The Journal of Open Source Software

RWNN: Fast and simple non-linear modelling using random weight neural networks in R

Søren B. Vilsen[1, 2]

1 Department of Mathematical Sciences, Aalborg University, Denmark **2** Department of Energy Technology, Aalborg University, Denmark

DOI: 10.21105/joss.03977

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Submitted: 03 December 2021
Published: 13 December 2021

Summary

In recent years neural networks, and variants thereof, have seen a massive increase in popularity. This increase is largely due to the flexibility of the neural network architecture, and their accuracy when applied to highly non-linear problems. However, due to the non-linear nature of the neural network architecture training the weights of the network using gradient based optimisation procedures, like back-propagation, does not guarantee a globally optimal solution. Furthermore, optimising the weights by back-propagation can be very slow process. Therefore, various simplifications of the general feed forward neural network architecture have been proposed, including the so called random vector functional link networks, also called random weight neural networks Cao et al. (2018) (sometimes referred to as extreme learning machines). RWNNs randomly assigns the weights of the network between the input-layer and the last hidden-layer, focusing on training the weights between the last hidden-layer and the output-layer. This simplification makes training an RWNN as fast, simple, and efficient as training a (regularised) linear model. However, by randomly drawing the weights between the input and last hidden-layer additional instability is introduced into the network. Therefore, extensions like deep RWNN (Henríquez & Ruz, 2018), sparse RWNN (Zhang et al., 2019), and ensemble deep RWNN (Shi et al., 2021) have been proposed to combat this instability.

Statement of need

`RWNN` is a general purpose implementation of RWNNs in `R` using `Rcpp` and `RcppArmadillo` ((Vilsen, 2021)). The `RWNN` package allows the user to create an RWNN of any depth, letting the user set the number of neurons and activation functions in each layer, choose the sampling distribution of the random weights, and during estimation of the output-weights it allows for Moore-Penrose inversion, ℓ_1 regularisation, and ℓ_2 regularisation. The network, as well as output weight estimation, is implemented in C++ through `Rcpp` and `RcppArmadillo`, making training the model simple, fast, and efficient.

Variants implemented in the package

Along with a standard RWNN implementation, a number of popular variations have also been implemented in the `RWNN` package:

- **ELM** (extreme learning machine) (Huang et al., 2006): A simplified version of an RWNN without a link between the input and output layer.
- **deep RWNN** (Henríquez & Ruz, 2018): An RWNN with multiple hidden layers, where the output of each hidden-layer is included as features in the model.

- **sparse RWNN** (Zhang et al., 2019): Applies sparse auto-encoder (ℓ_1 regularised) pre-training to reduce the number non-zero weights between the input and the hidden layer (the implementation generalises this concept to allow for both ℓ_1 and ℓ_2 regularisation).
- **ensemble deep RWNN** (Shi et al., 2021): An extension of deep RWNNs using the output of each hidden layer to create separate RWNNs. These RWNNs are then used to create an ensemble prediction of the target.

Furthermore, the `RWNN` package also includes general implementations of the following ensemble methods (using RWNNs as base learners):

- **Stacking**: Stack multiple randomly generated RWNN's, and estimate their contribution to the weighted ensemble prediction using k-fold cross-validation.
- **Bagging** (Sui et al., 2021): Bootstrap aggregation of RWNN's creates a number of bootstrap samples, sampled with replacement from the training-set. Furthermore, as in random forest, instead of using all features when training each RWNN, a subset of the features can be chosen at random.
- **Boosting**: Gradient boosting creates a series of RWNN's, where an element, k, of the series is trained on the residual of the previous $k-1$ RWNN's. It further allows for manipulation of the learning rate used to improve the generalisation of the boosted model. Lastly, like the implemented bagging method, the number of features used in each iteration can be chosen at random (also called stochastic gradient boosting).

A Bayesian sampling approach is also implemented using a simple Metropolis-Hastings sampler to sample hidden weights from the posterior distribution of the RWNN. The sampling approach can create multiple types of output such as a single RWNN using the MAP, an ensemble RWNN using stacking, and the entire posterior of the hidden weights.

Lastly, the `RWNN` package also includes a simple method for grid based hyperparameter optimisation using k-fold cross-validation.

References

Cao, W., Wang, X., Ming, Z., & Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing*, *275*, 278–287.

Henríquez, P. A., & Ruz, G. A. (2018). Twitter sentiment classification based on deep random vector functional link. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–6.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70*(1), 489–501.

Pao, Y.-H., Park, G.-H., & Sobajic, D. J. (1992). Learning and generalization characteristics of random vector Functional-link net. *Neurocomputing*, *6*, 163–180.

Schmidt, W. F., Kraaijveld, M. A., & Duin, R. P. W. (1992). Feedforward neural networks with random weights. *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, 1–4.

Shi, Q., Katuwal, R., Suganthan, P. N., & Tanveer, M. (2021). Random vector functional link neural network based ensemble deep learning. *Pattern Recognition*, *117*, 107978.

Sui, X., He, S., Vilsen, S. B., Teodorescu, R., & Stroe, D.-I. (2021). Fast and Robust Estimation of Lithium-ion Batteries State of Health Using Ensemble Learning. *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, 1–8.

Vilsen, S. B. (2021). Random Weight Neural Networks. In *GitHub repository:* https://github.com/svilsen/RWNN; GitHub.

Zhang, Y., Wu, J., Cai, Z., Du, B., & Yu, P. S. (2019). An unsupervised parameter learning model for RVFL neural network. *Neural Networks*, *112*, 85–97.