Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Máster en Automática y Robótica

Applied Artificial Intelligence

Assignment 5.2: MLP for Function Generalization

Josep María Barberá Civera (17048)

April 16, 2024

ASSIGNMENT 5.2

MLP for Function Generalisation

1.1 Exercise 1

Choose an adequate MLP structure and training set. Plot in the same figure the training set, the output of the MLP for the test set, and the ground truth $sin(x)$ function.

First of all it is necessary to plot the `training data` as well as the expected solution (which in this case is known: $sin(x)$ between $[0, 2\pi]$. In this way, thanks to the use of Matlab, Figure 1 has been obtained.

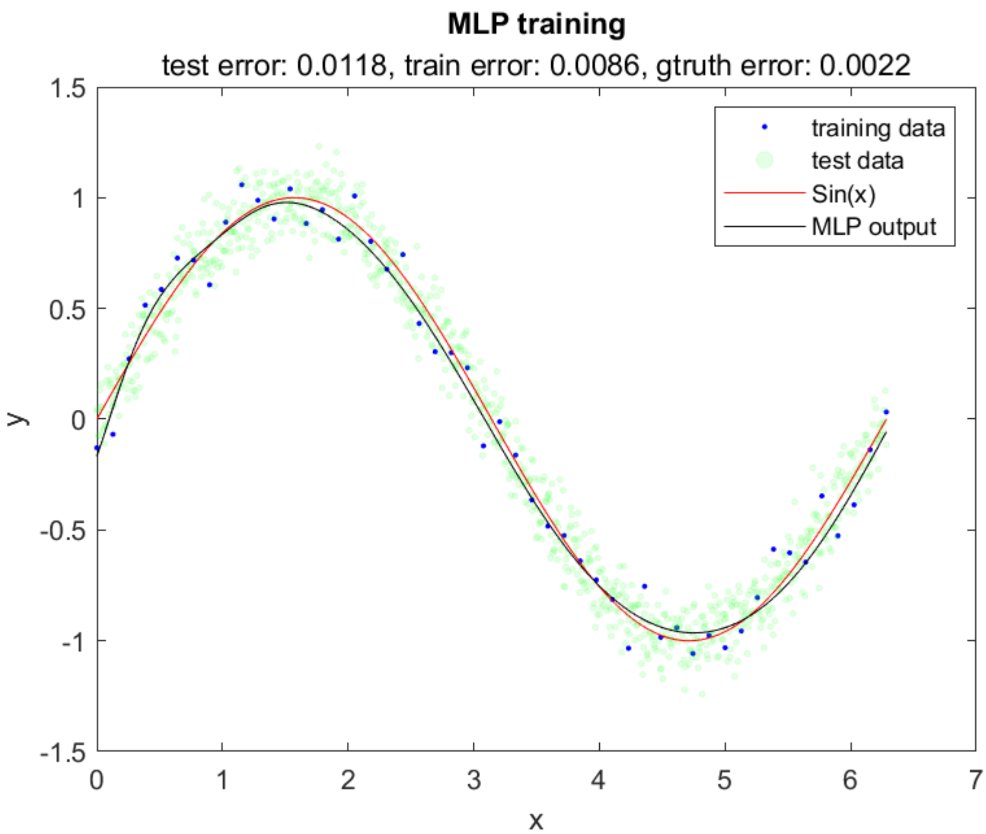

Figure 1: *Multi Layer Perceptron (MLP) training with Matlab. In blue the training data (50 values); in green the test data (1000 values); in red the "ground truth" (desired solution) and finally in black the output of the MLP trained with 4 neurons in the hidded layer. The errors are compiled in the subtitle.*

The initial structure chosen consists of four neurons in the hidden layer of the network. Further experiments will allow us to conclude which dataset and which architecture is most suitable for this problem. It is worth mentioning that Matlab offers functions that are very convenient for a fast deployment of the neural network, but they do not allow the desired flexibility. For example, in the following section, we wish to study the evolution of the error as the number of neurons increases. For this particular case, it is to be expected to see the `overfitting` that occurs when the number of neurons is excessive. However, this overfitting should be accompanied by a decreasing training error, which is not observed. We have tried to increase the number of `epochs` but this is only a limit, it is Matlab and its implementation of training that decides when to stop training depending on several other parameters not covered in the course.

Thanks to the Matlab visualisation tool, the structure is easily visible, resulting in the image shown in Figure 2.

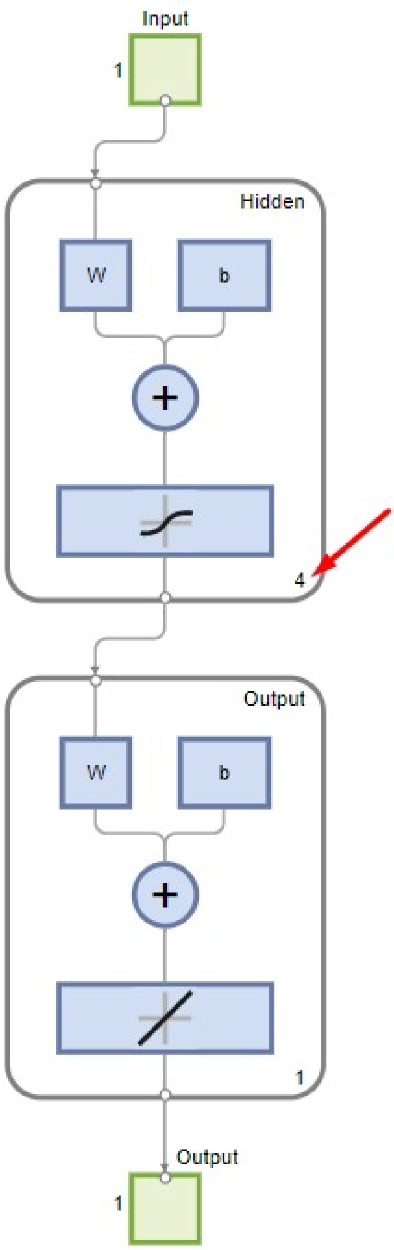

Figure 2: *Multi Layer Perceptron (MLP) structure. The number of neurons chosen is marked in red with an arrow.*

1.2 Exercise 2

Evaluate the evolution of the train error, the test error and the ground truth error in the following cases:

This is the most experimental part and has already been advanced in the previous section. The order of the sections has been slightly varied, with **the first section being the study of training with an increasing number of neurons** (when in fact it should be the last). We believe that this variation in order is not a problem for the study in question.

1.2.1 Part 1

Changing the net structure: number of neurons.

Figure 3 shows the graph with the results of the experiment. In particular, the three errors can be seen: training error, test error and ground truth error. And this for an increasing number of neurons (from 1 to 50).

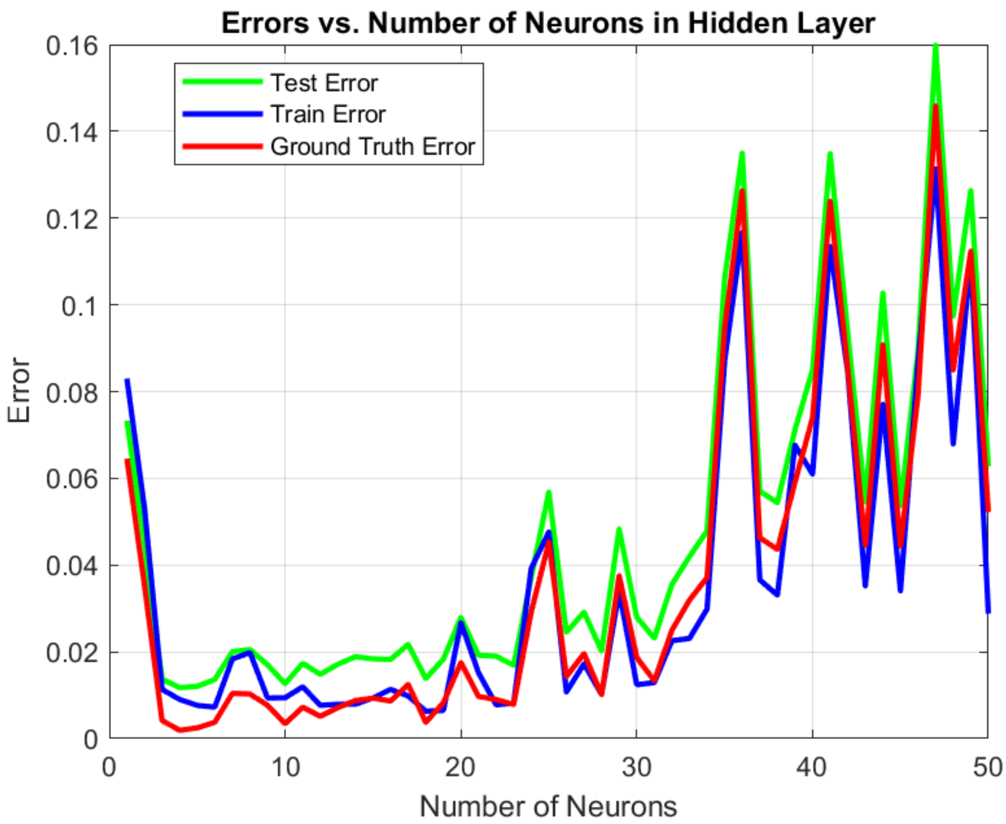

Figure 3: *Mean error over five tries evolution with increasing number of neurons in the hidden layer of the network.*

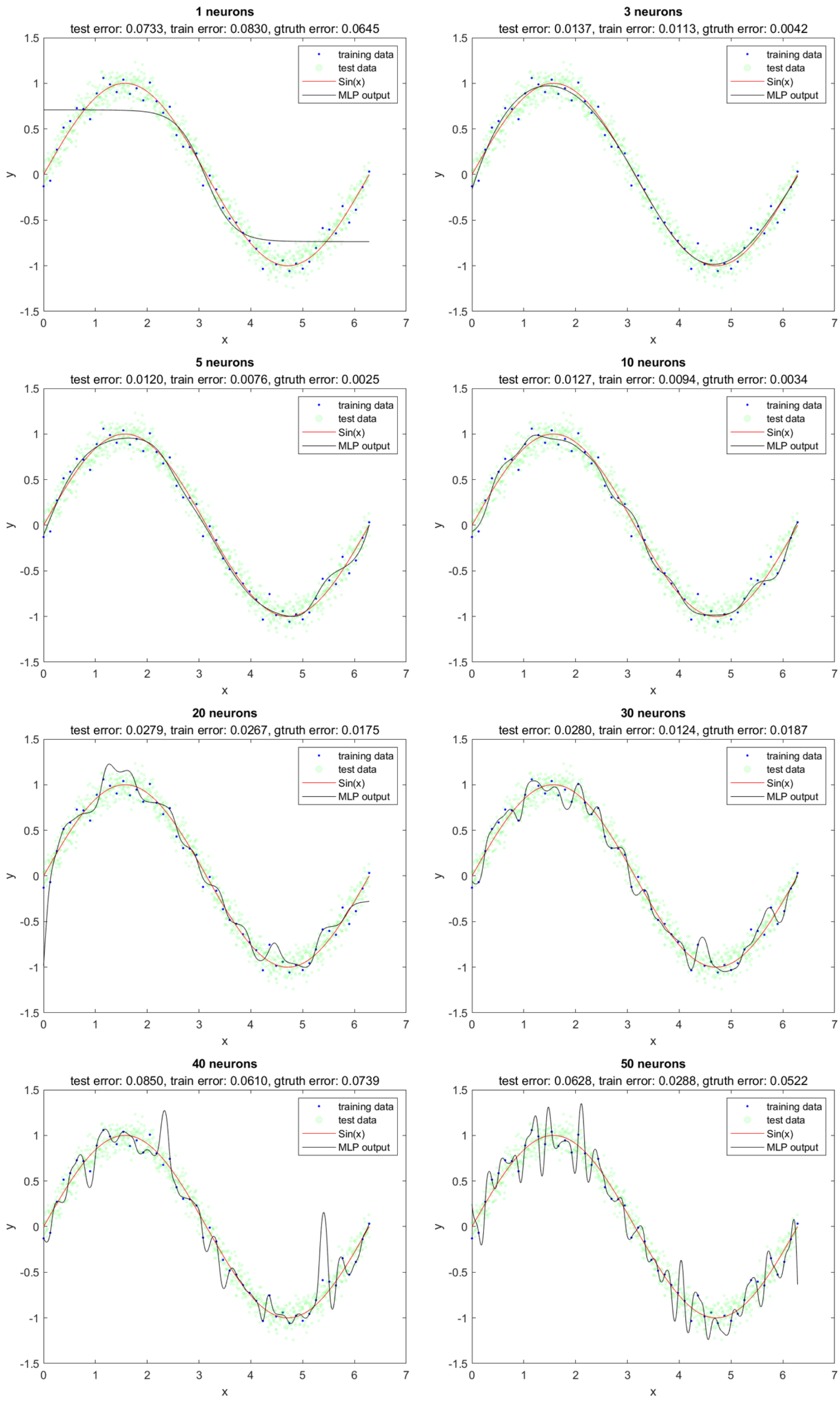

Figure 4: *Network outputs for test values according to the number of neurons with which they have been trained (from 1 to 50).*

In general, the behaviour is not as expected. One would expect that as the number of neurons increases, the training error should decrease to zero. Attempts have been made to remedy this by increasing the number of `epochs` but this is simply a maximum, not a minimum, so it is really Matlab that given several criteria (not just one) decides to terminate the training. This is done to optimise the training, but in our case it does not allow us to reproduce what we want.

On the other hand, it can be seen that increasing the number of neurons is not always good. In fact, **in this case, with 4 or 5 neurons the best results are achieved**, and increasing or decreasing the number of neurons is detrimental. We believe that this study may be necessary in general for each problem, in order to obtain an intuition about the size of the network needed to capture the behaviour of the data.

For this part, the following hyperparameters have been selected:

- **Training function**: `trainlm` (Levenberg-Marquardt)
- **Desired number of epochs**: 100
- **Performance function**: Mean Squared Error (MSE)
- **Performance target value:** 0
- μ **(target value):** $1e+10$
- **Gradient (target value):** $1e-07$

1.2.2 Part 2

Changing the training data: number of samples.

In a similar way to the previous section, we iterate the training of the MLP but this time increasing the number of training data. We start with 5 data equispaced between 0 and 2π until we end up with 100 data. The training was repeated 5 times and the average of the five iterations was calculated.

A similar behaviour to that expected is observed in Figure 5 in which the test error instead has a slight increase until it stagnates around 0.01 while the error of the desired (or real - ground truth) data decreases until it is almost zero. Contrary to what is desired, the training error does not start with a zero value and we do not know why. It is true that as the number of data increases, the error decreases rapidly and shows the described behaviour of a slight increase.

- **Training function**: `trainlm` (Levenberg-Marquardt)
- **Desired number of epochs**: 50
- **Performance function**: Mean Squared Error (MSE)

- **Performance target value:** 0
- **Max fail (Maximum Number of Validation Increases):** 50
- **Gradient (min target value):** $1e-10$

Figure 5: *Mean error over five tries evolution with increasing number of samples in the training set.*

Figure 6 shows different phases of training for an increasing number of samples. Interestingly, with only five samples the neural network is not able to memorise the data (which is very surprising and anomalous), but with 25 samples it behaves as expected.

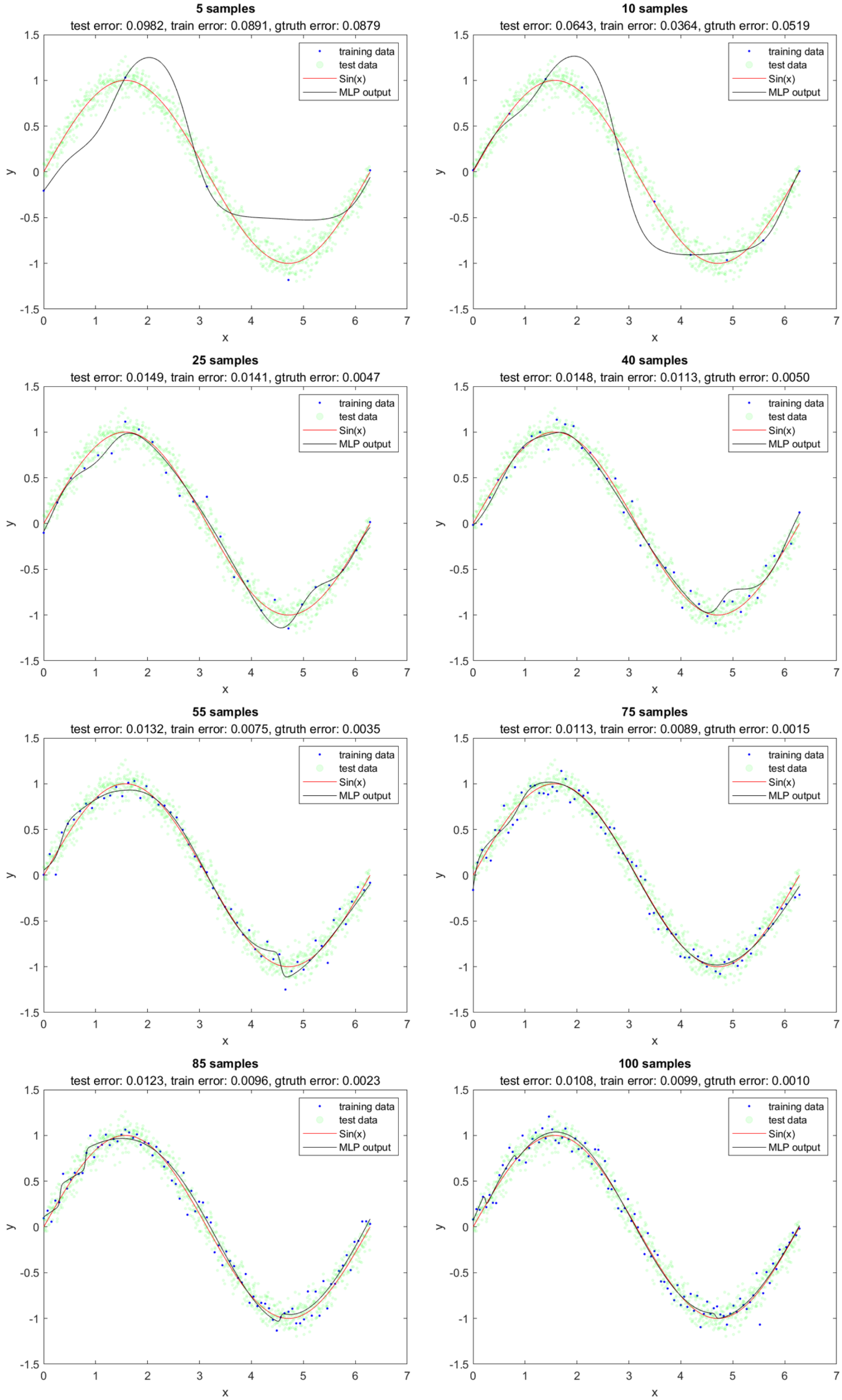

Figure 6: *Network outputs for test values according to the number of samples used in the training.*

1.2.3 Part 3

Changing the training parameters: initial values, number of epochs, optimisation algorithm, etc.

Finally, we proceed to study how the initialisation of the training affects it. In other words, in the optimisation process, the initial values condition to a large extent the obtaining of global or local optima.

To test this, we proceeded to create a network with the same structure as in the previous section and repeat the training and validation process 100 times. The result can be seen in Figure 7.

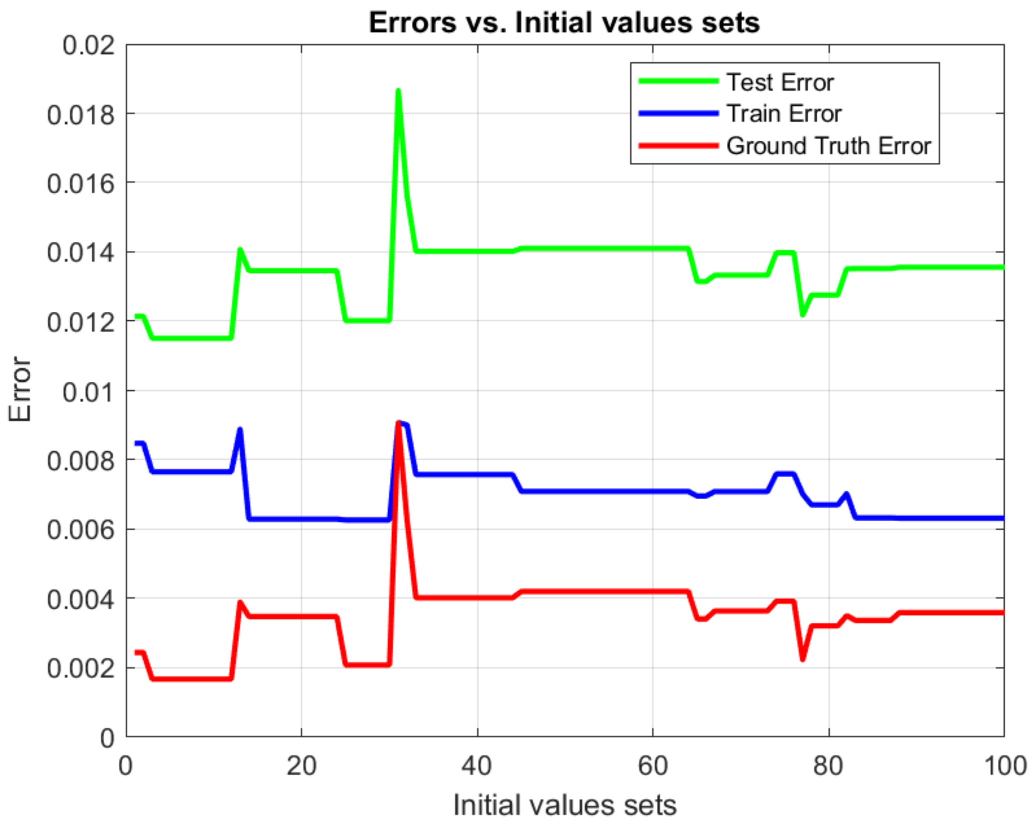

Figure 7: *Error evolution under different initial values for the same MLP and same training and test data.*

We have repeated the experiment, but this time including in the iterations the creation of the network itself. The results are shown in Figure 8. It can be seen that the errors are higher and that there is a greater variability in each iteration. Except for the increase in computation time, we do not know the underlying reason, which could be, among others, that the creation of the network involves the randomness of many other parameters that for Figure 7 are fixed and only the initial values are modified.

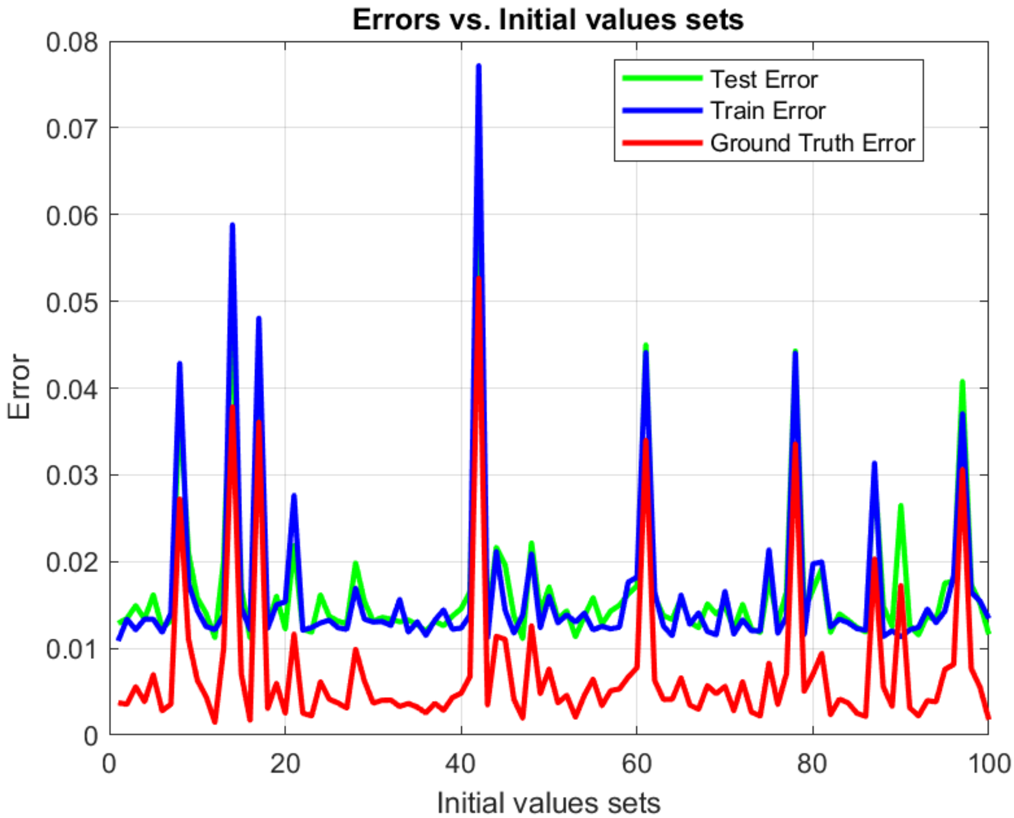

Figure 8: *Network outputs for test values according to the number of samples used in the training.*

1.3 Relevant Code

The code prepared for this assignment is shown in the next pages in a wider style.

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           Master in Robotics
%                    Applied Artificial Intelligence
%
% Assinment 5.2:  Function Generalization - MLP
% Student: Josep Barbera Civera
% ID: 17048
% Date: 14/04/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 1.- Choose an adequate MLP structure and training set.
%   Plot in the same figure the training set, the output of the MLP
%   for the test set, and the ground truth sin function.
%
% Evaluate the evolution of the train error, the test error and
%   the ground truth error in the following cases:
%
% 2.- Changing the training parameters:
%   initial values, (# of epochs, optimization algorithm)
%
% 3.- Changing the training data:
%   # of samples (order of samples)
%
% 4.- Changing the net structure:
%   # of neurons

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1. Training set, MLP structure and initial plots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
generate_data = false;              %%%%% <- change here as needed! %%%%%
plot_specific_MLP_training = false; %%%%% <- change here as needed! %%%%%
if(generate_data)
    %% Training data (N_train, x_train, y_train)
    N_train = 50;
    x_train = linspace(0, 2*pi, N_train);
    for i = 1:N_train
         y_train(i) = sin(x_train(i)) + normrnd(0, 0.1);
    end

    %% Test data (N_test, x_test, y_test, y_gtruth)
    N_test = 1000;
    x_test = linspace(0, 2*pi, N_test);
    y_gtruth = sin(x_test); % ground truth
    for i = 1:N_test
         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
    end
end

if(plot_specific_MLP_training)
```

```
    %% MLP structure
    trainFcn = 'trainlm';
    hiddenSizes = 4;
    net = feedforwardnet (hiddenSizes, trainFcn);
    %% MLP training
    [net, tr] = train(net, x_train, y_train);
    %% MLP testing
    % Test error
    MLP_test = net(x_test);
    perf_test = perform(net,y_test, MLP_test);
    % Train error
    MLP_train = net(x_train);
    perf_train = perform(net,y_train,MLP_train );
    % Ground truth error
    perf_gtruth = perform(net, y_gtruth, MLP_test);


    %% Plot data
    % Plot training data
    plot(x_train, y_train, 'b.', 'DisplayName', "training data");
    hold on;
    % Plot test data
    s1 = scatter(x_test, y_test, 5,'DisplayName', "test data", ...
                'MarkerFaceColor','g','MarkerEdgeColor','g');
    alpha(s1,.1)
    hold on;
    % Plot truth data
    plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
    hold on;
    % Plot MLP output for test data
    plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
    title("MLP training");
    my_subtitle = sprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f", ...
                        perf_test, perf_train, perf_gtruth);
    subtitle(my_subtitle);
    xlabel("x");
    ylabel("y");
    legend("training data", "test data", "Sin(x)", "MLP output");
    figure_name = sprintf("test_error_%.3g_train_error_%.3g_gtruth_error_%.3g.png", ...
                        perf_test, perf_train, perf_gtruth);
    saveas(gcf, figure_name);
    hold off;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2. Net structure
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
folder_name = "net_structure";
if ~exist(folder_name, 'dir')
    mkdir(folder_name);
end
```

```

generate_data_2 = false; %%%%% <- change here as needed! %%%%%

if(generate_data_2)
    %% Training data (N_train, x_train, y_train)
    N_train = 50;
    x_train = linspace(0, 2*pi, N_train);
    for i = 1:N_train
         y_train(i) = sin(x_train(i)) + normrnd(0, 0.1);
    end

    %% Test data (N_test, x_test, y_test, y_gtruth)
    N_test = 1000;
    x_test = linspace(0, 2*pi, N_test);
    y_gtruth = sin(x_test); % ground truth
    for i = 1:N_test
         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
    end
end

test_net_structure = false; %%%%% <- change here as needed! %%%%%
neurons = 1:50;
if (test_net_structure)
    for i = 1:numel(neurons)
        fprintf("Training with %d neurons in the hidden layer\n", neurons(i));
        trainFcn = 'trainlm';
        net = feedforwardnet (neurons(i), trainFcn);
        net.trainParam.epochs = 1000;

        for j = 1:5
            [net, tr] = train(net, x_train, y_train);
            % Test error
            MLP_test = net(x_test);
            error_t(j) = mean((y_test-MLP_test).^2);
            % per_error_t = perform(net, y_test, MLP_test);
            % Train error
            MLP_train = net(x_train);
            error_tr(j) = mean((y_train-MLP_train).^2);
            % per_error_tr = perform(net, y_train, MLP_train);
            % Ground truth error
            error_gt(j) = mean((y_gtruth-MLP_test).^2);
            % per_error_gt = perform(net, y_gtruth, MLP_test);
            % fprintf(['Error_t: %f, Percent Error_t: %f, Error_tr: %f, ' ...
            %     'Percent Error_tr: %f, Error_gt: %f, Percent Error_gt: %f\n'], ...
            %     error_t(j), per_error_t, error_tr(j), per_error_tr, ...
            %     error_gt(j), per_error_gt);
        end
        test_error(i) = mean(error_t);
```

```
        train_error(i) = mean(error_tr);
        gtruth_error(i) = mean(error_gt);
        % Plot training data
        plot(x_train, y_train, 'b.', 'DisplayName', "training data");
        hold on;
        % Plot test data
        s1 = scatter(x_test, y_test, 5,'DisplayName', "test data", ...
                     'MarkerFaceColor','g','MarkerEdgeColor','g');
        alpha(s1,.1)
        hold on;
        % Plot truth data
        plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
        hold on;
        % Plot MLP output for test data
        plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
        my_title = sprintf("%d neurons", neurons(i));
        title(my_title);
        my_subtitle = sprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f", ...
                              test_error(i), train_error(i), gtruth_error(i));
        subtitle(my_subtitle);
        xlabel("x");
        ylabel("y");
        legend("training data", "test data", "Sin(x)", "MLP output");
        hold off;
        figure_name = sprintf("/%d_neurons_in_MLP.png", neurons(i));
        saveas(gcf, strcat(folder_name, figure_name));
        fprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f\n", ...
                test_error(i), train_error(i), gtruth_error(i));
        disp("************************************");
    end
    % Plot errors
    plot(neurons, test_error, 'g-', 'LineWidth', 2, 'DisplayName', 'Test Error');
    hold on;
    plot(neurons, train_error, 'b-', 'LineWidth', 2, 'DisplayName', 'Train Error');
    plot(neurons, gtruth_error, 'r-', 'LineWidth', 2, 'DisplayName', 'Ground Truth Error');

    % Title and labels
    title('Errors vs. Number of Neurons in Hidden Layer');
    xlabel('Number of Neurons');
    ylabel('Error');
    legend('Location', 'best');

    % Adjust figure
    grid on;
    figure_name = sprintf("/error_plot.png");
    saveas(gcf, strcat(folder_name, figure_name));
```

```
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3. Number of samples
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
folder_name = "samples_number";
if ~exist(folder_name, 'dir')
    mkdir(folder_name);
end

test_samples_number = false; %%%%% <- change here as needed! %%%%%
if (test_samples_number)

    %% Test data (N_test, x_test, y_test, y_gtruth) (only once)
    N_test = 1000;
    x_test = linspace(0, 2*pi, N_test);
    y_gtruth = sin(x_test); % ground truth
    for i = 1:N_test
         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
    end

    %% Net structure
    trainFcn = 'trainlm';
    net = feedforwardnet (5, trainFcn);
    net.trainParam.epochs = 50;
    net.trainParam.min_grad = 1e-10;
    net.trainParam.max_fail = 50;
    net.trainParam.goal = 0;
    net.trainParam.showWindow = false;

    data_number = 5:5:100;
    for i = 1:numel(data_number)
        %% Training data (N_train, x_train, y_train)
        N_train = data_number(i);
        disp(N_train)
        x_train = linspace(0, 2*pi, N_train);
        for k = 1:N_train
             y_train(k) = sin(x_train(k)) + normrnd(0, 0.1);
        end

        fprintf("Training with %d samples\n", data_number(i));

        for j = 1:5
            [net, tr] = train(net, x_train, y_train);
            % Test error
            MLP_test = net(x_test);
            error_t(j) = mean((y_test - MLP_test).^2);
            % Train error
            MLP_train = net(x_train);
            error_tr(j) = mean((y_train - MLP_train).^2);
            % Ground truth error
```

```
            error_gt(j) = mean((y_gtruth - MLP_test).^2);
        end

        test_error(i) = mean(error_t);
        train_error(i) = mean(error_tr);
        gtruth_error(i) = mean(error_gt);
        % Plot training data
        plot(x_train, y_train, 'b.', 'DisplayName', "training data");
        hold on;
        % Plot test data
        s1 = scatter(x_test, y_test, 5,'DisplayName', "test data", ...
                     'MarkerFaceColor','g','MarkerEdgeColor','g');
        alpha(s1,.1)
        hold on;
        % Plot truth data
        plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
        hold on;
        % Plot MLP output for test data
        plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
        my_title = sprintf("%d samples", data_number(i));
        title(my_title);
        my_subtitle = sprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f", ...
                              test_error(i), train_error(i), gtruth_error(i));
        subtitle(my_subtitle);
        xlabel("x");
        ylabel("y");
        legend("training data", "test data", "Sin(x)", "MLP output");
        hold off;
        figure_name = sprintf("/%d_samples_in_MLP.png", data_number(i));
        saveas(gcf, strcat(folder_name, figure_name));
        fprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f\n", ...
                test_error(i), train_error(i), gtruth_error(i));
        disp("***********************************");
    end
    % Plot errors
    plot(data_number, test_error, 'g-', 'LineWidth', 2, 'DisplayName', 'Test Error');
    hold on;
    plot(data_number, train_error, 'b-', 'LineWidth', 2, 'DisplayName', 'Train Error');
    plot(data_number, gtruth_error, 'r-', 'LineWidth', 2, 'DisplayName', 'Ground Truth Error');

    % Title and labels
    title('Errors vs. Number of Samples in the Training Set');
    xlabel('Number of Training Samples');
    ylabel('Error');
    legend('Location', 'best');

```

```
    % Adjust figure
    grid on;
    figure_name = sprintf("/error_plot.png");
    saveas(gcf, strcat(folder_name, figure_name));
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4. Initial Values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
folder_name = "initial_values";
if ~exist(folder_name, 'dir')
    mkdir(folder_name);
end

test_initial_values = true; %%%%% <- change here as needed! %%%%%
if (test_initial_values)
    %% Training data (N_train, x_train, y_train)
    N_train = 50;
    disp(N_train)
    x_train = linspace(0, 2*pi, N_train);
    for i = 1:N_train
         y_train(i) = sin(x_train(i)) + normrnd(0, 0.1);
    end
    %% Test data (N_test, x_test, y_test, y_gtruth) (only once)
    N_test = 1000;
    x_test = linspace(0, 2*pi, N_test);
    y_gtruth = sin(x_test); % ground truth
    for i = 1:N_test
         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
    end

    %% Net structure
    trainFcn = 'trainlm';
    net = feedforwardnet (5, trainFcn);
    net.trainParam.epochs = 50;
    net.trainParam.min_grad = 1e-10;
    net.trainParam.max_fail = 50;
    net.trainParam.goal = 0;
    net.trainParam.showWindow = false;
    initial_values = 1:100;
    for i = 1:numel(initial_values)
        fprintf("Training in the %d iteration\n", initial_values(i));
        [net, tr] = train(net, x_train, y_train);
        % Test error
        MLP_test = net(x_test);
        test_error(i) = mean((y_test - MLP_test).^2);
        % Train error
        MLP_train = net(x_train);
        train_error(i) = mean((y_train - MLP_train).^2);
        % Ground truth error
        gtruth_error(i) = mean((y_gtruth - MLP_test).^2);
```

```

        % Plot training data
        plot(x_train, y_train, 'b.', 'DisplayName', "training data");
        hold on;
        % Plot test data
        s1 = scatter(x_test, y_test, 5,'DisplayName', "test data", ...
                     'MarkerFaceColor','g','MarkerEdgeColor','g');
        alpha(s1,.1)
        hold on;
        % Plot truth data
        plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
        hold on;
        % Plot MLP output for test data
        plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
        my_title = sprintf("Training number %d", initial_values(i));
        title(my_title);
        my_subtitle = sprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f", ...
                              test_error(i), train_error(i), gtruth_error(i));
        subtitle(my_subtitle);
        xlabel("x");
        ylabel("y");
        legend("training data", "test data", "Sin(x)", "MLP output");
        hold off;
        figure_name = sprintf("/%d_initial_values_in_MLP.png", initial_values(i));
        % saveas(gcf, strcat(folder_name, figure_name));
        fprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f\n", ...
                test_error(i), train_error(i), gtruth_error(i));
        disp("***********************************");
    end
    % Plot errors
    plot(initial_values, test_error, 'g-', 'LineWidth', 2, 'DisplayName', 'Test Error');
    hold on;
    plot(initial_values, train_error, 'b-', 'LineWidth', 2, 'DisplayName', 'Train Error');
    plot(initial_values, gtruth_error, 'r-', 'LineWidth', 2, 'DisplayName', 'Ground Truth Error');
    % Title and labels
    title('Errors vs. Initial values sets');
    xlabel('Initial values sets');
    ylabel('Error');
    legend('Location', 'best');
    % Adjust figure
    grid on;
    figure_name = sprintf("/error_initial_values.png");
    saveas(gcf, strcat(folder_name, figure_name));
end
```

The need has arisen to unify the figures generated with Matlab in a programmatic way, for which the following code has been implemented in Python.

Listing 1: *Python code for image grouping*

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

figure_directory = "."
numbers_to_select = [1, 3, 5, 10, 20, 30, 40, 50]
fig, axes = plt.subplots(4, 2, figsize=(20, 10))
plt.subplots_adjust(wspace=-1, hspace=0.01)
axes = axes.flatten()

for i, num in enumerate(numbers_to_select):
    figure_name = f"{figure_directory}/{num}_neurons_in_MLP.png"
    img = mpimg.imread(figure_name)
    axes[i].imshow(img)
    axes[i].axis('off')

for j in range(len(numbers_to_select), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.savefig("selected_figures_grid.png")
plt.show()
```