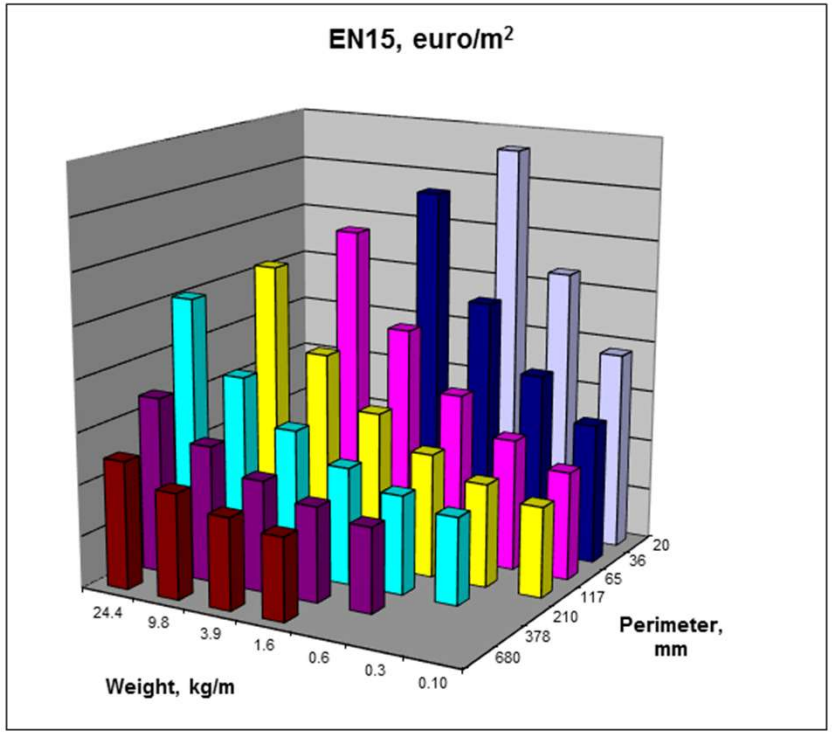
Linear regression and neural network application example: cost estimation of anodized extruded aluminium profiles

Anodizing is a process in which the thickness of the (natural) oxidation layer on aluminium is electrolytically increased. Aluminium profiles are anodized in large automatic plants, in which the profiles are manually loaded to the process and finally unloaded. Significant costs are related to labor, energy, chemicals and their handling, rent, maintenance, and initial investment. – Based on the process it can be concluded that the costs can be allocated to products based on their area, weight and oxidation layer thickness.

Cost data obtained from plant cost data collection system for various profiles:

	A	B	C	D
1	**Data**			
2	[µm]	[mm]	[kg]	[euro/m2]
3	**Thickne**	**Perimeter**	**Weight**	**Cost**
4	5	20	0.09	5.6
5	10	36	0.2	5.2
6	15	65	0.6	5.4
7	20	117	1.6	5.4
8	5	210	3.9	4.1
9	10	378	9.8	4.1
10	15	680	24.4	4.2
11	20	65	0.10	3.8



Linear regression and neural network application example: cost of anodizing extruded aluminium profiles

- We will fit a linear regression model and a neural network to the data and examine the characteristics of the results.
- Our inputs are coating layer Thickness, profile Perimeter (relates to coated area), and Weight (per meter).
- Cost in euros / area coated is the output.
- Our example data is in file AnodizingData.mat

[μm]	[mm]	[kg]	[euro/m2]
Thickness	**Perimeter**	**Weight**	**Cost**
5	20	0.09	5.6
10	36	0.2	5.2
15	65	0.6	5.4
20	117	1.6	5.4
5	210	3.9	4.1
10	378	9.8	4.1
15	680	24.4	4.2
20	65	0.10	3.8
5	117	0.3	2.7
10	210	0.6	2.9
15	65	0.2	4.1
20	117	2.2	5.9
5	210	9.2	5.4
10	20	0.025	4.0
15	36	0.2	5.6
20	65	0.6	5.8
5	680	12.0	2.9
10	65	0.25	3.9
15	117	0.6	4.0
20	210	1.6	4.1

Assignment

Fit a linear regression model and a neural network to your line simulation data from course MEC-E1080 and find out how well the models work. Fitting a model (linear regression or neural network) to data obtained using another model (simulation) is sometimes called metamodeling.

- Model inputs are the factors that you varied in simulation experiments
- Outputs are the production amounts
- Generate more data with your simulation model if necessary
- Test both models with data that was not used in training. Especially inputs outside training range are interesting
- Do regression analysis also with standardized data
- For neural network fitting use nnstart -> nftool
- For processing time variation use CV value (= standard deviation / average; for Erlang CV = $1/\sqrt{k}$)

Assignment – Discussion of results

- For the regression analysis, analyze the performance of the model based on the report KPIs
- Make conclusions based on the regression analysis with standardized data
- For neural network results, report R^2 and Standard Error. You can calculate these the same way as we did “manually” using Excel in the exercise – see RegressionAnalysisExample.xls in the course material
- Compare the R^2 and standard error for both models
- You can even list the errors of the models’ outputs and the target values and analyze their distribution, min, and max values

Assignment – additional study for extra points

- For neural network test different
  - Shares of training/validation/test data
  - Number of parallel nodes
  - Other training methods
- Compare network with two outputs to the results obtained with separate networks for production amounts and throughput time
  - Two (or more) outputs in the network require corresponding number of target data

Matlab and nftool hints

Data can be exported and imported in many ways from/to Matlab. For example, simply as follows:

- Data to Matlab: New Variable -> paste data (copied from Excel) to the appearing table
- Results from Matlab: Open data item in Matlab workspace and copy contents (to Excel) or save to file.
- Network training algorithms are not optimal. So, you may need to reinitialize the weights and retrain network several times to find a good fit
- When you find a good fit, select “Export Model” and name it e.g. “Good”
- To use the network, write for example Good.Network(test) on the Matlab command line and you will get the network output for input set “test”. Or try Good.Network(AnodiInputs) in the exercise session.
- Write y = Good.Network(AnodiInputs) to create variable y containing the outputs values to workspace. Now you have the example results available for your own analysis.

…Matlab and nftool hints

- You can Save Workspace – containing network and results – for later use.
- View network weights: for our simple network (default name in nftool script is net) type net.b{1}, net.b{2}, net.IW{1,1} or net.LW{2,1} on Matlab command line
- To create and deploy your network from Matlab command line, select Generate Code -> Generate Comprehensive Training Script. Run the generated script (and edit it as appropriate). After running it, you will find the results in the workspace.
- Note that Matlab first normalizes the data and then trains the network. Therefore, you need to do this normalization and de-normalization if you use the coefficients in your own applications

…Matlab and nftool hints

- Note that the R value in nftool is correlation between model outputs and output (target) values in your data. You could calculate it "manually" using for example Excel's CORREL(*array1*;*array2*) function. Squaring correlation does not necessarily produce exactly the same value found for R^2 using the equations shown on lecture notes and calculated in the Excel examples. This takes place if the averages of the measured values and (neural network) outputs are not the same. So, calculate the R^2 for neural network as it is done for linear regression to make them comparable. Standard errors for the different models are directly comparable.
- You can read more about the difference between R^2 and correlation for example here: http://www.win-vector.com/blog/2011/11/correlation-and-r-squared/