=> To Be Discussed (Option - Validating Neural Network Packages in R with NNbenchmark)

by Author One, Author Two

Abstract An abstract of less than 150 words. Possible format: 1) the overall purpose of the study and the research problem(s) you investigated; 2) the basic design of the study; 3) major findings or trends found as a result of your analysis; and, 4) a brief summary of your interpretations and conclusions.

Introduction

R Statistical Software, as any opensource platform, has relied on its contributors to keep it up to date with the latest developments. One of the many is neural networks. Neural networks are a class of models, based on the brain's own connections system, in the growing field of machine learning. Before, they were a theory with not much practical implementation, partly because of how computationally demanding their algorithms are. A neural network algorithm involves complicated calculations for each gradual "step" into the direction of the optimal solution, namely, determining the gradient of the cost function recursively. Parameters are adjusted accordingly for each iteration from this information. Manually computing partial gradients from complex nonlinear models is taxing. With the help of modern technology and its high-level code, implementing a neural network is relatively simple.

However, the utilization of computers should not only be done effectively but also efficiently. Instead of calculating first-order derivatives, and moving incrementally forward by a predetermined learning rate, it is better to adjust the size of each step according to its curvature. The second-order derivative provides this information. Numerical methods can take this further. The Hessian, a matrix of the second derivatives, is approximated instead of perfectly derived. Such algorithms are known as Quasi-Newton algorithms. Broyden-Fletcher-Goldfarb-Shanno (BFGS) is a well-known example of an algorithm from this class. We believed that these second-order algorithms would perform better than first-order algorithms in terms of finding the optimal solution.

Regardless of our belief, it is crucial to conduct a thorough examination to assess the quality of these training algorithms in R. There is much code, but barely any comparison. In particular, packages that provide neural network of the perceptron type (one input layer, one normalized layer, one hidden layer with a nonlinear activation function that is usually tanh(), one normalized layer, one output output layer) for regression purpose (i.e. NN(X1, ., Xn) = E[Y] were the focus of this research. At the very least, this research will serve as a framework for future research on neural network benchmarking.

Methodology

Our research process was largely divided into teh following 3 phases.

Phase 1 - Preparation

Datasets => NEED TO BE FINISHED
All the datasets used are nonlinear. Linear data sets are more simple and can even be solved with OLS (Ordinary Least Squares) regression. This is why we believe to truly set apart the ability of neural networks we needed to go beyond linear regression. Varying difficulties between data sets helped to classify further package's algorithms accuracy. One site was used for 3 of the multivariate data sets. Sonja Surjanovic and Derek Bingham of Simon Fraser University created this resourceful website to evaluate the design and analysis of computer models. Links to each dataset and their level of difficulty:
- http://www.sfu.ca/~ssurjano/fried.html (Friedman - average)
- http://www.sfu.ca/~ssurjano/detpep10curv.html (Dette - medium)
- http://www.sfu.ca/~ssurjano/ishigami.html (Ishigami - high)

The other multivariate dataset, Ref153, was taken from ... 3 of the univariate datasets were taken from NIST at: https://www.itl.nist.gov/div898/strd/nls/nls_main.shtml. Gauss1 and Gauss2 have a low level of difficulty to solve. Gauss3 is average. Dmod1, Dmod2 are from ... Dreyfus1 is a pure neural network which has no error. This can make it difficult for algorithms that assume an error exists. Dreyfus2 is Dreyfus1 with errors. NeuroOne from ...

Wood ... **Packages**
Searching through the thousands of packages title or the package description one by one would have taken a long time. With **RWsearch** (Kiener, 2020) we were able to automate the process. All packages that have "neural network" as a keyword in the package title or in the package description were included.

Packages <= need to update, RJreferences already updated.

(to use when needed in text, testing how long references will be) 1 **AMORE** (Limas et al., 2020), 2 **ANN2** (Lammers, 2020), 3 **appnn** (Família et al., 2015), 4 **autoencoder** (Dubossarsky and Tyshetskiy, 2015), 5 **automl** (Boulangé, 2020), 6 **BNN** (Jia, 2018), 7 **brnn** (Rodriguez and Gianola, 2020), 8 **Buddle** (Kim, 2020), 9 **CaDENCE** (Cannon, 2017a), 10**cld2** (Ooms, 2018), 11**cld3** (Ooms, 2020), 12**condmixt** (Carreau, 2020), 13**DamiaNN** (Siniakowicz, 2016), 14**deep** (Mayer, 2019), 15**deepdive** (Balakrishnan, 2020), 16**deepnet** (Rong, 2014), 17**deepNN** (Taylor, 2020), 18**DNMF** (Jia and Zhang, 2015), 19**elmNNRcpp** (Mouselimis and Gosso, 2018), 20**ELMR** (Petrozziello, 2015), 21**EnsembleBase** (Mahani and Sharabiani, 2016), 22**evclass** (Denoeux, 2017), 23**gamlss.add** (Stasinopoulos et al., 2020), 24**gcForest** (Jing, 2018), 25**GMDH** (Dag and Yozgatligil, 2016), 26**GMDH2** (Dag et al., 2019), 27**GMDHreg** (Tilve, 2019), 28**gnn** (Hofert and Prasad, 2020), 29**grnn** (Chasset, 2013a), 30**h2o** (LeDell et al., 2020), 31**hybridEnsemble** (Ballings et al., 2015), 32**isingLenzMC** (Suzen, 2016), 33**keras** (Allaire and Chollet, 2019), 34**kerasR** (Arnold, 2017), 35**leabRa** (Titz, 2017), 36**learNN** (Quast, 2015), 37**LilRhino** (Barton, 2019), 38**minpack.lm** (Elzhov et al., 2016), 39**MachineShop** (Smith, 2020), 40**monmlp** (Cannon, 2017b), 41**neural** (Nagy, 2014), 42**neuralnet** (Fritsch et al., 2019), 43**NeuralNetTools** (Beck, 2018), 44**NeuralSens** (Portela González et al., 2020), 45**NlinTS** (Hmamouche, 2020), 46**nlsr** (Nash and Murdoch, 2019), 47**nnet** (Ripley, 2020), 48**nnetpredint** (Ding, 2015), 49**nnfor** (Kourentzes, 2019), 50**nntrf** (Aler and Valls, 2020), 51**nnli2bRcpp** (Nikolaidis, 2020), 52**onnx** (Tang and ONNX Authors, 2018), 53**OptimClassifier** (Perez-Martin et al., 2020), 52**OSTSC** (Dixon et al., 2017), 53**pnn** (Chasset, 2013b), 54**polyreg** (**?**), 55**predictoR** (with contributions from Diego Jimenez A. and D., 2019), 56**qrnn** (Cannon, 2019), 57**QuantumOps** (**?**), 58**quarrint** (Barthelemy et al., 2016), 59**radiant.model** (Nijs, 2020), 60**rasclass** (Wiesmann and Quinn, 2016), 61**rcane** (Suresh et al., 2018), 62**regressoR** (Rodriguez R., 2019), 63**rminer** (Cortez, 2020), 64**rnn** (Quast and Fichou, 2019), 65**RSNNS** (Bergmeir, 2019), 66**ruta** (Charte et al., 2019), 67**simpleNeural** (Dernoncourt, 2020), 68**snnR** (Wang et al., 2017), 69**softmaxreg** (Ding, 2016), 70**Sojourn.Data** (Hibbing and Lyden, 2019), 71**spnn** (Ebrahimi, 2020), 72**TeachNet** (Steinbuss, 2018), 73**tensorflow** (Allaire and Tang, 2019), 74**tfestimators** (Allaire et al., 2018), 75**trackdem** (Bruijning et al., 2020), 76**TrafficBDE** (Chatzopoulou et al., 2018), 77**tsensembler** (**?**), 78**validann** (Humphrey, 2017), 80**zFactor** (Reyes, 2019).

Phase 2 - Exploration of each package and development of template

Exploration
However, not all packages that had the keyword were fit for the scope of our research. Some didn't have any functions to make neural networks. They were simply meta-packages. Others were not regression neural networks of the perceptron type or were only made for specific purposes. We learned this through reading documentation and trying out example code. **Template => TO REVISE AFTER 2020 CODE**
As we inspected the packages, we developed a template for benchmarking. This template's structure is as follows:
(1) Set up of environment - loading packages, setting directory, options;
(2) Summary of datasets;
(3) A loop over datasets which contained (a) setting parameters for a specific dataset (b) selecting benchmark options (c) the training of a neural network with a package's tuned functions (d) calculation of RMSE and MAE (e) plot each training over one initial graph, then plot the best result (f) adding results to the appropriate *.csv file and (g) clearing up environment for next loop; and
(4) Clearing up the environment for the next package. (5) It is optional to print warnings.
This process was made easier with tools from the NNbenchmark package. (FROM LAST YEAR, HOPEFULLY WE CAN SAY OTHERWISE THIS YEAR:) It is not on CRAN yet and can instead be found at `https://github.com/pkR-pkR/NNbenchmark`. Our templates for each package can be found in the companion repository, `https://github.com/pkR-pkR/NNbenchmarkTemplates`.

Phase 3 - Collection of and analysis of results

Collection After the templates were finished, the packages were looped on all datasets. Results were collected in the directory of the templates repository. **Analysis** To rank the how well a package converged and its speed, we developed the following method: 1. The results datasets are loaded into the R environment as one large list. The dataset names, package:algorithm names and all 10

run numbers, durations, and RMSE are extracted from that list 2. For the duration score (DUR), the duration is averaged by dataset. 3 criteria for the RMSE score by dataset are calculated: a. The minimum value of RMSE for each package:algorithm as a measure of their best performance b. The median value of RMSE for each package:algorithm as a measure of their average performance, without the influence of outliers c. The spread of the RMSE values for each package which is measured by the difference between the median and the minimum RMSE (d51) 3. Then, the ranks are calculated for every dataset and the results are merged into one wide dataframe. a. The duration rank only depends on the duration. b. For minimum RMSE values, ties are decided by duration mean, then the RMSE median c. For median RMSE values, ties are decided by the RMSE minimum, then the duration mean d. The d51 rank only depends on itself 4. A global score for all datasets is found by a sum of the ranks (of duration, minimum RMSE, median RMSE, d51 RMSE) of each package:algorithm for each dataset 5. The final table is the result of ranking by the global minimum RMSE scores for each package:algorithm

To rank how easy or not a package was to use (TO BE DISCUSSED FURTHER): - Functionality (util): scaling, input, output, trace - Documentation (docs): examples, structure/functions, vignettes

Results

Tables (NOTE: FINAL MEASURE FOR CONVERGENCE - RMSE RANKS? OR A COMBINATION OF OTHER MEASURES? As in Christophe's recent email: L1 MAE(), L2 RMSE(), Linfinity (WAE))

(ALSO: THE FOLLOWING IS SIMPLY ALPHABETIC LIST FOR ALL TESTED, I WILL DIVIDE THE TABLE INTO 4: 2nd ORDER always recommended, 1st ORDER recommended, 1st ORDER not recommended, untested packages)

Table X: Ratings

No	Name (package::algorithm)	RMSE	DUR	UTIL	DOCS	OVERALL
1	**AMORE**::train.ADAPTgd					
	AMORE::train.ADAPTgdwm					
	AMORE::train.BATCHgd					
	AMORE::train.BATCHgdwm					
2	**automl**					
3	**ANN2**::neuralnetwork.sgd					
	ANN2::neuralnetwork.adam					
	ANN2::neuralnetwork.rmsprop					
4	**brnn**					
5	**CaDENCE**					
6	**deepnet**::gradientdescent					
7	**elmNNRcpp**					
8	**ELMR**					
9	**h2o**::deeplearning					
10	**keras**					
11	**kerasformula**					
12	**kerasR**					
13	**minpack.lm**::nlsLM					
14	**MachineShop**::fit.NNetModel()					
15	**monmlp**::fit.BFGS					
	monmlp::fit.Nelder-Mead					
16	**neural**::mlptrain					
17	**neuralnet**::backprop					
	neuralnet::rprop+					
	neuralnet::rprop-					
	neuralnet::sag					
	neuralnet::slr					
18	**nlsr**::nlxb					
19	**nnet**::nnet.BFGS					
20	**qrnn**::qrnn.fit					
21	**radiant.model**::radiant.model					
22	**rcane**::rlm					
23	**rminer**::fit					
24	**RSNNS**::BackpropBatch					
	RSNNS::BackpropChunk					
	RSNNS::BackpropMomentum					
	RSNNS::BackpropWeightDecay					
	RSNNS::Quickprop					
	RSNNS::Rprop					
	RSNNS::SCG					
	RSNNS::Std-Backpropagation					
25	**ruta**					
26	**simpleNeural**::sN.MLPtrain					
27	**snnR**					
28	**softmaxreg**					
29	**tensorflow**::AdadeltaOptmizer					
	tensorflow::AdagradOptmizer					
	tensorflow::AdamOptmizer					
	tensorflow::FtrlOptmizer					
	tensorflow::GradientDescent					
	tensorflow::MomentumOptmizer					
30	**tfestimators**					
31	**tsensembler**					
32	**validann**::Nelder-Mead					
	validann::BFGS					
	validann::CG					
	validann::L-BFGS-B					
	validann::SANN					
	validann::Brent					

(THE FOLLOWING IS JUST AN ALPHABETICALLY ORDERED LIST OF CURRENTLY UNTESTED PACKAGES)

Table 2: Review of Ommitted Packages

No	Name (package)	Category	Comment
1	**appnn**	-	
2	**autoencoder**	-	
3	**BNN**	-	
4	**Buddle**	-	
5	**cld2**	-	
6	**cld3**	-	
7	**condmixt**	-	
8	**DALEX2**	-	
9	**DamiaNN**	-	
10	**DChaos**	-	
11	**deepNN**	-	
12	**DNMF**	-	
13	**EnsembleBase**	-	
14	**evclass**	-	
15	**gamlss.add**	-	
16	**gcForest**	-	
17	**GMDH**	-	
18	**GMDH2**	-	
19	**GMDHreg**	-	
20	**grnn**	-	
21	**hybridEnsemble**	-	
22	**isingLenzMC**	-	
23	**leabRa**	-	
24	**learNN**	-	
25	**LilRhino**	-	
26	**NeuralNetTools**	-	tools for neural networks
27	**NeuralSens**	-	tools for neural networks
28	**NlinTS**	NA	Time Series
29	**nnetpredint**	-	confidence intervals for NN
30	**nnfor**	NA	Times Series, uses neuralnet
31	**onnx**	-	provides an open source format
32	**OptimClassifier**	NA	choose classifier parameters, nnet
33	**OSTSC**	-	solving oversampling classification
34	**pnn**	NA	Probabilistic
35	**polyreg**	-	polyregression ALT to NN
36	**predictoR**	NA	shiny interface, neuralnet
37	**QuantumOps**	NA	classifies MNIST, Schuld (2018)
38	**quarrint**	NA	specified classifier for quarry data
39	**rasclass**	NA	classifier for raster images, nnet?
40	**regressoR**	NA	a manual rich version of predictoR
41	**rnn**	NA	Recurrent
42	**Sojourn.Data**	NA	sojourn Accelerometer methods, nnet?
43	**spnn**	NA	classifier, probabilistic
44	**TeachNet**	NA	classifier, selfbuilt, slow
45	**trackdem**	NA	classifier for particle tracking
46	**TrafficBDE**	NA	specific reg, predicting traffic
47	**zFactor**	NA	'compressibility' of hydrocarbon gas

Discussions (NOTE TO MENTORS: based on latest 2019 Run 04 or Run 03) A. Recommended: 2nd order algorithms Out of all the algorithms, these second algorithms generally performed better in terms of convergence despite being set to a much lower number of iterations, 200, than the first-order algorithms. Moreover, they performed better in terms of speed. The best in this class were **minpack.lm** and **nlsr**, tied at rank number 1. The Levenberg-Marquardt (LM) algorithm used is fast and converges well. stats::nls() is used. However, these packages require a handwritten formula that may not be ideal for certain situations. A more popular package for neural networks is nnet. This might be because it is part of base R. It implements the BFGS algorithm with stats::optim().

Ranked directly after are some packages that depend on nnet or use the same functions. They differ in how well they decide initial parameters. rminer (rank 4), MachineShop (rank 5), and radiant.model (rank 7) use nnet. Note, radiant.model has its iterations set to 10000, which originally made it slower yet converge better. We used a modified version of the package. At rank 6 is validann's BFGS algorithm using stats::optim(). Its use of optim's L-BFGS-B ranked at number 9 with CaDENCE's use of optim's

BFGS. **monmlp**, from the same author as CaDENCE (Alex Cannon), uses the package **optimx**'s BFGS (Nash and Varadhan, 2020).

Alex Cannon also implemented a quantile regression neural network in qrnn with stats::nlm(). It requires more iterations and is not as fast compared to the other second-order algorithms. However, it is a valuable implementation of quantile regression. Last but not least is **brnn**'s Gauss Newton algorithm which ranks at number 8. brnn is easy to use but does not converge as well due to a hidden constraint: a missing first parameter. Furthermore, brnn's algorithm minimizes the sum of squared errors and a penalty on parameters instead of just the sum of squared errors. This may prevent parameters to get highly correlated, especially with an almost degenerated Jacobian matrix.

B. Recommended: 1st order algorithms validann optim CG RSNNS SCG h2o back-propagation RSNNS Rprop ANN2 adam CaDENCE Rprop -SLOW deepnet BP AMORE ADAPTgdwm AMORE ADAPTgd ANN2 sgd automl trainwgrad ANN2 rmsprop RSNNS BackpropChunk RSNNS Back-WeightDecay RSNNS Std_Backpropagation RSNNS BackpropMomentum automl trainwpso validann optim NelderMead snnR Semi Smooth Newton RSNNS BackpropBatch validann optim SANN monmlp optimx Nelder Mead

C. Not recommended: 1st order algorithms <- DISCUSS CUTOFF By package ELMR, elmNNRcpp - fast ELM algorithms. Unfortunately, can't finetune, does not converge well. neuralnet: a large ammount of iterations, slow, erratic failures tensorflow: NOT EASY TO USE, subsequently keras, tfestimators, ruta ... user needs to understand the language However, advanced users might be able to highly specify a neural network to their needs (customization?)

By algorithm: neuralnet rprop+ neuralnet rprop- neuralnet slr - once ranked well with 100000 iterations AMORE BATCHgd CaDENCE pso psoptim - need to reconfigure? elmNNRcpp - fast, no iterations RSNNS Quickprop (?) AMORE BATCHgdwm tensorflow MomentumOptimizer tensorflow AdamOptimizer ELMR - fast, no iterations tensorflow GradientDescentOptimizer keras rmsprop keras adagrad keras sgd keras adadelta tensorflow AdagradOptimizer keras adam tensorflow FtrlOptimizer neuralnetwork sag tensorflow AdadeltaOptimizer neuralnet backprop - note, might not actually reflect standings, somehow from template to template the learning rate disappeared. Will fix this in future runs

D. Untested => TO DO - LIST

Conclusion => TO DO AFTER 2020 CODE

Future work

As the alogrithms for neural networks continue to grow, there will always be more to validate. For current algorithms in R, our research should be extended to encompass more types of neural networks and their data formats (classifier neural networks, recurrent neural networks, and so on). Different rating schemes and different parameters for package functions can also be tried out.

Acknowledgements

This work was possible due to the support of Google during Summer of Code years 2019 and 2020.

Bibliography

R. Aler and J. Valls. *nntrf: Supervised Data Transformation by Means of Neural Network Hidden Layers*, 2020. URL https://CRAN.R-project.org/package=nntrf. R package version 0.1.0. [p2]

J. Allaire and F. Chollet. *keras: R Interface to 'Keras'*, 2019. URL https://CRAN.R-project.org/package=keras. R package version 2.2.5.0. [p2]

J. Allaire and Y. Tang. *tensorflow: R Interface to 'TensorFlow'*, 2019. URL https://CRAN.R-project.org/package=tensorflow. R package version 2.0.0. [p2]

J. Allaire, Y. Tang, K. Ushey, and K. Kuo. *tfestimators: Interface to 'TensorFlow' Estimators*, 2018. URL https://CRAN.R-project.org/package=tfestimators. R package version 1.9.1. [p2]

T. Arnold. *kerasR: R Interface to the Keras Deep Learning Library*, 2017. URL https://CRAN.R-project.org/package=kerasR. R package version 0.6.1. [p2]

R. Balakrishnan. *deepdive: Deep Learning for General Purpose*, 2020. URL https://CRAN.R-project.org/package=deepdive. R package version 1.0.1. [p2]

M. Ballings, D. Vercamer, and D. Van den Poel. *hybridEnsemble: Build, Deploy and Evaluate Hybrid Ensembles*, 2015. URL https://CRAN.R-project.org/package=hybridEnsemble. R package version 1.0.0. [p2]

J. Barthelemy, T. Carletti, L. Collier, V. Hallet, M. Moriame, and A. Sartenaer. *quarrint: Interaction Prediction Between Groundwater and Quarry Extension Using Discrete Choice Models and Artificial Neural Networks*, 2016. URL https://CRAN.R-project.org/package=quarrint. R package version 1.0.0. [p2]

T. Barton. *LilRhino: For Implementation of Feed Reduction, Learning Examples, NLP and Code Management*, 2019. URL https://CRAN.R-project.org/package=LilRhino. R package version 1.2.0. [p2]

M. W. Beck. *NeuralNetTools: Visualization and Analysis Tools for Neural Networks*, 2018. URL https://CRAN.R-project.org/package=NeuralNetTools. R package version 1.5.2. [p2]

C. Bergmeir. *RSNNS: Neural Networks using the Stuttgart Neural Network Simulator (SNNS)*, 2019. URL https://CRAN.R-project.org/package=RSNNS. R package version 0.4-12. [p2]

A. Boulangé. *automl: Deep Learning with Metaheuristic*, 2020. URL https://CRAN.R-project.org/package=automl. R package version 1.3.2. [p2]

M. Bruijning, M. D. Visser, C. A. Hallmann, and E. Jongejans. *trackdem: Particle Tracking and Demography*, 2020. URL https://CRAN.R-project.org/package=trackdem. R package version 0.5.2. [p2]

A. J. Cannon. *CaDENCE: Conditional Density Estimation Network Construction and Evaluation*, 2017a. URL https://CRAN.R-project.org/package=CaDENCE. R package version 1.2.5. [p2]

A. J. Cannon. *monmlp: Multi-Layer Perceptron Neural Network with Optional Monotonicity Constraints*, 2017b. URL https://CRAN.R-project.org/package=monmlp. R package version 1.1.5. [p2]

A. J. Cannon. *qrnn: Quantile Regression Neural Network*, 2019. URL https://CRAN.R-project.org/package=qrnn. R package version 2.0.5. [p2]

J. Carreau. *condmixt: Conditional Density Estimation with Neural Network Conditional Mixtures*, 2020. URL https://CRAN.R-project.org/package=condmixt. R package version 1.1. [p2]

D. Charte, F. Charte, and F. Herrera. *ruta: Implementation of Unsupervised Neural Architectures*, 2019. URL https://CRAN.R-project.org/package=ruta. R package version 1.1.0. [p2]

P.-O. Chasset. *grnn: General regression neural network*, 2013a. URL https://CRAN.R-project.org/package=grnn. R package version 0.1.0. [p2]

P.-O. Chasset. *pnn: Probabilistic neural networks*, 2013b. URL https://CRAN.R-project.org/package=pnn. R package version 1.0.1. [p2]

A. Chatzopoulou, K. Koupidis, and C. Bratsas. *TrafficBDE: Traffic Status Prediction in Urban Places using Neural Network Models*, 2018. URL https://CRAN.R-project.org/package=TrafficBDE. R package version 0.1.0. [p2]

P. Cortez. *rminer: Data Mining Classification and Regression Methods*, 2020. URL https://CRAN.R-project.org/package=rminer. R package version 1.4.5. [p2]

O. Dag and C. Yozgatligil. *GMDH: Short Term Forecasting via GMDH-Type Neural Network Algorithms*, 2016. URL https://CRAN.R-project.org/package=GMDH. R package version 1.6. [p2]

O. Dag, E. Karabulut, and R. Alpar. *GMDH2: Binary Classification via GMDH-Type Neural Network Algorithms*, 2019. URL https://CRAN.R-project.org/package=GMDH2. R package version 1.5. [p2]

T. Denoeux. *evclass: Evidential Distance-Based Classification*, 2017. URL https://CRAN.R-project.org/package=evclass. R package version 1.1.1. [p2]

D. Dernoncourt. *simpleNeural: An Easy to Use Multilayer Perceptron*, 2020. URL https://CRAN.R-project.org/package=simpleNeural. R package version 0.1.3. [p2]

X. Ding. *nnetpredint: Prediction Intervals of Multi-Layer Neural Networks*, 2015. URL https://CRAN.R-project.org/package=nnetpredint. R package version 1.2. [p2]

X. Ding. *softmaxreg: Training Multi-Layer Neural Network for Softmax Regression and Classification*, 2016. URL https://CRAN.R-project.org/package=softmaxreg. R package version 1.2. [p2]

M. Dixon, D. Klabjan, and L. Wei. *OSTSC: Over Sampling for Time Series Classification*, 2017. URL https://CRAN.R-project.org/package=OSTSC. R package version 0.0.1. [p2]

E. Dubossarsky and Y. Tyshetskiy. *autoencoder: Sparse Autoencoder for Automatic Learning of Representative Features from Unlabeled Data*, 2015. URL https://CRAN.R-project.org/package=autoencoder. R package version 1.1. [p2]

R. Ebrahimi. *spnn: Scale Invariant Probabilistic Neural Networks*, 2020. URL https://CRAN.R-project.org/package=spnn. R package version 1.2.1. [p2]

T. V. Elzhov, K. M. Mullen, A.-N. Spiess, and B. Bolker. *minpack.lm: R Interface to the Levenberg-Marquardt Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds*, 2016. URL https://CRAN.R-project.org/package=minpack.lm. R package version 1.2-1. [p2]

C. Família, S. R. Dennison, A. Quintas, and D. A. Phoenix. *appnn: Amyloid Propensity Prediction Neural Network*, 2015. URL https://CRAN.R-project.org/package=appnn. R package version 1.0-0. [p2]

S. Fritsch, F. Guenther, and M. N. Wright. *neuralnet: Training of Neural Networks*, 2019. URL https://CRAN.R-project.org/package=neuralnet. R package version 1.44.2. [p2]

P. R. Hibbing and K. Lyden. *Sojourn.Data: Supporting Objects for Sojourn Accelerometer Methods*, 2019. URL https://CRAN.R-project.org/package=Sojourn.Data. R package version 0.1.0. [p2]

Y. Hmamouche. *NlinTS: Models for Non Linear Causality Detection in Time Series*, 2020. URL https://CRAN.R-project.org/package=NlinTS. R package version 1.3.8. [p2]

M. Hofert and A. Prasad. *gnn: Generative Neural Networks*, 2020. URL https://CRAN.R-project.org/package=gnn. R package version 0.0-2. [p2]

G. B. Humphrey. *validann: Validation Tools for Artificial Neural Networks*, 2017. URL https://CRAN.R-project.org/package=validann. R package version 1.2.1. [p2]

B. Jia. *BNN: Bayesian Neural Network for High-Dimensional Nonlinear Variable Selection*, 2018. URL https://CRAN.R-project.org/package=BNN. R package version 1.0.2. [p2]

Z. Jia and X. Zhang. *DNMF: Discriminant Non-Negative Matrix Factorization*, 2015. URL https://CRAN.R-project.org/package=DNMF. R package version 1.3. [p2]

X. Jing. *gcForest: Deep Forest Model*, 2018. URL https://CRAN.R-project.org/package=gcForest. R package version 0.2.7. [p2]

P. Kiener. *RWsearch: Lazy Search in R Packages, Task Views, CRAN, the Web. All-in-One Download*, 2020. URL https://CRAN.R-project.org/package=RWsearch. R package version 4.8.0. [p2]

J. Kim. *Buddle: A Deep Learning for Statistical Classification and Regression Analysis with Random Effects*, 2020. URL https://CRAN.R-project.org/package=Buddle. R package version 2.0.1. [p2]

N. Kourentzes. *nnfor: Time Series Forecasting with Neural Networks*, 2019. URL https://CRAN.R-project.org/package=nnfor. R package version 0.9.6. [p2]

B. Lammers. *ANN2: Artificial Neural Networks for Anomaly Detection*, 2020. URL https://CRAN.R-project.org/package=ANN2. R package version 2.3.3. [p2]

E. LeDell, N. Gill, S. Aiello, A. Fu, A. Candel, C. Click, T. Kraljevic, T. Nykodym, P. Aboyoun, M. Kurka, and M. Malohlava. *h2o: R Interface for the 'H2O' Scalable Machine Learning Platform*, 2020. URL https://CRAN.R-project.org/package=h2o. R package version 3.30.0.1. [p2]

M. C. Limas, J. B. O. Mere, A. G. Marcos, F. J. M. de Pison Ascacibar, A. V. P. Espinoza, F. A. Elias, and J. M. P. Ramos. *AMORE: Artificial Neural Network Training and Simulating*, 2020. URL https://CRAN.R-project.org/package=AMORE. R package version 0.2-16. [p2]

A. S. Mahani and M. T. Sharabiani. *EnsembleBase: Extensible Package for Parallel, Batch Training of Base Learners for Ensemble Modeling*, 2016. URL https://CRAN.R-project.org/package=EnsembleBase. R package version 1.0.2. [p2]

B. L. Mayer. *deep: A Neural Networks Framework*, 2019. URL https://CRAN.R-project.org/package=deep. R package version 0.1.0. [p2]

L. Mouselimis and A. Gosso. *elmNNRcpp: The Extreme Learning Machine Algorithm*, 2018. URL https://CRAN.R-project.org/package=elmNNRcpp. R package version 1.0.1. [p2]

A. Nagy. *neural: Neural Networks*, 2014. URL https://CRAN.R-project.org/package=neural. R package version 1.4.2.2. [p2]

J. C. Nash and D. Murdoch. *nlsr: Functions for Nonlinear Least Squares Solutions*, 2019. URL https://CRAN.R-project.org/package=nlsr. R package version 2019.9.7. [p2]

J. C. Nash and R. Varadhan. *optimx: Expanded Replacement and Extension of the 'optim' Function*, 2020. URL https://CRAN.R-project.org/package=optimx. R package version 2020-4.2. [p6]

V. Nijs. *radiant.model: Model Menu for Radiant: Business Analytics using R and Shiny*, 2020. URL https://CRAN.R-project.org/package=radiant.model. R package version 1.3.10. [p2]

V. Nikolaidis. *nnlib2Rcpp: A Collection of Neural Networks*, 2020. URL https://CRAN.R-project.org/package=nnlib2Rcpp. R package version 0.1.2. [p2]

J. Ooms. *cld2: Google's Compact Language Detector 2*, 2018. URL https://CRAN.R-project.org/package=cld2. R package version 1.2. [p2]

J. Ooms. *cld3: Google's Compact Language Detector 3*, 2020. URL https://CRAN.R-project.org/package=cld3. R package version 1.3. [p2]

A. Perez-Martin, A. Perez-Torregrosa, M. Vaca-Lamata, and A. J. Verdu-Jover. *OptimClassifier: Create the Best Train for Classification Models*, 2020. URL https://CRAN.R-project.org/package=OptimClassifier. R package version 0.1.5. [p2]

A. Petrozziello. *ELMR: Extreme Machine Learning (ELM)*, 2015. URL https://CRAN.R-project.org/package=ELMR. R package version 1.0. [p2]

J. Portela González, A. Muñoz San Roque, and J. Pizarroso Gonzalo. *NeuralSens: Sensitivity Analysis of Neural Networks*, 2020. URL https://CRAN.R-project.org/package=NeuralSens. R package version 0.2.0. [p2]

B. Quast. *learNN: Examples of Neural Networks*, 2015. URL https://CRAN.R-project.org/package=learNN. R package version 0.2.0. [p2]

B. Quast and D. Fichou. *rnn: Recurrent Neural Network*, 2019. URL https://CRAN.R-project.org/package=rnn. R package version 0.9.8. [p2]

A. R. Reyes. *zFactor: Calculate the Compressibility Factor 'z' for Hydrocarbon Gases*, 2019. URL https://CRAN.R-project.org/package=zFactor. R package version 0.1.9. [p2]

B. Ripley. *nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models*, 2020. URL https://CRAN.R-project.org/package=nnet. R package version 7.3-14. [p2]

P. P. Rodriguez and D. Gianola. *brnn: Bayesian Regularization for Feed-Forward Neural Networks*, 2020. URL https://CRAN.R-project.org/package=brnn. R package version 0.8. [p2]

O. Rodriguez R. *regressoR: Regression Data Analysis System*, 2019. URL https://CRAN.R-project.org/package=regressoR. R package version 1.1.8. [p2]

X. Rong. *deepnet: deep learning toolkit in R*, 2014. URL https://CRAN.R-project.org/package=deepnet. R package version 0.2. [p2]

D. Siniakowicz. *DamiaNN: Neural Network Numerai*, 2016. URL https://CRAN.R-project.org/package=DamiaNN. R package version 1.0.0. [p2]

B. J. Smith. *MachineShop: Machine Learning Models and Tools*, 2020. URL https://CRAN.R-project.org/package=MachineShop. R package version 2.4.0. [p2]

M. Stasinopoulos, B. Rigby, V. Voudouris, and D. Kiose. *gamlss.add: Extra Additive Terms for Generalized Additive Models for Location Scale and Shape*, 2020. URL https://CRAN.R-project.org/package=gamlss.add. R package version 5.1-6. [p2]

G. Steinbuss. *TeachNet: Fits Neural Networks to Learn About Backpropagation*, 2018. URL https://CRAN.R-project.org/package=TeachNet. R package version 0.7.1. [p2]

A. Suresh, S. Acharekar, H. Chao, and S. Y. Biradar. *rcane: Different Numeric Optimizations to Estimate Parameter Coefficients*, 2018. URL https://CRAN.R-project.org/package=rcane. R package version 1.0. [p2]

M. Suzen. *isingLenzMC: Monte Carlo for Classical Ising Model*, 2016. URL https://CRAN.R-project.org/package=isingLenzMC. R package version 0.2.5. [p2]

Y. Tang and ONNX Authors. *onnx: R Interface to 'ONNX'*, 2018. URL https://CRAN.R-project.org/package=onnx. R package version 0.0.2. [p2]

B. Taylor. *deepNN: Deep Learning*, 2020. URL https://CRAN.R-project.org/package=deepNN. R package version 1.0. [p2]

M. V. Tilve. *GMDHreg: Regression using GMDH Algorithms*, 2019. URL https://CRAN.R-project.org/package=GMDHreg. R package version 0.2.0. [p2]

J. Titz. *leabRa: The Artificial Neural Networks Algorithm Leabra*, 2017. URL https://CRAN.R-project.org/package=leabRa. R package version 0.1.0. [p2]

Y. Wang, P. Lin, Z. Chen, Z. Bao, and G. J. M. Rosa. *snnR: Sparse Neural Networks for Genomic Selection in Animal Breeding*, 2017. URL https://CRAN.R-project.org/package=snnR. R package version 1.0. [p2]

D. Wiesmann and D. Quinn. *rasclass: Supervised Raster Image Classification*, 2016. URL https://CRAN.R-project.org/package=rasclass. R package version 0.2.2. [p2]

O. R. R. with contributions from Diego Jimenez A. and A. N. D. *predictoR: Predictive Data Analysis System*, 2019. URL https://CRAN.R-project.org/package=predictoR. R package version 1.1.0. [p2]

Author One
Affiliation
line 1
line 2
author1@work

Author Two
Affiliation
line 1
line 2
author2@work