Sequential Extended Kalman Filtering

Training Neural Networks Using
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
Training Neural Networks Using Sequential Extended Kalman Filtering†

Edward S. Plumer
Los Alamos National Laboratory, MS F663, Los Alamos NM, 87545. (esp@glitner.lanl.gov)

Abstract

Recent work has demonstrated the use of the extended Kalman filter (EKF) as an alternative to gradient-descent backpropagation when training multi-layer perceptrons. The EKF approach significantly improves convergence properties but at the cost of greater storage and computational complexity. Feldkamp et al. have described a decoupled version of the EKF which preserves the training advantages of the general EKF but which reduces the storage and computational requirements. This paper reviews the general and decoupled EKF approaches and presents sequentialized versions which provide further computational savings over the batch forms. The usefulness of the sequentialized EKF algorithms is demonstrated on a pattern classification problem.

1 Introduction

There has been much recent interest in the application of Kalman filtering techniques to the training of multi-layer perceptron neural networks (MLP). Backpropagation (BP), the most popular algorithm for training such networks, suffers from slow convergence, especially for large network structures. Further, BP is ill-suited to handling highly time-correlated input sequences.

Direct application of the extended Kalman filter (EKF) to MLP training was demonstrated by Singhal and Wu [7]. Their implementation, referred to herein as the global extended Kalman filter (GEKF), provided better performance than backpropagation but at the expense of significantly greater computational and storage complexity. A simplification to the GEKF called the multiple extended Kalman algorithm (MEKA) was studied by Shah and Palmieri [6]. This variant trained each node of the network using a separate EKF and resulted in substantially reduced complexity.

Arguing that such independent filters can lead to unstable behavior and poor performance, Feldkamp and Puskorius [4] have detailed a modification of the GEKF called the decoupled extended Kalman filter (DEKF) which reduces the required storage and computational complexity while still preserving the salient features of the GEKF.

This paper presents sequential versions of both the GEKF and DEKF algorithms. In linear Kalman filters, such sequential forms have been employed to reduce the computational complexity and eliminate the need for matrix inversions [1]. The derivations presented herein extend these sequential forms to the case of nonlinear extended Kalman filtering. The computational complexity and efficacy of these various EKF algorithms are then compared.

2 General Extended Kalman Filter

In training neural networks using EKF methods, we model the system that gives rise to the input/output training pairs as an MLP and treat the weights $w(n)$ as unknown "states" of the system. An estimate of the estimation error covariance $P(n)$ is also maintained. The desired output responses $y(n)$ are then considered to be "output measurements" from which the correct states are estimated asymptotically using standard extended Kalman filtering theory. Any noise in the desired output response is modeled as a zero-mean white gaussian process with covariance $R$. Since the ideal weights are usually constant, the time-update portion of the EKF algorithm is trivial. However, non-stationary systems can be modeled by introducing artificial process noise and assuming that the ideal set of weights evolves as a Wiener process with covariance $Q$.

†This work was performed under the auspices of the U.S. Department of Energy.
As in backpropagation, the output-prediction error is computed after each input data presentation according to 
\[ e(n) = y(n) - \phi(\hat{y}(n), x(n)) \]
where the network output is given by \( \phi \). Subsequently, backpropagation of unit basis vectors is used to build up, one row at a time, the output-prediction Jacobian 
\[ H = \frac{\partial \phi(\hat{y}(n), x(n))}{\partial w} \]

### 2.1 Batch Measurement Update

Once \( e \) and \( H \) have been computed, the estimated weight vector and error covariance matrix are updated according to:

\[
\begin{align*}
\Gamma &= P(n)H^T \\
\Delta &= R + H\Gamma \\
K &= \Gamma\Delta^{-1} \\
\hat{w}(n+1) &= \hat{w}(n) + Ke \\
P(n+1) &= P(n) - K\Gamma^T + Q
\end{align*}
\]

These equations define the batch form of the global extended Kalman filter (batch-GEKF) since the update combines the effect of all the measurements at the current time step. Due to the matrix multiplications and the matrix inversion, this batch update can be computationally tedious.

### 2.2 Sequential Measurement Update

Sequential update forms have been used in linear Kalman filtering to reduce the computational burden of the measurement update by processing the output measurements individually [1]. This section presents the analogous result for the extended Kalman filter. This sequential form provides exactly the same weight update history as the batch form but at a computational savings. Details of the relevant derivations are provided elsewhere [3].

The necessary condition for applying the sequential algorithm is that the measurement noises be uncorrelated resulting in 
\[ R = \text{diag}[r_1, \ldots, r_{N_y}] \]
where \( N_y \) is the number of output measurements. Given this condition, The sequential GEKF consists of first initializing the variables \( \Delta \hat{w}_0 = 0 \) and \( P_0 = P(n) \). Second, the recursive updates

\[
\begin{align*}
\gamma_j &= P_{j-1}h_j^T \\
\lambda_j &= r_j + h_j\gamma_j \\
\tilde{k}_j &= \frac{\gamma_j}{\lambda_j} \\
\Delta \hat{w}_j &= \Delta \hat{w}_{j-1} + \tilde{k}_j(e_j - h_j \Delta \hat{w}_{j-1}) \\
P_j &= P_{j-1} - \tilde{k}_jh_j^T
\end{align*}
\]

are computed for each measurement \( j = 1, \ldots, N_y \). Finally, the estimates are updated to give \( \hat{w}(n+1) = \hat{w}(n) + \Delta \hat{w}_{N_y} \) and \( P(n+1) = P_{N_y} + Q \). Note that the gains \( \tilde{k} \) in equation 2 are only indirectly related to the batch gain matrix \( K \).

The steps in equation 2 are almost identical to the simpler update performed when there is only one measurement. The important difference is the presence of the term \(-h_j \Delta \hat{w}_{j-1}\), the meaning of which is evident after some manipulation:

\[
e_j - h_j \Delta \hat{w}_{j-1} = (y_j - \hat{y}_j) - h_j(\hat{w}_{j-1} - \hat{w}_0) = y_j - \phi(\hat{w}_0, x) - \frac{\partial \phi}{\partial \hat{w}}(\hat{w}_{j-1} - \hat{w}_0)
\]

The final expression is a first order Taylor expansion for the modified error \( e'_j = y_j - \phi(\hat{w}_{j-1}, x) \). Therefore, the additional term accounts for the fact that the weight estimate \( \hat{w}_j \), and therefore the corresponding output measurement prediction \( \hat{y}_j \), changes at each step. However, actually recomputing the measurement error \( e'_j \) would require multiple forward propagations of the network and, in fact, would lead to a different weight update than the batch form.
3 Decoupled Extended Kalman Filter

Regardless of whether the batch or sequential form of the GEKF is chosen, the matrix $P$ must still be stored and updated. This $N_w \times N_w$ matrix can become extremely large, even for moderate sized networks. Feldkamp and Puskorius have described a modification to the GEKF called the decoupled extended Kalman filter (DEKF) [4] which remedies this problem. The DEKF algorithm consists of partitioning the set of weights into $G$ groups each having $N_g$ weights: $w = [w_1^T, \ldots, w_G^T]^T$. The error-covariance $P = \text{diag}[P_1, \ldots, P_G]$ and output-prediction Jacobian $H = [H_1, \ldots, H_G]$ are likewise partitioned.

3.1 Batch Measurement Update

In the batch form of the DEKF, given by Feldkamp and Puskorius, each group $g = 1, \ldots, G$ is updated according to:

$$
\begin{align*}
\Gamma_g &= P_g(n)H_g^T \\
A &= R + \sum_{g=1}^{G} H_g \Gamma_g \\
K_g &= \Gamma_g A^{-1} \\
\tilde{w}_g(n+1) &= \tilde{w}_g(n) + K_g e \\
P_g(n+1) &= P_g(n) - K_g \Gamma_g^T + Q_g
\end{align*}
$$

Since only the block diagonal terms of the covariance matrix are maintained, there is a significant savings in required computations. However, the DEKF is not equivalent to the use of independent Kalman filters for each group.

3.2 Sequential Measurement Update

In order to further reduce the number of required calculations by eliminating the matrix multiplications and inversions, we now describe a sequential version of the DEKF algorithm. Whereas the sequential and batch forms of the GEKF produce the same weight updates, the sequential-DEKF algorithm does not result in exactly the same weight update as the batch-DEKF [3].

The sequential-DEKF requires first initializing the variables $\Delta \tilde{w}_{g,0} = 0$ and $P_{g,0} = P_g(n)$ for each group $g = 1, \ldots, G$. Second, computing

$$
\begin{align*}
\gamma_{g,j} &= P_{g,j-1}h_{g,j}^T \\
\lambda_j &= r_j + \sum_{g=1}^{G} h_{g,j} \gamma_{g,j} \\
\bar{k}_{g,j} &= \frac{\gamma_{g,j}}{\lambda_j} \\
\Delta \tilde{w}_{g,j} &= \Delta \tilde{w}_{g,j-1} + \bar{k}_{g,j} \left( e_n - \sum_{g=1}^{G} h_{g,j} \Delta \tilde{w}_{g,j-1} \right) \\
P_{g,j} &= P_{g,j-1} - \bar{k}_{g,j} \gamma_{g,j} \\
\end{align*}
$$

for $j = 1, \ldots, N_y$ and $g = 1, \ldots, G$. Finally, updating the estimates $\tilde{w}_g(n+1) = \tilde{w}_g(n) + \Delta \tilde{w}_{g,N_y}$ and $P_g(n+1) = P_g(n) + Q_g$. Again note the presence of the term $-\sum_{g=1}^{G} h_n \Delta \tilde{w}_{g,N_y-1}$ which is necessary in order to eliminate the need for recomputing $\phi(\tilde{w}_j, x)$ after each weight update. This term has been omitted in other attempts at sequentializing the DEKF [2].

4 Comparison of Calculation Complexity

The main purpose of the sequentialized forms of the GEKF and DEKF is the reduction of the number of required calculations. Since the time-update, computation of measurement error, and calculation of the
output-prediction Jacobian are common to the four algorithms presented, the required number of calculations for those steps will not be considered. Rather, only the number of multiplications required for the measurement updates will be compared. The relevant expressions, given in Table 1, assume that the updates are performed exactly as described in equations 1–4 and that the batch matrix inversions are performed explicitly using LU decomposition and backsubstitution. In the tabulated expressions, the variables are: the number of network outputs \( N_y \), the number of weight groups \( G \), the number of weights in each group \( N_g \), and the total number of weights \( N_w \).

Observe that for \( N_y > 1 \), the sequential algorithms always require fewer multiplications than the equivalent batch form. Furthermore, the sequential forms become more advantageous as the number of outputs is increased. Finally, note that the relative effect of sequentialization is more pronounced in the DEKF than in the GEKF. Graphical comparison of these trends is provided in reference [3].

Table 1: Comparison of Number of Required Multiplications.

<table>
<thead>
<tr>
<th></th>
<th>Batch Form</th>
<th>Sequential Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEKF</td>
<td>( N_y^3 + 2N_wN_y^2 + 2N_yN_w^2 + N_wN_y )</td>
<td>( N_y (2N_y^2 + 4N_w + 1) - N_w )</td>
</tr>
<tr>
<td>NDEKF</td>
<td>( N_y^3 + 2N_wN_y^2 + 2N_y \sum_{g=1}^{G} N_g^2 + N_wN_y )</td>
<td>( N_y \left( 2 \sum_{g=1}^{G} N_g^2 + 4N_w + 1 \right) - N_w )</td>
</tr>
</tbody>
</table>

5 Simulation Results

In order to demonstrate the relative performance of the four EKF algorithms, results of a simple pattern classification problem are provided. Since the batch and sequential forms of the GEKF produce the same training history, these are not considered separately. However, results of using standard backpropagation (BP) are included. "Node-decoupling" is assumed for both DEKF trials.

The specific example provided is a pattern classification problem having four interlocking regions [7][4]. The desired classification boundaries are shown in Fig 1. An MLP with 2 inputs, 4 outputs and a single hidden layer with 20 nodes was trained to classify these regions. Each network output was assigned one of the four classes and a maximum-output criteria was used to determine the predicted class.

Initially, the MLP weights were set to random values with the same initial weight vector used for each of the four algorithms. At each training step, a random pattern was drawn from within the pattern space. Each network output was then given a desired output value of +0.8 if the pattern fell within its assigned class and -0.8 otherwise. The BP algorithm used a learning rate of \( \mu = 0.05 \), whereas the three EKF algorithms used learning parameters \( P(0) = 10I, Q = 0.00001 I, \) and \( R = 0.7 I \).

At selected times during the training, the network was tested on a 90 \( \times \) 90 grid of test points and the corresponding RMS error

\[
RMS = \sqrt{\frac{1}{8100} \sum_{p=1}^{8100} \sum_{j=1}^{4} (y_j(x_p) - \hat{y}_j(x_p))^2}
\]

was computed for each algorithm. The RMS convergence results are plotted in Fig. 2. These curves are plotted on a number-of-pattern-presentations time scale. This choice, which arguably is more relevant for online applications than a number-of-calculations time scale, explains discrepancies between these and earlier learning curves [4].

The RMS error level at which the correct classification boundaries were recognizable was about 0.53. All three EKF algorithms attained this level of accuracy after about 2000 iterations with similar convergence characteristics. On the other hand, BP required at least 50,000 iterations to obtain the same level of RMS error.
Training was terminated after 100,000 pattern presentations at which time the learned classification regions were established by determining the predicted class for each point on the test-set grid. The classification results for the four algorithms are shown in Fig. 3. Again, all three EKF forms resulted in similar classification boundaries, all of which were superior to those obtained using BP.

6 Conclusions

This paper has presented sequential forms of both the global and decoupled EKF algorithms for training multi-layer perceptron neural networks. These sequential forms differ from the linear sequential Kalman filters described in the literature by an added term necessary to compensate for the effect of the non-linear measurement function.

The sequential-GEKF was claimed to be exactly equivalent to the batch-GEKF but require as few as half the number of multiplications. More importantly, no matrix-matrix multiplications nor matrix inversions are required, the latter of which can be particularly tedious to implement in dedicated hardware.

The sequential-DEKF provides an even greater savings over the batch-DEKF and is particularly simple to implement, especially in the node-decoupled form. The sequential-DEKF, however, is only an approximation to the batch-DEKF and therefore does not result in exactly the same training history. However, the example provided, as well as a number of others investigated by the author, have demonstrated that the sequential-DEKF and batch-DEKF have very similar convergence properties and result in similar performance of the trained networks.

Apart from the requirement that the measurement noise process be uncorrelated, no structural conditions are assumed in deriving the sequential forms. Although only static learning problems were discussed, the EKF forms can be adapted to recurrent learning problems by computing the matrix $H$ recursively [5].

References


Fig. 1. Desired classification boundaries.

Fig. 2. Learning curves for EKF forms.

Fig. 3. MLP classification boundaries using: (a) BP, (b) GEKF, (c) batch-DEKF, (d) seq-DEKF.