Torch-based Multi-Layer Perceptron

AA1 A.Y. 2016/2017
Project type A

Fabrizio Baldini
fabrizio.baldini@protonmail.com

August 27th 2018

Abstract

A simple MLP was developed in Lua within the Torch framework and employed for both MONKS and ML-CUP. The network was driven by LMS steepest gradient descent backpropagation, with momentum and Tikhonov regularization terms. 5-fold Cross-Validation was employed for model selection.

1 Introduction

The main aim of the project was to familiarize myself with the backpropagation algorithm, as well as getting experience in dealing with concrete matters of Machine Learning, like model selection, the bias-variance tradeoff and how to avoid overfitting.

The employed algorithm is a Multi-Layer Perceptron with a single hidden layer, trained by backpropagation using a steepest descent batch gradient method over normalized input. Momentum and L2 regularization are also supported. Several conditions for early stopping are checked at each epoch.

An important assumption was made with regards to the ML-CUP dataset: features of the input patterns were normalized independently of the target labels, but otherwise all dimensions of the respective domain spaces were projected with the same coefficient (symmetry was assumed between domain dimensions and between image dimensions).

2 Method

Tensors and basic mathematical operations over Tensors from the Torch framework were employed to implement the neural network. No ready-made algorithms were used in any capacity.

2.1 Network features

Batch/online: batch processing was chosen for its stability properties. Online was initially implemented but ultimately discarded for its more jagged behaviour.

Activation function: a sigmoidal function was chosen and employed in all test cases. In general, the software allows for the definition of arbitrary activation functions both for the input-to-hidden and hidden-to-output layers, assuming they are provided with the respective derivatives.

Regularization: The algorithm can employ Tikhonov regularization.
To note: considering how MONKS problems are conditioned (with many irrelevant features from input patterns), a LASSO (or better yet Elastic Net) regularization scheme would have likely been productive; however, it was not implemented due to time constraints as it would have required adding support for a coordinate descent optimization strategy.

Momentum: Momentum can be applied adding the gradient delta from the previous epoch.

Error metrics: While the algorithm always trains under a Least Mean Squares strategy, error traces can be produced with any arbitrary measure, knowing that individual contributions from the patterns will be averaged over the epoch.

Postprocessing: Error signals backpropagated during training lie within an LMS strategy and assume all labels have been normalized to the output range of the outermost activation function (e.g. $(0, 1)$ for logistic sigmoid, $(-1, 1)$ for arctangent). An arbitrary postprocessing function (e.g. thresholding for classification, or de-normalization) can be applied to the output when assessing the network's performance or otherwise outside of training.

Preprocessing: The network assumes all input to have been normalized to the $[0, 1]$ interval. Specifically for the case of MONKS, 1-of-k (one-hot) encoding was used for the input patterns.

Early stop: The network stops training as soon as:

- Training error, averaged over the last 10 epochs, falls under a (configurable) threshold;
- Training error, averaged over the last 10 epochs and the 10 epochs before those, starts increasing;
- Training error, averaged over the last 10 epochs and the 10 epochs before those, changes by less than a (configurable) threshold;
- If a validation set is provided, VL error, averaged over the last 10 epochs and the 10 epochs before those, starts increasing;
- If a validation set is provided, VL accuracy hits 100%.

Otherwise, training will stop after a set number of epochs have lapsed.

K-fold Cross-Validation: A standard K-fold validation schema is supported by the network, computing mean and standard deviation of the final VL error. Input patterns are shuffled before the partitioning.

Starting conditions: Each training session actually consists of 5 trials run with different starting conditions (randomized weights). The best result is kept out of the 5, where best means minimal in terms of final VL error if present, or final TR error otherwise.

2.2 Domain-specific provisions

Preprocessing: For MONKS, 1-of-k (one-hot) encoding was used as normalization technique. For ML-CUP17, input features were projected to the $[0, 1]$ interval.

Validation: For all datasets, 5-Fold Cross Validation was used for the purpose of model selection. A hyperparameter grid was explored, choosing the combination yielding the least final VL error for its respective folding.

Preliminary trials: The number of neurons for the network was determined by hand outside of the grid search. It was in fact very easy to see that too few neurons inevitably produced catastrophically bad results no matter the hyperparameters, while too many neurons led to no discernible improvement, if not to overfitting.

The epoch cutoff was also determined by hand and fixed for the grid after looking at the general shape of the error curve during trial runs.

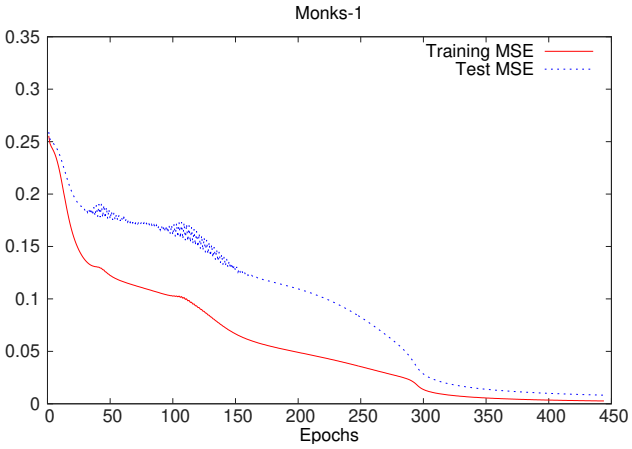
3 Experiments

3.1 MONKS

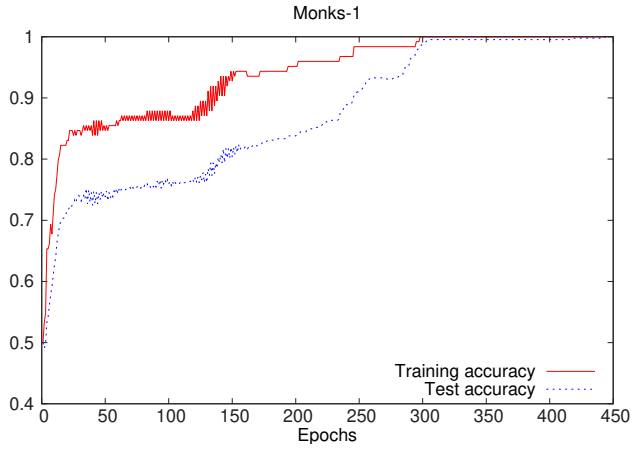
Task	Units	η	Momentum	λ	MSE (TR/TS)	Accuracy % (TR/TS)
MONKS 1	4	15	1	1.0e-4	2.68e-3 / 8.22e-3	100 / 100
MONKS 2	2	15	1.0e-2	1.0e-3	4.36e-2 / 5.11e-2	100 / 100
MONKS 3	1	10	1.0	1.0e-4	4.57e-2 / 4.55e-2	95.1 / 95.4

Table 1: MONKS results.

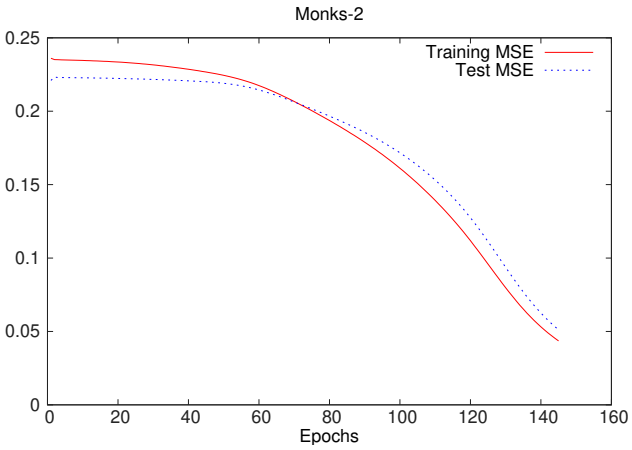
Please note: the results for MONKS 1 show a distinct instability in the convergence process. This is not due to a fault within the algorithm implementation, but in the extremely high learning rate chosen by model selection. In fact, see figure 4 in the appendix for the behaviour observed with the same hyperparameters except on a lower learning rate.



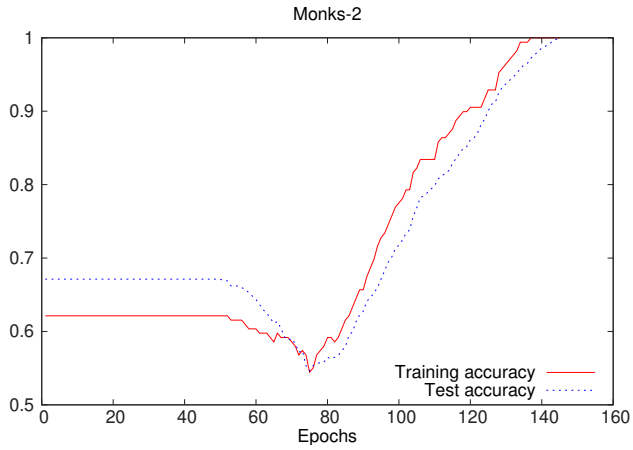
(a) MONKS 1 error



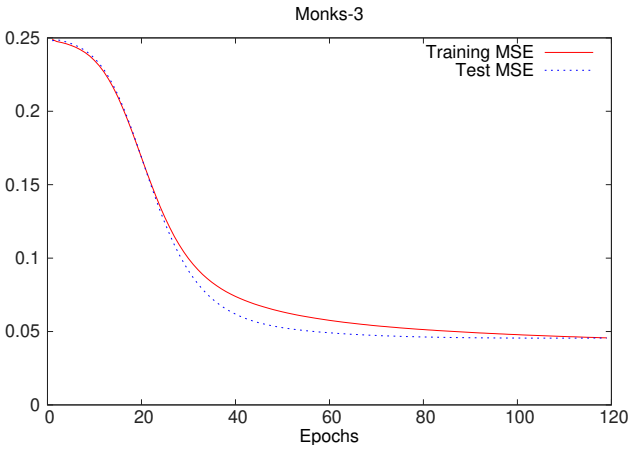
(b) MONKS 1 accuracy



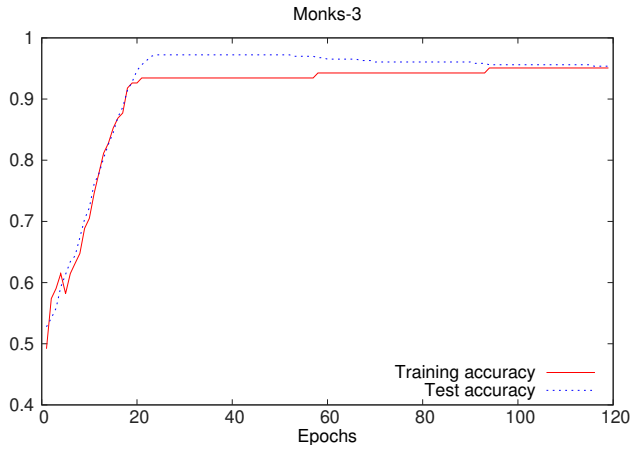
(c) MONKS 2 error



(d) MONKS 2 accuracy



(e) MONKS 3 error



(f) MONKS 3 accuracy

Figure 1: MONKS results

3.2 ML-CUP17

Model selection was carried out as 5-fold Cross-Validation.

A few preliminary investigations were run by hand to determine the least required number of units and epoch cutoff.

The final model was chosen simply as the one minimizing VL MEE as per the respective folding's average during grid search. See table 2 for the exact hyperparameters tested and table 3 for the selected model. See figure 3 for training error graphs.

3.2.1 Trends

Three main shapes in the error traces can be identified, roughly corresponding to employed values of learning rates:

- For $\eta = 0.1$ we observe a particularly gentle knee that looks almost like a linear function. This is reasonable, as such a low learning rate is bound to yield very slow improvements. See figure 2 (a) for an example.
- For $\eta = 0.5$ or 1 we get a very clean knee, although convergence stops just shy of an MEE of 8 uinits. See figure 2 (b) for an example.
- For $\eta = 10$ or 15 the error plot curve shows a more complex shape, highlighting what is likely a local error minimum around 8 units that lower learning rates don't overcome. Almost all runs in this class managed to converge to the better minimum of around 4 MEE units except for a few unlucky ones, not reported, likely due to unfavourable starting conditions. See figure 2 (c) for an example.
- One specific degenerate case is worth reporting, registered for $\eta = 15$, $\lambda = 0.1$ and momentum coefficient of 1, likely due to the excessive learning rate and momentum driving the gradient descent to overshoot minima. See figure 2 (d) .

Parameter	Values
η	15, 10, 1, 0.5, 0.1
Momentum	1, 0.1, 0.01, 0.001, 0.0001
λ	1, 0.1, 0.01, 0.001, 0.0001

Table 2: ML-CUP17 hyperparameters search grid

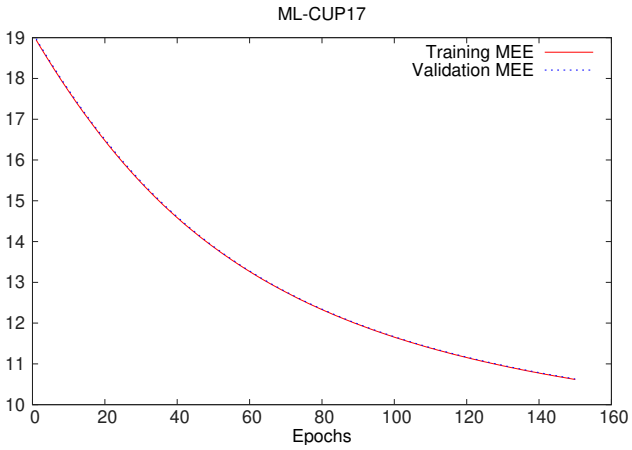
3.2.2 Performance

A 5-fold CV run (which means 25 individual backpropagation convergences, given each training run was made up of 5 different random starts as per section 2) takes up to 4 minutes on the provided TR set to complete a full 150 epochs. Underlying hardware is a 3.7Ghz Intel Core i5-3570K with 8GB DDR3 1333Mhz RAM.

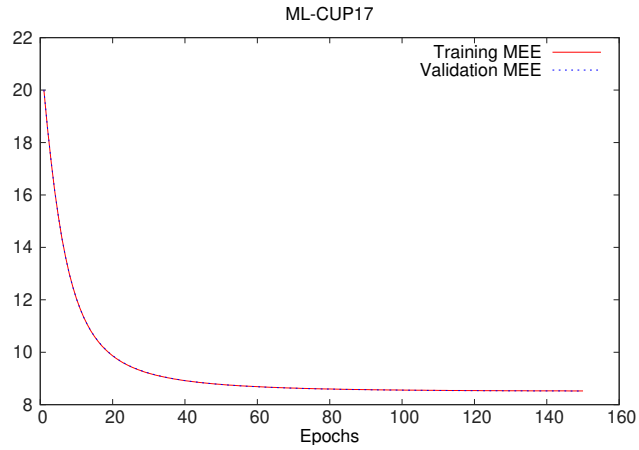
3.2.3 Final model

Units	η	Momentum	λ	VL MEE	TR MEE
1	15	0.01	1.0e-4	(4.25 +/- 3.21e-2)	(4.15 +/- 3.02e-2)

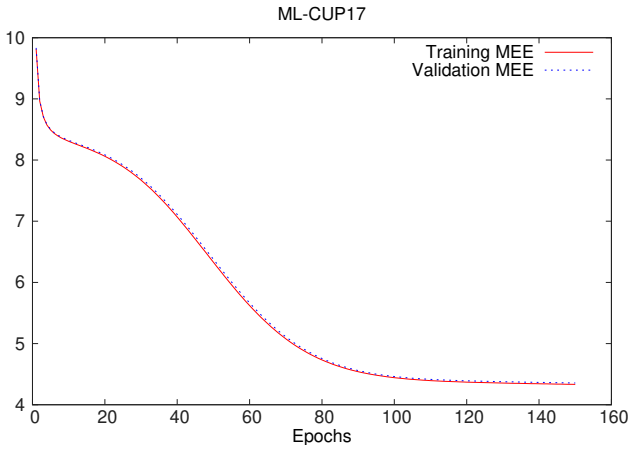
Table 3: ML-CUP17: final model hyperparameters



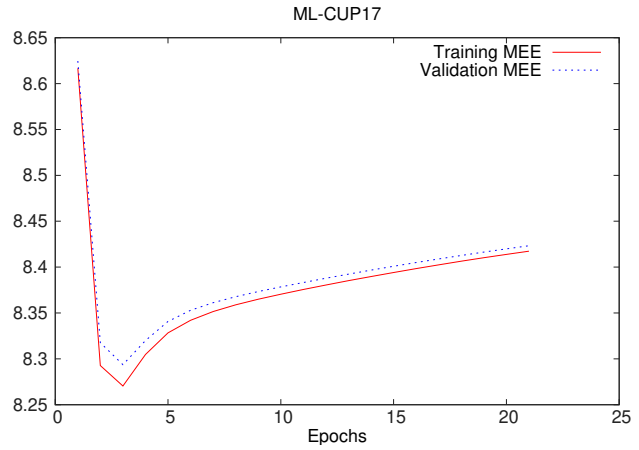
(a) $\eta = 0.1$, $\lambda = 0.01$, m $= 0.01$



(b) $\eta = 1$, $\lambda = 0.01$, m $= 0.1$

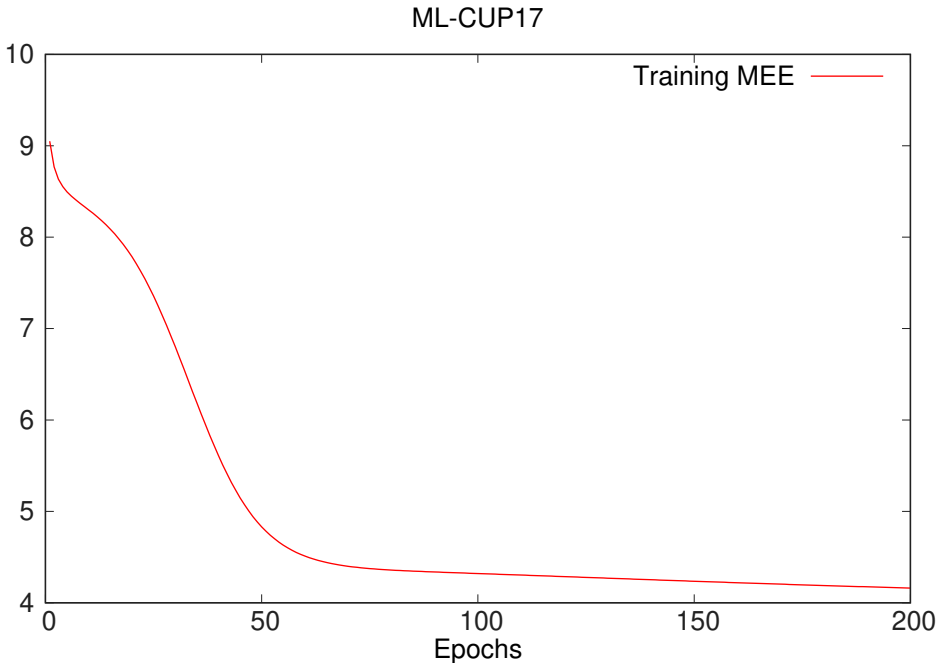


(c) $\eta = 10$, $\lambda = 0.001$, m $= 0.1$

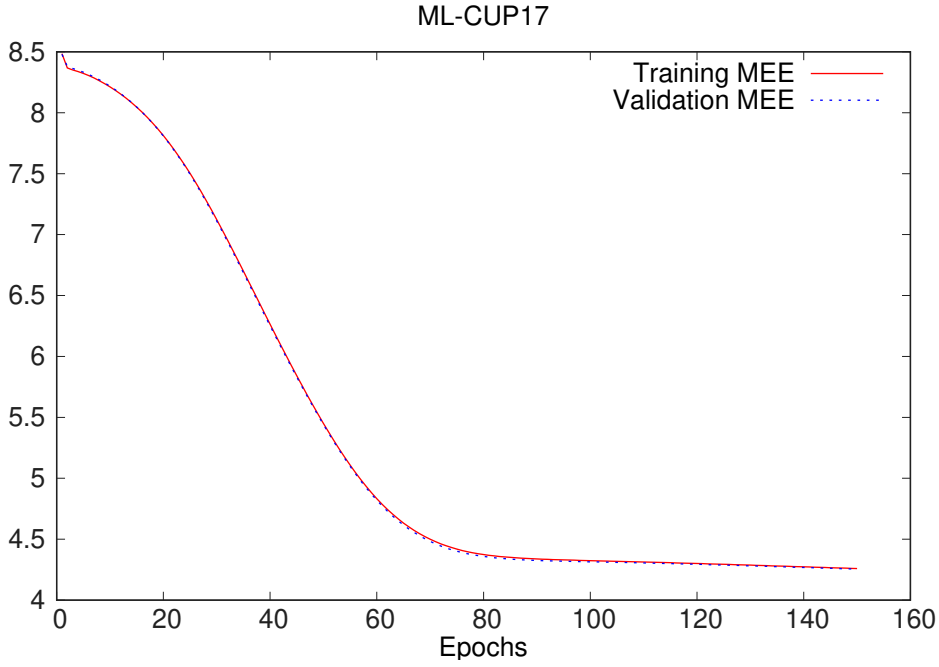


(d) $\eta = 15$, $\lambda = 0.1$, m $= 1$

Figure 2: Some interesting cases for ML-CUP17



(a) ML-CUP17: training error for final parameters, full TR set



(b) ML-CUP17: training error during CV training run for winning model

Figure 3: ML-CUP17: selected model

4 Conclusions

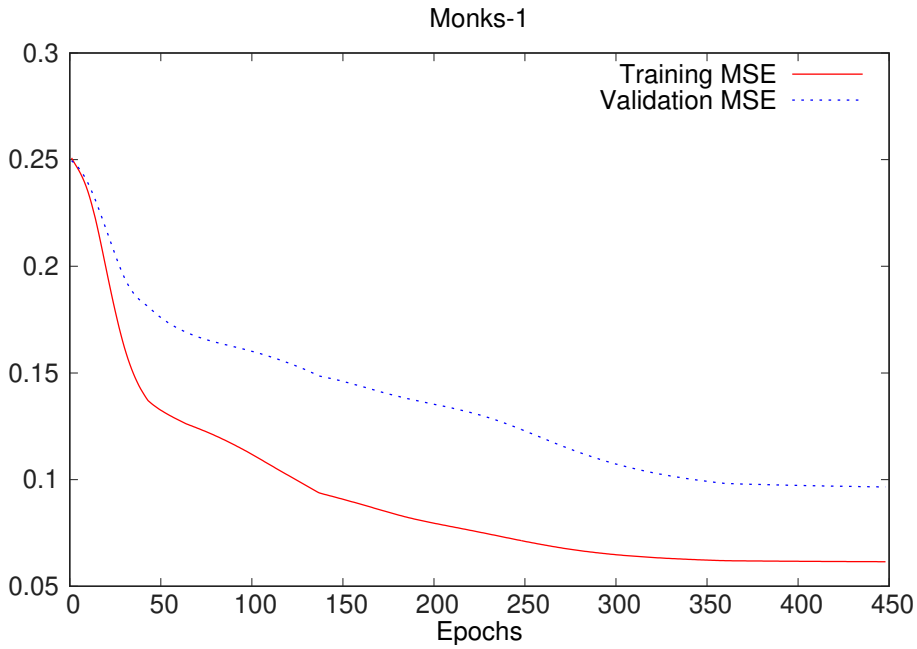
BLIND TEST RESULTS:
`FB-18_abstract.txt`
`FB-18_ML-CUP17-TS.csv`

Miscellaneous

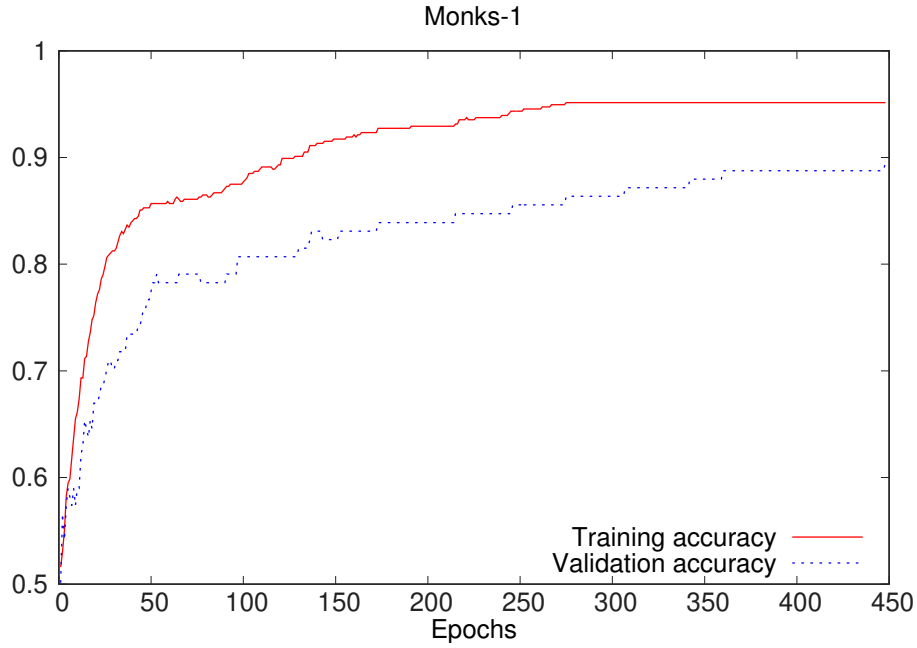
Code is included in a tarball but it is also available at `https://github.com/Aanok/torch-mlp`. Source files are well commented and an HTML luadoc is also provided to document usage and interface.

I agree to the disclosure and publication of my name, and of the results with preliminary and final ranking.

Appendix



(a) MSE



(b) Accuracy

Figure 4: MONKS 1: $\eta = 15$, $\lambda =$ 1e-4, m $= 1$

References

[1] Tom M. Mitchell, *Machine Learning*, McGraw-Hill Science/Engineering/Math, 1997

[2] Simon Kaykin, *Neural Netowrks and learning machines*, Pearson Education, Inc, Upper Saddle River, New Jersey 07458, 3rd edition, 2009.