Implementing Autoencoders using H2O

By **Amey Varangaonkar** - October 27, 2017 - 12:00 am

3 min read

Note

*This excerpt is taken from the book **Neural Networks with R**, Chapter 7, Use Cases of Neural Networks – Advanced Topics, written by **Giuseppe Ciaburro** and **Balaji Venkateswaran**. In this article, we see how R is an effective tool for neural network* modelling, *by implementing autoencoders using the popular H2O library.*

An autoencoder is an ANN used for learning without efficient coding control. The purpose of an autoencoder is to learn coding for a set of data, typically to reduce dimensionality. Architecturally, the simplest form of autoencoder is an advanced and non-recurring neural network very similar to the MLP, with an input level, an output layer, and one or more hidden layers that connect them, but with the layer outputs having the same number of input level nodes for rebuilding their inputs.

In this section, we present an example of implementing Autoencoders using H2O on a movie dataset.

> The dataset used in this example is a set of movies and genre taken from https://grouplens.org/datasets/movielens

We use the **movies.csv** file, which has three columns:

- **movieId**
- **title**
- **genres**

There are 164,979 rows of data for clustering. We will use `h2o.deeplearning` to have the **autoencoder** parameter fix the clusters. The objective of the exercise is to cluster the movies based on genre, which can then be used to recommend similar movies or same genre movies to the users. The program uses `h20.deeplearning`, with the autoencoder parameter set to T:

```
library("h2o")

setwd ("c://R")
#Load the training dataset of movies
movies=read.csv ( "movies.csv", header=TRUE)
head(movies)

model=h2o.deeplearning(2:3,
                  training_frame=as.h2o(movies),
                  hidden=c(2),
                  autoencoder = T,
                  activation="Tanh")
summary(model)
```

```
features=h2o.deepfeatures(model,
                  as.h2o(movies),
                  layer=1)

d=as.matrix(features[1:10,])
labels=as.vector(movies[1:10,2])
plot(d,pch=17)
text(d,labels,pos=3)
```

Now, let's go through the code:

```
library("h2o")
setwd ("c://R")
```

These commands load the library in the R environment and set the working directory where we will have inserted the dataset for the next reading. Then we load the data:

```
movies=read.csv( "movies.csv", header=TRUE)
```

To visualize the type of data contained in the dataset, we analyze a preview of one of these variables:

```
head(movies)
```

The following figure shows the first 20 rows of the movie dataset:

Now we build and train `model`:

```
model=h2o.deeplearning(2:3,
                  training_frame=as.h2o(movies),
                  hidden=c(2),
                  autoencoder = T,
                  activation="Tanh")
```

Let's analyze some of the information contained in `model`:

```
summary(model)
```

This is an extract from the results of the `summary()` function:

In the next command, we use the `h2o.deepfeatures()` function to extract the nonlinear feature from an h2o dataset using an H2O deep learning model:

```
features=h2o.deepfeatures(model,
                  as.h2o(movies),
                  layer=1)
```

In the following code, the first six rows of the features extracted from the model are shown:

```
> features
DF.L1.C1 DF.L1.C2
1 0.2569208 -0.2837829
2 0.3437048 -0.2670669
3 0.2969089 -0.4235294
4 0.3214868 -0.3093819
5 0.5586608 0.5829145
6 0.2479671 -0.2757966
[9125 rows x 2 columns]
```

Finally, we plot a diagram where we want to see how the model grouped the movies through the results obtained from the analysis:

```
d=as.matrix(features[1:10,])
labels=as.vector(movies[1:10,2])
plot(d,pch=17)
text(d,labels,pos=3)
```

The plot of the movies, once clustering is done, is shown next. We have plotted only 100 movie titles due to space issues. We can see some movies being closely placed, meaning they are of the same genre. The titles are clustered based on distances between them, based on genre.

Given a large number of titles, the movie names cannot be distinguished, but what appears to be clear is that the model has grouped the movies into three distinct groups.

If you found this excerpt useful, make sure you check out the book Neural Networks with R, containing an interesting coverage of many such useful and insightful topics.

Amey Varangaonkar

Data Science Enthusiast. A massive science fiction and Manchester United fan. Loves to read, write and listen to music.