Training without Gradients - A Filtering Approach

Isaac Yaesh
Control Dept.
Elbit Systems
Ramat Hasharon, Israel
itzhak.yaesh@elbitsystems.com

Natan Grinfeld
Control Dept.
Elbit Systems
Ramat Hasharon, Israel
nati.grinfeld@elbitsystems.com

Abstract—A particle filtering approach is suggested for the training of multi-layer neural networks without utilizing gradients calculation. The network weights are considered to be the components of the estimated state-vector of a noise driven linear system, whereas the neural network serves as the measurement function in the estimation problem. A simple example is used to provide a preliminary demonstration of the concept, which remains to be further studied for training deep neural networks.

I. INTRODUCTION

Artificial Neural Networks (ANN) are motivated by the nonlinear, complex and parallel structure of the human brain and are used in various applications such as control of nonlinear systems, system identification and classification. Traditional neural networks architectures include single and multi layer feed-forward networks as well as recurrent networks and they are based on interconnections of neurons of the same type. The feed-forward networks are usually trained by the back propagation algorithm (see e.g. [1]). However, back-propagation based training may face vanishing gradients as the result of the products of e.g. partial derivative of hyperbolic tangent functions which possess gradients in the range of (0, 1). A few solutions have been proposed to alleviate the vanishing gradient problem, such as using ReLU activation functions, as well as pre-training one network level at a time and fine tuning using back propagation. A widely used approach to deal with the vanishing gradient problem is the one of ResNet which involves splitting the network to chunks and direct passes which actually make deep networks. ResNet indeed reduces much of the vanishing gradient problem but as they are in a way a cascade of shallow networks, they do not avoid the calculation of gradients. One approach which avoids calculation of gradients is the ADMM (Alternating Direction Method of Multipliers [2]), where the network weights optimization problem is broken into a sequence of minimization sub-steps where each step can be globally optimized, however, originally not guaranteeing global convergence of the overall scheme. More recently such convergence was established [3] under some conditions. Yet we would like to suggest the entirely different approach of Particle Filtering with implicit guarantee of global convergence and no need in calculating gradients. To this end, it is useful to recall a relation between filtering and back-propagation. The latter relation has been established in e.g. [4]. It was shown there, that $H_{\infty}$-optimal filtering HF is closely related to back-propagation based training of neural networks. A similar result appears in [5]. Note that in HF the maximum ratio between the energy of the network output prediction errors and the energy of disturbances (i.e. observation noise) is minimized as oppose to the Extended Kalman filter (EKF) which is a finite dimensional approximation to the optimal Minimal Mean Square Error (MMSE) estimate for the network weights. However, in the case of general multi-layer networks, gradients are still needed both in EKF and HF as it will be shown in the next section. In contrast, in Affine Functional Networks (AFN), a special case of Functional Neural Networks [6] with an affine relation between network output and its weights, the use of gradients is not needed. For this case, in [7] an $H_{\infty}$-filter has been derived, which was shown to be superior over the Kalman filter, being effective and a possible solution to the training of AFN. In AFN [7] the network weights are regarded as the components of the state-vector, which is estimated using either a standard Kalman filter or $H_{\infty}$ filter, both avoiding the use of gradients. Unfortunately, in most of the applications, multi-layer networks with a nonlinear relation rather than a linear or affine one between the network weights and output are used, and both the Kalman filter and $H_{\infty}$-optimal filters are to be replaced by approximations (e.g. Extended Kalman Filter - EKF) requiring the calculation of gradients.

It is well known the MMSE is just a conditional expectation $\hat{x} = E\{x(t)/Y(t)\}$ of the state-vector $x(t)$ (i.e. the weights vector) on the history of the measurements $Y = \{y(\tau), \tau \leq t\}$ (i.e. training example outputs). In the general nonlinear networks case, the conditional expectation requires an infinite-dimensional filter [8] which can be approximated in various ways, such as the EKF which can, however, even diverge in some special nonlinear cases. The most generally effective and widely used approximation of the MMSE is the particle filter (see e.g. [9]) (PF), where a finite-dimensional version of the conditional probability distribution function (PDF) is estimated where it is represented by points on the multidimensional graph known as particles.

II. PROBLEM FORMULATION

Consider the following training problem:

$$y(t) = g(x(t), u(t)) + v(t)$$  \hspace{1cm} (1)
where \( x \) is the weights vector in a neural network \( g(x, u) \) that maps the input vector \( u \) to the output \( y \). In we denote by \( u(t) \) the \( t \)th input pattern and by \( y(t) \) the corresponding observed output pattern. The signal \( v(t) \) represents either the observation error or the modeling error.

Our aim is to estimate the weights vector \( x \) from the measurements \( y(t), t = 1, 2, ..., N \), given an initial estimate \( x_0 \). Although the weights vector \( x \) is constant (but unknown), following we model it by

\[
x(t + 1) = x(t) + w(t)
\]

where \( w(t) \) is the uncertainty in \( x \) assumed to be a normally distributed white noise sequence with intensity \( Q \), namely

\[
E\{w(t)w(\tau)\} = Q\delta(t, \tau)
\]

where \( \delta \) is the Kronecker delta function, namely attains the value of 1 for \( t = \tau \) and zero elsewhere, and where \( Q \) may serve as a tuning parameter chosen according to the trade off between fast convergence and accuracy. Similarly, \( v \) is modeled also as a white noise sequence independent of \( w \), where \( E\{v(t)v(\tau)\} = R\delta(t, \tau) \) with \( R \) being another parameter for tuning. Using a sequential estimation scheme, we may want to estimate \( x(k) \) by \( \hat{x}(k) \) on the basis of accumulated measurements \( Y(t) = \{y(\tau), \tau \in \{0, 1, 2, ..., t\}\} \), so that eventually at \( t = N \) for large enough \( N \), we will get that \( \hat{x}(t) \approx x \). We define the error signal at step \( t \) by

\[
e(t) = x(t) - \hat{x}(t)
\]

and also define the following cost function

\[
\bar{J} = E\{e(t)e^T(t)\}
\]

Namely, we would like to minimize the Mean Square Estimation error covariance at each moment of the estimation process, namely at each training instance. At this point, one may wonder, whether the assumption of normal distribution of \( w \) leading to a normally distributed weights vector \( x \) can be justified. In our example we will see that this indeed the case. This assumption is supported to some extent in [10].

**Remark**: In the sequel we freely switch between e.g. \( \hat{x}(t) \) and \( x_t \) meaning the same, when it serves the simplicity of representations.

### III. Training Algorithm

It is well known (e.g. [8]) that,

\[
\hat{x}(t) = E(x|Y) := \int_{-\infty}^{\infty} p(x|Y)dx
\]

where \( p(x|Y) \) is the conditional probability density of \( x \) given \( Y \) and \( E(x|Y) \) is the corresponding conditional expectation of the weights (i.e. state-vector) on the training examples outputs given by (1). Note that \( \hat{x} \) is a function of \( Y \). For the sake of completeness we further note that this result readily follows considering a general estimator of the form \( \hat{x} = f(y) \). We would like to minimize the following loss function

\[
E\{(x - f(y))^2\} = \int_{-\infty}^{\infty} p(y) \int_{-\infty}^{\infty} (x - f(y))^2 p(x|y) dx dy
\]

The result of (6) readily follows then from minimization of \( \int_{-\infty}^{\infty} (x - f(y))^2 p(x|y) dx \).

Since (1) defines a nonlinear relation between the weights and the training examples, we resort to using the Particle filtering algorithm that can deal with such a nonlinear situation without needing to calculate any gradients. As with the Kalman filter, the Particle filter [9] employs two basic steps, the first one being a prediction step which takes the previous posterior (i.e. the conditional estimate \( p(x_t|Y_{t-1}) \) and using the state-vector evolution model of (2), namely, \( p(x_t|x_{t-1}) \), computes

\[
p(x_t|Y_{t-1}) = \int p(x_t|x_{t-1}) p(x_{t-1}|Y_{t-1}) dx_{t-1}
\]

The second step, again similarly to the Kalman filter, is the measurement update step which is based on the Chapman-Kolmogorov identity

\[
p(x_t|Y_t) = P(x_t|Y_{t-1})p(y_t|x_t)/p(y_t|Y_{t-1})
\]

where the denominator of the latter equation, is computed applying

\[
p(y_t|Y_{t-1}) = \int p(y_t|x_t) p(x_t|Y_{t-1}) dx_t
\]

Equations (7)-(9) summarize the basic concepts of the Particle Filtering algorithm, where the probability density functions are represented by a finite number of samples of the state vector (also known as 'particles'), rather than the functions themselves. However, to make it work, some additional considerations such as importance sampling and re-sampling are to be employed. We note that we did not try to implement the algorithm of Particle Filtering, but rather used the one implemented within the control toolbox of MATLAB.
and the input vector by
\[
\mathbf{u}(t) = \begin{bmatrix} \xi(t-1) & \xi(t-2) \end{bmatrix}^T
\]
we note that the output of the oscillator is given by \( u(t) \), where
\[
\begin{align*}
g(x, u(t)) &= x_1 + x_2 u_2(t) + x_3 u_2(t)^2 \\
&\quad + x_4 u_1(t) + x_5 u_1(t)^2 := C(u(t)) x(t)
\end{align*}
\]
where
\[
C(u(t)) = \begin{bmatrix} 1 & u_2(t) & u_2(t)^2 & u_1(t) & u_1(t)^2 \end{bmatrix}
\]
Noting, however, that having the noise corrupted outputs of the oscillator only, rather than its inner states, for training purposes, actually
\[
\mathbf{u}(t) = \begin{bmatrix} y(t-1) & y(t-2) \end{bmatrix}
\]
The training experiment is, therefore, based on running the Henon oscillator as a ‘black-box’ while we log its outputs \( y(t) \) of \( u(t) \). The delayed versions of \( y(t) \) are then used in \( \mathbf{u}(t) \) to form the network input vectors. Finally, the pairs \( \{ y(t), u(t) \} \) are the training examples that we use as inputs to the Particle Filter based training algorithm. We note that the same example was used in \( \mathbf{u}(t) \) using both a Kalman filter and \( H_\infty \) filter. Here we use a Particle filter which is capable of training also general ANN rather than FNN, but we choose to test PF using this simple example, to verify that PF is not inferior to the Kalman filter and \( H_\infty \) filter which are suited to deal with the linear case, and are not approximate solutions, in contrast to the PF which is not exact due to its finite number of particles.

The training results which were obtained using \( Q = 0.5 \sqrt{\Delta T} \), where \( \Delta T = 0.016 \) and \( R = 0.2 \). The training results are depicted in Fig. 1, where we compare the time-history trained weights to the nominal true values. A satisfactory convergence of the weights to the true values is observed, following an initial transient. We next perform a simulation of the equation
\[
\begin{align*}
\xi(t) &= \hat{x}_1 + \hat{x}_2 \xi(t-2) + \hat{x}_3 \xi(t-2)^2 \\
&\quad + \hat{x}_4 \xi(t-1) + \hat{x}_5 \xi(t-1)^2
\end{align*}
\]
The results of simulating the latter equation is shown in Fig. 2. clearly showing the classical Henon map.

V. CONCLUSIONS

A particle filtering based training concept for training general artificial neural networks, alleviating the difficulty involved in gradient based training processes such as back propagation is suggested. While testing the suggested approach to a deep learning problem is necessary to establish its tractability, the suggested approach was demonstrated by considering a simple problem which has the same structure as general problems have, but due to its affinity in the weights, is solvable also using a Kalman filter and thus has a known global minimum. Therefore, the fact that the particle filter approach, in spite of being an approximate one (due to the finite number of particles), performed as good as the Kalman filter, should serve as an initial proof of concept and may encourage testing more involved training problems to the extent of dealing with deep learning problems. The fact that the particle filter may be implemented in a scalable manner further motivates study of deep learning applications.

REFERENCES

[1] Simon Haykin. Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.
[2] Taylor Gavin, Ryan Buremeister, Zheng Xu, Bharath Sing, Ankit Patel, and Goldstein Tom. Training neural networks without gradients: A scalable admm approach. In arXiv:1605.02026v1 [cs.LG], 2016.
[3] Junxiang Wang, Fuxun Yu, Xiang Chen, and Liang Zhao. Admm for efficient deep learning with global convergence. In KDD ’19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining., pages 39–44, 2019.
[4] Babak Hassibi, Ali H. Sayed, and Thomas Kailath. Lms is \( h_\infty \) optimal. In , volume 1, pages 74 – 79 vol.1, 01 1994.
[5] Babak Hassibi and Thomas Kailath. \( h_\infty \) optimal training algorithms and their relation to backpropagation. In Advances in Neural Information Processing Systems, pages 191–198, 1995.
[6] Enrique Castillo, Angel Cobo, JM Gutierrez, and E Pruneda. An introduction to functional networks with applications. In A Neural Based Paradigm. Kluwer Academic Publishers, 1998.
[7] Isaac Yaesh, Noelia Sánchez Maroño, and Uri Shaked. Robust terminal filtering in an \( h_\infty \) setting-application to affine functional networks training. In 2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEi), pages 1–5. IEEE, 2014.
[8] Andrew H Hazewinkel. Stochastic processes and filtering theory. Courier Corporation, 2007.
[9] M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing, 50:174–188, 2002.
[10] I Bellido and Fiesler E. Do back propagation. In Proceedings of ICANN, 1993.
[11] Noelia Sánchez Maroño, Oscar Fontenla-Romero, Amparo Alonso-Betanzos, and Bertha Gutiérrez-Galindo. Self-organizing maps and functional networks for local dynamic modeling. In , pages 39–44, 01 2003.
[12] Michel Hénon. A two-dimensional mapping with a strange attractor. In The Theory of Chaotic Attractors, pages 94–102. Springer, 1976.
[13] Penalkumar Engineer, Rajshri Velmurugan, and Sachin Patkar. Scalable implementation of particle filter-based visual object tracking on network-on-chip (noc). In J Real-Time Image Proc 17, 1117–1134, pages 1117–1134, 2020.
Fig. 1. Convergence of weights

Fig. 2. Simulation results with training results for weights