ACTL4305/5305: Week 9 Lab - Neural Network (Questions)

Week 9

Learning Objectives

- Training neural networks in R using various packages, for both classification and regression tasks.
- Enhancing neural network performance through tuning: comprehending the associated tuning parameters and employing techniques such as early stopping, dropout, or weight regularization.

1 Neural Network

This week covers two case studies: one focused on a regression problem and the other on a classification problem. We will introduce you to two packages for implementing neural networks:

- `neuralnet()`: This package is considered classical and is particularly well-suited for beginners due to its simplicity and user-friendliness. (from `neuralnet` package)
- `keras` package: Originally developed in Python, the R implementation of the `keras` package provides a user-friendly interface for designing and training neural networks, along with extensive customization options.

Additionally, there are other valuable packages for working with neural networks, including `h2o` and `caret`.

2 A Regression Problem

2.1 Data Manipulation

The data we use is a subset of `freMTPL2freq` from `CASdatasets`.

In the dataset `freMTPL2freq` risk features and claim numbers were collected for 677991 motor third-part liability policies (observed on a year). We only consider a subset of `freMTPL2freq` including 40000 observations for training and 10000 observations for testing. Our task is to predict the number of claims.

`freMTPL2freq` contains 11 columns (with `IDpol`):

- `IDpol` The policy ID (used to link with the claims dataset).
- `ClaimNb` Number of claims during the exposure period.
- `Exposure` The exposure period.
- `Area` The area code.
- `VehPower` The power of the car (ordered categorical).

- `VehAge` The vehicle age, in years.
- `DrivAge` The driver age, in years (in France, people can drive a car at 18).
- `BonusMalus` Bonus/malus, between 50 and 350: 100 means malus in France.
- `VehBrand` The car brand (unknown categories).
- `VehGas` The car gas, Diesel or regular.
- `Density` The density of inhabitants (number of inhabitants per km2) in the city the driver of the car lives in.
- `Region` The policy regions in France (based on a standard French classification)

2.1.1 Import Data

```r
library(neuralnet) #neural network (slow)
library(tidyverse)
library(ROCR) #AUC plot
library(pdp)
```

```r
load("Train-set.RData")
load("Test-set.RData")

traindata<-newtrain
testdata<-newtest
#str(traindata)
```

2.1.2 Task: Data normalization for numeric variables

One of the most important procedures when forming a neural network is data normalization. This involves adjusting the data to a common scale so as to accurately compare predicted and actual values. Failure to normalize the data will typically result in the prediction value remaining the same across all observations, regardless of the input values.

We can do this in two ways in R:

- Scale the data frame automatically using the scale function in R.
- Transform the data using a max-min normalization technique.

We implement both techniques below but choose to use the **max-min normalization** technique.

2.1.3 Task: Dummy coding for categorical variables

For categorical variables, we apply dummy coding.

2.2 Task: Training a Neural Network Model

To train a neural network, we use `neuralnet` function.

Notes:

- We use `neuralnet` to 'regress' the dependent `ClaimNb` variable against the other independent variables. Here we should consider feature selection problem.
- Setting the number of hidden layers to (2,2) based on the `hidden=(2,2)` formula.
- The `linear.output` variable is set to `TRUE` for regression problem. For classification problem, set it to be `FALSE`. If it is `FALSE`, then you can set the **activation function** by `act.fct`. The activation function can be 'logistic' for the logistic function and 'tanh' for tangent hyperbolicus.
- `err.fct` defines a differentiable function that is used for the calculation of the error. 'sse' and 'ce' which stand for the sum of squared errors and the cross-entropy can be used.
- The `threshold` is set to 0.05, meaning that if the change in error during an iteration is less than 5%, then no further optimization will be carried out by the model (`stepmax` is another stopping criteria).
- There are several types of `algorithm` you can use. e.g. 'backprop' refers to backpropagation, 'rprop+' and 'rprop-' refer to the resilient backpropagation with and without weight backtracking,
- Deciding on the number of hidden layers in a neural network is not an exact science. In fact, there are instances where accuracy will likely be higher without any hidden layers. Therefore, trial and error plays a significant role in this process. One possibility is to compare how the accuracy of the predictions change as we modify the number of hidden layers.

2.3 Task: Testing The Accuracy Of The Model

As already mentioned, our neural network has been created using the training data. We then compare this to the test data to gauge the prediction of the neural network. Note for neural network, we need to back-scale the predicted values to **its original scale**!

3 A Classification Problem (Credit Risk Modeling)

Now, we use the dataset introduced in Week 3 Lab, so we should change the response variable to a categorical variable. In this section, we will focus on training our neural networks using the `keras` package. Known for its flexibility and user-friendliness, Keras provides an extensive range of neural network architectures and customization options.

```r
library(keras)
library(tensorflow)

load("train_credit.RData") #70%
load("test_credit.RData") #30%

num_var<-c(1,5,12:23)

train_data_label <- as.numeric(xtrain0$default)-1

train_feature <- xtrain0[, -24]
test_feature <- xtest0[, -24]
```

3.0.1 Task: Training a Neural Network Model

You can refer to the `UC Business Analytics R Programming Guide` for references on R implementations of `keras` pacakge. To get started, we recommend exploring a simple architecture as a starting point.