Artificial Neural Networks

Project 1: Regression and Classification with MLPs

Lood, Cédric
Master of Bioinformatics

May 31, 2016

1 Context

The analysis presented in this report was done for the class of Artificial Neural Networks at KU Leuven (Spring 2016). It consists in practical implementations of Multi-layer perceptrons (MLP) to solve regression and classification problems in a supervised setting, with a brief foray into unsupervised learning for dimensionality reduction. The implementation was done in the MatLab environment (2015a) using the neural networks toolbox. The scripts for each of the sections can be found in the annex to this report.

2 Regression

The problem we need to solve here is to estimate a (unknown) non-linear function from a given dataset of 13600 datapoints uniformly sampled. Importantly to note, the sampled data is noiseless. The dependent variable was generated using the student number (r0575791) and the two dependent variables $X_1, X_2 \in [0, 1]$ (figure 1 for a rendering of the dataset generated).

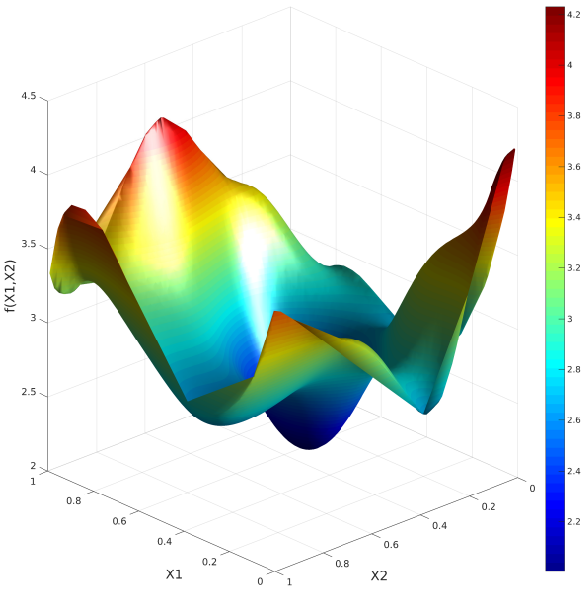

Figure 1: Visualization of the full dataset

For the creation of the training, validation, and test set, I first randomized the dataset and then proceeded to use the built-in matlab function *dividerand* which creates 3 lists of non-repeating indices of size 1000 that can then be used to subset the original dataset into the 3 subsets. The visualization of the sets on figure 2 reassures that the sampling was done correctly.

Although it is possible to use more than one layer for MLPs, I limited myself here at 1 hidden layer, comforted by the fact that they consist of universal approximator given that I used a non-linear transfer function. I also obtained good results previously (see homework 1) for function approximation using a single hidden layer. The choices I was faced with to optimize the architecture are the number of hidden units, the transfer function, and the choice of the training algorithm.

For the rest, the network has 2 input neurons connected to the hidden neurons, whose outputs are linearly combined by a single output neuron.

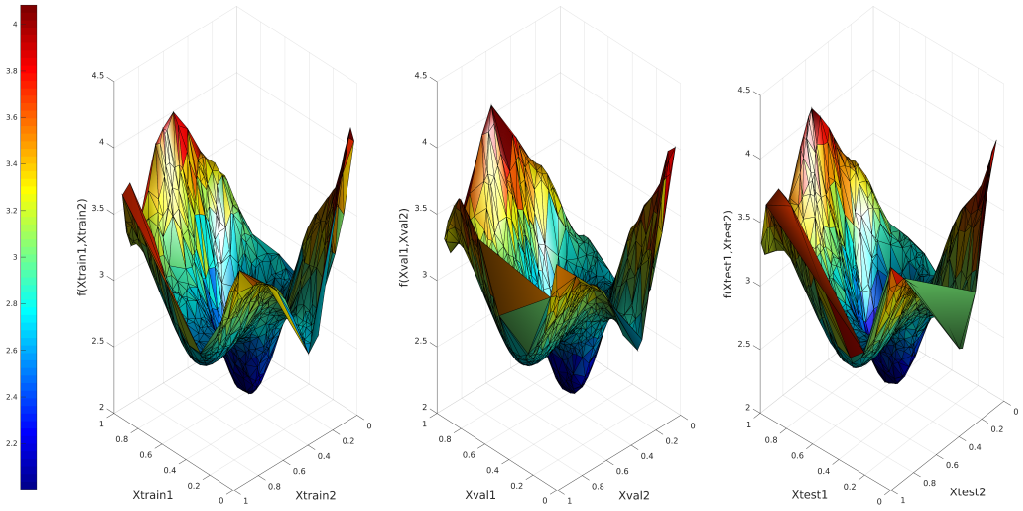

Figure 2: From left to right: training, validation, and test datasets (1000 points each)

For the training algorithm, I based my choice on consideration such as the size of the dataset. Since I worked on an interactive computing node of the VSC [1], I did not have to worry to much about the memory requirements (even though a 1000x1000 Hessian matrix can also be handled by a laptop computer without any troubles), and did not investigate conjugate gradient methods. Instead, I chose to focus on the Levenberg-Marquardt method, which is known for its fast convergence.

On the figure 3, I report the finding in terms of validation MSE (no usage of the training set at this stage). I performed a systematic search for the correct amount of hidden neurons using *transig* and *logsig*. On the same figure, the Test MSE for the selected architecture of 60 hidden neurons, and a *tansig* transfer function is reported.

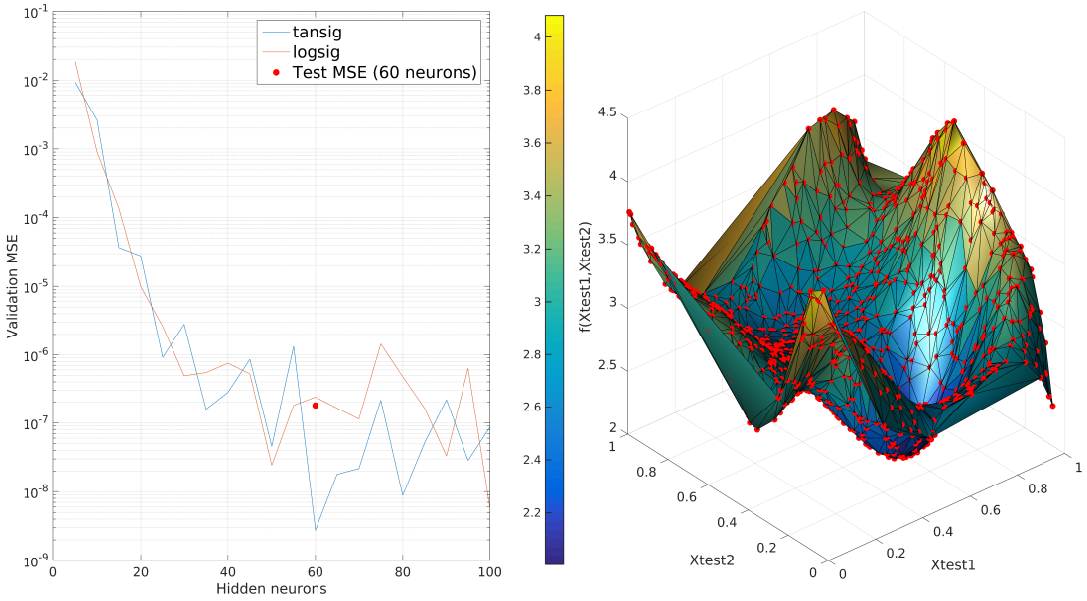

Figure 3: Validation MSE curves and test surface + prediction (red balls)

[1] Vlaams Super Computer

Computation of the difference between the predicted values and actual values for the test set revealed the extreme precision of the function estimation. The errors are in the order of 10^-5 to 10^-9, the largest error (in absolute value) is 0.015. Although this is indeed excellent, it is possible to improve the process by using more of the available data. Using cross validation on 80% of the dataset for example and set aside 20% of the dataset for test purposes.

3 Classification

In this section, we focus on a classification task related to red and white wine quality. The dataset was subset according to the instructions with my student number r0575791. Hence I had the task of classifying white wines with positive class 6, and negative class 7.

I tried to use the *feedforwardnet* but kept getting into troubles where all the predictions of my trained network were of one class, so I found a more modern alternative *patternnet*. I first reformulated my target, 6 and 7, as 1 and 0. This is compatible with the sigmoid function used in the output neuron. To use this as a classifier, one can simply take the rounded value of the output. Since the output is always in $[0, 1]$, the rounded value will be in $\{0, 1\}$, which maps naturally to the dataset.

For architecture of the network, I could not really see a large influence in the number of hidden units (see table 1), there was some support for an architecture with 15 hidden units. The training function used was the scaled conjugate gradient, default in *patternnet* since *trainlm* was causing errors during the training.

	1	2	3	4	5	10	15	20	25	30
CCR_test	71.75	74.67	72.40	74.67	73.37	71.1	75.97	74.35	73.05	73.37
CCR_val	73.37	75.00	75.00	75.32	75.32	75.32	76.99	76.62	76.62	74.67

Table 1: Correct classification ration (test and validation) for different architectures

The results of the PCA analysis are shown on figure 4. Using the rotate 3D functions in Matlab, I could not see any evidence of obvious patterns. The first PC explains about 30% of the variability in the data, then it drops down almost linearly for the next components. I selected the 6 first PC, to have 80%+ of the variability explained to reduce the dimension of the dataset.

Finally, I reconstructed the dataset using these 6 components. Using that reduced dataset, I trained another patternnet with different numbers of hidden units. The results were less variable than in the case above, but also a bit worse. The best CCR on the validation set was 73.37, while the best CCR on the test set was 71.75. It would then seem that the reduction does not impact too much the results of the classifier, so the user/designer of the neural network in this case could opt for it.

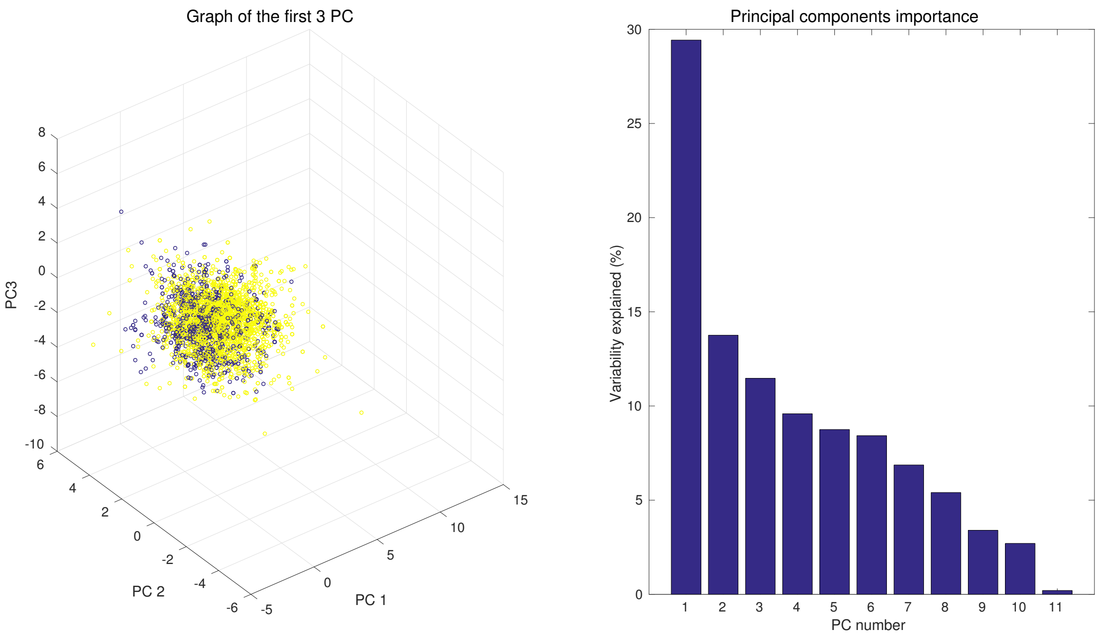

Figure 4: On the left, projection using the 3 first PC, on the right, the components' importance

Appendices

A Regression

```
clc, clear all, close all;
addpath 'export_fig'; % export pdf: https://github.com/altmany/export_fig
load '../datasets/regression.mat';

% generation of own data from student number r0575791
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tnew = (9*T1 + 7*T2 + 7*T3 + 5*T4 + 5*T5)/(9 + 7 + 7 + 5 + 5);

% plotting the dataset
%%%%%%%%%%%%%%%%%%%%%%
figure('Color', [1 1 1]);
tri = delaunay(X1, X2);
h = trisurf(tri, X1, X2, Tnew);
l = light('Position', [-50 -15 29]);
lighting phong; shading interp;
colormap Jet; colorbar EastOutside;
xlabel('X1','FontSize',14);
ylabel('X2','FontSize',14);
zlabel('f(X1,X2)','FontSize',14);

export_fig('regression_dataset.pdf');

% random permutation of the data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dataset = [X1 X2 Tnew];
dataset = dataset(randperm(size(dataset,1)),:);
ratio = 1/3;
[trainInd, valInd, testInd] = dividerand(3000, ratio, ratio, ratio);
```

```
30
31 % training set definition
32 Xtrain = [dataset(trainInd,1)'; dataset(trainInd,2)'];
33 Ytrain = dataset(trainInd,3)';
34
35 % validation set definition
36 Xval = [dataset(valInd,1)'; dataset(valInd,2)'];
37 Yval = dataset(valInd,3)';
38
39 % test set definition
40 Xtest = [dataset(testInd,1)'; dataset(testInd,2)'];
41 Ytest = dataset(testInd,3)';
42
43 % plotting surfaces
44 figure('Color', [1 1 1]);
45
46 % plot train set
47 subplot(1,3,1);
48 tri = delaunay(Xtrain(:,1), Xtrain(:,2));
49 h = trisurf(tri, Xtrain(:,1), Xtrain(:,2), Ytrain);
50 l = light('Position', [-50 -15 29]);
51 lighting phong; colormap Jet;
52 xlabel('Xtrain1','FontSize',14);
53 ylabel('Xtrain2','FontSize',14);
54 zlabel('f(Xtrain1,Xtrain2)','FontSize',14);
55
56 % plot validation set
57 subplot(1,3,2);
58 tri = delaunay(Xval(:,1), Xval(:,2));
59 h = trisurf(tri, Xval(:,1), Xval(:,2), Yval);
60 l = light('Position', [-50 -15 29]);
61 lighting phong; colormap Jet;
62 xlabel('Xval1','FontSize',14);
63 ylabel('Xval2','FontSize',14);
64 zlabel('f(Xval1,Xval2)','FontSize',14);
65
66 % plot test set
67 subplot(1,3,3);
68 tri = delaunay(Xtest(:,1), Xtest(:,2));
69 h = trisurf(tri, Xtest(:,1), Xtest(:,2), Ytest);
70 l = light('Position', [-50 -15 29]);
71 lighting phong; colormap Jet; colorbar EastOutside;
72 xlabel('Xtest1','FontSize',14);
73 ylabel('Xtest2','FontSize',14);
74 zlabel('f(Xtest1,Xtest2)','FontSize',14);
75
76 export_fig('regression_trainingsets.pdf');
77
78 % Estimation of number of hidden neurons
79 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80
81 % with sigmoid transfer function
82 n_hiddenn = 2:2:200;
83 mse_validation_logsig = zeros(100,1);
84 i = 1;
85 for nn = n_hiddenn
86     net = feedforwardnet(nn);
87     net.trainParam.showWindow = false;
88     net.trainParam.epochs = 5000;
89     net.divideFcn = 'dividetrain';
90     net.layers{1}.transferFcn = 'logsig';
91     net = train(net, Xtrain, Ytrain, 'UseParallel', 'yes');
```

```
    pred = sim(net, Xval);
    mse_validation_logsig(i) = perform(net, Yval, pred);
    i = i + 1;
end

% with hyperbolic tangent transfer function
n_hiddenn = 2:2:200;
mse_validation_tansig = zeros(100,1);
i = 1;
for nn = n_hiddenn
    net = feedforwardnet(nn);
    net.trainParam.showWindow = false;
    net.trainParam.epochs = 5000;
    net.divideFcn = 'dividetrain';
    net.layers{1}.transferFcn = 'tansig';
    net = train(net, Xtrain, Ytrain, 'UseParallel', 'yes');
    pred = sim(net, Xval);
    mse_validation_tansig(i) = perform(net, Yval, pred);
    i = i + 1;
end

figure('Color', [1 1 1]);
subplot(1,2,1);
semilogy(5:5:100, mse_validation_tansig);
hold on;
semilogy(5:5:100, mse_validation_logsig);
grid on;
xlabel('Hidden neurons','FontSize',14);
ylabel('Validation MSE','FontSize',14);

% Evaluation on test set
%%%%%%%%%%%%%%%%%%%%%%%%

net = feedforwardnet(60);
net.trainParam.showWindow = true;
net.trainParam.epochs = 5000;
net.divideFcn = 'dividetrain';
net.layers{1}.transferFcn = 'tansig';
net = train(net, Xtrain, Ytrain, 'UseParallel', 'yes');
pred = sim(net, Xtest);
performance = perform(net, Ytest, pred);

hold on;
plot(60, performance, 'r*');
legend('tansig', 'logsig', 'Test MSE (60 neurons)');

export_fig('regression_logtan_error.pdf');

% Plot param selection + Plot surface of test set + prediction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tets MSE for parameter selection
figure('Color', [1 1 1]);
subplot(1,2,1);
semilogy(5:5:100, mse_validation_tansig);
hold on;
semilogy(5:5:100, mse_validation_logsig);
grid on;
xlabel('Hidden neurons','FontSize',14);
ylabel('Validation MSE','FontSize',14);
hold on;
plot(60, performance, 'r*');
legend('tansig', 'logsig', 'Test MSE (60 neurons)');
```

```
154
155 % surface plot + prediction
156 subplot(1,2,2);
157 tri = delaunay(Xtest(1,:), Xtest(2,:));
158 h = trisurf(tri, Xtest(1,:), Xtest(2,:), Ytest);
159 l = light('Position', [-50 -15 29]);
160 lighting phong; colormap winter; colorbar EastOutside;
161 xlabel('Xtest1','FontSize',14);
162 ylabel('Xtest2','FontSize',14);
163 zlabel('f(Xtest1,Xtest2)','FontSize',14);
164 hold on;
165 scatter3(Xtest(1,:), Xtest(2,:), pred, 'r', 'filled');
```

B Classification

```
1  clc, clear all, close all;
2  addpath 'export_fig'; % export pdf: https://github.com/altmany/export_fig
3  rng(7); % setting random seed
4
5  % student number = r0575791, datasets initialization
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  csv_import = importdata('../datasets/winequality-white.csv');
8  data = csv_import.data;
9  cpos = data(data(:,12) == 6,:); cpos_size = length(cpos);
10 cneg = data(data(:,12) == 7,:); cneg_size = length(cneg);
11
12 X = [cpos(:,1:(end-1)) ; cneg(:,1:(end-1))]'; % label removed (6,7)
13 Y = [ones(cpos_size,1) ; zeros(cneg_size, 1)]'; % replaced by 1 and 0
14 stdx = mapstd(X);
15
16 % creation of training, validation, and test sets
17 n = cpos_size + cneg_size;
18 [train_ind, val_ind, test_ind] = dividerand(n, 0.8, 0.1, 0.1);
19
20 % Neural net training
21 %%%%%%%%%%%%%%%%%%%%%%
22 i = 1; ccr_val = zeros(10,1); ccr_test = zeros(10,1);
23 for hnn = [1 2 3 4 5 10 15 20 25 30]
24     net = patternnet(hnn);%
25     net.divideFcn = 'divideind';
26     net.trainParam.showWindow = false;
27     net.divideParam.trainInd = train_ind;
28     net.divideParam.valInd = val_ind;
29     net.divideParam.testInd = test_ind;
30     net = train(net, stdx, Y);
31
32     pred = round(sim(net, stdx(:,val_ind)));
33     ccr_val(i) = sum(pred == Y(:,val_ind))*100/length(val_ind);
34     pred = round(sim(net, stdx(:,test_ind)));
35     ccr_test(i) = sum(pred == Y(:,test_ind))*100/length(test_ind);
36     i = i + 1;
37 end
38
39 % PCA
40 %%%%%
41 [coeff,scores,¬,¬,explained] = pca(X(:,train_ind)','Centered', true, ...
       'VariableWeights','variance');
42 figure('Color',[1 1 1]);
43 subplot(1,2,1);
44 scatter3(scores(:,1), scores(:,2), scores(:,3), 15, Y(train_ind));
```

```
45 title('Graph of the first 3 PC'); xlabel('PC 1'); ylabel('PC 2'); ...
       zlabel('PC3');
46 subplot(1,2,2);
47 bar(explained);
48 title('Principal components importance'); xlabel('PC number'); ...
       ylabel('Variability explained (%)');
49
50 % reconstruction of whole dataset with 6 components
51 reduced_dim = coeff(:,1:6);
52 reduced_data = X' * reduced_dim;
53 reduced_data = reduced_data';
54 stdxr = reduced_data;%mapstd(reduced_data);
55
56 i = 1; ccr_val = zeros(10,1); ccr_test = zeros(10,1);
57 for hnn = [1 2 3 4 5 10 15 20 25 30]
58     net = patternnet(hnn);%
59     net.divideFcn = 'divideind';
60     net.trainParam.max_fail = 50;
61     net.trainParam.min_grad=1e-10;
62     net.trainParam.showWindow = true;
63     net.divideParam.trainInd = train_ind;
64     net.divideParam.valInd = val_ind;
65     net.divideParam.testInd = test_ind;
66     net = train(net, stdxr, Y);
67
68     pred = round(sim(net, stdxr(:,val_ind)));
69     ccr_val(i) = sum(pred == Y(:,val_ind))*100/length(val_ind);
70     pred = round(sim(net, stdxr(:,test_ind)));
71     ccr_test(i) = sum(pred == Y(:,test_ind))*100/length(test_ind);
72     i = i + 1;
73 end
74
75 [max_ccr_test, pos_test] = max(ccr_test);
76 [max_ccr_val, pos_val] = max(ccr_val);
```