CSG2341 Intelligent Systems

Workshop 5: Cross-validation testing of a neural network for regression

Related Objectives from the Unit outline:

- Identify appropriate intelligent system solutions for a range of computational intelligence tasks.
- Demonstrate the ability to apply computational intelligence techniques to a range of tasks normally considered to require computational intelligence.

Learning Outcomes:

After completing this workshop, you should be able to demonstrate an understanding of the resources required to implement a problem solution based on a multi-layer perceptron (MLP), to describe and analyse the steps in designing a neural network to solve a regression problem, and to use a computer package to develop such a classifier. You should also be able to use cross-validation to select suitable network parameters.

Background

As you learned from lectures, it is important to be able to estimate the true performance of a neural network that has been trained to solve classification or prediction (regression) problems. Last week, you trained MLPs to solve a classification problem, and use cross-validation to estimate the performance of the trained network. This week, you will do something similar with a regression problem.

As you know, the aim in a classification problem is to determine, from an input pattern, which out of a set of alternative types (classes) of instance the pattern represents. The performance figure that is relevant in this type of task is the expected % of correct classifications. We can use cross-validation testing to estimate this percentage.

For a regression problem, the aim is to predict some unknown numerical value, based on know numerical or categorical inputs. The performance figure that is relevant for this kind of task is the average error in the prediction. We can also use cross-validation testing to estimate this average error.

Using this estimate, you can then optimise the design of the network and the learning parameters, similar to what you did in last week's exercise.

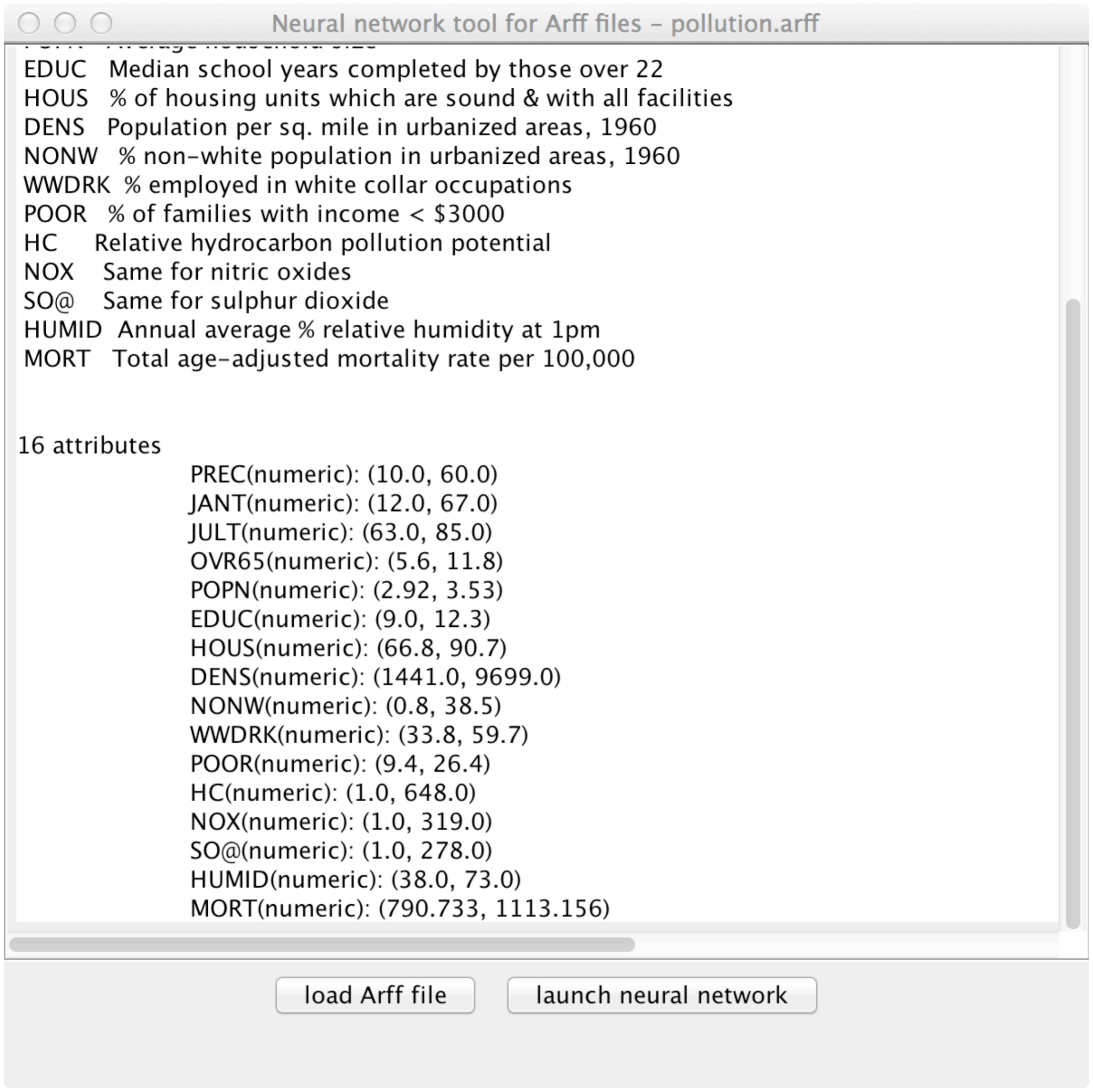
Task:

Step 1

On BlackBoard, you will find *NNArffToolDist.zip* (the same tool you used last week) and pollution.arff. Download these and unzip NNArffTool.

Step 2

Start NNArffTool and load in pollution.arff. See last week's instructions if you have forgotten how to do this. You should see a window like this:



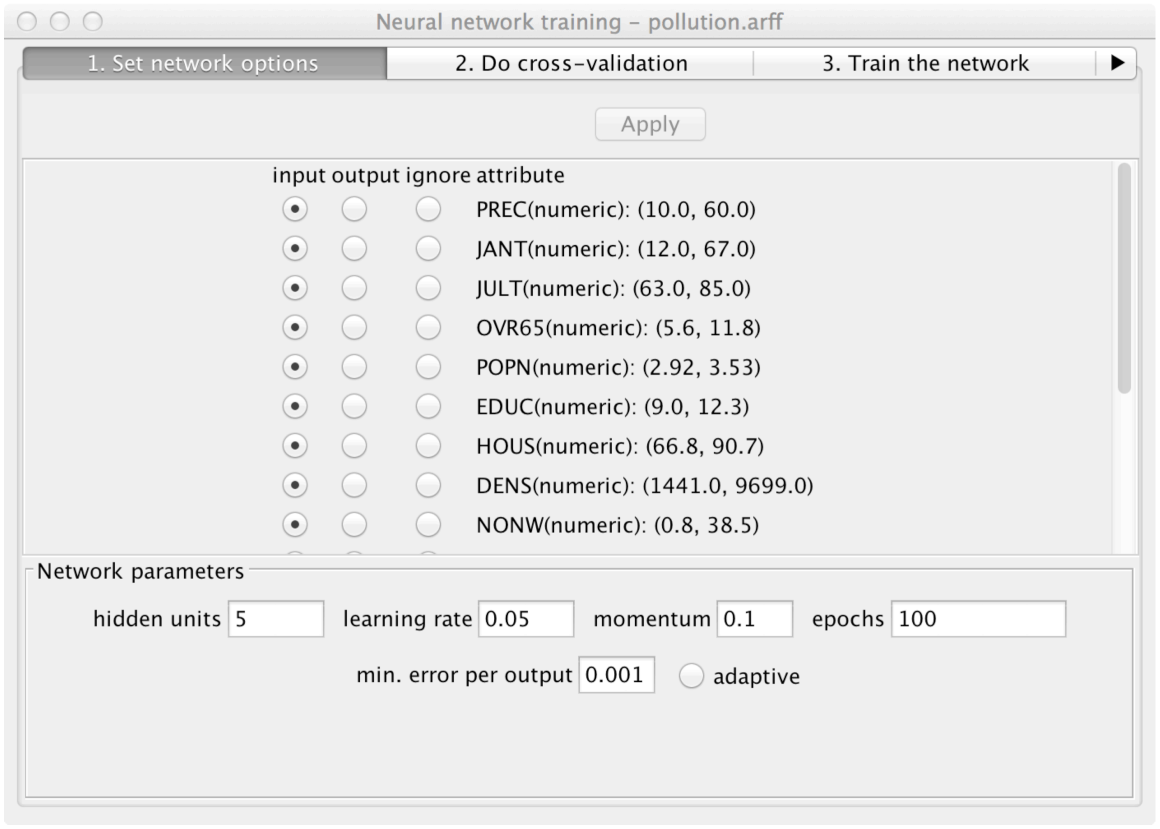
```
EDUC   Median school years completed by those over 22
HOUS   % of housing units which are sound & with all facilities
DENS   Population per sq. mile in urbanized areas, 1960
NONW   % non-white population in urbanized areas, 1960
WWDRK  % employed in white collar occupations
POOR   % of families with income < $3000
HC     Relative hydrocarbon pollution potential
NOX    Same for nitric oxides
SO@    Same for sulphur dioxide
HUMID  Annual average % relative humidity at 1pm
MORT   Total age-adjusted mortality rate per 100,000


16 attributes
                PREC(numeric): (10.0, 60.0)
                JANT(numeric): (12.0, 67.0)
                JULT(numeric): (63.0, 85.0)
                OVR65(numeric): (5.6, 11.8)
                POPN(numeric): (2.92, 3.53)
                EDUC(numeric): (9.0, 12.3)
                HOUS(numeric): (66.8, 90.7)
                DENS(numeric): (1441.0, 9699.0)
                NONW(numeric): (0.8, 38.5)
                WWDRK(numeric): (33.8, 59.7)
                POOR(numeric): (9.4, 26.4)
                HC(numeric): (1.0, 648.0)
                NOX(numeric): (1.0, 319.0)
                SO@(numeric): (1.0, 278.0)
                HUMID(numeric): (38.0, 73.0)
                MORT(numeric): (790.733, 1113.156)
```

There are 60 instances in this dataset. There are 16 attributes. The last attribute, MORT is the thing we are trying to predict – mortality rate – based on demographic data plus pollution data.

Step 3

Click on “launch neural network”. A new window will open that looks like this:



input	output	ignore	attribute
●	○	○	PREC(numeric): (10.0, 60.0)
●	○	○	JANT(numeric): (12.0, 67.0)
●	○	○	JULT(numeric): (63.0, 85.0)
●	○	○	OVR65(numeric): (5.6, 11.8)
●	○	○	POPN(numeric): (2.92, 3.53)
●	○	○	EDUC(numeric): (9.0, 12.3)
●	○	○	HOUS(numeric): (66.8, 90.7)
●	○	○	DENS(numeric): (1441.0, 9699.0)
●	○	○	NONW(numeric): (0.8, 38.5)

Network parameters

hidden units 5 learning rate 0.05 momentum 0.1 epochs 100

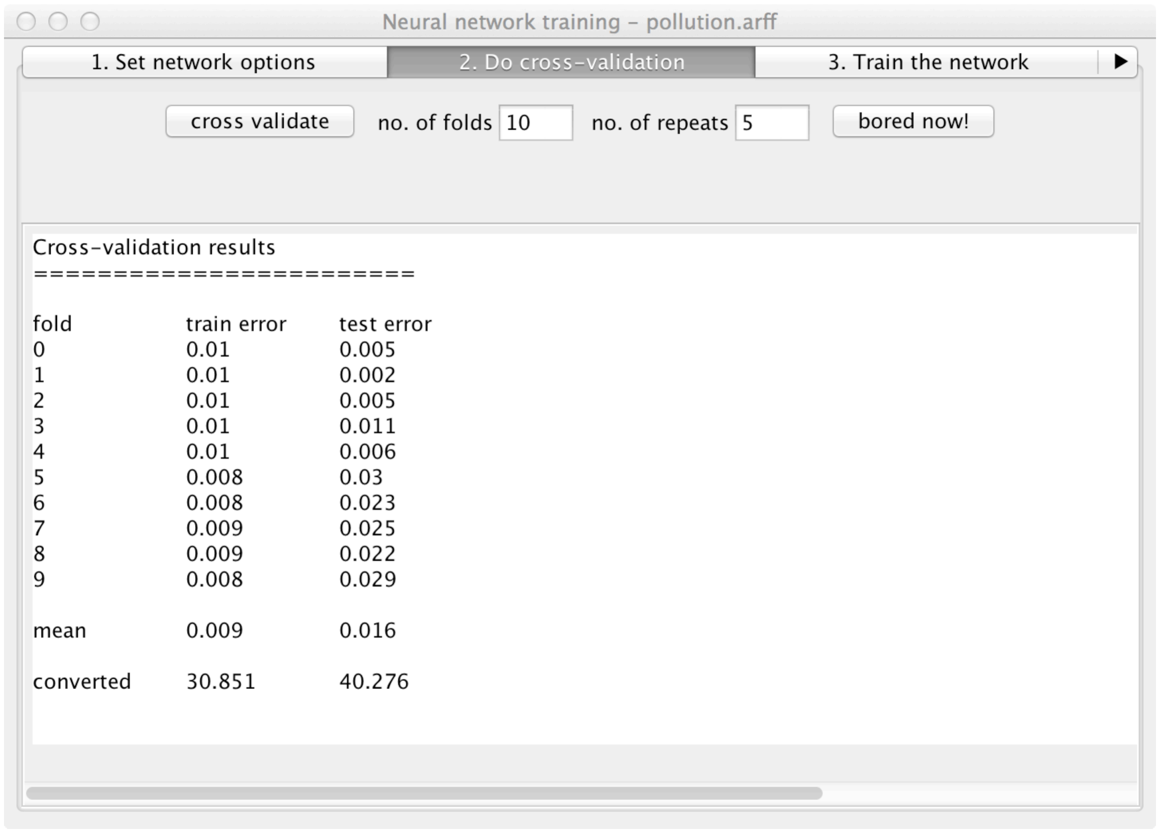
min. error per output 0.001 ○ adaptive

As in last week’s exercise, an MLP has been created for you with the right number of inputs and outputs, and the data has been converted into a form suitable for use with this network.

Before you look, with View the trained network, can you figure out how many input units and how many output units this network will have?

Step 4

Set the epochs to 1000, and then click on the “Do cross-validation” tab and then on the “cross validate” button. This will take a few minutes – just wait. You should see something like:



no. of folds 10   no. of repeats 5

```
Cross-validation results
========================

fold        train error    test error
0           0.01           0.005
1           0.01           0.002
2           0.01           0.005
3           0.01           0.011
4           0.01           0.006
5           0.008          0.03
6           0.008          0.023
7           0.009          0.025
8           0.009          0.022
9           0.008          0.029

mean        0.009          0.016

converted   30.851         40.276
```

This is similar to last week, but this time it reports on mean errors instead of predication rates. Look at the "converted" row. The last two figures tell us that, over a number of trials, the average error (actually the square root of mean-squared error) on training data was about 30.851 and on **test data** (the important figure) a bit worse at about 40.276 (the values for MORT are around 1000, so this seems like a reasonably small error – around 1 part in 25).

Step 5

Now we come to your bit! Your task is to use cross-validation testing to find the smallest subset of input attributes that you can, which is still at least gives a converted mean error on the test data of less than 41.

This could perhaps be a way to get an idea of which factors are most important in determining mortality – are the pollution factors important? Or is it more or less determined by the demographics? Note that this kind of procedure doesn't prove anything conclusively – for example, it might be that both demographics and pollution data are important, and that you could predict mortality from either by itself.

To do this, use the "Set network options" tab to "ignore" some attributes – YOU HAVE TO CLICK ON "Apply" to give effect to the changes you make.

Two possible strategies are

1. Train the network, using the Train the network tab, and then View the network using the View the trained network tab, and look for attributes that

have “weak” weights. These might be ones that could be left out. See if the remaining attributes work well but using the Do cross-validation tab. Repeat the process with different sets of attributes; or

2. Start with just a few attributes with “strong” weights and add new ones until you get to less than 41 for the cross-validated mean error.

Suggestions: To save time, you might set the number of folds to 5 and the number of repeats to 3 in the cross-validation panel (this is not so accurate, but a lot faster). You could also try setting the learning rate fairly high – say 0.25.

When you have completed this workshop, submit your findings to your tutor using the submission facility on BlackBoard.