A Study on Various Deep Learning Algorithms with R

박정민, Jeong Min Park, Department of Statistics, Ewha Womans University

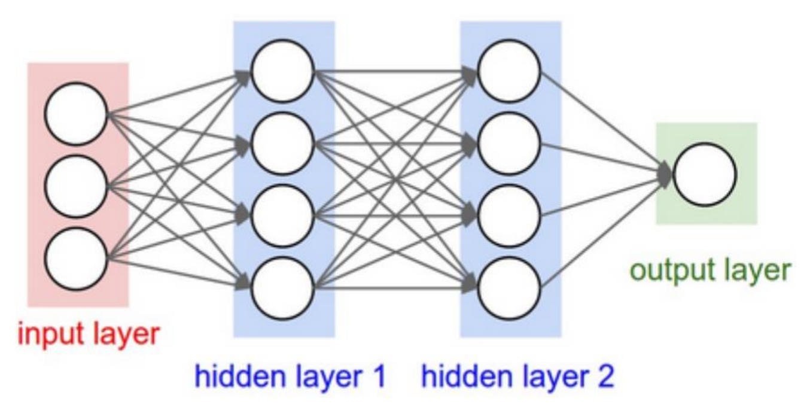
Ⅰ. Introduction

In recent days, deep learning is getting more popular in almost every field, such as bioinformatics, medical images, finance, and so on. Although most softwares for the deep learning algorithms are written in Python, nowadays the realization of deep learning is also possible in R. This paper provides a guideline for the users for deep learning with R among various R packages for deep learning.

First, we briefly introduce various algorithms of deep neural networks and review state-of-art softwares for deep learning with R - `deepnet`, `NeuralNetTools`, `automl`, `autoencoder`, `keras`, `RcppDL`, `mxnet`, `h2o`. Then using the Portuguese bank marketing data, we compare the performance of various R functions for deep learning.

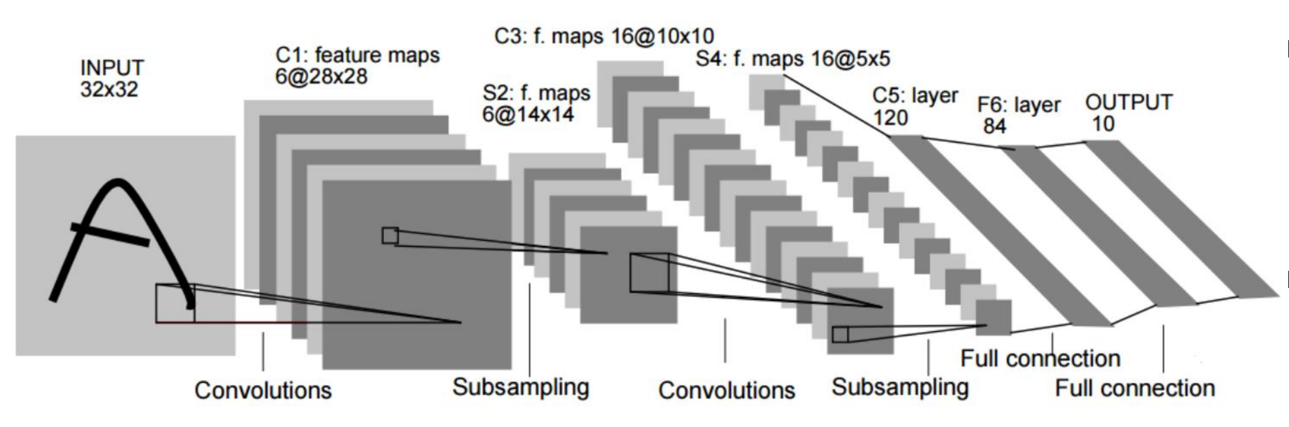
Ⅱ. Deep Learning Models

1. DNN (Deep Neural Network)



- made up of 3 layers – input layer, hidden layer, output layer
- uses fully-connected layers resulting in large amount of computation
- optimized by back-propagation

2. CNN (Convolution Neural Network)



- formed by a convolution layer, pooling layer, fully-connected layer
- often used in analyzing visual imagery or speech recognition
- significantly reduces the number of parameters; filters share same weights
- tuning the model is possible by choosing optimal hyperparameters

3. RNN (Recurrent Neural Network)

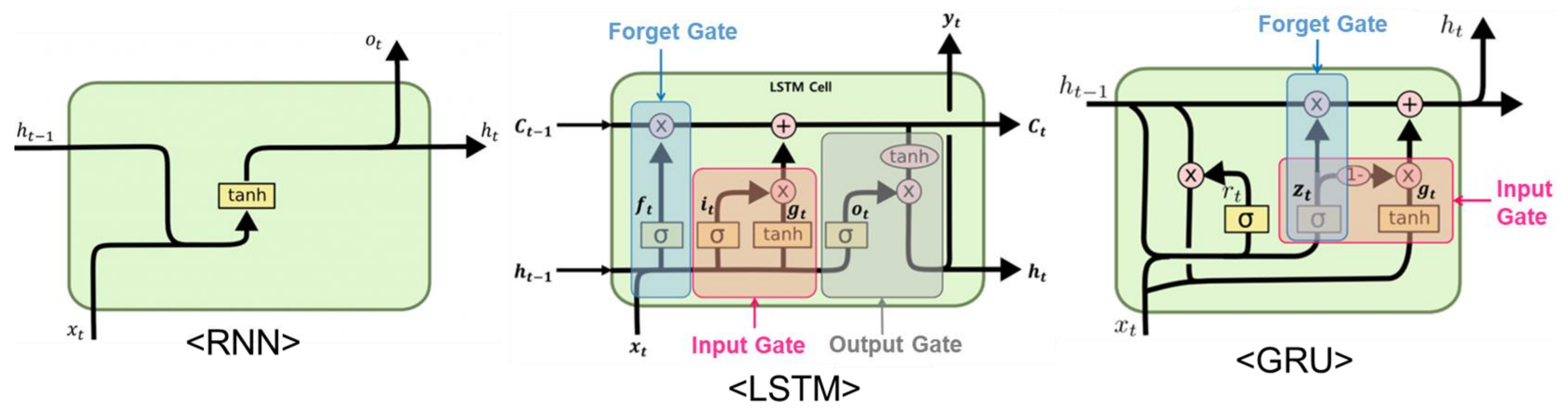
- able to handle sequential data
- the hidden state has the ability to remember
- capable of handling tasks such as handwriting recognition, speech recognition, sentiment analysis, and so on

4. LSTM (Long Short-term Memory)

- overcomes vanishing gradient problem in RNN
- the hidden state has 3 gates; forget gate, input gate, output gate

5. GRU (Gated Recurrent Units)

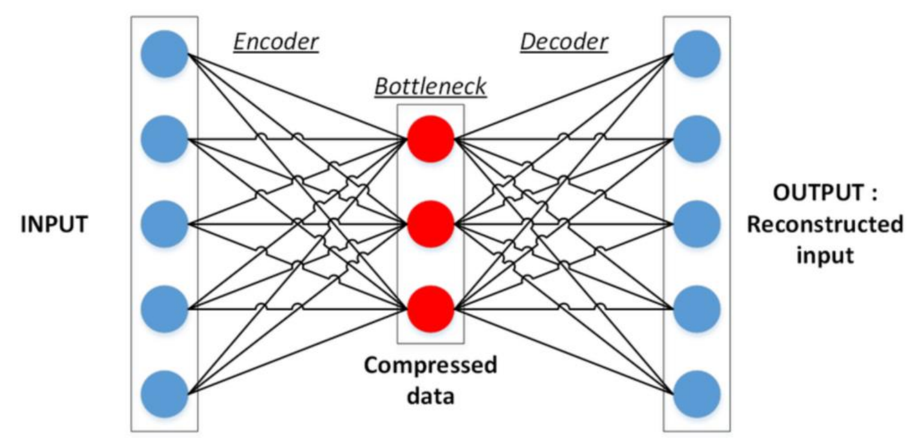
- deep neural network derived from RNN and LSTM
- 2 gates; reset gate, update gate



<RNN> <LSTM> <GRU>

6. AE (Auto-Encoder)

- reconstructs original data
- consisted of an encoder and decoder
- reduce dimensionality, compress data, and finds most important features

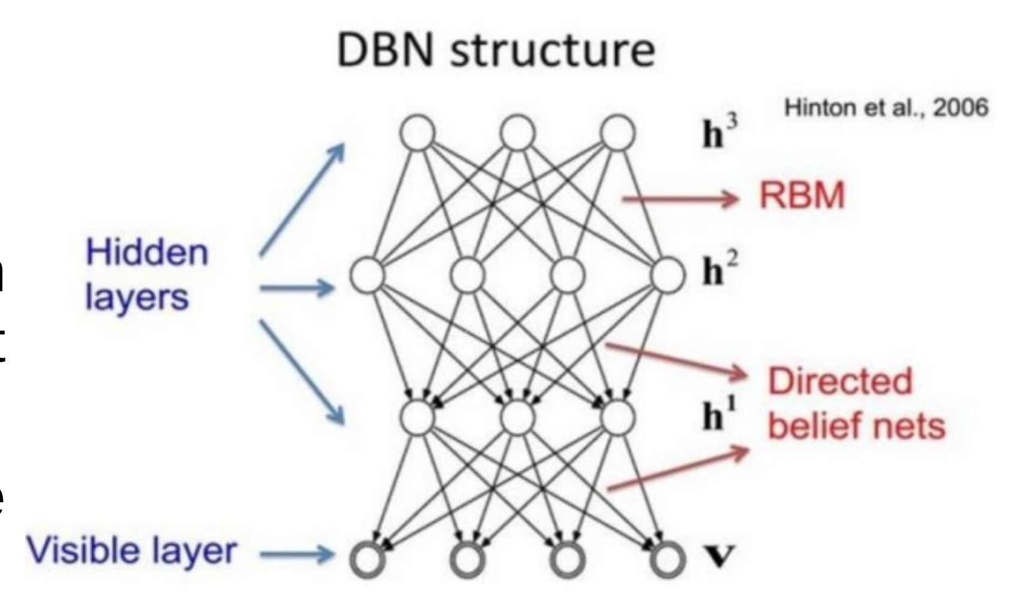


7. RBM (Restricted Boltzmann Machine)

- shallow 2 layer neural net; one visible layer, one hidden layer
- feature extractor whose goal is to recreate the input as accurately as possible, based on KL Divergence method

8. DBN (Deep Belief Network)

- formed by stacking RBMs
- works by fine-tuning the entire input in succession; solves vanishing gradient problem
- capable of detecting patterns in the data



Ⅲ. Comparison of various R packages

Data Description

The data used in the paper is a bank marketing data related with direct marketing campaigns of Portuguese banking institution. The data is consisted of 45,211 observations and 17 variables. The target variable is a binary variable indicating whether the client has subscribed a term deposit or not.

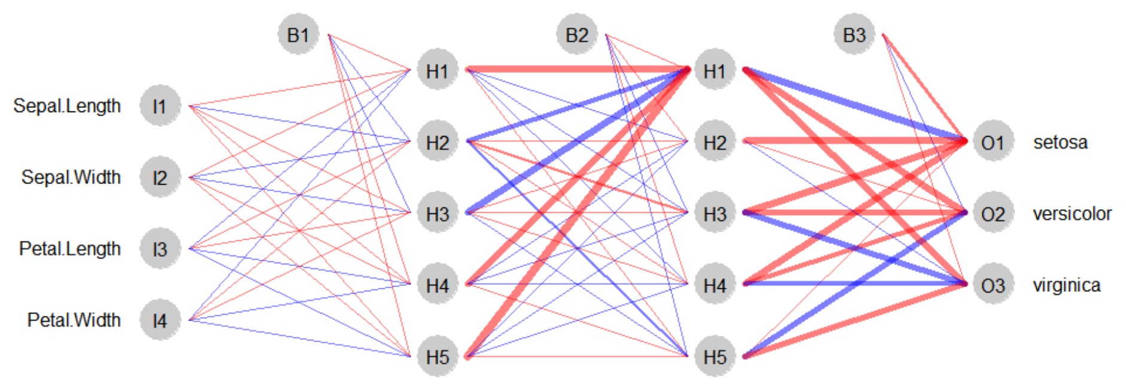
DNN model

3 hidden layers with 20, 40, 10 nodes in each layer

1. `deepnet` : R-based, enable the implementation of various deep learning architectures such as RBM, DBN, and AE

DBN accuracy : 72.5%	computation time : 46.35 seconds
SAE accuracy : 85.2%	computation time : 8 seconds

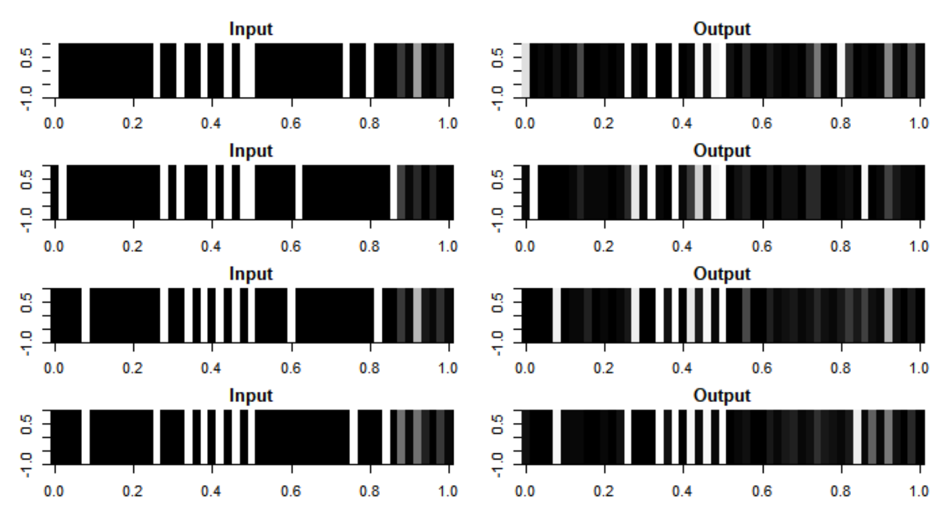
2. `NeuralNetTools` : R-based, visualization and analysis tool for neural networks



3. `automl` : R-based, allow automatic machine learning and tunes the model within the user-specified options

DNN accuracy : 80.3% computation time : 7.23 minutes

4. `autoencoder` : R-based, enable the implementation of sparse autoencoder, visualize features learned by autoencoder



computation time : 2.06 minutes

5. `keras` : Python based, run on CPU or on GPU, use tensorflow as backend tensor calculation, provide a user-friendly API, support arbitrary network architectures, provide built-in CNN, RNN, and any combination of both networks

DNN accuracy : 81.2% computation time : 8.25 seconds

6. `RcppDL` : based on Rcpp, allow implementation of basic machine learning methods such as DA, SDA, RBM, and DBN

RBM rmse : 0.3113 computation time : 13.23 seconds

7. `mxnet` : based on Rcpp, flexible and efficient tool for computing deep learning models

DNN accuracy : 59.8% computation time : 22.53 seconds

8. `h2o` : based on JAVA, offer various supervised/unsupervised machine learning algorithms and deep neural networks

DNN accuracy : 88.6% computation time : 1.16 seconds

	based	DNN	CNN	RNN	LSTM	GRU	AE	RBM	DBN	viz.
deepnet	R	O					O	O	O	
NeuralNetTools	R									O
automl	R	O								
autoencoder	R	O					O			O
keras	Python	O	O	O	O	O				
RcppDL	Rcpp						O	O	O	
mxnet	Rcpp	O	O	O						O
h2o	JAVA	O								

Ⅳ. Results & Conclusion

According to the results derived from the Portuguese bank data, we were able to compare the performance of various R functions for deep learning. The `h2o` package showed the best performance among all the packages in terms of both accuracy and computation time.

Most of the packages support the basic DNN model and `keras` supports the most models. Visualization of filters or structure of the deep learning models are also provided by some packages.