M345S17 QUANTITATIVE METHODS IN RETAIL FINANCE 2019 COURSEWORK 1

Fraud Model using Artificial Neural Networks

In this exercise you will download a data set of fraudulent credit card transactions and build an artificial neural network (ANN) for prediction. You will use the R statistical language for your work.

Tony Bellotti

TIMESCALE

Your coursework must be submitted by 4pm on Thursday 31st January 2019.

INSTRUCTIONS

1. Download the R workspace file from Blackboard. This file contains a single data frame `D2` of fraud data. Please see Appendix A for details.

2. Produce summary statistics and/or histograms for the variables `Time`, `Amount` and `Class`. What do these empirical distributions tell you about the data?

3. Randomly split the data frame into a training and test data set with a 2:1 ratio of sample size, but ensure that approximately the same ratio of frauds to non-frauds is in the training and test set.
   Hint: you can use the `sample` function.

   Save your training and test data set – you will need them in CW2.

4. Use an ANN with a single hidden layer to model fraud with `Time`, `Amount` and `V1` to `V28` as predictor variables. Train only on the training data set.
   See Appendix B for information about the `nnet(nnet)` function you can use to implement the ANN.

5. Construct a precision-recall (PR) curve and compute the area under the precision-recall curve (AUPRC), when applying the ANN to the test data set.
   See Appendix B for information about the `pr.curve(PRROC)` function you can use.

6. Repeat steps 4 & 5 for different numbers of hidden neurons. Report AUPRC in each case, but you only need to show the PR curve for one ANN in your report.

7. Which is the best number of hidden neurons to use for this problem? Explain why. In this case, if an alarm rate of no more than 0.5% is required, what is the maximum recall that can be achieved?

8. Your coursework must be submitted as:
   a) a paper copy of your solutions to the student office and
   b) an electronic submission, along with an R script giving the commands you used to complete the coursework, emailed to a.bellotti@imperial.ac.uk with subject heading "M345S17 CW1". Your R script should include annotated comments describing what it is doing at each step.

Remember to include all results and the R code you used in your report.

APPENDIX A: FRAUD DATA

This data is taken from an anonymous UK credit card provider. Each observation refers to a credit card transaction and includes variables as listed in the table below.

The file `CCfraud.RData` consists of a single data frame `D2` of 50247 credit card transactions. It can be loaded into R using the R `load` function; eg I used

```
load("C:/Users/abellott/CCfraud.RData")
```

VARIABLE	DESCRIPTION
`Time`	Time (in seconds) at which the transaction took place.
`Amount`	The value of the transaction (denomination unknown).
`V1` to `V28`	All other predictor variables have been anonymized and transformed into these 28 predictor variables.
`Class`	Default status: • 0 if not fraud. • 1 if this is a fraudulent transaction.

Note that this data set has been manipulated so that a large proportion of non-fraudulent transactions have been randomly removed.

APPENDIX B: HELP IN R

You may use any package and functions to complete this coursework. However, some guidance is given in this appendix.

Artificial Neural Networks

You can use the `nnet` function in the `nnet` package. You will need to first install this package (look under the Tools menu in RStudio), then load it using

```
library(nnet)
```

For neural networks, it is advisable to *standardize* variables in the input layer so they have approximately the same scale. There are several ways to do this but for this project, rescale your data so that all variables have values between 0 and 1. You can use the `scale` function to do this.

The ANN can then be called with

```
nn1 <- nnet(formula, data=TrainData, size=n, maxit=500)
```

where `size` is the number of hidden nodes and `maxit` gives the maximum number of iterations of the gradient descent algorithm.

Warning! By default `nnet` starts off with random weights between neurons and it has a tendency to converge to local minima. You will need to find a way to handle this problem so that you can be sure you have an optimal or near-optimal solution.

Once an ANN is built, the predict function can be used:

```
preds <- predict(nn1, newdata=TestData)
```

The precision-recall curve

You can use the `pr.curve` function in the `PRROC` package. You will need to first install this package (look under the Tools menu in RStudio), then load it using

```
library(PRROC)
```

Then construct the curve and get AUPRC using:-

```
pr <- pr.curve(scores.class0 = preds[TestData$Class==1],
scores.class1 = preds[TestData$Class==0], curve = T)
plot(pr)
```

ensuring that *`TestData`* is the same data frame that was used to compute the predictions in *`preds`*. The object `pr` contains a matrix of recall and precision values used to plot the PR curve (see `help(pr.curve)` for details).