is: 6080. It will vary at each running, near to 6000 epochs.



Exercise 5: Modify the activation function for the neurons from sigmoid to hyperbolic tangent. The hyperbolic tangent is given by the following equation: tanh.

What can be observed about the predicted values? Justify the results.

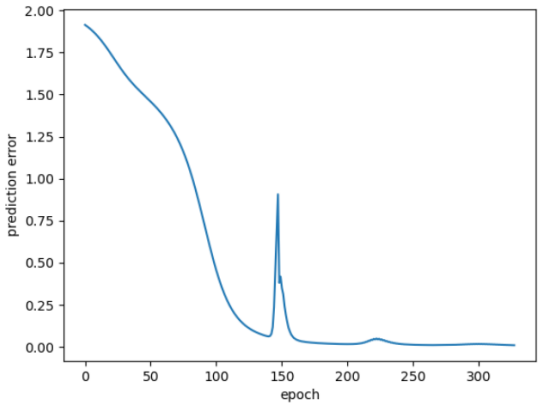


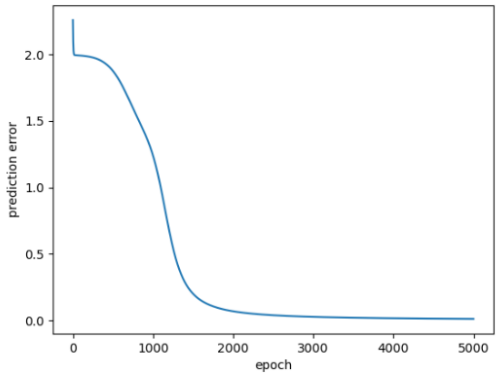
https://medium.com/@omkar.nallagoni/activation-functions-with-derivative-and-python-code-sigmoid-vs-tanh-vs-relu-44d23915c1f4

Answer:

According to the graph of tanh function, we can know that in the range of (-2, 2), the derivative is greater than in other ranges. The return value is close to -1 for value smaller than -2, and 1 for value greater than 2. This feature makes it change rapidly in range (-2,2). What's more, compared to sigmoid, it can reach minus number. A sigmoid will always return a positive number. Therefore, sigmoid can only converge to the correct answer, but tanh can reach it.

By experimenting, we can see that tanh takes about 200 epochs to have the acceptable error. While for sigmoid on the right side, it needs much more.





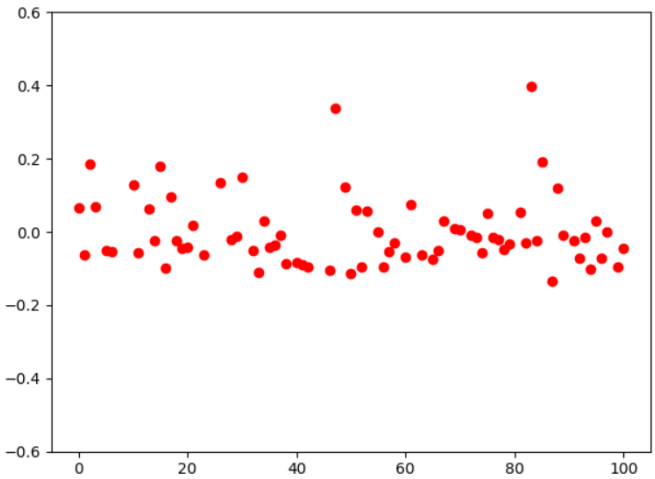
Exercise 6: How many hyper-parameters are involved in the ANN architecture existent of a XOR gate. Is the ANN performances influenced by the random initialization of the different variables?

Answer:

A hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training the data[1].

In this model, there are 'number of epochs' and 'learning rate' that are hyper-parameters. According to this definition, 'inputSize', 'noNeuronLayer1' and 'noNeuronLayer2' are also hyperparameters. They are decided by the model. The weights and the biases of different layers can be initialized randomly and will not greatly influence the results, so they are not hyperparameters.

In order to know the influence of random initialization, I calculate the M(median minimum steps) for 100 tests to achieve a prediction error inferior to 0.01. I use the D[i](difference of K[i] (minimum steps of each single test) and M), divided by M. We can see from the figure below that most of the test is in the range of 20% of difference compare to M. There are only 2 tests in 100 tests at about 40% steps more than M. The conclusion is that the random initialization will not influence too much the result.



Exercise 7: By using Python and some dedicated machine learning libraries (i.e.,Tensorsflow with Keras API) write a novel script that re-implements the ANN architecture of the AND gate

developed at Application 1 (consider as activation function sigmoid). Extend the code in order to predict the output for the XOR gate in Application 2. Both networks will use Adam as optimizer.

Answer:

The script can be found with name ‘exercise1_7.py’. For the AND gate I used only a two neurons input layer and one neuron output layer. For XOR gate I add a two-neuron hidden layer and use ‘tanh’ as activation function.

```
def train_AND_model(X_data, y_data):
    model = Sequential()
    model.add(Dense(1, input_dim=2, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X_data, y_data, epochs=15000, verbose=2)
    print(model.predict_proba(X_data))

def train_XOR_model(X_data, y_data):
    model = Sequential()
    model.add(Dense(2, input_dim=2, activation='tanh'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    model.fit(X_data, y_data,  epochs=5000, verbose=2)
    print(model.predict_proba(X_data))
```

Reference:

[1] Hyperparameter Tuning with Python: Complete Step-by-Step Guide | Towards Data Science