



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

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.

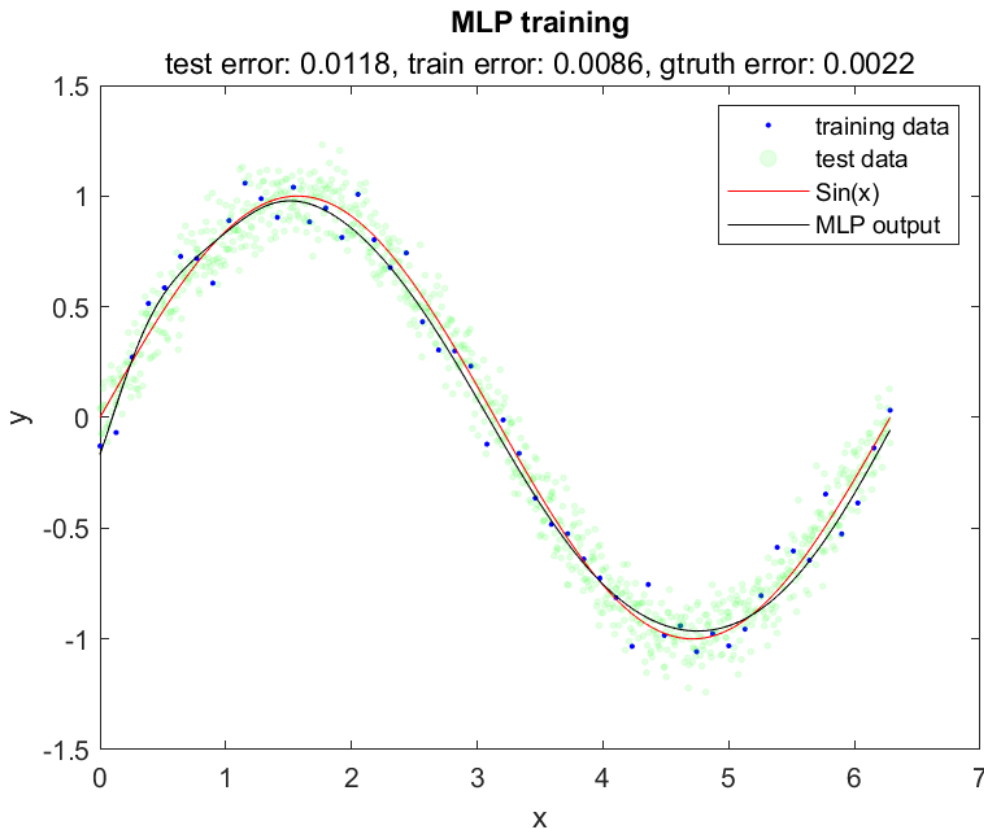


Figure 1: Multi Layer Perceptron (MLP) training with Matlab. In blue the training data (50 values); in green the test data (1000 values); in red the “ground truth” (desired solution) and finally in black the output of the MLP trained with 4 neurons in the hidden layer. The errors are compiled in the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

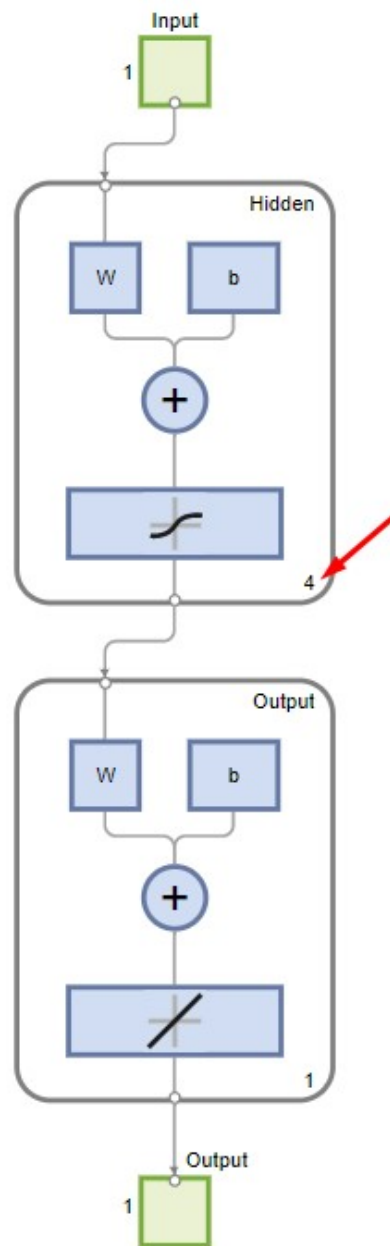


Figure 2: Multi Layer Perceptron (MLP) structure. The number of neurons chosen is marked in red with an arrow.

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).

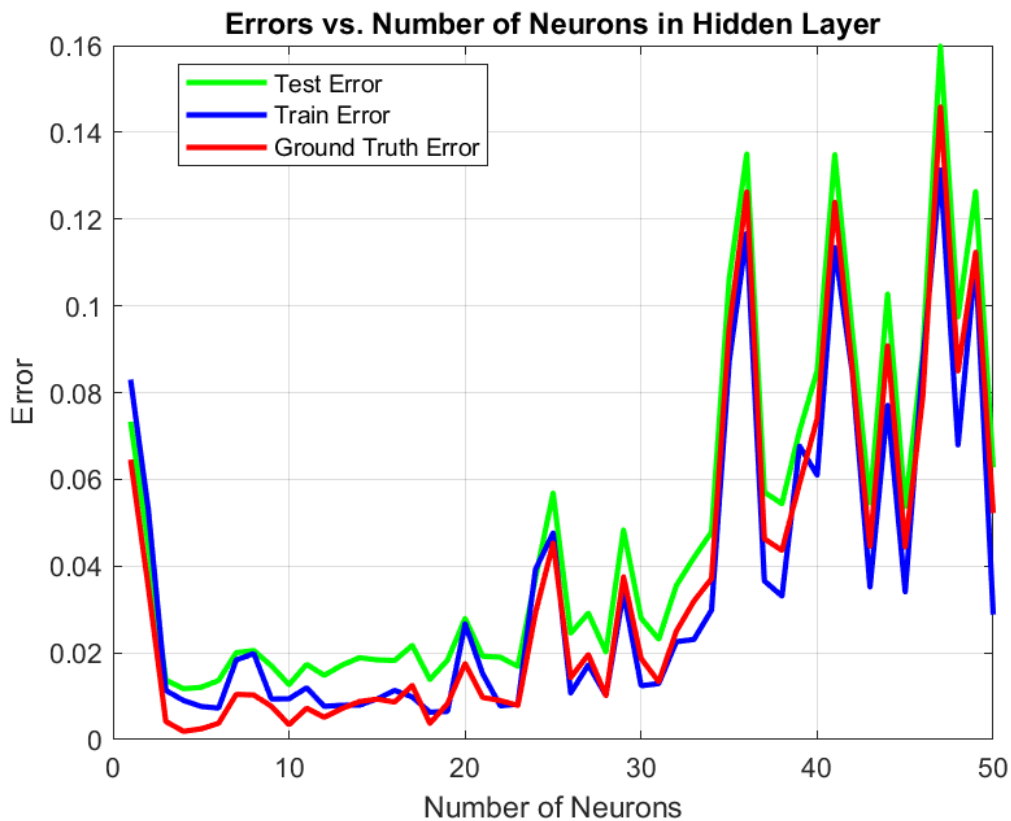


Figure 3: Mean error over five tries evolution with increasing number of neurons in the hidden layer of the network.

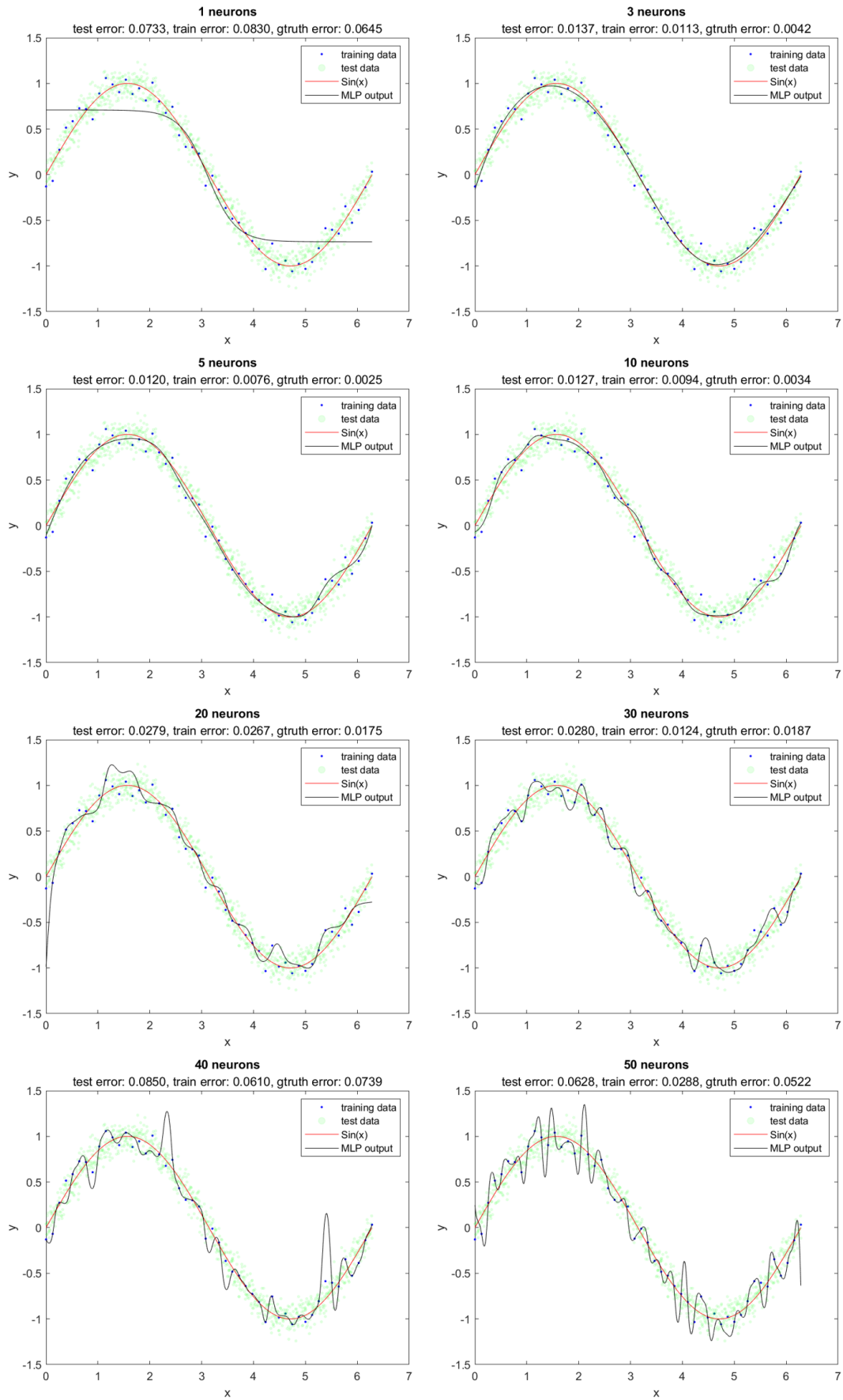


Figure 4: Network outputs for test values according to the number of neurons with which they have been trained (from 1 to 50).

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.

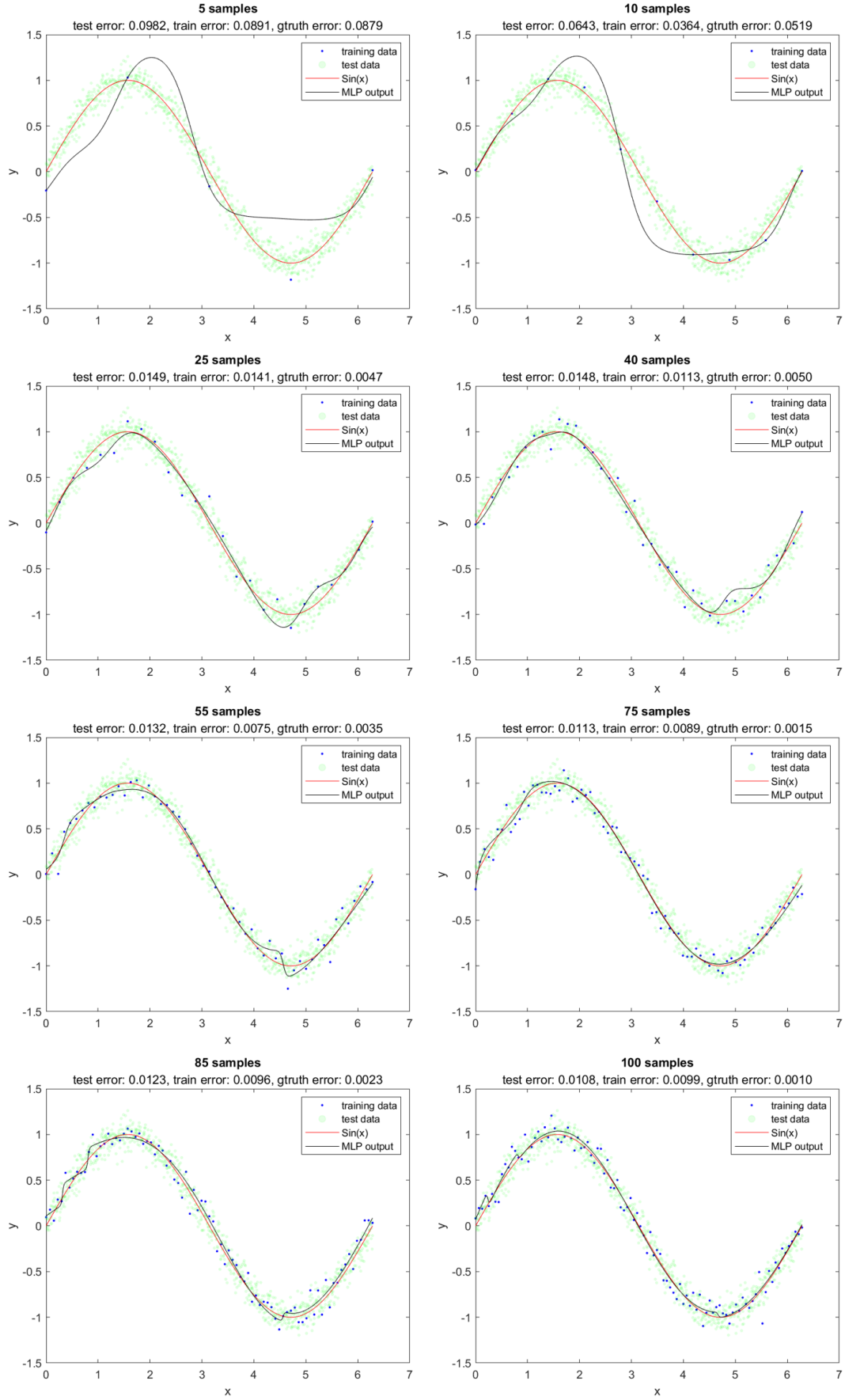


Figure 6: Network outputs for test values according to the number of samples used in the training.

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.

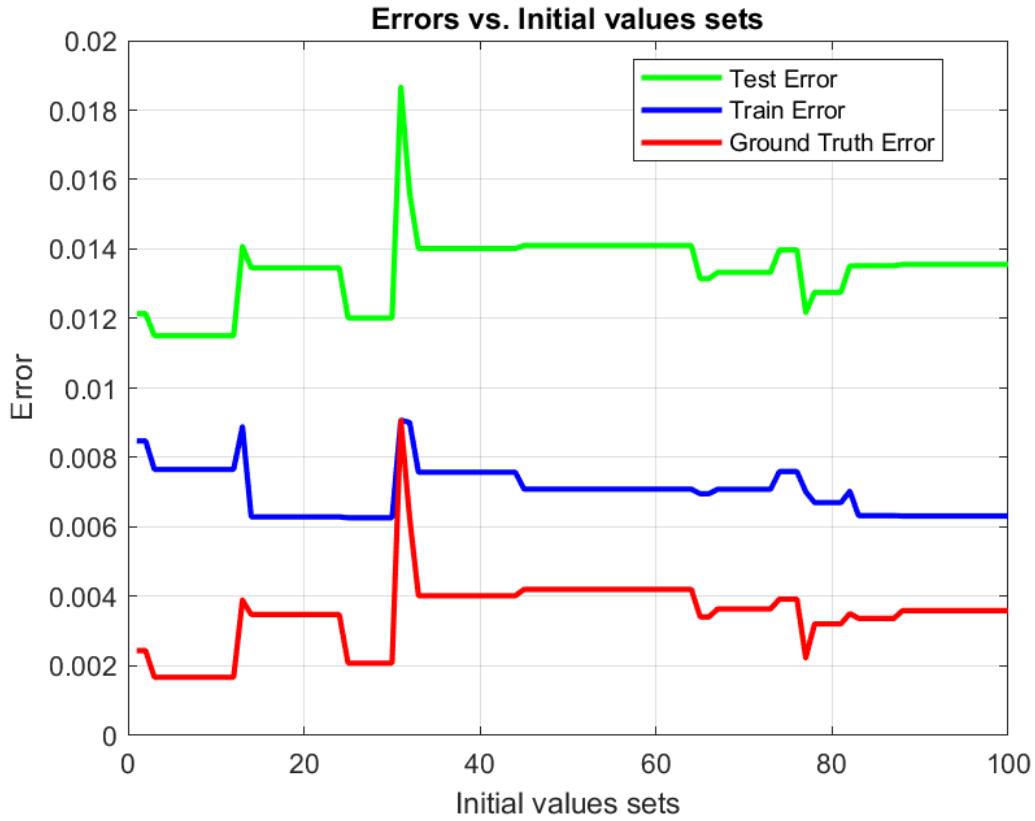


Figure 7: Error evolution under different initial values for the same MLP and same training and test data.

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.

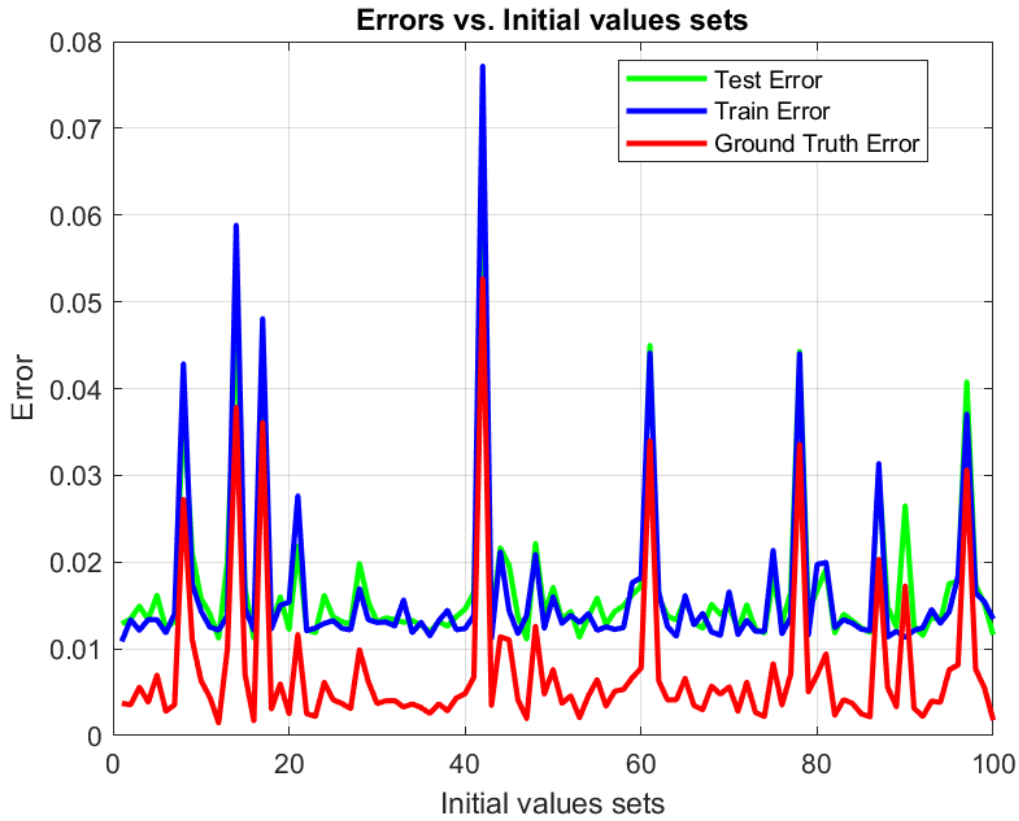


Figure 8: *Network outputs for test values according to the number of samples used in the training.*

1.3 Relevant Code

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

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %                               Master in Robotics
3 %                               Applied Artificial Intelligence
4 %
5 % Assinment 5.2: Function Generalization - MLP
6 % Student: Josep Barbera Civera
7 % ID: 17048
8 % Date: 14/04/2024
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10
11 % 1.- Choose an adequate MLP structure and training set.
12 % Plot in the same figure the training set, the output of the MLP
13 % for the test set, and the ground truth sin function.
14 %
15 % Evaluate the evolution of the train error, the test error and
16 % the ground truth error in the following cases:
17 %
18 % 2.- Changing the training parameters:
19 %   initial values, (# of epochs, optimization algorithm)
20 %
21 % 3.- Changing the training data:
22 %   # of samples (order of samples)
23 %
24 % 4.- Changing the net structure:
25 %   # of neurons
26
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 % 1. Training set, MLP structure and initial plots
29 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30 generate_data = false;          %%%% <- change here as needed!
31 %%%%
32 plot_specific_MLP_training = false; %%%% <- change here as needed!
33 %%%%
34 if(generate_data)
35     %% Training data (N_train, x_train, y_train)
36     N_train = 50;
37     x_train = linspace(0, 2*pi, N_train);
38     for i = 1:N_train
39         y_train(i) = sin(x_train(i)) + normrnd(0, 0.1);
40     end
41
42     %% Test data (N_test, x_test, y_test, y_gtruth)
43     N_test = 1000;
44     x_test = linspace(0, 2*pi, N_test);
45     y_gtruth = sin(x_test); % ground truth
46     for i = 1:N_test
47         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
48     end
49 end
50
51 if(plot_specific_MLP_training)

```

```

50 %% MLP structure
51 trainFcn = 'trainlm';
52 hiddenSizes = 4;
53 net = feedforwardnet (hiddenSizes, trainFcn);
54 %% MLP training
55 [net, tr] = train(net, x_train, y_train);
56 %% MLP testing
57 % Test error
58 MLP_test = net(x_test);
59 perf_test = perform(net,y_test, MLP_test);
60 % Train error
61 MLP_train = net(x_train);
62 perf_train = perform(net,y_train,MLP_train );
63 % Ground truth error
64 perf_gtruth = perform(net, y_gtruth, MLP_test);
65
66
67 %% Plot data
68 % Plot training data
69 plot(x_train, y_train, 'b.', 'DisplayName', "training data");
70 hold on;
71 % Plot test data
72 s1 = scatter(x_test, y_test, 5, 'DisplayName', "test data", ...
73             'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g');
74 alpha(s1, .1)
75 hold on;
76 % Plot truth data
77 plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
78 hold on;
79 % Plot MLP output for test data
80 plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
81 title("MLP training");
82 my_subtitle = sprintf("test error: %.4f, train error: %.4f, gtruth
83                       error: %.4f", ...
84                       perf_test, perf_train, perf_gtruth);
85 subtitle(my_subtitle);
86 xlabel("x");
87 ylabel("y");
88 legend("training data", "test data", "Sin(x)", "MLP output");
89 figure_name = sprintf("test_error_%.3g_train_error_%.3
90                       g_gtruth_error_%.3g.png", ...
91                       perf_test, perf_train, perf_gtruth);
92 saveas(gcf, figure_name);
93 hold off;
94 end
95 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
96 % 2. Net structure
97 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
98 folder_name = "net_structure";
99 if ~exist(folder_name, 'dir')
100     mkdir(folder_name);
101 end

```

```

100
101 generate_data_2 = false; %%%% <- change here as needed! %%%%
102
103 if(generate_data_2)
104     % Training data (N_train, x_train, y_train)
105     N_train = 50;
106     x_train = linspace(0, 2*pi, N_train);
107     for i = 1:N_train
108         y_train(i) = sin(x_train(i)) + normrnd(0, 0.1);
109     end
110
111     % Test data (N_test, x_test, y_test, y_gtruth)
112     N_test = 1000;
113     x_test = linspace(0, 2*pi, N_test);
114     y_gtruth = sin(x_test); % ground truth
115     for i = 1:N_test
116         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
117     end
118 end
119
120 test_net_structure = false; %%%% <- change here as needed! %%%%
121 neurons = 1:50;
122 if (test_net_structure)
123     for i = 1:numel(neurons)
124         fprintf("Training with %d neurons in the hidden layer\n",
125             neurons(i));
126         trainFcn = 'trainlm';
127         net = feedforwardnet (neurons(i), trainFcn);
128         net.trainParam.epochs = 1000;
129
130         for j = 1:5
131             [net, tr] = train(net, x_train, y_train);
132             % Test error
133             MLP_test = net(x_test);
134             error_t(j) = mean((y_test-MLP_test).^2);
135             % per_error_t = perform(net, y_test, MLP_test);
136             % Train error
137             MLP_train = net(x_train);
138             error_tr(j) = mean((y_train-MLP_train).^2);
139             % per_error_tr = perform(net, y_train, MLP_train);
140             % Ground truth error
141             error_gt(j) = mean((y_gtruth-MLP_test).^2);
142             % per_error_gt = perform(net, y_gtruth, MLP_test);
143             % fprintf(['Error_t: %f, Percent Error_t: %f, Error_tr: %f,
144                 '...
145                 'Percent Error_tr: %f, Error_gt: %f, Percent Error_gt:
146                 %f\n'], ...
147                 error_t(j), per_error_t, error_tr(j), per_error_tr,
148                 ...
149                 error_gt(j), per_error_gt);
150         end
151         test_error(i) = mean(error_t);

```

```

148     train_error(i) = mean(error_tr);
149     gtruth_error(i) = mean(error_gt);
150     % Plot training data
151     plot(x_train, y_train, 'b.', 'DisplayName', "training data");
152     hold on;
153     % Plot test data
154     s1 = scatter(x_test, y_test, 5, 'DisplayName', "test data", ...
155                 'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g');
156     alpha(s1, .1)
157     hold on;
158     % Plot truth data
159     plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
160     hold on;
161     % Plot MLP output for test data
162     plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
163     my_title = sprintf("%d neurons", neurons(i));
164     title(my_title);
165     my_subtitle = sprintf("test error: %.4f, train error: %.4f,
166                          gtruth error: %.4f", ...
167                          test_error(i), train_error(i),
168                          gtruth_error(i));
169
170     subtitle(my_subtitle);
171     xlabel("x");
172     ylabel("y");
173     legend("training data", "test data", "Sin(x)", "MLP output");
174     hold off;
175     figure_name = sprintf("/%d_neurons_in_MLP.png", neurons(i));
176     saveas(gcf, strcat(folder_name, figure_name));
177     fprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f
178            \n", ...
179            test_error(i), train_error(i), gtruth_error(i));
180     disp("*****");
181 end
182 % Plot errors
183 plot(neurons, test_error, 'g-', 'LineWidth', 2, 'DisplayName', 'Test
184      Error');
185 hold on;
186 plot(neurons, train_error, 'b-', 'LineWidth', 2, 'DisplayName', '
187      Train Error');
188 plot(neurons, gtruth_error, 'r-', 'LineWidth', 2, 'DisplayName', '
189      Ground Truth Error');
190
191 % Title and labels
192 title('Errors vs. Number of Neurons in Hidden Layer');
193 xlabel('Number of Neurons');
194 ylabel('Error');
195 legend('Location', 'best');
196
197 % Adjust figure
198 grid on;
199 figure_name = sprintf("/error_plot.png");
200 saveas(gcf, strcat(folder_name, figure_name));

```

```

194 end
195
196
197 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
198 % 3. Number of samples
199 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
200 folder_name = "samples_number";
201 if ~exist(folder_name, 'dir')
202     mkdir(folder_name);
203 end
204
205 test_samples_number = false; %%%% <- change here as needed! %%%%
206 if (test_samples_number)
207
208     %% Test data (N_test, x_test, y_test, y_gtruth) (only once)
209     N_test = 1000;
210     x_test = linspace(0, 2*pi, N_test);
211     y_gtruth = sin(x_test); % ground truth
212     for i = 1:N_test
213         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
214     end
215
216     %% Net structure
217     trainFcn = 'trainlm';
218     net = feedforwardnet (5, trainFcn);
219     net.trainParam.epochs = 50;
220     net.trainParam.min_grad = 1e-10;
221     net.trainParam.max_fail = 50;
222     net.trainParam.goal = 0;
223     net.trainParam.showWindow = false;
224
225     data_number = 5:5:100;
226     for i = 1:numel(data_number)
227         %% Training data (N_train, x_train, y_train)
228         N_train = data_number(i);
229         disp(N_train)
230         x_train = linspace(0, 2*pi, N_train);
231         for k = 1:N_train
232             y_train(k) = sin(x_train(k)) + normrnd(0, 0.1);
233         end
234
235         fprintf("Training with %d samples\n", data_number(i));
236
237         for j = 1:5
238             [net, tr] = train(net, x_train, y_train);
239             % Test error
240             MLP_test = net(x_test);
241             error_t(j) = mean((y_test - MLP_test).^2);
242             % Train error
243             MLP_train = net(x_train);
244             error_tr(j) = mean((y_train - MLP_train).^2);
245             % Ground truth error

```

```

246         error_gt(j) = mean((y_gtruth - MLP_test).^2);
247     end
248
249     test_error(i) = mean(error_t);
250     train_error(i) = mean(error_tr);
251     gtruth_error(i) = mean(error_gt);
252     % Plot training data
253     plot(x_train, y_train, 'b.', 'DisplayName', "training data");
254     hold on;
255     % Plot test data
256     s1 = scatter(x_test, y_test, 5, 'DisplayName', "test data", ...
257                 'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g');
258     alpha(s1, .1)
259     hold on;
260     % Plot truth data
261     plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
262     hold on;
263     % Plot MLP output for test data
264     plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
265     my_title = sprintf("%d samples", data_number(i));
266     title(my_title);
267     my_subtitle = sprintf("test error: %.4f, train error: %.4f,
268                          gtruth error: %.4f", ...
269                          test_error(i), train_error(i),
270                          gtruth_error(i));
271
272     subtitle(my_subtitle);
273     xlabel("x");
274     ylabel("y");
275     legend("training data", "test data", "Sin(x)", "MLP output");
276     hold off;
277     figure_name = sprintf("/%d_samples_in_MLP.png", data_number(i));
278     saveas(gcf, strcat(folder_name, figure_name));
279     fprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f
280            \n", ...
281            test_error(i), train_error(i), gtruth_error(i));
282     disp("*****");
283 end
284 % Plot errors
285 plot(data_number, test_error, 'g-', 'LineWidth', 2, 'DisplayName', '
286     Test Error');
287 hold on;
288 plot(data_number, train_error, 'b-', 'LineWidth', 2, 'DisplayName', '
289     Train Error');
290 plot(data_number, gtruth_error, 'r-', 'LineWidth', 2, 'DisplayName', '
291     Ground Truth Error');
292
293 % Title and labels
294 title('Errors vs. Number of Samples in the Training Set');
295 xlabel('Number of Training Samples');
296 ylabel('Error');
297 legend('Location', 'best');

```



```

292 % Adjust figure
293 grid on;
294 figure_name = sprintf("/error_plot.png");
295 saveas(gcf, strcat(folder_name, figure_name));
296 end
297
298
299 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
300 % 4. Initial Values
301 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
302 folder_name = "initial_values";
303 if ~exist(folder_name, 'dir')
304     mkdir(folder_name);
305 end
306
307 test_initial_values = true; %%%% <- change here as needed! %%%%
308 if (test_initial_values)
309     %% Training data (N_train, x_train, y_train)
310     N_train = 50;
311     disp(N_train)
312     x_train = linspace(0, 2*pi, N_train);
313     for i = 1:N_train
314         y_train(i) = sin(x_train(i)) + normrnd(0, 0.1);
315     end
316     %% Test data (N_test, x_test, y_test, y_gtruth) (only once)
317     N_test = 1000;
318     x_test = linspace(0, 2*pi, N_test);
319     y_gtruth = sin(x_test); % ground truth
320     for i = 1:N_test
321         y_test(i) = y_gtruth(i) + normrnd(0, 0.1);
322     end
323
324     %% Net structure
325     trainFcn = 'trainlm';
326     net = feedforwardnet (5, trainFcn);
327     net.trainParam.epochs = 50;
328     net.trainParam.min_grad = 1e-10;
329     net.trainParam.max_fail = 50;
330     net.trainParam.goal = 0;
331     net.trainParam.showWindow = false;
332     initial_values = 1:100;
333     for i = 1:numel(initial_values)
334         fprintf("Training in the %d iteration\n", initial_values(i));
335         [net, tr] = train(net, x_train, y_train);
336         % Test error
337         MLP_test = net(x_test);
338         test_error(i) = mean((y_test - MLP_test).^2);
339         % Train error
340         MLP_train = net(x_train);
341         train_error(i) = mean((y_train - MLP_train).^2);
342         % Ground truth error
343         gtruth_error(i) = mean((y_gtruth - MLP_test).^2);

```

```

344
345 % Plot training data
346 plot(x_train, y_train, 'b.', 'DisplayName', "training data");
347 hold on;
348 % Plot test data
349 s1 = scatter(x_test, y_test, 5, 'DisplayName', "test data", ...
350             'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g');
351 alpha(s1, .1)
352 hold on;
353 % Plot truth data
354 plot(x_test, y_gtruth, 'r-', 'DisplayName', "Sin(x)");
355 hold on;
356 % Plot MLP output for test data
357 plot(x_test, MLP_test, 'k-', 'DisplayName', "MLP output");
358 my_title = sprintf("Training number %d", initial_values(i));
359 title(my_title);
360 my_subtitle = sprintf("test error: %.4f, train error: %.4f,
361                       gtruth error: %.4f", ...
362                       test_error(i), train_error(i),
363                       gtruth_error(i));
364 subtitle(my_subtitle);
365 xlabel("x");
366 ylabel("y");
367 legend("training data", "test data", "Sin(x)", "MLP output");
368 hold off;
369 figure_name = sprintf("/%d_initial_values_in_MLP.png",
370                       initial_values(i));
371 % saveas(gcf, strcat(folder_name, figure_name));
372 fprintf("test error: %.4f, train error: %.4f, gtruth error: %.4f
373        \n", ...
374        test_error(i), train_error(i), gtruth_error(i));
375 disp("*****");
376 end
377 % Plot errors
378 plot(initial_values, test_error, 'g-', 'LineWidth', 2, 'DisplayName'
379       , 'Test Error');
380 hold on;
381 plot(initial_values, train_error, 'b-', 'LineWidth', 2, 'DisplayName'
382       , 'Train Error');
383 plot(initial_values, gtruth_error, 'r-', 'LineWidth', 2, '
384       DisplayName', 'Ground Truth Error');
385 % Title and labels
386 title('Errors vs. Initial values sets');
387 xlabel('Initial values sets');
388 ylabel('Error');
389 legend('Location', 'best');
390 % Adjust figure
391 grid on;
392 figure_name = sprintf("/error_initial_values.png");
393 saveas(gcf, strcat(folder_name, figure_name));
394 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*

```
1 import matplotlib.pyplot as plt
2 import matplotlib.image as mpimg
3
4 figure_directory = "."
5 numbers_to_select = [1, 3, 5, 10, 20, 30, 40, 50]
6 fig, axes = plt.subplots(4, 2, figsize=(20, 10))
7 plt.subplots_adjust(wspace=-1, hspace=0.01)
8 axes = axes.flatten()
9
10 for i, num in enumerate(numbers_to_select):
11     figure_name = f"{figure_directory}/{num}_neurons_in_MLP.png"
12     img = mpimg.imread(figure_name)
13     axes[i].imshow(img)
14     axes[i].axis('off')
15
16 for j in range(len(numbers_to_select), len(axes)):
17     fig.delaxes(axes[j])
18
19 plt.tight_layout()
20 plt.savefig("selected_figures_grid.png")
21 plt.show()
```