A Variable Memory Length Auto Encoder

Shamahil Ibunu, Samuel Weller, Clive Cheong Took
Department of Electronic Engineering
Royal Holloway, The University of London
Egham, United Kingdom
{PHEE004, WEEE012}@live.rhul.ac.uk, Clive.CheongTook@rhul.ac.uk

***Abstract*—Auto-encoders typically require batch learning to be effective. There is a lack of online learning mechanisms for auto-encoders. To address this shortcoming in the literature, we propose an auto-encoder that can not only learn on a sample-by-sample basis without back-propagation but also has a memory to benefit from past learning. The memory can be adapted to fit the current state of the data by varying the memory length of the auto-encoder. Simulation supports our approach, especially when the data is nonstationary.**

***Index Terms*—Multi channel least mean square algorithm, auto-encoder, feed forward neural network, fractional memory length.**

I. Introduction

Feature extraction lies at the heart of auto-encoders. It is this ability to extract useful information without labelled data that has made auto-encoders so popular. As such, they have been widely used in various applications ranging from image compression [1], to pre-training of deep neural networks [2], to denoising [3].

To maximise feature extraction, the weights of auto-encoders are typically learned and adjusted after the presentation of all (or some) examples in the training set. In other words, auto-encoders are optimal when batch (or mini batch) learning is undertaken. However, there are cases where this is not always possible such as in streaming data or one-short learning. Small data typically pose a problem for batch learning. To this end, we address the problem of small data by proposing novel online learning for auto-encoders. To achieve the online learning of auto-encoders, we consider the multi-channel least mean square (MLMS) algorithm proposed in [4]. However, the MLMS algorithm is not a neural network; it can be regarded as a single multivariate perceptron and its architecture is not appropriate for encoding and decoding data so that feature extraction can be performed. More appropriate architectures are offered by the time-delay neural networks (TDNN) proposed in [5] and the non-linear auto-regressive networks with exogenous inputs (NARX) [6].

Both TDNN and NARX, however, cannot track local changes in the training data, especially when the environment for generating the data is non-stationary. This is due to their inability to adapt at how much data is used to solve current task. In other words, the recurrent use of past data is "fixed" by the maximum delay modelled in both TDNN and NARX. To this end, we propose a novel variable memory length auto-encoder, which can:

1) Auto-encode its input requiring only on its current and past samples of the current data rather than the whole dataset. In this way, it operates in an online fashion rather than undertakes batch learning;
2) Adapt the maximum delay on the past samples considered so that non-stationarity of the data can be accounted for.

This paper is organised as follows. Section I sets the scene in terms of shortcomings in the literature and the motivation for our proposed method, whereas Section II provides the necessary background to derive our proposed auto-encoder as well as its details. Simulation results are provided in Section III, followed by the conclusion in Section IV.

II. Online Learning

To derive our proposed variable memory length auto-encoder, we first need to revisit the MLMS algorithm and then illustrate how the variable memory length works.

A. Multi-Channel LMS

The MLMS introduced in 1985 [4] has been particularly successful in signal processing due to its efficacy in real-time processing in audio applications [7]. We exploit the MLMS algorithm to update the synaptic weights of our proposed auto-encoder by minimising the following cost function:

$$\begin{aligned} J(n) &= \sum_{\ell=1}^{K} J_\ell(n) = \sum_{\ell=1}^{K} |e_\ell(n)|^2 \\ &= \sum_{\ell=1}^{K} |d_\ell(n) - y_\ell(n)|^2 \end{aligned} \tag{1}$$

where $e_\ell(n)$, $y_\ell(n)$, and $d_\ell(n)$, denote the error, the estimated output, and the teaching data respectively. For generality, 'k'', 'ℓ'' and '$(n)'$' reflect the kth input index and the ℓth index output and the time index respectively. Observe that the time index in Eq. (1) models the *instantaneous* quadratic error, thereby enabling online learning. It is noteworthy to distinguish that the summation in (1) corresponds to summing all errors over one example rather than summing all errors over all examples. Thus, (1) is neither the batch nor the mini batch quadratic error, but the *instantaneous* quadratic error. The kth output $y_k(n)$ can be computed as

$$y_\ell(n) = \sum_{k=1}^{K} \mathbf{w}_{k\ell}^T(n)\mathbf{x}_k(n) \qquad \ell = 1, 2...K \tag{2}$$

where $\mathbf{w}_{k\ell}(n)$ is the learning weight vector corresponding to the input vector $\mathbf{x}_k(n)$. More specifically,

$$\mathbf{w}_{k\ell}(n) = [w_{k\ell,1}(n), w_{k\ell,2}(n), \ldots, w_{k\ell,L}(n)]^T \quad (3)$$
$$\mathbf{x}_k(n) = [x_k(n), x_k(n-1), \ldots x_k(n-L+1)]^T \quad (4)$$

where the superscript $(.)^T$ denotes the transpose operator. The memory length L dictates how much past data is used to estimate $d_k(n)$ and models the maximum delay considered in TDNN or NARX. Alternatively, all the outputs of the MLMS algorithm can be calculated as

$$\mathbf{y}(n) = \mathbf{W}^T(n)\mathbf{x}(n)$$

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ . \\ . \\ . \\ y_K(n) \end{bmatrix} = \begin{bmatrix} \mathbf{w}_{11} & \mathbf{w}_{12} & . & . & . & \mathbf{w}_{1K} \\ \mathbf{w}_{21} & \mathbf{w}_{22} & . & . & . & \mathbf{w}_{2K} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \mathbf{w}_{K1} & \mathbf{w}_{K2} & . & . & . & \mathbf{w}_{KK} \end{bmatrix}^T \begin{bmatrix} \mathbf{x}_1(n) \\ \mathbf{x}_2(n) \\ . \\ . \\ . \\ \mathbf{x}_K(n) \end{bmatrix} \quad (5)$$

The ℓth column of $\mathbf{W}$ can be updated as

$$\mathbf{w}_\ell(n+1) = \mathbf{w}_\ell(n) - \mu \nabla J_\ell$$
$$\mathbf{w}_\ell(n+1) = \mathbf{w}_\ell(n) + \mu \mathbf{x}(n) e_\ell(n) \quad (6)$$

Notice that Eq. (5) updates a vector of weight elements, whereby each element $\mathbf{w}_{k\ell}(n)$ is also a vector of memory length L, i.e. Eq. (3). To track the *local* changes within the data $\mathbf{x}$, we now proceed on illustrating how the optimal memory length L can be learnt in the next section.

B. The Variable Memory Length Algorithm

The variable *tap* length algorithm was proposed to optimise the memory length L of the least mean square (LMS) algorithm [8]. This was achieved by minimising the instantaneous quadratic $e^2(n)$ similar as in Eq. (1), albeit for one channel instead of several channels as in $\sum_{\ell=1}^{K} e_\ell^2(n)$ - as proposed in this paper.

To guide the MLMS algorithm towards the optimal memory length, a pair of errors are used, i.e.

$$e_{\ell,N}(n) = d_\ell(n) - \sum_{k=1}^{K} \mathbf{w}_{k\ell,1:N}^T(n)\mathbf{x}_{k,1:N}(n) \quad (7)$$
$$e_{\ell,L}(n) = d_\ell(n) - \sum_{k=1}^{K} \mathbf{w}_{k\ell,1:L}^T(n)\mathbf{x}_{k,1:L}(n) \quad (8)$$

where the subscript $(.)_{1:\tau}$ means that the first τ elements of the vector are used. Similarly, the subscript $(.)^\tau$ indicates that the error was computed based on the first τ elements of the input vector in (4). Note that $1 \leq N \leq L$, where L represent the actual memory length and $N = L - \Delta$ where Δ is small positive integer.

These two errors in (7)-(8) can then be employed to update the memory length in the form of the fractional memory length f_t as

$$f_t(n+1) = f_t(n) - \beta \left[\sum_{\ell=1}^{K} e_{\ell,L}^2(n) - e_{\ell,N}^2(n) \right] \quad (9)$$

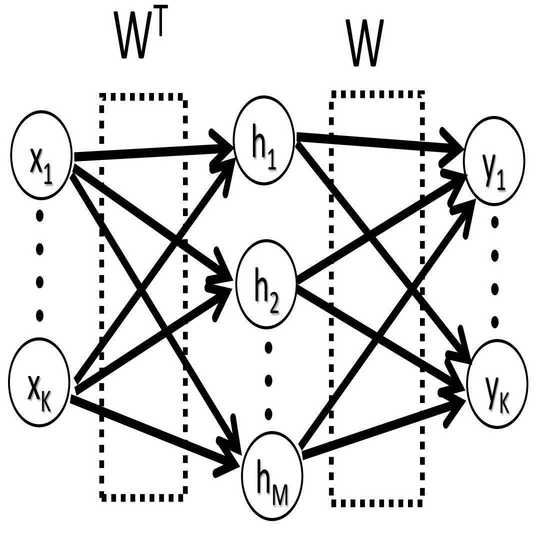

Fig. 1. Structure of proposed auto-encoder without backpropagation.

where β is the learning step size. As its name indicates the fractional memory length is not guaranteed to be an integer. To address this issue, the memory length is instead updated as

$$\begin{aligned} L(n+1) &= \text{floor}\Big(f_t(n)\Big) \quad \text{if} \quad |L(n) - f_t(n)| > \delta \\ &= L(n) \qquad\qquad \text{otherwise} \end{aligned} \quad (10)$$

where floor(.) rounds down its argument to the nearest integer and δ is a small positive integer. This concludes the variable memory length for our proposed auto-encoder, which is next described.

C. The proposed variable memory length auto encoder

Recall that MLMS does not have the structure of a neural network but that of a multi-channel perceptron. As such, it lacks the depth of an auto-encoder. The lack of depth implies that MLMS cannot auto-encode its input to its output. To this end, our auto-encoder is designed as shown in Fig. 1. More specifically, the weights $\mathbf{W}$ in (5) are optimised by MLMS as in (6), and the same weights connect the hidden layer to the input layer by transposing $\mathbf{W}$. In this way, there is no need to derive the back-propagation mechanism, which entails the following advantages:

- A reduction in computational cost to better cope with streaming data or applications that require real-time applications;
- A lower risk of overfitting the data, since there are less parameters to be optimised;
- An enhanced adaptability (and quicker convergence) of the auto-encoder for non-stationary data.

With the structure shown in Fig. 1, our proposed auto-encoder can not only auto-encodes its input, but also benefit from the online learning of MLMS. More importantly, our proposed auto-encoder can also adapt its memory length as described in Section II-B to track changes within the data.

III. Simulation

This section assesses the performance of our proposed method on two types of machine learning problems: time series prediction and image reconstruction. The nonlinear autoregressive exogenous (NARX) algorithm was considered as benchmark in the time series prediction, whereas a feedforward auto-encoder with the same architecture as in Fig. 1 was considered in the image reconstruction problem, albeit optimised with back-propagation.

A. Time series prediction

For this particular task, a one-step ahead forecasting was performed on two nonstationary signals, i.e 4D Saito signal [9] in Fig. 4 and 3D Lorenz signal [10] in Fig. 5. Fig. 2 shows the sample autocorrelation for both signals. Notice that the autocorrelation for Lorenz tends to degrade much quicker to zero (below 20%) than Saito. In other words, the degree of non-stationarity for the Saito signal is much stronger. As such, we expect that our proposed auto-encoder will tend to perform better on the Saito signal than the Lorenz signal.

1) 4D Saito Signal: The topology used for both NARX and our proposed auto-encoder was 4 (inputs)- 2 (hidden) - 4 (outputs). Fig. 3 shows how the memory length of the proposed auto-encoder varies across time. Notice that when the Saito signal has abrupt changes to high magnitudes, the error also increases forcing the memory of our auto-encoder to have instantaneous decrease in length. It is these abrupt changes that pose a challenge for NARX to adapt as shown in Fig. 4. The mean square error (MSE) also confirms the superiority of our approach, i.e. 1.09×10^{-2} dB for our method and 1.23×10^{-2} dB for NARX.

2) 3D Lorenz Signal: The topology used for both NARX and our proposed auto-encoder was set to 3 (inputs)- 2 (hidden) - 3 (outputs). In this particular experiment, the advantage of our proposed method is less obvious as shown in Fig. 5, as the degree of non-stationarity of Lorenz is much weaker than Saito. As expected, the difference in MSE was also less prominent, i.e. 5.11×10^{-2} dB (our method) and 5.12×10^{-2} dB (NARX).

B. Image reconstruction

This experiment assessed the traditional feedfoward auto-encoder that operates on mini batch learning compared to our online learning of our proposed auto-encoder. To demonstrate the efficacy of online learning, a *single* image was considered from the well-known MNIST dataset (28 x 28 pixels) to undertake one-shot learning. The topology of the neural networks was set to 28 (inputs) - 14 (hidden) - 28 (outputs). The memory length of our proposed method was fixed at 7 samples. For a fair comparison, the image was partitioned into 28 x 7 pixels and fed 4 times to the traditional autoencoder as inputs. Fig. 6 shows the images reconstructed by both methods. The Pearson correlation coefficients between the original and reconstructed images were 0.37 (37%) and 0.63 (63%) for the benchmark and our method respectively.

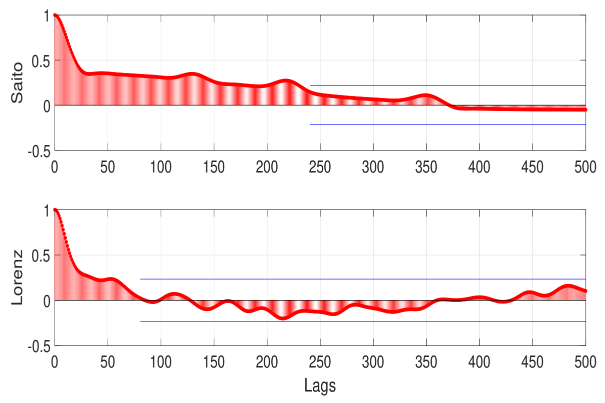

Fig. 2. Sample auto-correlation to illustrate the higher degree of nonstationarity of Saito signal [9] than Lorenz signal [10].

IV. Conclusion

We have proposed an auto-encoder which can learn on a sample-by-sample basis to cope with real-time applications. To put our auto-encoder on parity with batch-learning, we have introduced memory for auto-encoders so that they benefit from their past learning (albeit on limited data rather than having access to the whole dataset). To enhance the adaptability of our proposed method, we have borrowed the concept of variable tap length from adaptive filtering to neural networks. Simulation supports our approach for both time series prediction and image processing.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] Y. Cai and Z. Cai and M. Zeng and X. Liu and J. Wu and G. Wang, "A novel deep learning approach: Stacked evolutionary auto-encoder," *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2018.

[3] W. Xiong and B. Du and L. Zhang and L. Zhang and D. Tao, "Denoising auto-encoders toward robust unsupervised feature representation," *International Joint Conference on Neural Networks (IJCNN)*, pp. 4721–4728, 2016.

[4] S. J. Elliott and P. A. Nelson, "Algorithm for multichannel lms adaptive filtering," *Electronics Letters*, vol. 21, no. 21, pp. 979–981, 1985.

[5] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[6] E. Diaconescu, "The use of narx neural networks to predict chaotic time series," *WSEAS Transactions on Computer Research*, vol. 3, 03 2008.

[7] Y. Huang and J. Benesty, *Audio Signal Processing for Next-Generation Multimedia Communication Systems*. USA: Kluwer Academic Publishers, 2004.

[8] Yu Gong and C. F. N. Cowan, "An lms style variable tap-length algorithm for structure adaptation," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2400–2407, 2005.

[9] P. Arena and L. Fortuna and G. Muscato and M. G. Xibilia, "Neural networks in multidimensional domains," *Lecture Notes in Control and Information Sciences (Springer Verlag)*, vol. 234, 1998.

[10] Saha, D. C. and Ray, Anirban and Chowdhury, A. Roy, "Electronic circuit simulation of the lorenz model with general circulation," *International Journal of Physics*, vol. 2, no. 5, pp. 124–128, 2014.

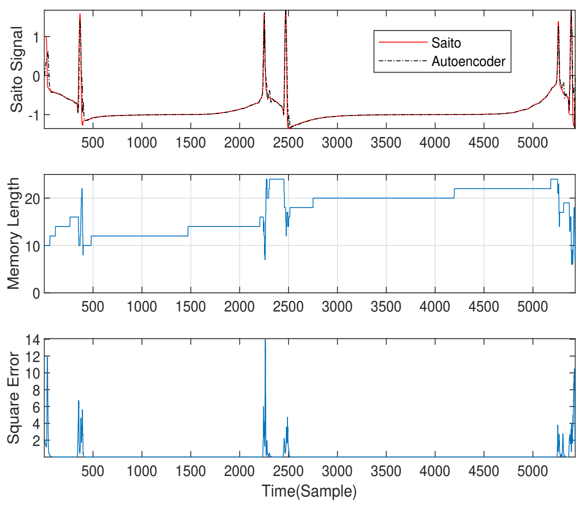

Fig. 3. Evolution of the memory length of our proposed method.

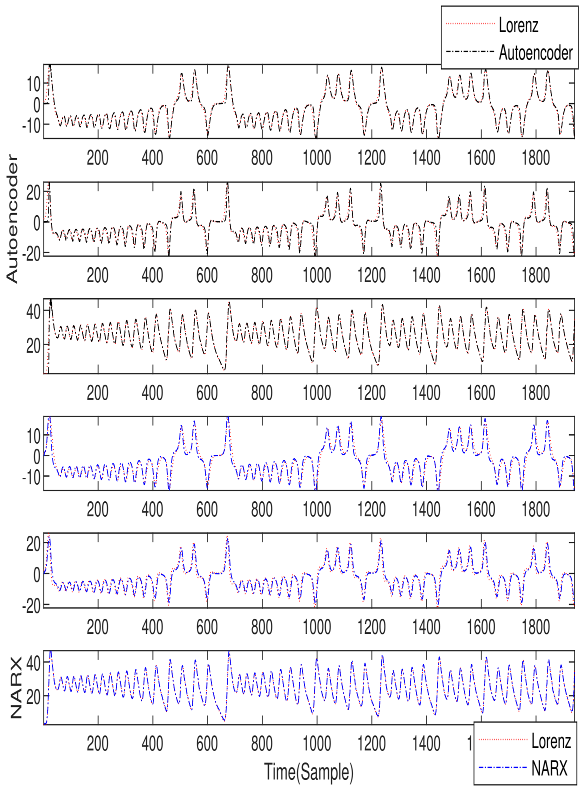

Fig. 5. Performance of proposed method against NARX on 3D Lorenz signal.

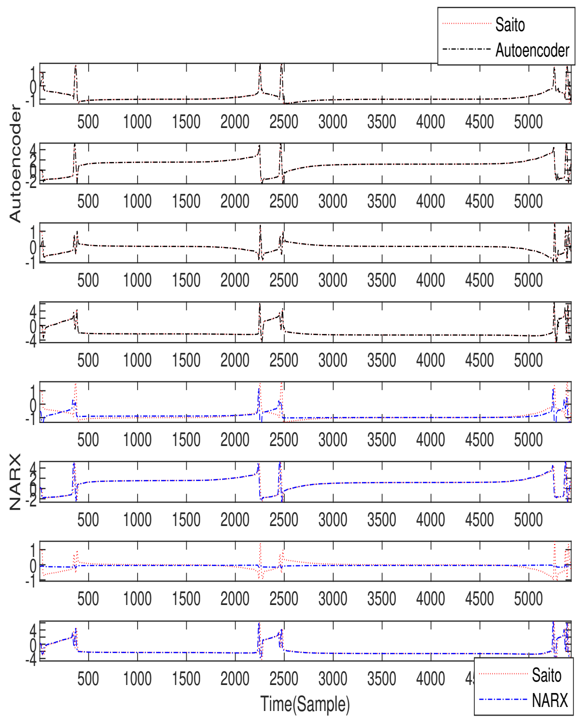

Fig. 4. Performance of proposed method against NARX on 4D Saito signal.

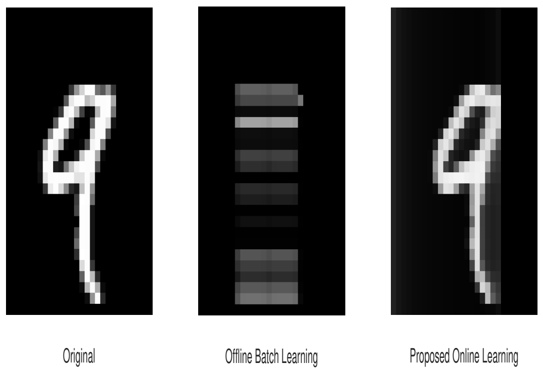

Fig. 6. Image reconstruction in one-shot learning: a traditional auto-encoder against our method.