Documentation of Laboratory Task - Study of a Two-Layer Perceptron

Adam Szostek, Index Number 331443
Krzysztof Wyrzykowski, Index Number 331455

11 lutego 2025

1.1 Description of the Algorithm's Operation

The task of the perceptron is to solve a regression problem, which involves approximating the value of the function $f(x)$. A perceptron consists of neurons connected by weights. Neurons are grouped into layers, so that each neuron in a given layer is connected to every neuron in the next layer. A perceptron consists of one input layer, one output layer, and at least one hidden layer. The input layer contains as many neurons as the number of arguments the approximated function takes. The number of neurons in the output layer corresponds to the number of function outputs. Hidden layers can contain any number of neurons, and their number and arrangement affect the perceptron's performance. The model's output is determined by the formula:

$$\hat{f}(x) = w^\top \phi(x) + b$$

where:

- w – weight vector (model parameters),
- $\phi(x)$ – output feature vector from the hidden layer (data transformation),
- b – bias.

The algorithm's goal is to minimize the error between the actual function values $f(x)$ and the predicted values $\hat{f}(x)$ by appropriately updating the weights w and bias b. For the regression task, the cost function is the Mean Squared Error (MSE), expressed by the formula:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{f}(x_i) - y_i \right)^2$$

where:

- N – number of samples in the training set,
- y_i – actual function value for sample x_i,
- $\hat{f}(x_i)$ – predicted value by the model for sample x_i.

The perceptron's objective is to minimize this cost function through iterative adjustment of the weights w and bias b.

A key aspect of the perceptron's operation is the addition of an activation function in the hidden layers, as this introduces non-linearity. The hidden layer's activation output a_{hidden}, considering the activation function, is calculated as follows:

$$z_{\text{hidden}} = W_{\text{hidden}} \cdot x + b_{\text{hidden}}$$

$$a_{\text{hidden}} = \phi(z_{\text{hidden}})$$

where:

- W_{hidden} – weights of the hidden layer,
- b_{hidden} – biases of the hidden layer,
- $\phi(\cdot)$ – activation function (e.g., ReLU, tanh, sigmoid).

The network's output is computed as a linear combination of the hidden layer's activations:

$$\hat{f}(x) = W_{\text{output}} \cdot a_{\text{hidden}} + b_{\text{output}}$$

The perceptron's training process involves minimizing the cost function using the backpropagation algorithm and gradient descent. Weight updates follow the rule:

$$W \leftarrow W - \eta \cdot \frac{\partial \text{MSE}}{\partial W}$$

$$b \leftarrow b - \eta \cdot \frac{\partial \text{MSE}}{\partial b}$$

where η is the learning rate.

Gradient computation occurs in two steps:

1. **Forward Pass:** Calculate the output values for the hidden and output layers.

2. **Backward Pass:** Determine the gradients of the cost function with respect to the weights and biases.

For training the perceptron, a modified gradient descent method called stochastic gradient descent (SGD) was used, which computes gradients on random samples from the training set of a fixed size. This approach significantly improves the time taken by the model training process and, in some cases, also enhances the quality of the obtained results.

Additionally, the implemented perceptron was augmented with the ability to train using the evolutionary strategy ES (1+1) with the 1/5 success rule. To correctly implement the algorithm, the model's weight vectors are flattened into a single matrix along with the biases. This way, the strategy can properly explore the solution space in search of the perceptron's optimal parameters.

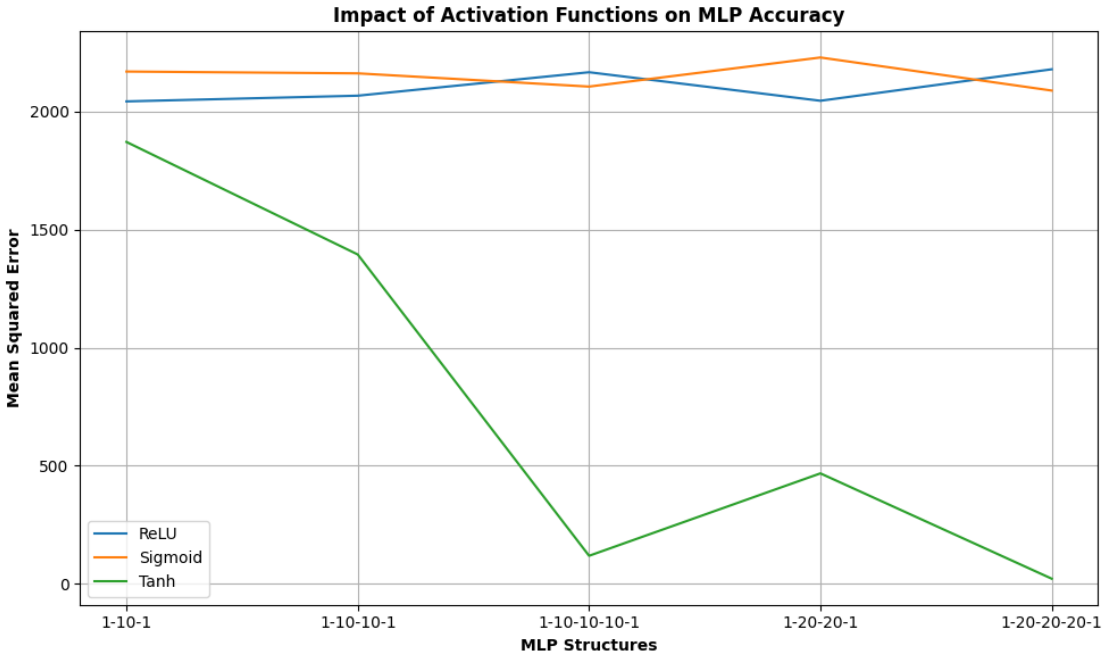
1.2 Planned Numerical Experiments

To test the implemented perceptron in a regression problem, data from the following function was used:

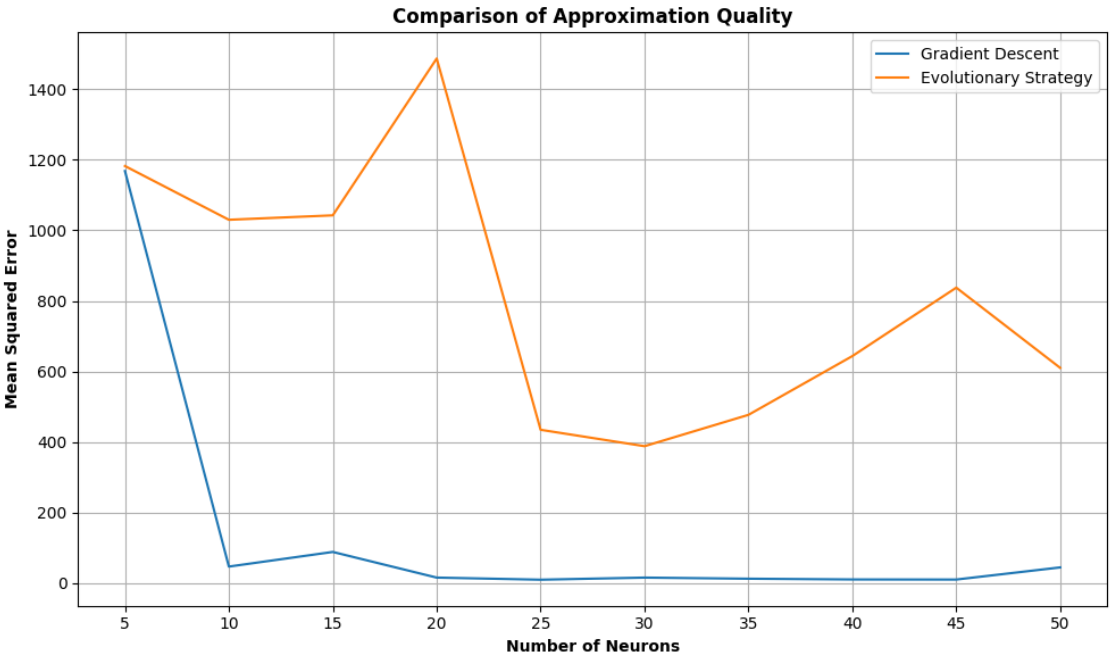
$$f\colon [-10, 10] \to R, \quad f(x) = x^2 \cdot \sin(x) + 100 \cdot \sin(x) \cdot \cos(x)$$

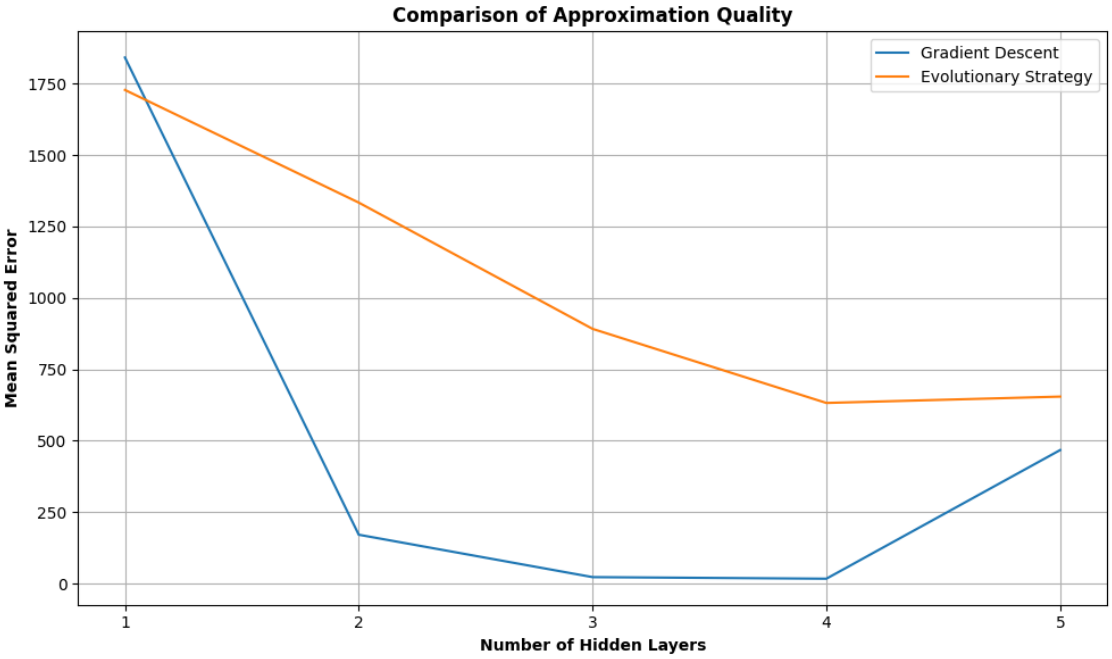
The perceptron's task is to approximate the function based on data samples. During the experiments, the influence of the number of hidden layers and the number of neurons within them on the perceptron's prediction quality was examined. To analyze the results, plots were created showing test points, perceptron predictions, and a curve representing the approximated function. The impact of the two implemented training methods (ES and SGD) on the test set error was also investigated depending on different numbers of neurons in three hidden layers and the number of hidden layers themselves, followed by a plot of the obtained results. Additionally, to select the best activation function, plots were created showing the dependence of the average error on nonlinearity for different perceptron structures. Each plot presents averaged data from five training trials.

1.3 Obtained Results

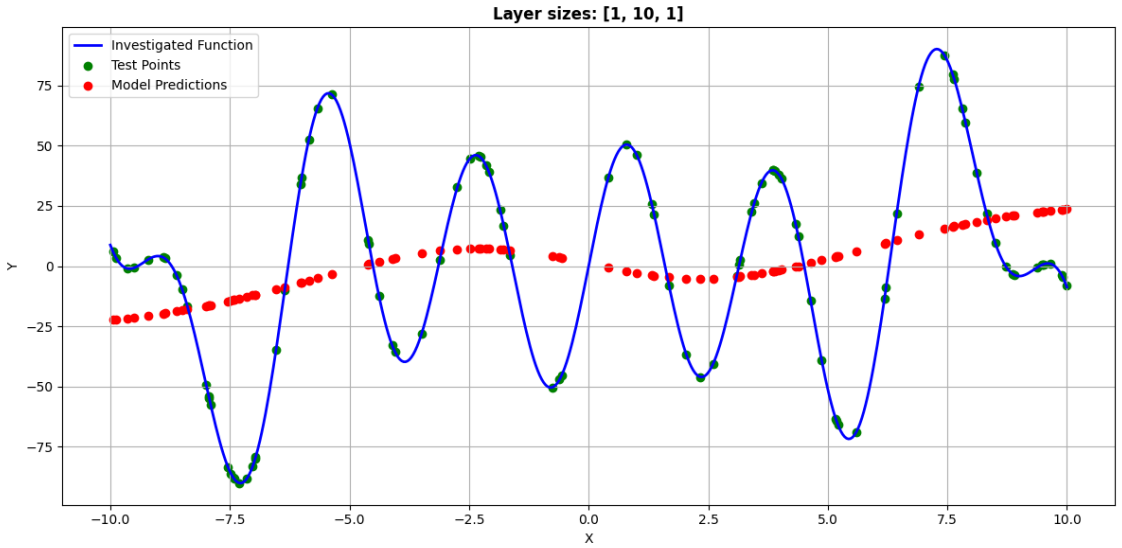


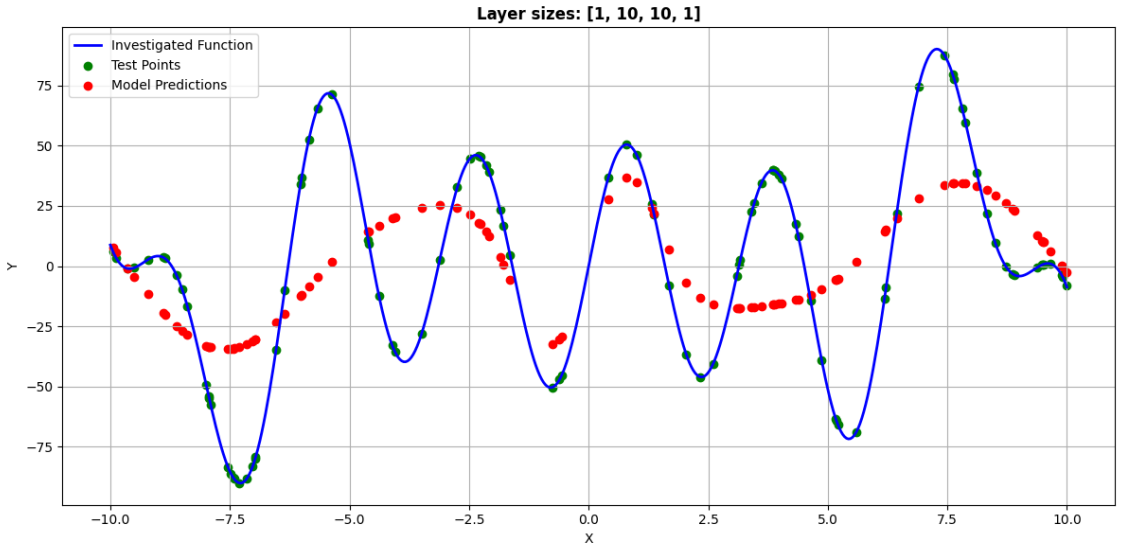
In the plot, it is clearly visible that the hyperbolic tangent performs the best, hence it was used as the activation function in further studies.

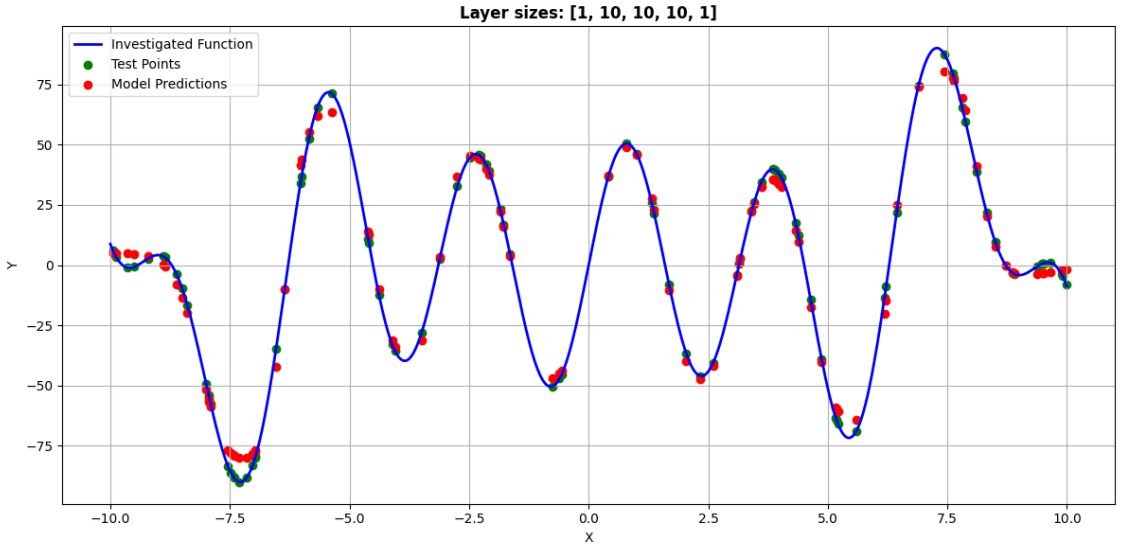

Comparison of Approximation Quality
Gradient Descent
Evolutionary Strategy
Mean Squared Error
Number of Neurons
0
200
400
600
800
1000
1200
1400
5
10
15
20
25
30
35
40
45
50

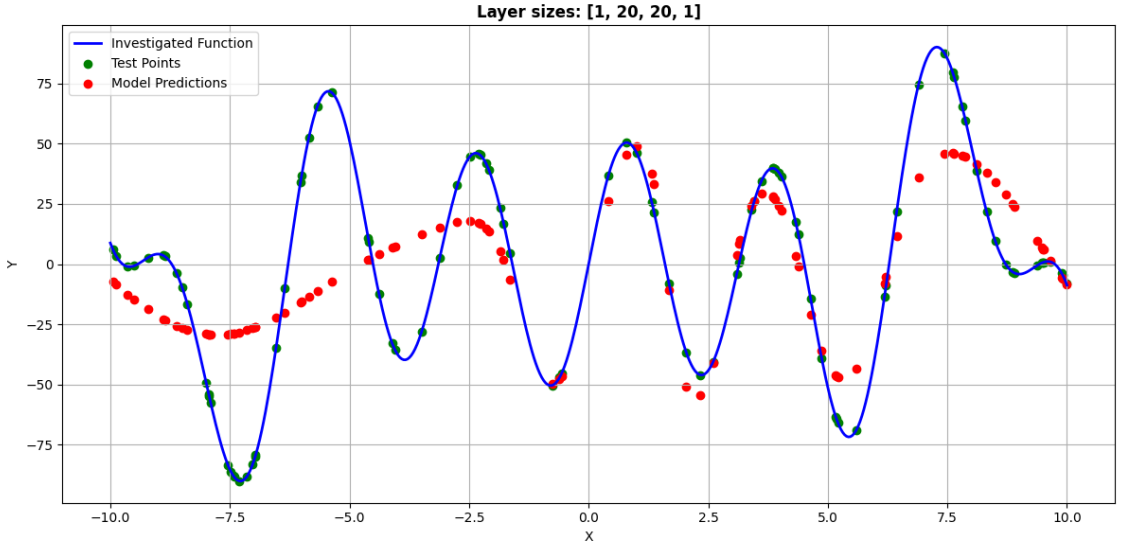

Comparison of Approximation Quality
Gradient Descent
Evolutionary Strategy
Mean Squared Error
Number of Hidden Layers
0
250
500
750
1000
1250
1500
1750
1
2
3
4
5

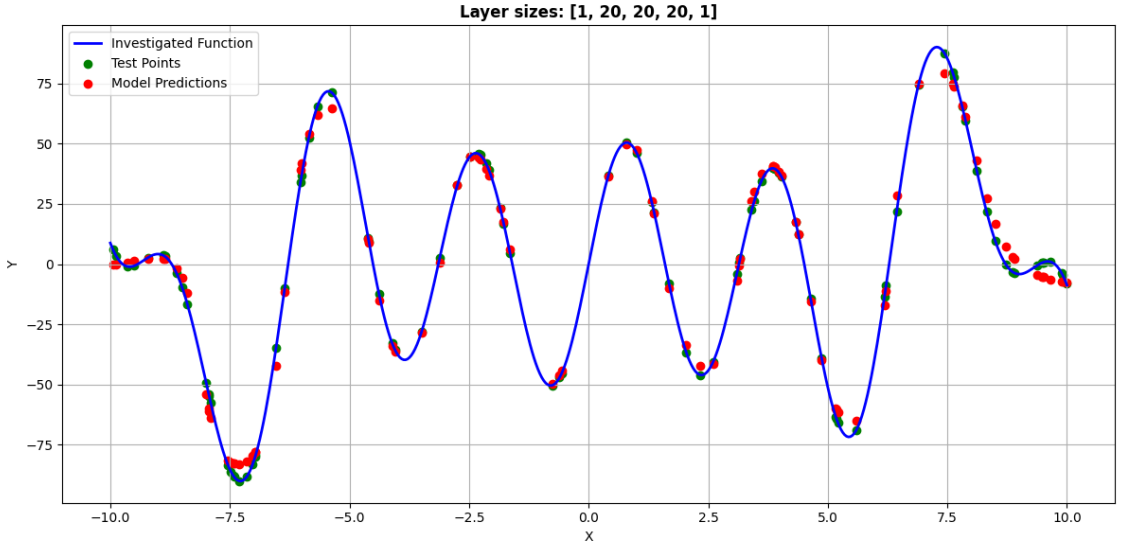
In the plot showing the relationship between mean squared error and the number of neurons in hidden layers as well as the number of layers themselves, it is evident that the gradient method provides much more stable results than the evolutionary strategy ES. At the same time, the obtained error is also smaller for SGD. Based on the study results, it was decided that, in this specific case, testing a larger number of neurons in hidden layers than 20 and more than three hidden layers is unnecessary, as it does not significantly improve the quality of the obtained approximation. In further studies, the model was trained using the gradient method (SGD).






Layer sizes: [1, 10, 10, 10, 1]
Investigated Function
Test Points
Model Predictions
Y
X


Layer sizes: [1, 20, 20, 1]
Investigated Function
Test Points
Model Predictions
Y
X


Layer sizes: [1, 20, 20, 20, 1]
Investigated Function
Test Points
Model Predictions
Y
X

In the plots presenting the quality of predictions depending on the applied structure of hidden layers, it can be observed that the most accurate approximation is achieved with three hidden layers.

1.4 Conclusions from the Obtained Results

Based on the plots comparing the training methods based on the number of neurons in hidden layers and the number of layers themselves, it can be clearly stated that the gradient training method achieves better results (smaller error) than the ES strategy. Additionally, the evolutionary strategy behaves much less stably than SGD. This is because the algorithm relies heavily on randomness, unlike the gradient method, which in most cases ensures the finding of the sought minimum during optimization. Observing the curve representing the error obtained from the model trained by SGD alongside the plot comparing the number of neurons, it can be concluded that for the given regression problem, the number of neurons affects the model's quality up to a certain point, after which no observable change in the approximation error occurs. This point occurs precisely at 20 neurons in the hidden layer, and for each subsequent value, as mentioned earlier, no difference in error can be observed. Similarly, with the number of layers, choosing a model with more than three layers also does not reduce the error.

Analyzing the plots presenting the model's approximations for different combinations of neurons and layers, it can be concluded that both the number of hidden layers and the number of neurons influence the model's quality. However, based on earlier studies (plot of error versus the number of neurons and layers), it is known that these parameters matter only up to a certain point—beyond which increasing them does not improve the results. For one hidden layer, the model is unable to accurately approximate the studied function, as seen from the shape of the curve representing the perceptron's predictions—it is only slightly similar to the approximated function. After increasing the number of layers to two, there is a significant improvement in approximation quality, although it is still imperfect. Comparing the results of the model with three hidden layers (20 neurons each) with the model with two hidden layers (20 neurons each), it is clearly visible that the model with three layers achieves a more accurate approximation. In this case, it suggests that the number of layers has a greater impact on the model's quality than the number of neurons. The most accurate model is the perceptron with three hidden layers of twenty neurons each, which is slightly more precise in predictions than the previously described one.

In summary, the conducted studies have established that the gradient method (SGD) is significantly more stable and effective than the evolutionary algorithm. Additionally, the influence of parameters defining the model's structure on the approximation quality was examined. For the studied regression problem, it was determined that the best results are achieved by a model with three hidden layers, each containing twenty neurons. The above results clearly indicate that selecting the parameters of a multilayer perceptron has a significant impact on its results, but different problems require different approaches and parameter values.