Research on hyperparameters of a neural network

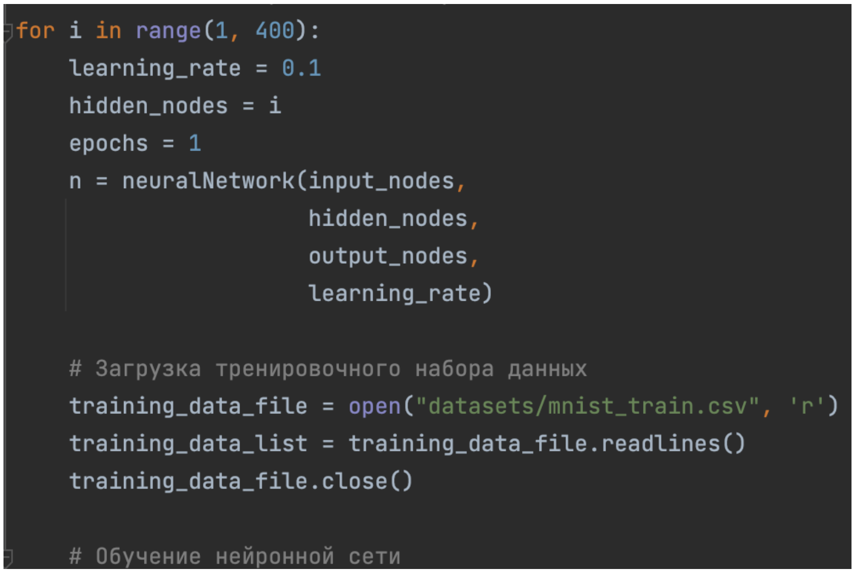
Aim

Using the code and dataset provided, define perceptron (written numbers recognition) hyperparameters that improve the quality of its work on the MNIST dataset:

a) learning level
b) number of epochs
c) number of neurons in the intermediate layer

Approach

Using a Python programming language, with the help of which the neural network (N.N.) was implemented, we could run some automated tests to find the best parameters. An instance of a neural network can be placed into a loop, so each iteration we will pass a new numeric value to a certain parameter and save every performance of the network, as a result, into a JSON file. Then make some analysis of received data.



```
for i in range(1, 400):
    learning_rate = 0.1
    hidden_nodes = i
    epochs = 1
    n = neuralNetwork(input_nodes,
                      hidden_nodes,
                      output_nodes,
                      learning_rate)

    # Загрузка тренировочного набора данных
    training_data_file = open("datasets/mnist_train.csv", 'r')
    training_data_list = training_data_file.readlines()
    training_data_file.close()

    # Обучение нейронной сети
```

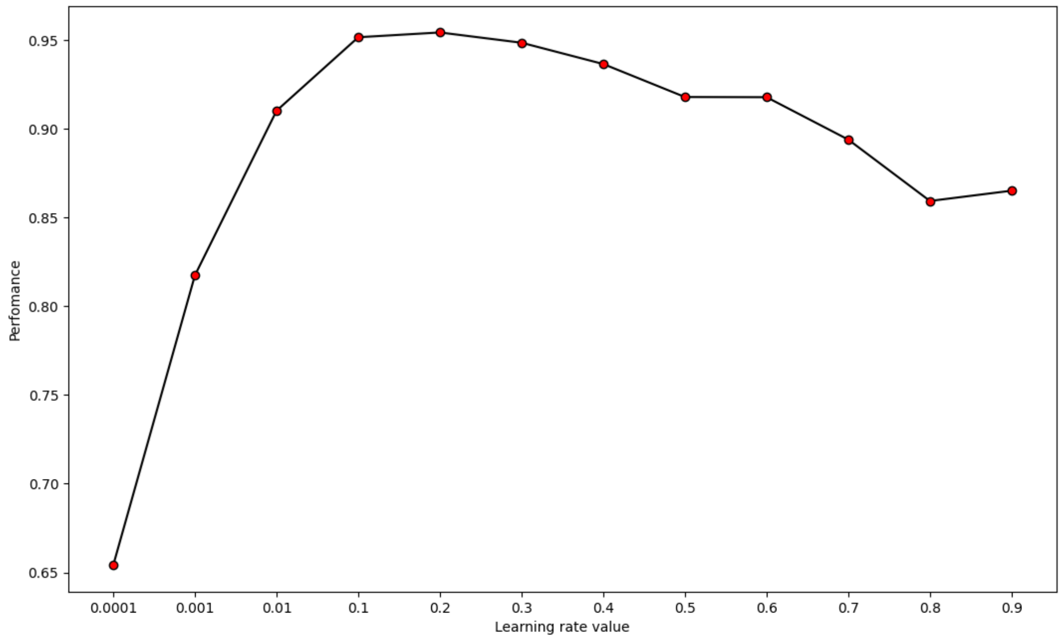
Example of hidden nodes are changed each iteration

Results

After multiple tests and hours of waiting, I can present next detailed overview:

Learning rate

Firstly I decided to find the best learning rate (L.R.) parameter which is the most important one for the whole learning process. Other parameters: hidden nodes - 200, number of epochs - 2.

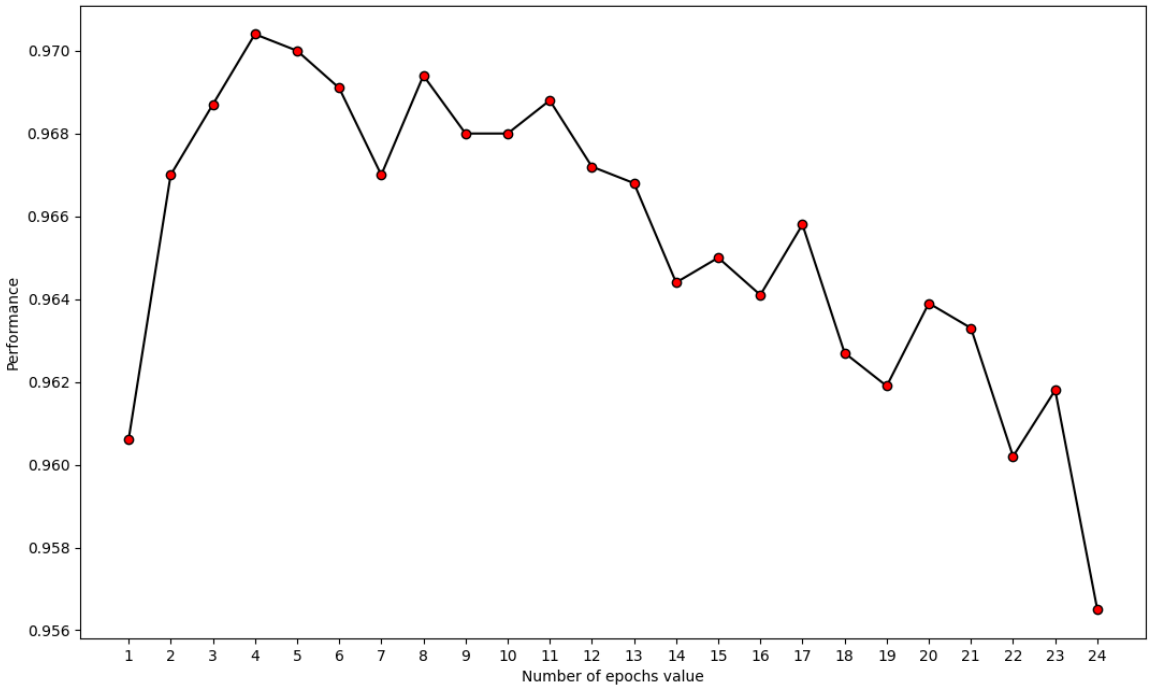


Graphed data of learning rate changes

As we can see on this plot, two main trends are the intensive increase of L.R. before X axis values of 0.1 and 0.2 and smooth fall till the end, after them. So we can consider that the desired parameter could be 0.1 or 0.2 which performed with ~0.95. We can notice that fractions after 0.2 lead to worse performance. This is because of the backpropagation aspect - while this process keeps going, weights are changed that much, so new weight isn’t equally distributed/centered. Later on multiple re-test of this logical part of our research was done, the graph looked almost the same. Now, in the next tests we can use the hyperparameter of 0.2, which is a learning rate.

Epochs

As we’ve agreed recently, a learning rate of 0.2 and 200 hidden nodes will be used in the following test.

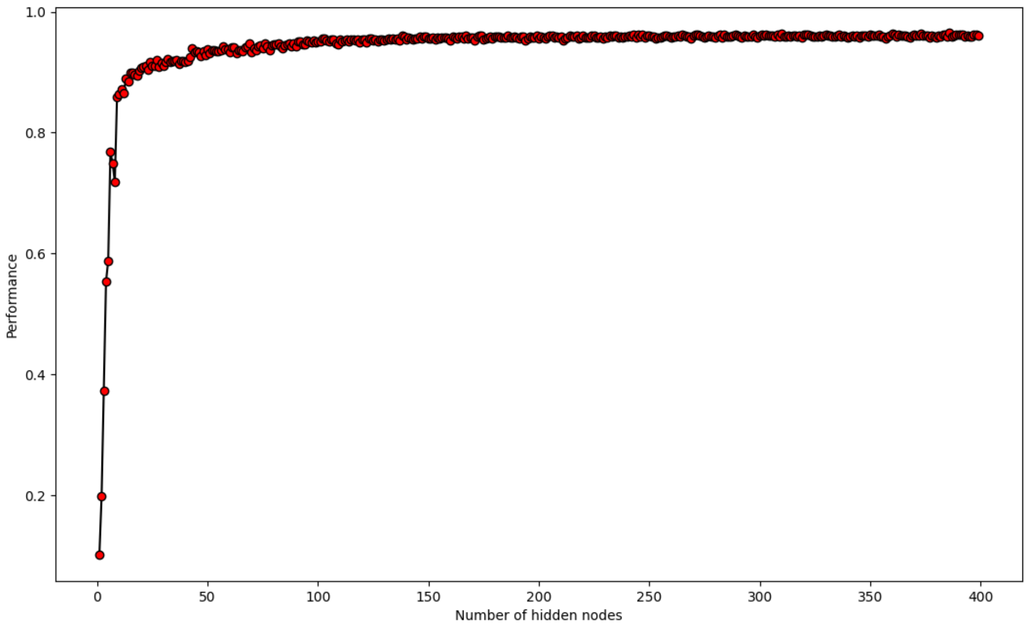


Graphed data of epochs number changes

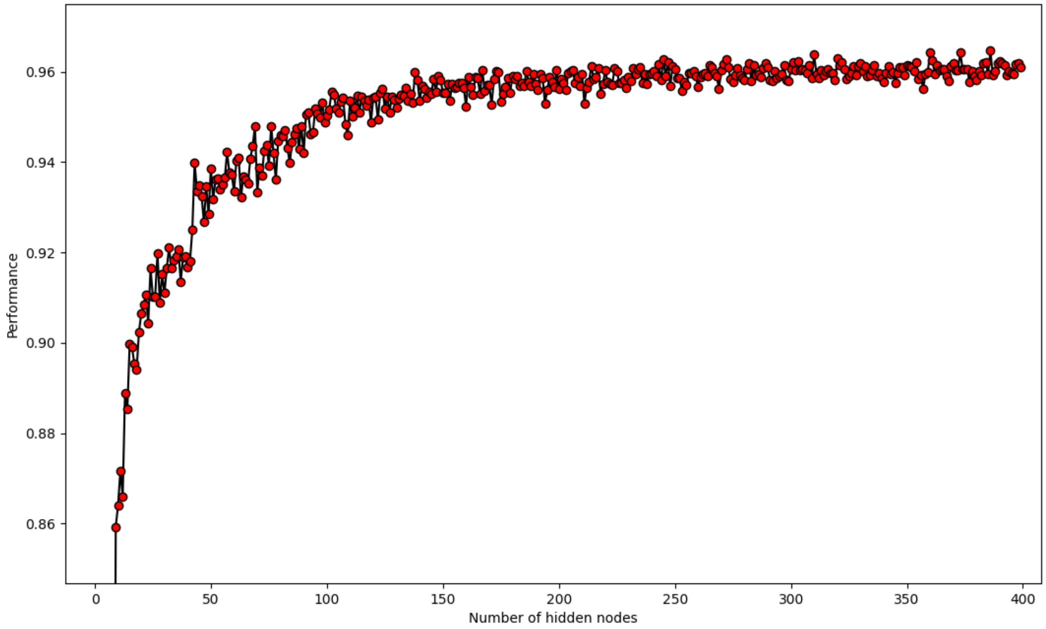
The first thing that can be noticed is the pretty chaotic behavior of the neural network. The main trend is a growth before 4 epochs and a decrease after it, but multiple jumps can be seen over all the graphs. The best performance can be achieved with the researched parameter of 4, which is ~0.961 and the lowest is ~0.956 - after 24 epochs. The conclusion can be as follows: mostly, higher number of epochs can lead to worse performance, the encountered phenomenon is called overtraining. Anyways, almost the fastest way - 2 is not bad at all, with an index of ~0.967, so next our tests will be done using it.

Hidden nodes

In the following research I decided to use from 1 to 400 hidden nodes and analyze the performance.



Graphed data of hidden nodes number changes



Zoomed graphed data of hidden nodes number changes

We can definitely state that a number of less than 9 nodes is not that good choice, because performance can be disappointing. The active raise continues until N.N. has ~150 neurons in its hidden layer, which can be the optimal searched value - performance is 0.956 and training time is short. Later, after X of 150, very slow growth can be noticed which leads to performance of approximately 0.9614.

Conclusion

After multiple loops and tests we’ve defined desired parameters for our perceptron. The best performance achieved during the research is ~0.977 - quite good result with parameters: *learning level - 0.2, number of epochs - 4, number of neurons in the intermediate layer - 200.* For more precise results, the loop which is iterating through all the parameters should be repeated that many times, so every result will have maximum similarity with the previous one.