

rnn: Recurrent Neural Network architectures in native R

Bastiaan Quast Dimitri Fichou
The Graduate Institute University of Giessen

June 21, 2018

Abstract

The R package `rnn` implements several Recurrent Neural Network (RNN) architectures in the R language. The native R implementations of these architectures allow scientists familiar with the R language, to develop an intuitive understanding of these architectures, something which is not possible with production frameworks, such as TensorFlow, PyTorch or CNTK.

1 About package `rnn` in R

The `rnn` package is available on CRAN at <https://cran.r-project.org/package=rnn> and can be installed using¹:

```
install.packages('rnn')
```

After installation, the package can be loaded using:

```
library(rnn)
```

A list of all the exported functions can be printed using:

```
ls('package:rnn')  
## [1] "bin2int"          "epoch_annealing"  "epoch_print"  
## [4] "int2bin"          "loss_L1"          "predictr"  
## [7] "run.finance_demo" "run.rnn_demo"     "trainr"
```

A list of all the functions - including non-exported ones - can be printed using:

¹The development version can be installed using `devtools::install_github('bquast/rnn')`

```
ls(getNamespace('rnn'), all.names=TRUE)

## [1] ".__NAMESPACE__."      ".__S3MethodsTable__." ".packageName"
## [4] "b2i"                  "backprop_gru"         "backprop_lstm"
## [7] "backprop_r"           "backprop_rnn"         "bin2int"
## [10] "clean_lstm"           "clean_r"              "clean_rnn"
## [13] "epoch_annealing"      "epoch_print"          "i2b"
## [16] "init_gru"             "init_lstm"            "init_r"
## [19] "init_rnn"             "int2bin"              "loss_L1"
## [22] "predict_gru"          "predict_lstm"         "predict_rnn"
## [25] "predictr"             "run.finance_demo"     "run.rnn_demo"
## [28] "trainr"               "update_adagrad"       "update_r"
## [31] "update_sgd"
```

The **rnn** has one dependency, the **sigmoid** package (Quast 2016), which is on CRAN at <https://cran.r-project.org/package=sigmoid>. The **sigmoid** package provides a collection of sigmoid functions such as the Rectified Linear Unit (**ReLU()**), **Gompertz()**, etc. Until version 0.8.0 of the **rnn** package, the sigmoid functions were included in the package, after which they were released as a separate package for more general use.

In addition to this, the **rnn** package includes a **Shiny** app demonstrating a Recurrent Neural Network analysis of a time series (Foreign Exchange rates). In order to run the app locally, the **Shiny** package needs to be installed.

2 trainr

The workhorse of the **rnn** package is the **trainr()** function, it trains a model based on input and output data, given the specified hyperparameters.

The documentation of the **trainr()** function can be called up using:

```
help('trainr')
```

Recurrent Neural Networks have the ability to learn bit-by-bit binary addition (including carrying over) with as little as 3 hidden nodes, whereas feed-forward neural networks would need many more.

First training data is generated, the training data is between 0-127, or an 8-bit binary.

```
set.seed(123) # for reproducible random numbers
X1 = sample(0:127, 50000, replace=TRUE)
X2 = sample(0:127, 50000, replace=TRUE)
```

The training data is used to generate the output data or labels.

```
Y <- X1 + X2
```

Following this, both the input data and the output data are converted into binary format, using the built-in `int2bin()` function.

```
X1 <- int2bin(X1, length=8)
X2 <- int2bin(X2, length=8)
Y <- int2bin(Y, length=8)
```

Finally, the two input variables are stored in a single 3 dimensional tensor. Where the first dimension contains observations, the second dimension time, and the third dimension variables.

```
X <- array( c(X1,X2), dim=c(dim(X1),2) )
```

The objects `X` and `Y` can now be fed to the `trainr()` function, which will output a trained model (stored here in the object called `m1`).

```
m1 <- trainr(Y=Y,
             X=X,
             learningrate = 1,
             hidden_dim   = 6 )

## Trained epoch: 1 - Learning rate: 1
## Epoch error: 0.194449702806897
```

`predictr`

Using a trained model to make predictions is done using the `predictr()` function.

Take the first two observation of the training data, the first input variable.

```
X[1:2,,1]

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    0    1    0    0    1    0    0
## [2,]    0    0    1    0    0    1    1    0
```

Observation 1 & 2; variable 1 in decimal format.

```
bin2int( X[1:2,,1] )

## [1] 36 100
```

Take the first two observation of the training data, the second input variable.

```
X[1:2,,2]

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    1    0    1    1    0    0    0
## [2,]    0    0    0    1    1    1    0    0
```

Observation 1 & 2; variable 2 in decimal format.

```
bin2int( X[1:2,,2] )

## [1] 27 56
```

Summing observation 1 of variable 1: 36, with observation 1 of variable 2: 119, gives 155.

Summing observation 2 of variable 1: 100, with observation 2 of variable 2: 75, gives 175.

Make predictions using the `predictr()` function.

```
round( predictr(model = m1,
                 X      = X[1:2,,]) )

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    1    1    1    1    1    0    0
## [2,]    0    0    1    1    1    0    0    1
```

Compared to the ground truth values.

```
Y[1:2,]

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    1    1    1    1    1    0    0
## [2,]    0    0    1    1    1    0    0    1
```

Or in decimal format.

```
bin2int(round( predictr(model = m1,
                       X      = X[1:2,,]) ) )

## [1] 63 156
```

Compared to the ground truth values.

```
bin2int( Y[1:2,] )

## [1] 63 156
```

3 Architectures

In addition to fully-connected Recurrent Neural Networks, `rnn` also supports Long Short-Term Memory (LSTM) Recurrent Neural Networks (Hochreiter and Schmidhuber 1997)

```
trainr(Y, X, network_type="lstm")
```

as well as Gated Recurrent Unit (Cho et al. 2014) architectures.

```
trainr(Y, X, network_type="gru")
```

Acknowledgments

Bastiaan Quast gratefully acknowledges support from the Swiss National Science Foundation under grant # (2013-2015). Dimitri Fichou gratefully acknowledges support ...

(Hebb 2005)(Linnainmaa 1970)(McCulloch and Pitts 1943)(Minsky 1952)(Rosenblatt 1958)(Rumelhart, Hinton, and Williams 1986)(Werbos 1974)(Widrow et al. 1960)

References

- Cho, Kyunghyun et al. (2014). “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078*.
- Hebb, Donald Olding (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.
- Linnainmaa, Seppo (1970). “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. In: *Master’s Thesis (in Finnish), Univ. Helsinki*, pp. 6–7.
- McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.
- Minsky, Marvin (1952). “A neural-analogue calculator based upon a probability model of reinforcement”. In: *Harvard University Psychological Laboratories, Cambridge, Massachusetts*.
- Quast, Bastiaan (2016). *sigmoid: Sigmoid Functions for Machine Learning*. R package version 0.3.0. URL: <https://cran.r-project.org/package=sigmoid>.
- Rosenblatt, Frank (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6, p. 386.

- Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, p. 533.
- Werbos, Paul (1974). "Beyond regression: new fools for prediction and analysis in the behavioral sciences". In: *PhD thesis, Harvard University*.
- Widrow, Bernard et al. (1960). *Adaptive "adaline" Neuron Using Chemical "memistors."*