Neural Networks in RStudio with the H2O Module

Raymond Fraizer
California State University, Sacramento
fraizerr@ecs.csus.edu

Abstract

The purpose of this project is to learn how to train and use an artificial neural network model in RStudio. I have chosen a dataset provided by a research team at the Pontifical Catholic University of Rio de Janeiro, Informatics Department. Ugulino et al. published their work in 2012. In their work on human activity recognition they provided a public dataset of 165633 samples from 4 three axis accelerometers worn by subjects doing 5 activities. For their work they used C4.5 with AdaBoost using 10 boosting iterations and 10-fold cross validation with which they achieved a 99.4% accuracy. In my work I tried a several different neural network configurations with one or two hidden layers, varying the number of nodes in the layers, and comparing my results with Ugulino’s team.

Keywords: Human Activity Recognition, Accelerometer, Machine Learning, Artificial Neural Network

1. Introduction

In order to learn how to build and train an artificial neural network in RStudio I first needed to install a package that could implement neural networks. After some searching online I was aware of the nnet, neuralnet, and H2O packages. From looking at the web site for the H2O software it seemed very flexible, and since we had an in class demo that used it, I chose to use H2O. Initially I thought to just try making a few networks with different numbers of nodes and layers expecting to see a trend to guide me to a better network size. But it soon became clear that would not work. So I looked into doing random grid searches for the network size as well as other configuration options like the activation unit to use with or without dropout, etc.

2. Data background

The dataset I choose is from a paper titled "Wearable Computing: Accelerometers’ Data Classification of Body Postures and Movements" by Ugulino et al. at the Pontifical Catholic University of Rio de Janeiro, Informatics Department[1]. This is a set of accelerometer data that has been cleaned up and provided as a benchmark set. The paper deals with identifying activities of test subjects that are wearing a set of accelerometers while performing certain tasks. There were five activities their model was to classify. In the paper the researchers used C4.5 with AdaBoost using 10 boosting iterations and 10-fold cross validation with which they achieved a 99.4% accuracy.

In my work I will try a few different neural network configurations, one or two hidden layers, varying the number of nodes in the layers, and compare my results with Ugulino’s team.

The dataset was downloaded from http://groupware.les.inf.puc-rio.br/har on 03/13/2017. The data is sorted, and one of the records was missing a field separator and still had a time stamp that was removed from all other data records. I added the missing separator and removed the time stamp on that record. I also randomly shuffled the records and removed the columns I would not be using. The data represents a classification problem with 5 classes. There are a total of 165633 records.

3. Related work

There have been other works using accelerometers either individually or in sets to monitor the motion of test subjects either while performing everyday activities or while performing scripted activities. The set of activities varies. The data collected usually goes through some kind of filtering and feature selection before being used as input to a classification model. A variety of software and machine learning algorithms have been tried as well.

For example de Vries et al.[2] used two accelerometers, one attached at the hip and one at the ankle. They used SPSS and R with the nnet module to preprocess the data and build three neural network models one for each location and the last using data from both locations. The hip model achieved 80.4% accuracy, the ankle model achieved 77.7% accuracy, while the combined data model achieved 83.0% accuracy.

Lubina et al.[3] used five accelerometers, two at the ankles, two at the waist(left and right sides), and one at the upper back. They had seven activities including going up and down stairs. They got accuracies of 97% for discriminating sitting and walking, 89% for standing,
72-75% for walking the stairs, and 56% and 38% respectively for standing up and sitting down.

Zhang et al.[4] used three publicly available data sets, “Opportunity”, “USC-HAD”, and “Daily and Sports Activities” rather then collect data themselves. They used WEKA to implement a deep neural network using the “deep belief network” code using the accelerator data directly without doing feature selection first. Their error rates for the three datasets respectively were, 0.177, 0.083, and 0.094.

Zebin et al.[5] also used a deep neural network on data that was not preprocessed for feature selection. However they used a distributed generalized linear modeling,... Naive Bayes, principal components analysis, k-means clustering,... distributed random forest, gradient boosting, and deep learning. As noted in the summary of features in “Deep Learning with H2O”[10], most of the data preprocessing is done automatically, like scaling, factorizing categorical features, and imputation of missing values. So normally you just load the dataset from the file, split it into training and test sets, possibly with a validation set as well, then start building a model.

5. How H2O was used

At first I was using RStudio to prepare the data for building a model and only using H2O to actually build and train the model, and run the prediction. But this was cumbersome having to convert between H2O objects and R data frames, and I had difficulty getting a confusion matrix for the categorical dataset I had. I tried the Web interface, H2O Flow. This was easy to use and showed me that the data was being processed alright when it was all being handled interactively through the web. After looking at the examples that came with H2O’s R package, I saw that they were using R strictly as a scripting interface to initialize, control, and shutdown the H2O instance. No data was being processed in R at all. Since adopting this approach I have made some individual models, but there are too many settings and a wide range of values to try. So I started to use random grid searches to find good configurations. The grid search also has a lot of parameters so the H2O package documentation[11] was very helpful in getting the grid search setup, as well as tuning individual neural networks. With smaller networks and fewer than 100 models per run, on my workstation a grid search took 45-75 minutes. But with large networks even limited to 100 models per run, a grid search took as much as 5 hours per run. So there were few runs with networks of a few hundred nodes. Also due to the time limitation I had not tried more than two hidden layers yet.

6. Conclusion

Small networks, 30 nodes or less, did not do as well with accuracies ranging from 92.4% to 94.4%. The 45-60 node networks all did better than 98% accuracy, but there was some over fitting. The large 400+ node networks that did not use dropout exceeded 99% accuracy, but with a lot of over fitting. Still these accuracies are for the test set, not the training set accuracies. As for the activation unit, without dropout Tanh was best and Rectifier the worst. However, with dropout it is reversed, Rectifier is the best and Tanh the worst. Maxout consistently scored in the middle regardless of dropout. Both the 11, 12 settings and the max_w2 setting, which deal with over fitting and rectifier stability respectively, seemed to have no effect.

Even without using an ensemble the neural network came close to the accuracy of the original work’s 99.4% accuracy for a decision tree with 10 iterations of AdaBoost and 10-fold validation.
7. References


