STEP BY STEP INSTRUCTIONS ON HOW TO IMPLEMENT NEUROET, CONDUCT SENSITIVITY ANALYSIS, AND EXTRACT A MODEL EQUATION FROM A TRAINED NETWORK.

Peter A Noble PhD

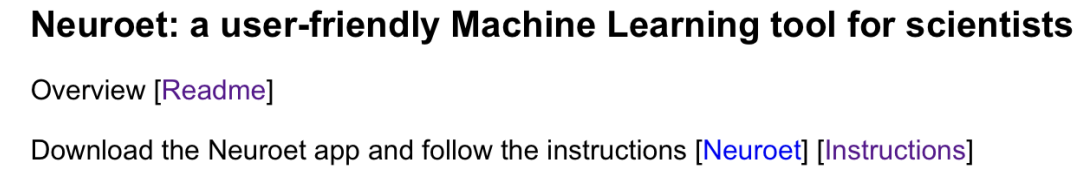
Overview

Neuroet is an easy-to-use artificial neural network (NN) package designed to assist with determining relationships among variables in complex ecological and biological systems. The package features a procedure to optimize the architecture of NNs by adjusting the number of neurons in the hidden layer, and a novel procedure to identify the input variable, or combinations of input variables, that is/are important for predicting outputs. The package also includes a method to extract equations defining relationships among the data (independent of the NN package). The performance of Neuroet has been assessed using benchmark standards for NNs.

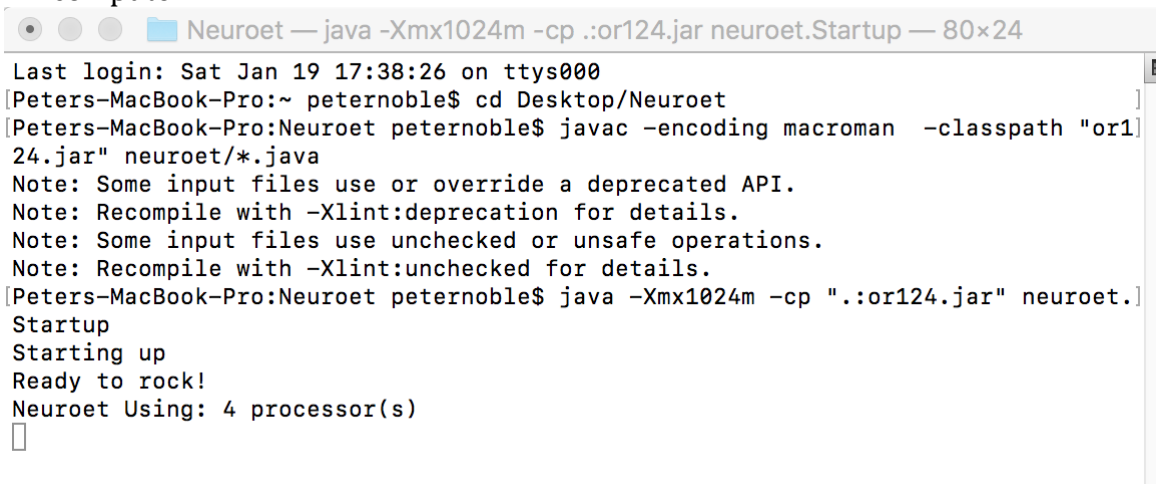
Noble, P.A. and E. Tribou. (2007) Neuroet: an easy-to-use artificial neural network for ecological and biological modelling. Ecological Modelling 203:87-98.

Instructions

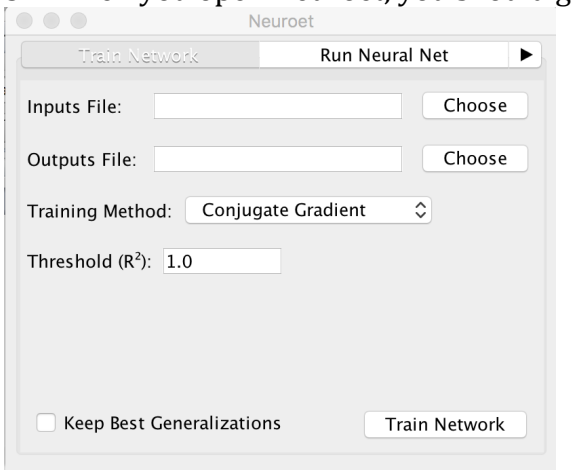
1. Go to the webpage: http://peteranoble.com/software.html
2. Find Neuroet (below).



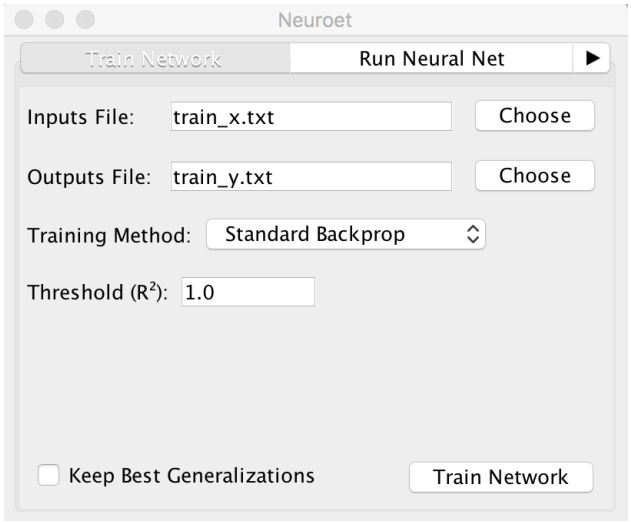
3. Follow the Instructions and download the application. Once you have compiled the java script in the terminal console, run it. Below is the log from my computer.



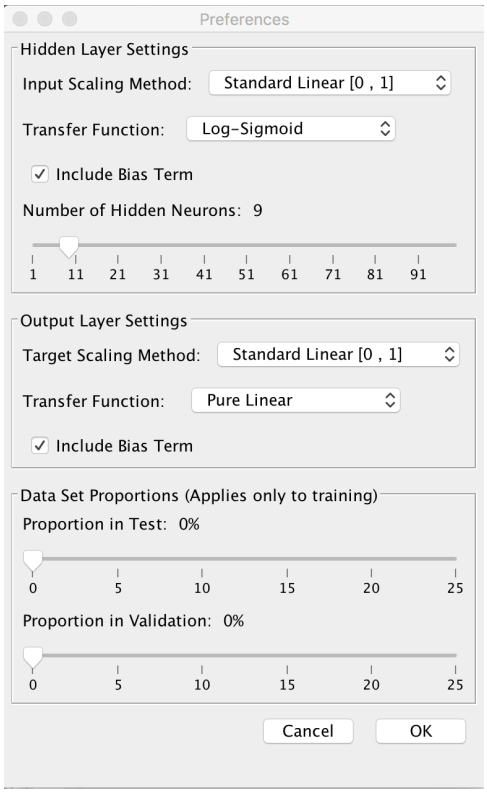
4. Download the sample training and testing files from the web site.
5. When you open Neuroet, you should get a panel that looks like this:



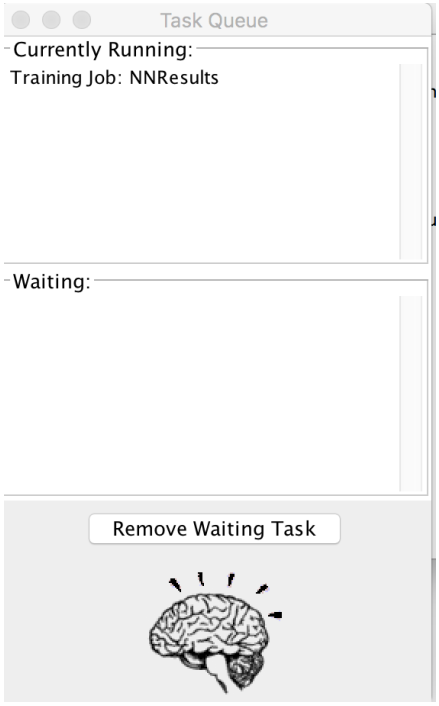
6. Find the training files on your Desktop using the 'Choose' button and set the training method to Standard Backprop (below):



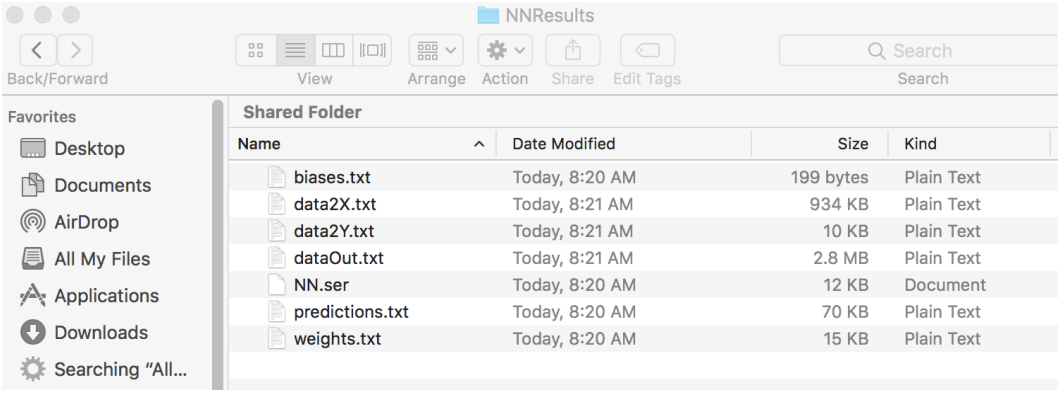
7. Up in the top bar of the menu under Edit, there is a 'Preferences' button. Press it to open the settings panel. Set the input scaling method to Standard Linear, the transfer function to Log-Sigmoid, check include bias term, set the number of hidden neurons to the square root of number of inputs (i.e., sqrt(83)=~9). Set the output layer scaling to Standard Linear and the transfer function to 'pure linear'. Click 'ok' (below):



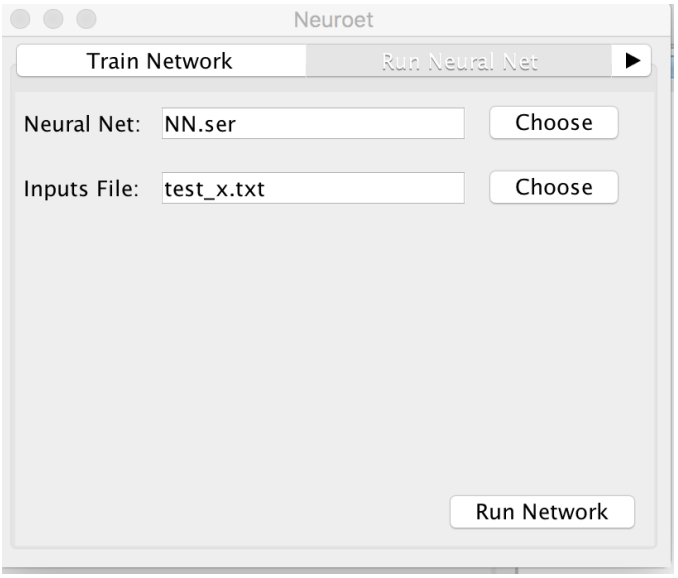
8. Click to run the neural network. A panel should emerge stating that the network is training (see below):



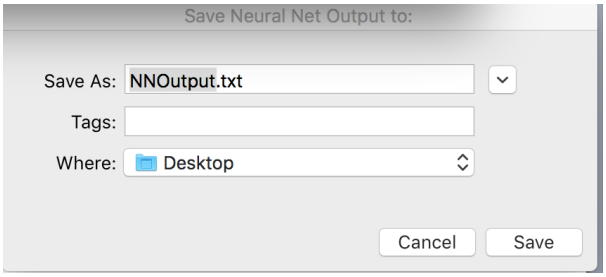
9. When it is finished, you will get a file folder containing 7 files. The NN.ser contains the compacted trained neural network that can be used for testing.



10. To see how well the network trained, test it using a test file. Click on the top bar of panel to 'Run the Neural Net'. Choose the file in the folder containing the NN.ser and the file containing the test file.



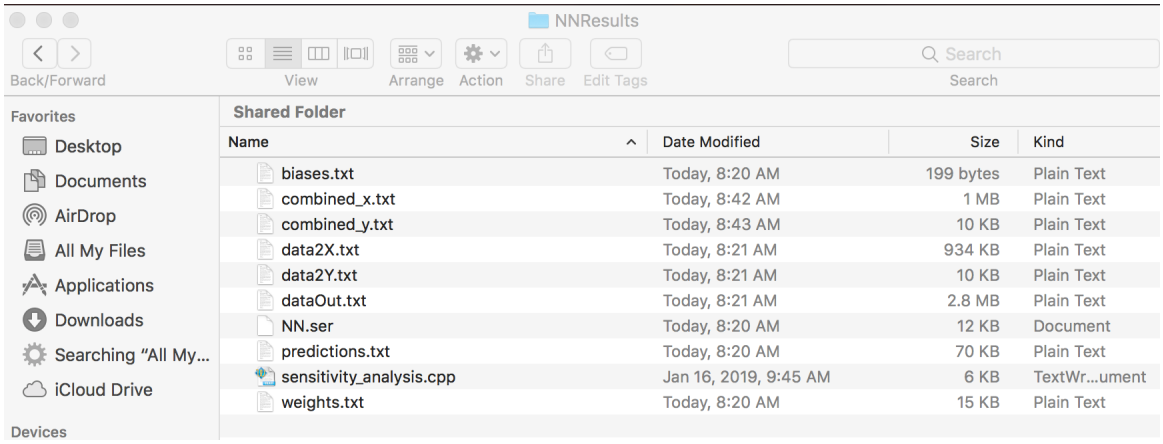
11. Provide a file name that will store the *Y*-predictions of the network.



Transfer to MS Excel and compare the predicted Y's to the actual Y's. The predictions in NNOutput.txt are in decimals. It is necessary to convert the decimals to integers using the round function (Round(x,0)). Determine the number of correct by making a compare function. That is: =IF(CF2=CI2,1,""). Then sum the column. In my example the number of correct predictions is 2278 out of 2460, which is about 93%.

Which of the input variables most contributes to the output predictions?

This can be determined by conducting sensitivity analysis. I wrote a C++ program to do this. Sensitivity analysis needs the weight and bias files and two files containing the inputs and outputs of the training and testing files combined. To make it easy, we will perform sensitivity analysis in the NNResults folder that was generated by Neuroet (since it contains the weights.txt and biases.txt files). Transfer the combined_x.txt and combined_y.txt to the folder and the sensitivity_analysis.cpp file. The folder should look like this:



To compile sensitivity_analysis.cpp, open a new terminal console and set the terminal the right directory using 'cd Desktop/NNResults'. Then type the following:

g++ sensitivity_analysis.cpp -o sensitivity analysis

A new icon should appear in the folder with the name 'sensitivity_analysis'. To run the application type the following:

./sensitivity_analysis 83 9 weights.txt biases.txt combined_x.txt combined_y.txt out.txt

The number 83 refers to the number of input neurons and the number 9 refers to the number of hidden neurons.

The program will output results to the console screen and to a file called 'out.txt'. Drop the file into MS Excel (see Sensitivity Analysis folder).

A	B	C	D	E	F	G
Variable	min/max	correct	total	%	sensitivity	Combined
Original		4627	4920	0.940		
Var1	0	4606	4920	0.936	0.00	0.03
Var1	1	4499	4920	0.914	0.03	
Var2	0	4615	4920	0.938	0.00	0.01
Var2	1	4601	4920	0.935	0.01	
Var3	0	4592	4920	0.933	0.01	0.13
Var3	1	4011	4920	0.815	0.13	
Var4	0	4595	4920	0.934	0.01	0.02
Var4	1	4546	4920	0.924	0.02	

The original ANN yielded 4627 out of a possible 4920 correct prediction, which represents about 94% accuracy. This will be used as the standard to compare to the models with variables set to zero or one.

For example, when all rows for Var1 were set to zero, it had little effect on the predicted outputs since 4606 were correct. When compared to the original model, the sensitivity was small (i.e., 0.00). However, when all the rows of Var1 were set to one, the number of correct classifications was 4499, which represents about a 3% change (i.e., 0.03). Summing the sensitivities of Var1 resulted in a combined sensitivity of 3%. Hence, we can conclude that Var1 contributed more to the original model than Var 2 (1%). In our model, Var83 most contributed to the original model (84%), followed by Var80 (44%) and Var29 (29%) and so on. One can order the variables by their contributions.

Note that different ANN models will yield different sensitivities. Therefore, the best practice is to repeat training and sensitivity analysis ten times and determine the global rank order of the variables. Since some variables do not contribute the ANN models, one can remove them (based on their global rank order), and build a new and potentially better model. Sensitivity analysis revealed that the top ten variables were: Var25, Var70, Var71, Var72, Var73, Var75, Var80, Var81, Var82, and Var83. When I reran ten ANN models using these 10 variables (10 input neurons, 4 hidden neurons, and one output), the R2 of the relationship between actual and predicted *Y*s improved from 94% to 95%.

Extracting the equation of one ANN model

One can extract the equation of a model by importing the biases and weight files into MS Excel (see extracting_model.xlsx). The spreadsheet 'datax' contains the normalized input variables. The spreadsheet 'data2x' contains the raw input variables. The spreadsheet 'datay' contains the actual output variables. The spreadsheet 'work' contains the model based on the normalized data while the spreadsheet 'work2_min_max' contains the model based on raw data. One can see the derived equations by clicking on the Cell R4 in the 'work2_min_max' spreadsheet. The final equation for the non-normalized data is:

Y' = ROUND(((1/(1+EXP(-1*(((Var25)*(-1.139223879
))+((Var70/106)*(-0.999696348
))+((Var71)*(-1.007919835
))+(((Var72-1)/(110-1))*(-2.679811195
))+((Var73/30)*(2.597915535
))+((Var75/34)*(1.649003437
))+((Var80)*(-3.74422832
))+((Var81/18)*(-5.068910678
))+(((Var82-1)/(19-1))*(0.48236017
))+((Var83)*(-3.470410625
))+(-2.150381483)))))*(-1.916110954
))+((1/(1+EXP(-1*(((Var25)* 4.616621931
)+((Var70 /106)* 3.680553783
)+((Var71)* -1.421481469
)+(((Var72 -1)/(110-1))* 8.718725142
)+((Var73 /30)* 3.740438456
)+((Var75 /34)*(-1.147018968
))+((Var80)* 0.837602533
)+((Var81 /18)* 1.056348211
)+(((Var82 -1)/(19-1))* 0.141720207
)+((Var83)* -1.925553932
)+(-9.639034549)))))* -2.55531288
)+((1/(1+EXP(-1*(((Var25)* 7.869828979
)+((Var70 /106)* -5.288615095
)+((Var71)* 5.499610334
)+(((Var72 -1)/(110-1))* 2.480118711
)+((Var73/30)* 1.787745799
)+((Var75 /34)*(-9.213198408
))+((Var80)*(-2.080480779
))+((Var81 /18)* 1.296235716
)+(((Var82 -1)/(19-1))*(-3.168973131
))+((Var83)*(-5.218634245
))+(-7.034892874)))))*(-1.013599402
))+((1/(1+EXP(-1*(((Var25)* 11.55577944
)+((Var70 /106)* -8.115061901
)+((Var71)* 4.080625213
)+(((Var72 -1)/(110-1))* 6.850812829
)+((Var73 /30)*(-2.761700694
))+((Var75 /34)*(-2.366867832
))+((Var80)* 16.92350498
)+((Var81 /18)*(-6.248190302
))+(((Var82 -1)/(19-1))* 10.12057878
)+((Var83)* 15.97258931
)+(-11.64077796)))))* 0.904326071
)+ 0.073241432,0)

I built a program to predict *Y*s using normalized input variables (see predict.cpp). One can compile this program by typing into the terminal console:

g++ predict.cpp -o predict

Then type the following into the terminal console:

./predict 83 9 weights.txt biases.txt train_x.txt train_y.txt out.txt

In this example, the number 83 refers to the number of input variables and the number 9 refers to the number of hidden neurons. The 'weights.txt' and 'biases.txt' are outputs from Neuroet. The 'train_x.txt' and 'train_y.txt' represent normalized training data sets and 'out.txt' is the output file containing the normalized predictions. Note, that in this example the normalized predictions must be rounded to zero or one in order to compare them to the actual data.