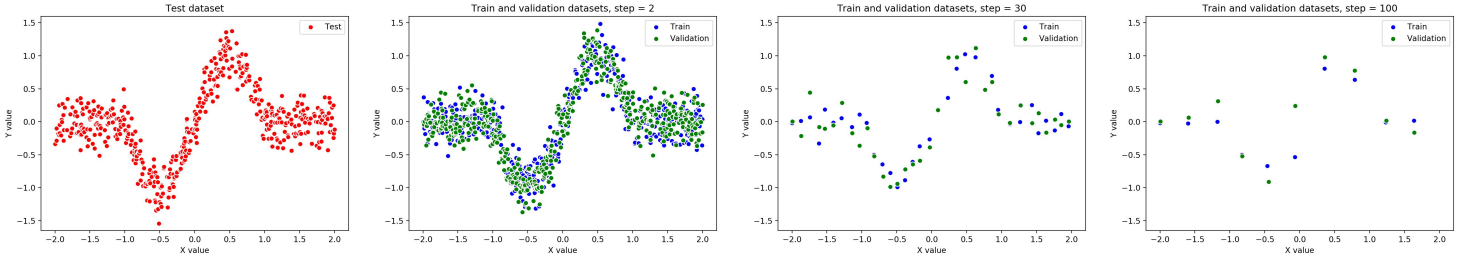
TRIED TPB-01 Report

Carl Robinson - 20th Nov 2017

Regression with Multi-layered Perceptrons - Part 1: Approximation of Expected Value

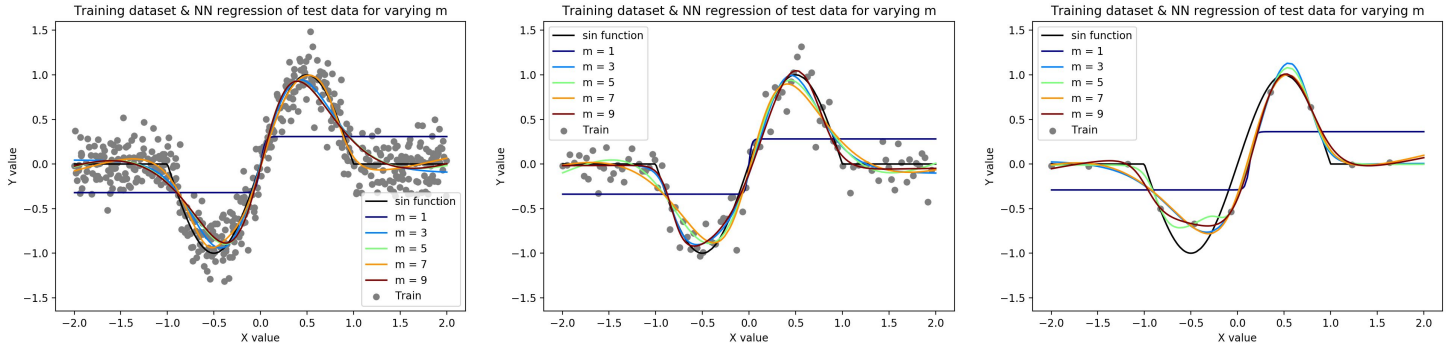
Objective: To put into practice a methodology for finding a regression function using multi-layered perceptrons.
Dataset: Simulated data created using sin function, Gaussian noise added. **Methods**: Multi-layered perceptrons.

Test dataset used in all cases (500 elements). Training datasets for selection step = 2, 30 and 100, noise sigma = 0.2



- As the selection step increases, the size and density of the training and validation sets decrease.

Test data regression using models of m hidden layer neurons, trained with datasets of step = 2, 30 and 100



- When m = 1 the model is too simple to be useful, regardless of the dataset size. However, for m ≥ 3, the models all closely approximate the sin function. It is difficult to determine which values of m perform better from these graphs.

Error of training, validation & test data regressions, for m hidden layer neurons, for varying dataset selection steps

step = 2 (graph)

M	Train	Validation	Test
1	0.42155	0.43123	0.42268
3	0.21196	0.20816	0.21187
5	0.21739	0.21716	0.2207
7	0.19626	0.19896	0.19708
9	0.21821	0.21911	0.22216

step = 3

M	Train	Validation	Test
1	0.43835	0.4218	0.42249
3	0.20648	0.18966	0.19425
5	0.22495	0.20603	0.22153
7	0.20511	0.19084	0.1969
9	0.21113	0.19471	0.20378

step = 4

M	Train	Validation	Test
1	0.40869	0.43419	0.42352
3	0.22073	0.23205	0.23115
5	0.19632	0.20778	0.19931
7	0.21215	0.22587	0.22286
9	0.19604	0.20523	0.19779

step = 5

M	Train	Validation	Test
1	0.41732	0.41615	0.42219
3	0.18436	0.19316	0.20198
5	0.18644	0.19156	0.19823
7	0.20051	0.19974	0.21071
9	0.18097	0.18991	0.19753

step = 10

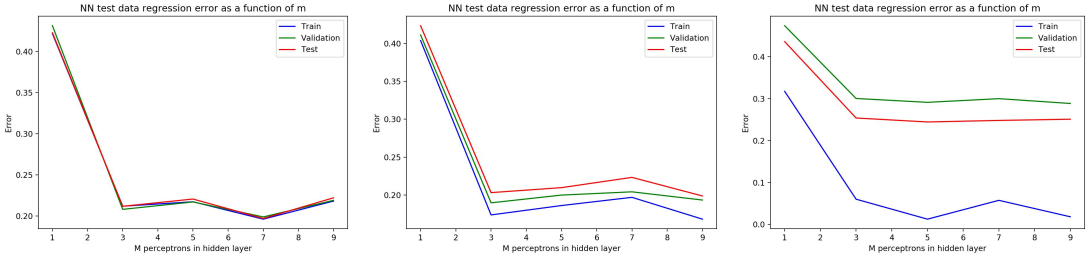
M	Train	Validation	Test
1	0.40367	0.41108	0.42325
3	0.17387	0.18982	0.20327
5	0.18624	0.20002	0.20983
7	0.19705	0.20432	0.22336
9	0.1683	0.19349	0.19886

step = 30 (graph)

M	Train	Validation	Test
1	0.37022	0.39856	0.42565
3	0.1301	0.19393	0.20366
5	0.13409	0.19171	0.20748
7	0.12915	0.18885	0.20787
9	0.16642	0.19522	0.23449

step = 100 (graph)

M	Train	Validation	Test
1	0.31725	0.47374	0.43579
3	0.06013	0.30026	0.25371
5	0.01253	0.29106	0.24427
7	0.05741	0.29985	0.24793
9	0.0182	0.28838	0.25085



- For step=2, regression graphs show minimum error is likely to be M=7, as it clearly fits the sin function the best. The error graphs and table confirm this to be the case.

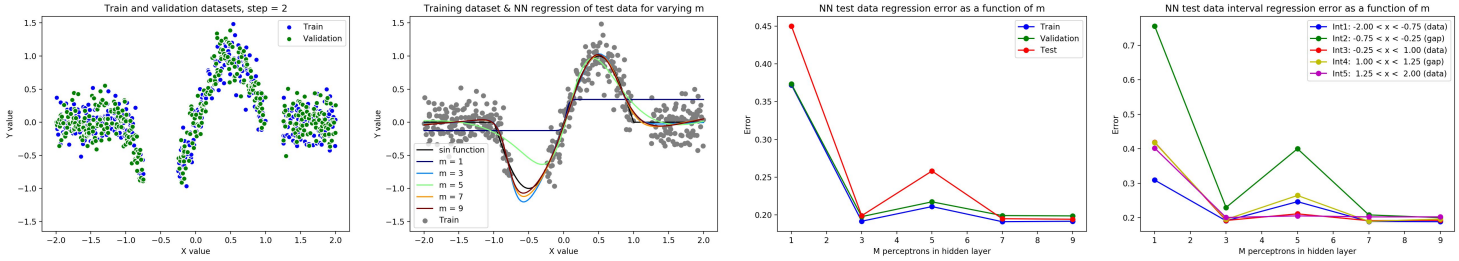
- For step=30, regression graphs are harder to interpret as the curves are more similar. M=3 or M=9 seem to be the closest fitting to the sin function. The error graphs and table confirm that M=9 has the lowest error.
- For step=100, the regression graphs do not clearly indicate which m value produces the closest fit. The error graphs and table show that M=5 has the lowest error, but only marginally.

From the error graphs/table we see that the optimal architecture for all step values (training set sizes) is as follows:

Step Value (low value = large training set)	2	3	4	5	10	30	100
Optimal Architecture (m neurons in hidden layer)	7	3	9	9	9	3	3

- With a large training dataset (step = 2, 4, 5, 10), the greater the number of neurons in the hidden layer, the lower the error is. This is because a large training set is more representative of the general case, so that a closer fitting model will generalise well to unseen test sets.
- There is an anomaly when step = 3, where the optimal number of hidden layer neurons is only 3 - this error value is very close to the error achieved when step = 9 though, and may be due to the random initialisation of weights in the network.
- With a very sparse datasets (step = 30 or 100) the fewer the number of neurons in the hidden layer, the lower the error is. This is because a higher number of neurons would cause more overfitting to the relatively few training samples, leading to the model generalising very poorly.
- From the error graphs, we see that the difference between the training error and the validation/test errors increases dramatically as the step increases (i.e. as the training set size decreases), as models trained on fewer data generalise more poorly. Note that while we often use the validation set to choose the correct model (e.g. the choice of m) to use with the test set (to discover the true performance of model), here we plot the test error for all m to show that test and validation errors follow each other closely.

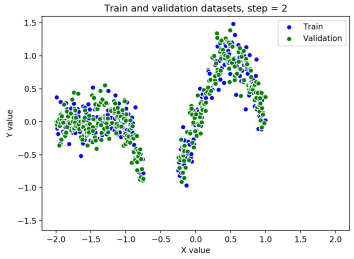
<u>Case 1: Intervals removed [-0.75 -0.25] U [1.00 1.25]</u>

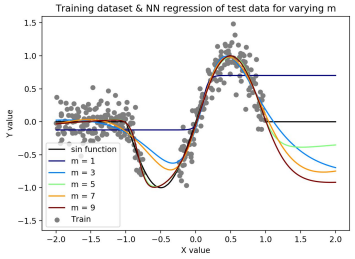


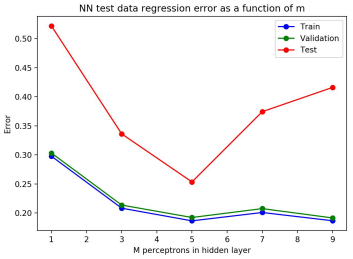
M	Train	Validation	Test	Test_int1	Test_int2	Test_int3	Test_int4	Test_int5
1	0.37213	0.37332	0.45009	0.30899	0.75614	0.41869	0.41765	0.4018
3	0.19166	0.19774	0.19897	0.19074	0.22955	0.19188	0.19478	0.20041
5	0.21103	0.21732	0.25815	0.24643	0.40004	0.21104	0.2644	0.20518
7	0.19103	0.19913	0.19496	0.18895	0.20794	0.19124	0.1892	0.20221
9	0.19159	0.1987	0.19425	0.1883	0.19991	0.19234	0.19591	0.2023

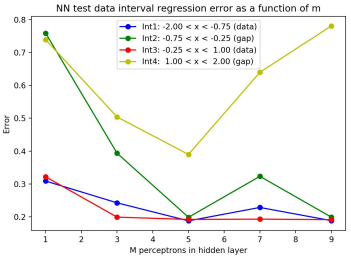
- The missing intervals in the training data distort the weights of the NN model. The test regression error is highest when m=1, then interestingly when m=5. The regression graph shows the regression line for m=5 ‘jumping the gap’ between intervals 1 and 3, producing a poor fit with the sin function over this interval.
- Intervals 2 and 4 have the highest error; those where data the training data has been removed. Interval 2 error is much higher than that of interval 4, as the sin function changes direction quickly over this interval. The model can only use the data provided, so it is unable to accurately predict this complex behaviour.
- The lowest error is found when m=3 or m=9. The m=3 model is likely to be highly generalisable as there are only a few neurons in the hidden layer, which prevents it from overfitting the data. Conversely, m=9 likely overfits the training data, but the training set is large (step=2), so there is enough training data to represent the test data well.

Case 2: Intervals removed [-0.75 -0.25] U [1.00 2.00]





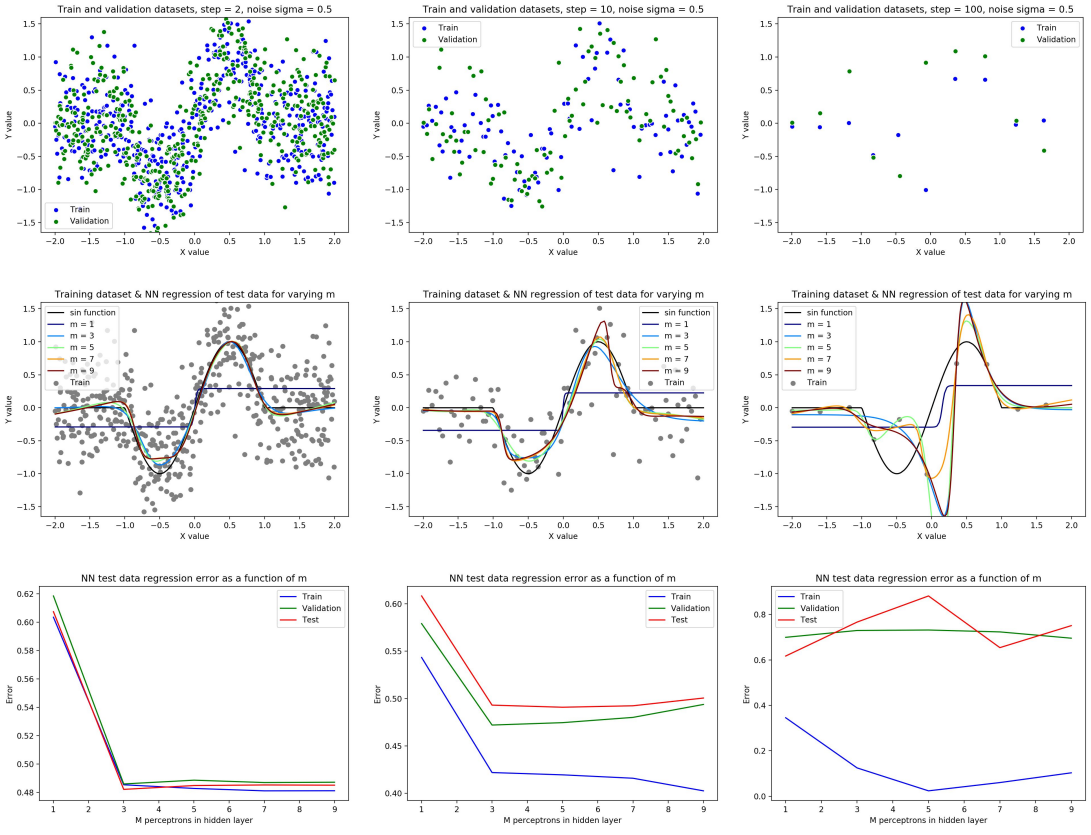




M	Train	Validation	Test	Test_int1	Test_int2	Test_int3	Test_int4
1	0.29773	0.30317	0.52174	0.30912	0.75834	0.32203	0.73852
3	0.20834	0.21373	0.33612	0.2429	0.39356	0.19948	0.50375
5	0.1866	0.19236	0.25351	0.18822	0.19903	0.19248	0.38999
7	0.20094	0.20763	0.37429	0.2289	0.32349	0.19338	0.63966
9	0.18677	0.19161	0.41601	0.1889	0.19955	0.19184	0.78042

- Conversely, for this dataset m=5 produces the lowest error of all m values. This suggests that the choice of m is highly dependent on the data involved, and an experimental approach is required to find the optimum.
- Again, the intervals where data the training data has been removed have the highest regression error. In particular the interval [1.00 2.00] has a very high error, as it is not bounded on both sides by training data points as in case 1. This means the weights of the model are no longer trained to represent values between two sets of data points, but instead are trained on ‘empty space’, and consequently tend to drift off in the direction they were last tending towards.

Noisier datasets where selection step = 2, 10 and 100, noise sigma = 0.5



- Here we increase the standard deviation of the noise added to the sin function values from 0.2 to 0.5. We see that for a large training dataset (step=2) the regression curves still fit the sin function fairly well for all values of m. The least-well-fitting is m=9, which ‘jumps the gap’ around [-0.7 -0.2] due to it overfitting the noise in that interval.
- This behaviour of overfitting the noise becomes more apparent with a reduced dataset (step=10), where we see the higher values of m in particular stretch outside the bounds of the sin function to overfit the noise. The models m=7 and m=9 skew to the left of the first curve around [-0.7 -0.2] and m=9 skews far above the second curve at [0.4 0.7].
- When we reduce the dataset size even more (step=100) the overfitting is so severe that we obtain a model that no longer represents the sin function at all, whichever value of m is used.
- The error graphs show that test and validation errors are lowest for low values of m, and rise steadily as m increases.