Course on Artificial Intelligence and Intelligent Systems

Example and exercise using an Excel based Neural Network package

Henning Christiansen
Roskilde University, Computer Science Dept.

1 Introduction

This notes explains, by means of an example, how to use an Excel-based system which simulates artificial neural nets.

The system is developed by Angshuman Saha and it is available at the following internet address

`http://www.geocities.com/adotsaha/NNinExcel.html`

In fact there are two systems, one for classification and one for prediction; here we show an example using the one for classification, and the prediction system seems to work pretty much in a similar way. You simply download a file, and if you got a recent version of Excel installed, you double-click, and it starts. You need, though, to click OK to allow macros. In brief, the system has the following features.

- You can define your own feed-forward, back-propagation networks with 1 or 2 hidden layers.
- You can enter training data, set various learning parameters, start a learning phase and see the results (error rate, etc.) presented in various ways, including graphically.
- You can afterwards type in specific input data for classification.

All what is needed to know in order to use the system is described in the first sheet `ReadMe`, and in this documents I describe only a little example developed in the system, so that you can get started with your own exercise.

The only thing that the system does not help you with is the preparation and entering of data, but I have based this example on data generated using Excel (Excel has a random number generator).

The present note is not intended to be read separately, and you are should have had an introduction to neural nets, and for your own applications, you should definitely study the description provided by Saha's system.

2 The example

Training data need to be produced by the user and entered manually into the Excel sheet, which means that it is not easy to enter, say, digitized pictures, although this should in principle be possible.

In the following we consider a small example which has the nice property that training data, including classifications, can be produced automatically.

We consider here samples of colours represented as triplets of red-green-blue (RGB) values, each being a decimal number between 0 and 1.

Such triplets are considered input data to the neural net, and output is a classification of which of the basic colours is dominant. This is defined simply by which of them that has the greatest value. We use letters `r`, `g`, `b` for classification, and the following table shows two samples together with their classification.

Red	Green	Blue	Dominant
0.8	0.4	0.5	`r`
0.7	0.8	0.9	`b`

So our task is to produce a set of such data, get them into Saha's system, and to train and test a network.

The networks will be successful if it has been able to learn the relatively simple mathematical function that determines the classification above. More precisely, that the function stored in the net after training resembles the "true" classification function.

3 Setting up the network type plus parameters for learning

We assume that we have downloaded a copy of the file `NNClass.xls` and that we have made a copy of that file which we call `NNClassColours.xls`. We open this file and click `UserInput` at the bottom of the window.

We need to set `Number of Inputs` which for our example is `3`. We will in this experiment use a very simple network to see how well it performs (then we can always make it more complicated if needed). Thus we select for `Number of Hidden Layers`, the option `1`; notice that when you click this field, a little menu pops up.

We need also specify the number of nodes in the hidden layer which we can do in the field `Hidden Layer sizes` where we select `2` — thus a very simple network.

We need also inform the system here about how large training data we plan to use. We stay with the default value `150`. We would like to work with both training and validation data, so we find `Training / Validation Set` in the sheet, press the neighbouring field wich initially reads `Use whole data as training set` and chose the other option `Partition data into Training / Validation set`.

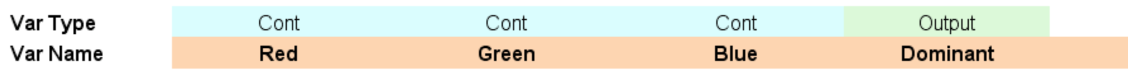
To specify that we want to use 100 samples as training data and 50 for validation, we chose option `2` and insert the number `50` in the relevant field in the bottom of this sheet.

There are a lot of other parameters that we just leave as they are.

Notice the big button labeled `Build Model`. Do not press it now, because you need to fill in data first.

4 Setting up variables

To get to the sheet where the data are to be entered, click `Data` at the bottom of the window. As you notice there are some sample data in the sheet already that are not relevant for our experiment, so let us select a sufficiently lange square with upper left corner `AC105` and use the `Edit` → `Clear` command to get rid of those data. Next we change the types and names of variables according to our own experiment. We insert the following:



Var Type	Cont	Cont	Cont	Output
Var Name	**Red**	**Green**	**Blue**	**Dominant**

The type `Cont` indicates that the variable ranges over real numbers, and `Output` indicates two things, namely that it is the output variable and, secondly, that is must be *categorical*, which means that it ranges over a set of names, which fits our classification into one of `r`, `g`, `b` which we refer to the variable `Dominant`.

Now we could in principle type our training data into the next 150 rows in the sheet, amounting to some 600 fields. However, we'll do something smarter...

5 Preparing and entering data

For this experiment, we'll generate artificial data with the value of the output variable determined by a mathematical function. One way of doing is this using Excel, and that is what we'll do here.

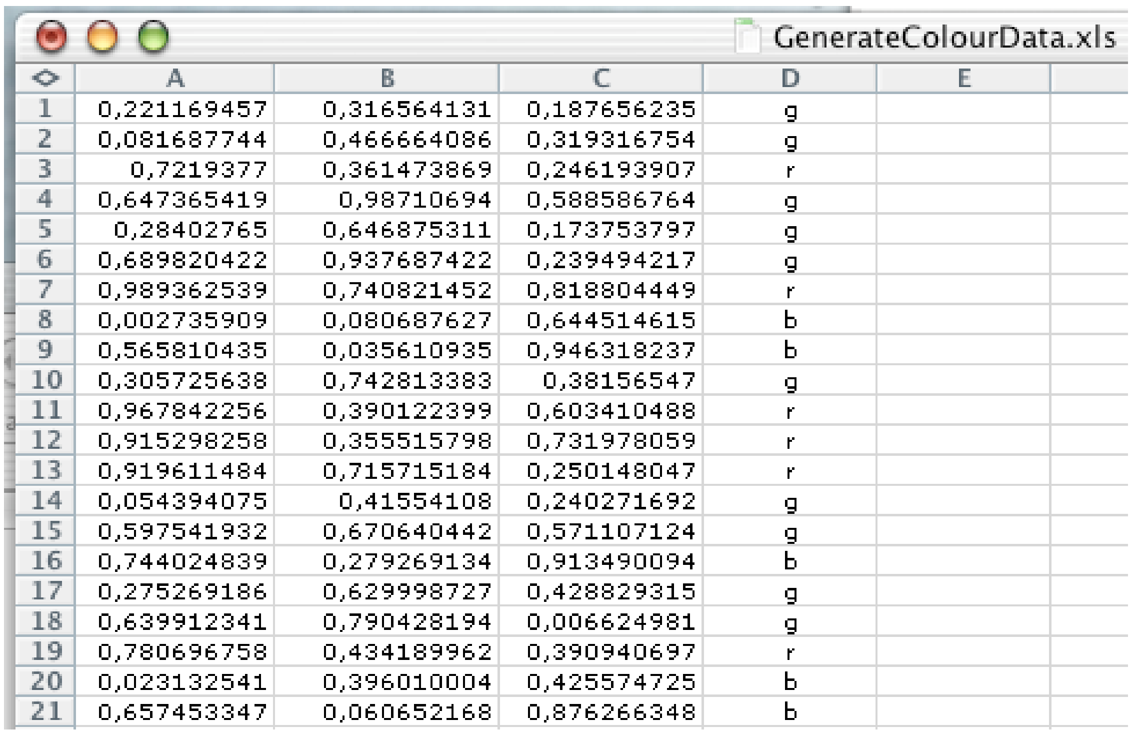
So we start a new Excel sheet, which we call `GenerateColourData.xls`, and enter the formula `=RAND()` in each of `A1`, `B1`, `C1`. This function will, when evaluated, return a random value between `0` and `1`. In order to calculate the output variable, i.e., the classification of which colour is dominant, we enter the following formula in `D1`:

```
=IF(AND(A1>B1,A1>C1),"r",IF(B1>C1,"g","b"))
```

This may result in the following picture:

<>	A	B	C	D	···
1	0.186224665	0.123761615	0.674147389	b	
2					
⋮					

Finally, let us copy row one and paste it into rows 2–150, and we have our data visible in front of us.



	A	B	C	D	E
1	0,221169457	0,316564131	0,187656235	g	
2	0,081687744	0,466664086	0,319316754	g	
3	0,7219377	0,361473869	0,246193907	r	
4	0,647365419	0,98710694	0,588586764	g	
5	0,28402765	0,646875311	0,173753797	g	
6	0,689820422	0,937687422	0,239494217	g	
7	0,989362539	0,740821452	0,818804449	r	
8	0,002735909	0,080687627	0,644514615	b	
9	0,565810435	0,035610935	0,946318237	b	
10	0,305725638	0,742813383	0,38156547	g	
11	0,967842256	0,390122399	0,603410488	r	
12	0,915298258	0,355515798	0,731978059	r	
13	0,919611484	0,715715184	0,250148047	r	
14	0,054394075	0,41554108	0,240271692	g	
15	0,597541932	0,670640442	0,571107124	g	
16	0,744024839	0,279269134	0,913490094	b	
17	0,275269186	0,629998727	0,428829315	g	
18	0,639912341	0,790428194	0,006624981	g	
19	0,780696758	0,434189962	0,390940697	r	
20	0,023132541	0,396010004	0,425574725	b	
21	0,657453347	0,060652168	0,876266348	b	

To get these data into our neural net sheet, `NNClassColours.xls`, we should copy the rows 1–150 of `GenerateColourData.xls`, and paste them into the `Data` sheet of `NNClassColours.xls`, more precisely a square whose upper left corner is `AC105`.

Copying data is difficult using MS products, and attempts to do this using Excel's standard commands such as `Paste Special` may occasionally lead to strange behaviour. So we stick to the classical technique that solves many such anomalies: Copy the data, and paste them into a window of a flat text editor and then copy the date once again from there.

In more details, we succeeded in getting the date over in the following way (it may work different under different versions of Excel and different operating systems):

1. Select the cells from `A1` to `D150` of `GenerateColourData.xls` and do `Copy`.
2. Open a flat text editor (we used `BBedit` under MacOs), and do `Paste`.
3. Do `Select All`.
4. Select cell `AC105` of `NNClassColours.xls` and do `Paste`.

Now you may be lucky that you have pasted a set of training data into your neural network machine `NNClassColours.xls`.

6 Training the neural network

To start the training of the net with the chosen parameters and training data you should first close the `GenerateColourData.xls` window (perhaps saving it). If you forget this step, very strange things may occasionally happen (there is only explanation to this: MS).

Now go to the `UserInput` sheet of `NNClassColours.xls`, and press the big button entitled `Build Model`. You can follow what happens in the bottom bar of the window, and when the learning phase has finished, you can see a nice presentation of the results in the sheet `Output`. The `Calc` sheet allows you to enter new data and have them classified.

Now, back to our particular example. In the `Output` sheet we observe that the net and the settings tend to have a fixed error rate of 10%. However, as we have used only one hidden layer and with only two nodes, this may not be a surprise.

The next thing to do, is then to experiment with other nets, making them more advanced, step by step, and see how much is needed to achieve better results.

7 Exercises

If you can invent another example for which the training data can be generated in a similar easy way, you are most welcome to work with that instead and modify the questions below accordingly. Otherwise continue with the example we have shown so far.

7.1

Redo from scratch the example explained above, so that you have your own Excel sheet ready for experimentation.

7.2

Add gradually more nodes in the hidden layer and/or try with one more layer as to find the smallest net that gives acceptable results.

7.3

For the solution that you found in the previous question, experiment to find out what is the optimal sizes of training and validation sets. (Optimal may mean as small as possible and that increasing the sizes does not give much improvements).

7.4

Will different values of other parameters influence the performance, and how?

7.5

This question is about *overfitting.* Explain to yourselves what this words means and try to organize you training data and parameters such that your trained model will exhibit this (undesired phenomenon).

7.6

Try to make the classification problem more complicated by extending the set of possible classifications, e.g., by saying that

- The dominant colour is `black` if sum of RGB is less that 0.3. (NB: This will most likely make larger training sets relevant).
- The dominant colour is `white` of sum of RGB is close to 2.7.
- Define in similar ways what it means for the dominating colours to be either `cyan`, `magenta`, or `yellow`.