The Journal of Open Source Education

mLEARn: An Implementation of Multi-layer Perceptron in C++

Kalu U. Ogbureke[1]

1 GRLMC, Rovira i Virgili University, Tarragona Spain

DOI: 10.21105/jose.00055

Software

- Review
- Repository
- Archive

Submitted: 22 May 2019
Published: 05 June 2019

Abstract

This paper presents `mLEARn`, an open-source implementation of multi-layer perceptron in C++. The techniques and algorithms implemented represent existing approaches in machine learning. `mLEARn` is written using simple C++ constructs. The aim of `mLEARn` is to provide a simple and extendable machine learning platform for students in courses involving C++ and machine learning. The source code and documentation can be downloaded from https://github.com/kalu-o/mLEARn.

Introduction

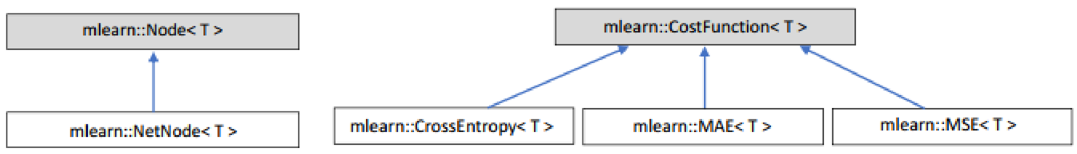
Artificial neural networks (ANNs) is an area of deep learning which has been well studied. This is because of their importance in many areas, from autonomous driving through speech technologies (Graves, Mohamed, & Hinton, 2013). The two main categories are usually recurrent (feedback) and feed-forward architectures (Bishop, 1995). Deep learning is currently an active research area in machine learning.

Statement of Need

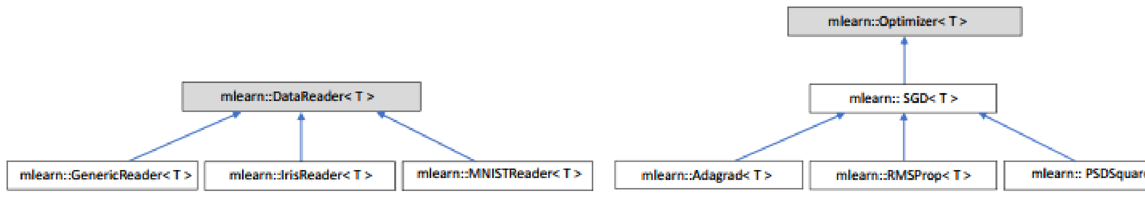
There are currently available popular deep learning frameworks such as MXNet (T. Chen et al., 2015), Caffe (Jia et al., 2014) and TensorFlow (Abadi et al., 2016). Students often use these as off-the-shelf machine learning tools and have little or no control over the codes. One of the reasons for this is because the codes are advanced and production ready. `mLEARn` addresses these as it can be used as an off-the-shelf machine learning tool. Furthermore, the coding style makes it easier to apply what was learnt in machine learning/C++ courses and extend the functionalities. These make it easier to understand machine learning algorithms from first principle and extend state-of-the-art.

Architecture of mLEARn

The classes implemented in `mLEARn` are Node, NetNode, Activation, CostFunction, Layer, Network, DataReader and Optimizer. The Node class is the fundamental data structure used; and NetNode is an extension of the Node class used for multi-layer perceptron. The Activation class handles activations in the network. Currently, the functions implemented are sigmoid, tanh, rectified linear unit (ReLU), leaky ReLU, identity, softmax and exponential linear unit (ELU). The CostFunction class is responsible for objective/loss functions. Cost functions implemented are mean squared error (MSE), mean absolute error (MAE) and cross entropy.



The Network class is a classic MLP consisting of sequences of layers, i.e. one or more hidden layers and an output layer. The DataReader class is the base class responsible for reading train/test dataset into features and labels. Three different readers are implemented, namely MNISTReader, GenericReader and IrisReader. The Optimizer class is the base class responsible for training algorithms. Three optimizers are currently implemented: mini-batch stochastic gradient descent (Kiefer & Wolfowitz, 1952; Ruder, 2016), adaptive gradient (Adagrad) (Duchi, Hazan, & Singer, 2011) and root mean square propagation (RMSProp) (Tieleman & Hinton, 2012).



A novel implicit adaptive learning rate method (PSDSquare) implemented is currently being considered for publication by IEEE. A number of new enhancements such as automatic differentiation, distributed computing and GPU support will be added in future.

Acknowledgements

The core work in neural networks and convergence was done as part of the DEA thesis "Training Multi-layer Perceptron using Genetic Algorithms", while the author was with GRLMC at Rovira i Virgili University, Tarragona. The author would like to acknowledge support for the speech and language technologies program.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In *12th usenix symposium on operating systems design and implementation (osdi 16)* (pp. 265–283). Retrieved from https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., et al. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, *abs/1512.01274*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1512.html#ChenLLLWWXXZZ15

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, *12*, 2121–2159. Retrieved from http://dl.acm.org/citation.cfm?id=1953048.2021068

Graves, A., Mohamed, A. R., & Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. *CoRR*, *abs/1303.5778*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1303.html/#abs-1303-5778

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., et al. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd acm international conference on multimedia*, MM '14 (pp. 675–678). New York, NY, USA: ACM. doi:10.1145/2647868.2654889

Kiefer, J., & Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, *23*, 462–466.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, *abs/1609.04747*. Retrieved from http://arxiv.org/abs/1609.04747

Tieleman, T., & Hinton, G. (2012). Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.