Population-based Global Optimisation Methods for Learning Long-term Dependencies with RNNs

Bryan Lim [1 2] **Stefan Zohren** [1 2] **Stephen Roberts** [1 2]

Abstract

Despite recent innovations in network architectures and loss functions, training RNNs to learn long-term dependencies remains difficult due to challenges with gradient-based optimisation methods. Inspired by the success of Deep Neuroevolution in reinforcement learning (Such et al., 2017), we explore the use of gradient-free population-based global optimisation (PBO) techniques – training RNNs to capture long-term dependencies in time-series data. Testing evolution strategies (ES) and particle swarm optimisation (PSO) on an application in volatility forecasting, we demonstrate that PBO methods lead to performance improvements in general, with ES exhibiting the most consistent results across a variety of architectures.

1. Introduction

With the increasing availability of high-frequency sensor data, recent trends in time series forecasting have explored the use of deep neural networks to make predictions from real-time data streams. Successful applications have also spanned a multitude of fields – including real-time human activity recognition based on wearable sensors in healthcare (Nweke et al., 2018), local rainfall prediction in weather forecasting (Chao et al., 2018), and high-frequency market microstructure prediction in finance (Zhang et al., 2019).

Recurrent neural networks (RNNs), in particular, have several properties that make them attractive for real-time predictions from a methodological standpoint. Firstly, RNNs learn complex cross-sectional and temporal relationships in a purely data driven manner, which is useful for complex datasets where the underlying data generation process is not well understood. In addition, RNNs also naturally retain information over time through the recursive update of an internal memory state. This helpful in cases where the exact length of relevant history is unknown, and architectures that rely on a fixed look-back window – such as convolutional neural networks (CNNs) – might not be fully capture all relevant information.

However, long-term dependency learning with RNNs remains difficult in practice, mainly due to inherent problems with backpropagation through time (BPTT) with stochastic gradient descent (SGD) – such as exploding/vanishing gradients seen in standard Elman RNN architectures (Hochreiter et al., 2001). Challenges still persist even with modern architectures which stabilise gradient flow – such as Long-short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) – with multiple lines of active research looking at both memory enhancements and training improvements to help RNNs learn long-term dependencies (Neil et al., 2016; Zhang et al., 2018; Trinh et al., 2018; Kanuparthi et al., 2019). Furthermore, standard minibatch SGD, where long trajectories are truncated into shorter sequences for minibatches, also runs the risk of excluding relevant information during training – as neural networks are unable to establish links between observations and historical drivers which lie outside the truncation window. Better performance for long-term dependency modelling could hence be achieved by exploring training methods that do not rely on gradient-based BPTT.

Gradient-free evolutionary computation techniques have previously been used to train deep neural networks in reinforcement learning, with methods such as Deep Neuroevolution (Such et al., 2017) exhibiting comparable results to standard gradient-based approaches. Inspired by this success, we investigate the use of population-based optimisation (PBO) algorithms – i.e. evolution strategies (Salimans et al., 2017) and particle swarm optimisation (Kennedy & Eberhart, 1995) – in RNN training, specifically to overcomes issues in learning long term dependencies with gradient-based methods. Focusing on applications in time series forecasting, we evaluate the use of PBO methods to train a variety of modern RNN architectures, demonstrating the

[1]Department of Engineering Science, University of Oxford, Oxford, United Kingdom [2]Oxford-Man Institute of Quantitative Finance, University of Oxford, Oxford, United Kingdom. Correspondence to: Bryan Lim <blim@robots.ox.ac.uk>, Stefan Zohren <zohren@robots.ox.ac.uk>, Stephen Roberts <sjrob@robots.ox.ac.uk>.

performance improvements over standard gradient-based stochastic backpropagation while maintaining a comparable computational budget – as measured by the number of feed-forward passes through the network during training.

2. Related Works

Architectural Innovations The bulk of research in long-term dependency learning has focused on architectural improvements – especially pertaining to the internal memory state of the RNN. The inclusion of the forget gate in Long-short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), for instance, reduces vanishing/exploding gradient issues by introducing *linear temporal paths* which facilitate gradient flow (Kanuparthi et al., 2019). More recently, Fourier Recurrent Units (FRUs) (Zhang et al., 2018) have been proposed, improving gradient flow via Fourier basis functions in its internal memory state. In other works, Neil et al. (2016) also introduced the Phased LSTM (P-LSTM) to address situations where sparse, asynchronous sensor updates infrequently contribute to predictions – using an additional time gate to control how often observations contribute to the LSTM's internal memory state. This helps to improve predictions for long event-based sequences, particularly where irregularly sampled data is present. While issues with gradient-based methods have been addressed in part, long-term dependency learning fundamentally remains an area of active research for RNNs, with performance improvements still being gained by enhancing standard training methods even for existing architectures (see below).

Modifications to Standard RNN Training An alternative class of methods investigates the enhancement of standard training methods (Goodfellow et al., 2016), namely the augmentation of loss functions or gradient flows during training (Trinh et al., 2018; Kanuparthi et al., 2019). In Trinh et al. (2018), a combination of truncated BPTT and an auxiliary loss function is adopted – generated by selecting a random anchor point during training, feeding internal states from that point into a separate prediction decoder, and back-propagating through the original RNN to a pre-determined truncation point. In contrast to PBO – which performs a full feed-forward pass through the network to compute losses – this approach stills applies backpropagation to truncated sequences, making it difficult to effectively learn dependencies beyond a specified window. Alternatively, Kanuparthi et al. (2019) explicitly decompose the LSTM recursion equations into a bounded linear and an unbounded polynomial gradient component, with the former being responsible for long-term dependency learning. As unbounded terms can dominate gradient backpropagation – and inadvertently hamper long-term dependency learning – they propose what they term the *h-detach* trick to suppress this term by stochastically dropping it during training. While effective, we note that this approach is solely restricted to the LSTM model, and PBO methods can be easily applied to any RNN architecture.

Evolutionary Algorithms in Reinforcement Learning Recent works in deep reinforcement learning have explored evolutionary algorithms as scalable alternatives to training deep neural networks (Such et al., 2017; Salimans et al., 2017). Using simple random Gaussian perturbations to mutate network weights at each training step, these methods utilise large populations of individuals to efficiently converge on the optimum coefficients ($\approx$ 1000 offspring in Such et al. (2017)) – all of which can be efficiently distributed on parallel workers. To maintain the speed of communication between workers for big networks with many weight parameters, they propose a simple compact representations of weights in each offspring of the population, saving down a single random seed which can be used to generate the full weight perturbation vector. While well-studied in renouncement learning, little work has been done to evaluate the efficacy of evolutionary algorithms in capturing long-term dependencies with RNNs. To the best of our knowledge, this paper is the first to examine the use PBO methods in the context of long-term dependency modelling – along with its implications on time series forecasting.

3. Population-based Global Optimisation Techniques

Population-based optimisation (PBO) methods are traditionally divided into two categories (Wu et al., 2019) – 1) evolutionary algorithms that mimic biological evolution, and 2) swarm intelligence approaches which simulate social behaviour of large groups of animals. For simplicity and ease of comparison, we interchangeably refer to population members in both evolutionary algorithms and swarm intelligence as *individuals* in this paper.

PBO methods in general comprise the following steps:

1. **Initialisation** – Create a default initial population of individuals and optimisation parameters, e.g. randomly distributing them over weight space or setting to 0.

2. **Population Update** – At each training iteration, the weights for each individual are updated based on their respective meta-heuristics – e.g. by mutation or particle movement.

3. **Score Computation** – Loss functions are then evaluated for each individual before control parameters are updated – i.e. the generation of offspring for ES and global/local optimum weights for PSO

4. Repeating steps 2 and 3 until convergence.

We next proceed to describe our specific implementations based on the general framework above.

3.1. Evolution Strategies

Given the comparable performance between both Deep Neuroevolution and Evolution Strategies, we adopt the simple ES implementation explored in the reinforcement learning application of Salimans et al. (2017) – an outline of which is presented in Algorithm 1 for reference.

Algorithm 1 Evolution Strategies

Input: Training data $\boldsymbol{x}$, Learning rate α, Noise Standard Deviation σ, Initial Weights θ_0

Initialise global optimal weights: $\boldsymbol{\theta}_g(0) = \theta_0$

for $k = 1$ **to** max iteration K **do**
  for $i = 1$ **to** N **do**

    Population Update:
    Sample $\boldsymbol{\epsilon}_i \sim N(0, I)$
    Update individuals $\boldsymbol{\theta}(i,k) \leftarrow \boldsymbol{\theta}_g(k) + \sigma\boldsymbol{\epsilon}_i$

    Score Computation:
    Compute Reward $R(i) = -\mathcal{L}\left(\boldsymbol{x}; \boldsymbol{\theta}(i,k)\right)$
  end for

  Set global weights:
  $\boldsymbol{\theta}_g(k+1) \leftarrow \boldsymbol{\theta}_g(k) + \frac{\alpha}{\sigma N} \sum_{i=1}^{N} R(i)\, \boldsymbol{\epsilon}_i$
end for

Here we define $\boldsymbol{\theta}(i,k) \in \mathbb{R}^C$ to be the vector of C RNN parameters for individual i at training iteration k, and $\mathcal{L}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ to be the loss function used for training given the input data and network parameters $\boldsymbol{\theta}$. We note that the loss function is computed here by conducting a full feed-forward pass across the network –avoiding any truncation of the data or minibatching beforehand.

3.2. Neuroparticle Swarm Optimisation

Given the relative simplicity of the mutation function used in ES, we also explore the use of more sophisticated population update rules through PSO – which we refer to as Neuroparticle Swarm Optimisation (NPSO) in the context RNN training.

Adopting the formulation of Shi & Eberhart (1998), a hyperparameter w is defined for inertial weights, and set the velocity $\boldsymbol{V}(i,k)$ and position $\boldsymbol{\theta}(i,k)$ as below for each training iteration.

$$\begin{aligned} \boldsymbol{\theta}(i,k) &= \boldsymbol{\theta}(i,k-1) + \boldsymbol{V}(i,k), \\ \boldsymbol{V}(i,k) &= w\, \boldsymbol{V}(i,k-1) \\ &\quad + c_1\, U_1(i,k)\ \left(\boldsymbol{\theta}_l(i,k-1) - \boldsymbol{\theta}(i,k-1)\right) \\ &\quad + c_2\, U_2(i,k)\ \left(\boldsymbol{\theta}_g(k-1) - \boldsymbol{\theta}(i,k-1)\right) \end{aligned} \tag{1}$$

where $c_1 = c_2 = 2$ are fixed constants, $U_1(i,k)$ and $U_2(i,k)$ are samples from standard uniform distributions $\mathcal{U}(0,1)$, $\boldsymbol{\theta}_l(i,k-1)$ is the best position observed locally by each particle, and $\boldsymbol{\theta}_g(k-1)$ is the best global position across all particles. A full description can be found in Algorithm 2 for additional clarity, noting that $\boldsymbol{\theta}_g(K)$ is used to generate forecasts at run-time.

Algorithm 2 Neuroparticle Swarm Optimisation

Input: Training data $\boldsymbol{x}$, Inertial Weight w,
Initial Weight Variance σ^2

Initialise $\forall i$:
$\boldsymbol{\theta}_g(0) = \boldsymbol{\theta}_l(i,0) = \boldsymbol{V}(i,0) = \boldsymbol{0}$, $\boldsymbol{\theta}(i,0) \sim N\left(0, \sigma^2 I\right)$
$\mathcal{L}^l_{min}(i) = \infty$, $\mathcal{L}^g_{min} = \infty$

for $k = 1$ **to** max iteration K **do**

  Population Update:
  $\boldsymbol{V}(i,k) \leftarrow \texttt{Update}(\boldsymbol{V}(i,k-1))$, using Equation (2)
  $\boldsymbol{\theta}(i,k) \leftarrow \boldsymbol{\theta}(i,k-1) + \boldsymbol{V}(i,k)$

  Score Computation:
  if $\mathcal{L}\left(\boldsymbol{x}; \boldsymbol{\theta}(i,k)\right) < \mathcal{L}_{min}(i)$ **then**

    $\boldsymbol{\theta}_l(i,k) \leftarrow \boldsymbol{\theta}(i,k)$
    $\mathcal{L}_{min}(i) \leftarrow \mathcal{L}\left(\boldsymbol{x}; \boldsymbol{\theta}(i,k)\right)$

    if $\mathcal{L}^l_{min}(i) < \mathcal{L}^g_{min}$ **then**
      $\boldsymbol{\theta}_g(k) \leftarrow \boldsymbol{\theta}_l(i,k)$
      $\mathcal{L}^g_{min} \leftarrow \mathcal{L}^l_{min}(i)$
    end if
  end if
end for

4. Experiments with Intraday Volatility Forecasting

To evaluate the effectiveness of the PBO in learning long-term dependencies, we apply our methods for training RNNs to the problem of volatility forecasting – a key area of interest in finance. Given the presence of volatility clustering at a daily time scales (Cont, 2001) and the evidence of intraday periodicity of returns volatility (Andersen & Bollerslev, 1997), volatility datasets present RNNs with a mixture of long-term and short-term relationships to be learnt – making them particularly relevant for our evaluation.

4.1. Description of Dataset

We consider the application of RNNs to forecasting 30-min intraday realised variances (Andersen et al., 2003) for FTSE 100 index returns. This was derived using 1-min index returns sub-sampled from Thomson Reuters Tick History Level 1 (TRTH L1) quote data from 4 January 2000 to 4 July 2018.

4.2. RNN Benchmarks

Tests are performed on a variety of modern RNN benchmarks as specified below:

- Standard LSTM (Hochreiter & Schmidhuber, 1997)
- Phased LSTM (P-LSTM) (Neil et al., 2016)
- Fourier Recurrent Unit (FRU) (Zhang et al., 2018)

As described in Section 2, both the P-LSTM and FRU are specifically designed for long-term dependency modelling – allowing us to determine if these relationships can be learnt using better architectures alone.

4.3. Training Methods

In addition, the following optimisation methods were tested in experiments:

- Stochastic Gradient Descent (SGD) with the Adam Optimiser(Kingma & Ba, 2015)
- Evolution Strategies (ES) (Salimans et al., 2017)
- Neuroparticle Swarm Optimisation (NPSO)

For the SGD approach, 100 iterations of random search are performed for hyperparameter optimisation with back-propagation performed up to a maximum of 300 epochs or convergence – making up a maximum of 30k feedforward passes through network during training. To explicitly consider the effects of short truncation windows, RNNs were only unrolled back 20 time steps for BPTT.

Using this to set the overall computational budget, evolutionary computation methods utilised a population of 30 particles over 50 training iterations – limiting to 20 iterations of random search for hyperparameter optimisation.

4.4. Results and Discussion

Network performance was evaluated using the mean-squared error (MSE) of one-step-ahead volatility forecasts, with results presented in Table 1 normalised by the MSE of the LSTM trained using SGD.

	SGD	ES	NPSO
LSTM	1.000	**0.189***	0.248
P-LSTM	11.272	0.189	**0.138***
FRU	5.446	**0.188***	268.441

Table 1. Normalised MSEs for Volatility Forecasts

From the MSEs reported, we can see that training RNNs using population-based approaches methods lead to significant improvements in predictive performance – with ES reducing MSEs by more than 80% on average. Performance improvements are also observed for architectures designed specifically with long-term dependencies in mind, overcoming the limitations with SGD.

While both ES and NPSO do lead to better RNN performance in general, apart from the NPSO-trained FRU which leads to large propagated errors, the simpler population update rules in ES appears to lead to more consistent results in general – with NPSO exhibiting a higher variance across the architectures. This could be attributed to the hyperparameter ranges selected for our initial population and inertial weights, and improved results can potentially be achieved through better hyperparameter search and varying the c_1 and c_2 parameters which are currently fixed.

Focusing on the SGD-trained models alone, we note that more sophisticated architectures underperformed compared to the standard LSTM for this specific volatility forecasting application. One possible reason is our use of very short truncated segments for BPTT – with RNNs unrolled for only 20 time steps – making it difficult for complex networks to learn the temporal relationships and resulting in overfitting.

5. Conclusions and Future Work

In this paper, we investigate the use of population-based global optimisation techniques for learning long-term dependencies with RNNs in time-series datasets. Testing this on an application in volatility forecasting, we observe that these gradient-free approaches help circumvent the issues observed with standard SGD optimisation, leading to better predictive performance across a variety of network architectures. While PBO does improve performance in general, simple evolution strategies appear to lead to more stable results in our specific application.

While our tests were performed on single workstations to ensure a comparable computational load to SGD, we note that ES is typically used with large distributed computing environments. As such, future extensions could achieve even better results by using ES with larger populations distributed over many parallel workers – unlocking the full potential of PBO for time series prediction tasks.

References

Andersen, T., Bollerslev, T., Diebold, F., and Labys, P. Modeling and forecasting realized volatility. *Econometrica*, 71(2):579–625, 2003.

Andersen, T. G. and Bollerslev, T. Intraday periodicity and volatility persistence in financial markets. *Journal of Empirical Finance*, 4(2):115 – 158, 1997. ISSN 0927-5398.

Chao, Z., Pu, F., Yin, Y., Han, B., , and Chen, X. Research on real-time local rainfall prediction based on MEMS sensors. *Journal of Sensors*, 2018.

Cont, R. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1:223–236, 2001.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 0899-7667.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

Kanuparthi, B., Arpit, D., Kerg, G., Ke, N. R., Mitliagkas, I., and Bengio, Y. h-detach: Modifying the LSTM gradient towards better optimization. In *International Conference on Learning Representations(ICLR)*, 2019.

Kennedy, J. and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Neil, D., Pfeiffer, M., and Liu, S.-C. Phased LSTM: Accelerating recurrent network training for long or event-based sequences. In *Proceedings of the 30th Conference on Neural Information Processing Systems, (NIPS)*, 2016.

Nweke, H. F., Teh, Y. W., Al-garadi, M. A., and Alo, U. R. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233 – 261, 2018. ISSN 0957-4174.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017. URL http://arxiv.org/abs/1703.03864.

Shi, Y. and Eberhart, R. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pp. 69–73, 1998.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep Neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR*, abs/1712.06567, 2017. URL http://arxiv.org/abs/1712.06567.

Trinh, T. H., Dai, A. M., Luong, M.-T., and Le, Q. V. Learning longer-term dependencies in RNNs with auxiliary losses. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Wu, G., Mallipeddi, R., and Suganthan, P. N. Ensemble strategies for population-based optimization algorithms – a survey. *Swarm and Evolutionary Computation*, 44:695 – 711, 2019. ISSN 2210-6502.

Zhang, J., Lin, Y., Song, Z., and Dhillon, I. S. Learning long term dependencies via fourier recurrent units. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Zhang, Z., Zohren, S., and Roberts, S. DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 2019.