Backpropagation Through Time: What It Does and How to Do It

Paul Werbos

Presenter: Po-Yao (Bernie) Huang
9/16/2015
Outline

• Back Propagation
  – Recap: Supervised Learning
  – Topology
  – Learning Rule
  – Implementation details

• Back Propagation Through Time
  – How it works
  – Implementation details

• Conclusion
Recap: Supervised Learning

- Example Task: Hand-written digit recognition
- Dataset (Training)
  - Size: 2000 example inputs of hand-written digits: \( T = 2000 \)
  - Feature: Each input is a 19x20 pixel image
    - \( X(t) = [X_1(t), X_2(t), \ldots, X_m(t)], m = 380, 1 \leq t \leq T \)
  - Label: a digit 0~9 (4 bits)
    - \( Y(t) = [Y_1(t), Y_2(t), \ldots, Y_n(t)], n = 4, 1 \leq t \leq T \)
- Artificial Neural Network
  - The network with topology \( S \) and parameters \( W \) outputs an estimate of \( \hat{Y}(t) \) based on \( X(t) \)
    - \( \hat{Y}(t) = N(X(t), W, S) \)
Recap: Supervised Learning

• Need to specify:
  – Network topology
    • How the network is connected
    • The perceptron functions
  – Learning rule:
    • The objective/error function
    • How to find the best weights $\mathbf{W}$ that $\hat{Y}(t)$ approximates $Y(t)$
    – Back Propogation !!

$s(z) = \frac{1}{1 + e^{-z}}$
$s'(z) = s(z)(1 - s(z))$
Network Topology

- Fully Connected network (in extreme)

\[ s(n_3) \quad s(n_6) \quad s(n_8) \]

- Simplified network (widely used)

\[ s(n_3) \quad s(n_6) \quad s(n_8) \]

\[ x_i = X_i, \quad 1 \leq i \leq m \]

\[ \text{net}_i = \sum_{j=1}^{i-1} W_{ij} x_j, \quad m < i \leq N + n \]

\[ x_i = s(\text{net}_i), \quad m < i \leq N + n \]

\[ Y_i = x_i + N_r, \quad 1 \leq i \leq n \]
Learning Rule

• Objective/Error Function

\[
E = \sum_{t=1}^{T} E(t) = \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{1}{2} (\hat{Y}_i(t) - Y_i(t))^2
\]

• Target
  – \text{argmin}_W E

• Approach: Back Propagation
  – Calculate derivatives of \( E \) with respect to all the weights (i.e. feedback to weights, \( F_{W_{ij}} \))
    • Core of Back Propagation: “Method of calculating derivatives exactly and efficiently in large systems”
  – Update the weights.
    • New \( W_{ij} = W_{ij} - \eta * F_{W_{ij}} \)
Back propagation

• Ordered Derivatives ( F_\text{z}_i: Feedback to z_i)

\[
\frac{\partial^+ \text{TARGET}}{\partial z_i} = \frac{\partial \text{TARGET}}{\partial z_i} + \sum_{j > i} \frac{\partial^+ \text{TARGET}}{\partial z_j} \cdot \frac{\partial z_j}{\partial z_i}
\]

\[F_{-z_i} = \frac{\partial E}{\partial z_i} + \sum_{j > i} F_{-z_i} \cdot \frac{\partial z_j}{\partial z_i}.
\]

• Example

- \( z_2 = 4 \cdot z_1 \)
- \( z_3 = 3 \cdot z_1 + 5 \cdot z_2 \)

Derivatives of \( z_3 \) w.r.t. \( z_1 \)
Partial: 3
Ordered: 23
Back propagation

\[ F_{W_{ij}} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial W_{ij}} = \frac{\partial E}{\partial \text{net}_j} x_i = F_{\text{net}_j} * x_i \]

\[ F_{\text{net}_j} = \frac{\partial E}{\partial \text{net}_j} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial \text{net}_j} = F_{x_j} * s'(\text{net}_j) \]

\[ F_{x_j} = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial \hat{Y}_{j-N}} + \sum_{k>j} \frac{\partial E}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial x_k} = F_{-\hat{Y}_{j-N}} + \sum_{k>j} F_{\text{net}_k} * W_{jk} = \delta_x \]

Remark:

\[ F_{-\hat{Y}(t)} = \frac{\partial E}{\partial \hat{Y}_{j(t)}} = \begin{cases} \hat{Y}_{j(t)} - Y_j(t), & 1 \leq j \leq n \\ 0, & \text{else} \end{cases} \]

\[ x_i = X_i, \quad 1 \leq i \leq m \]

\[ \text{net}_i = \sum_{j=1}^{i-1} W_{ij} x_j, \quad m < i \leq N + n \]

\[ x_i = s(\text{net}_i), \quad m < i \leq N + n \]

\[ Y_i = x_{i+N}, \quad 1 \leq i \leq n \]
Back propagation

Find Derivatives
\[ F_{W_{ij}} = F_{\text{net}_j} \times x_i \]
\[ F_{\text{net}_j} = F_{x_j} \times s'(\text{net}_j) \]
\[ F_{x_j} = F_{\hat{y}_{j-N}} + \sum_{k>j} F_{\text{net}_k} \times W_{jk} = \delta_{x_j} \]

Update weights
\[ W'_{ij} = W_{ij} - \eta \times F_{W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(\text{net}_j) x_i \]
Back propagation

\[ \delta = \hat{Y}_i(t) - Y_i(t), \]

Find Derivatives
\[
\begin{align*}
F_{W_{ij}} &= F_{net_j} * x_i \\
F_{\text{net}_j} &= F_{x_j} * s'(\text{net}_j) \\
F_{x_j} &= F_{\hat{y}_{j-N}} + \sum_{k>j} F_{\text{net}_k} * W_{jk} = \delta_{x_j}
\end{align*}
\]

Update weights
\[
W'_{ij} = W_{ij} - \eta * F_{W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(\text{net}_j)x_i
\]
Back propagation

\[ \delta_6 = W_{68} \delta \frac{\partial s}{\partial (n_8)} \]

Find Derivatives

\[ F_{W_{ij}} = F_{\text{net}_j} * x_i \]
\[ F_{\text{net}_j} = F_{x_j} * s'(\text{net}_j) \]
\[ F_{x_j} = F_{-\hat{y}_{j-N}} + \sum_{k>j} F_{\text{net}_k} * W_{jk} = \delta_{x_j} \]

Update weights

\[ W'_{ij} = W_{ij} - \eta * F_{W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(\text{net}_j)x_i \]
Back propagation

\[ Y_\delta = W_{78} \delta \frac{\partial s}{\partial (n_8)} \]

Find Derivatives
\[
\begin{align*}
F_{W_{ij}} &= F_{\text{net}_j} \ast x_i \\
F_{\text{net}_j} &= F_{x_j} \ast s' \left( \text{net}_j \right) \\
F_{x_j} &= F_{\hat{y}_{j-N}} + \sum_{k>j} F_{\text{net}_k} \ast W_{jk} = \delta_{x_j}
\end{align*}
\]

Update weights
\[
W'_{ij} = W_{ij} - \eta \ast F_{W_{ij}} = W_{ij} - \eta \delta_{x_j} s' \left( \text{net}_j \right) x_i
\]
Back propagation

\[ \delta_3 = W_{36} \delta_6 \frac{\partial s}{\partial (n_6)} + W_{37} \delta_7 \frac{\partial s}{\partial (n_7)} \]

Find Derivatives

\[ F_{W_{ij}} = F_{\text{net}_j} * x_i \]
\[ F_{\text{net}_j} = F_{x_j} * s'(\text{net}_j) \]
\[ F_{x_j} = F_{y_j-N} + \sum_{k>j} F_{\text{net}_k} * W_{jk} = \delta_{x_j} \]

Update weights

\[ W'_{ij} = W_{ij} - \eta * F_{W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(\text{net}_j) x_i \]
Back propagation

\[ \delta_4 = W_{46}\delta_6 \frac{\partial s}{\partial n_6} + W_{47}\delta_7 \frac{\partial s}{\partial n_7} \]

Find Derivatives

\[
\begin{align*}
F_W_{ij} &= F\_net_j \times x_i \\
F\_net_j &= F\_x_j \times s'(net_j) \\
F\_x_j &= F\_\hat{y}_{j-N} + \sum_{k>j}F\_net_k \times W_{jk} = \delta_{x_j}
\end{align*}
\]

Update weights

\[ W'_{ij} = W_{ij} - \eta \times F_W_{ij} = W_{ij} - \eta \delta_{x_j}s'(net_j)x_i \]
Back propagation

\[ \delta_5 = W_{56} \delta_6 \frac{\partial s}{\partial (n_6)} + W_{57} \delta_7 \frac{\partial s}{\partial (n_7)} \]

Find Derivatives

\[ F_\_W_{ij} = F_\_net_j \times x_i \]
\[ F_\_net_j = F_\_x_j \times s'(net_j) \]
\[ F_\_x_j = F_\_\hat{y}_{j-N} + \sum_{k>j} F_\_net_k \times W_{jk} = \delta_{x_j} \]

Update weights

\[ W'_{ij} = W_{ij} - \eta \times F_\_W_{ij} = W_{ij} - \eta \delta_{x_j} s'(net_j)x_i \]

\[ \hat{y} \]
Back propagation

\[ W'_{13} = W_{13} - \eta \delta_3 \frac{\partial s}{\partial (n_3)} x_1 \]
\[ W'_{23} = W_{23} - \eta \delta_3 \frac{\partial s}{\partial (n_3)} x_2 \]

Find Derivatives
\[ F\_W_{ij} = F\_net_j * x_i \]
\[ F\_net_j = F\_x_j * s'(net_j) \]
\[ F\_x_j = F\_\hat{y}_{j-N} + \sum_{k>j} F\_net_k * W_{jk} = \delta_{x_j} \]

Update weights
\[ W'_{ij} = W_{ij} - \eta * F\_W_{ij} = W_{ij} - \eta \delta_{x_j} s'(net_j) x_i \]
**Back propagation**

\[ W'_{14} = W_{14} - \eta \delta_4 \frac{\partial s}{\partial (n_4)} x_1 \]

\[ W'_{24} = W_{24} - \eta \delta_4 \frac{\partial s}{\partial (n_4)} x_2 \]

Find Derivatives

\[ F_{-W_{ij}} = F_{-net_j} * x_i \]

\[ F_{-net_j} = F_{-x_j} * s'(net_j) \]

\[ F_{-x_j}(\delta_{x_j}) = F_{-y_j-N} + \sum_{k>j} F_{-net_k} * W_{jk} = \delta_{x_j} \]

Update weights

\[ W'_{ij} = W_{ij} - \eta * F_{-W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(net_j)x_i \]
Back propagation

Find Derivatives

\[ F_{W_{ij}} = F_{\text{net}_j} * x_i \]
\[ F_{\text{net}_j} = F_{x_j} * s'(\text{net}_j) \]
\[ F_{x_j} = F_{\tilde{y}_{j-N}} + \sum_{k>j} F_{\text{net}_k} * W_{jk} = \delta_{x_j} \]

Update weights

\[ W'_{ij} = W_{ij} - \eta * F_{W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(\text{net}_j) x_i \]
Back propagation

\[ W'_{36} = W_{36} - \eta \delta_6 \frac{\partial s}{\partial (n_6)} x_3 \]
\[ W'_{46} = W_{46} - \eta \delta_6 \frac{\partial s}{\partial (n_6)} x_4 \]
\[ W'_{56} = W_{56} - \eta \delta_6 \frac{\partial s}{\partial (n_6)} x_5 \]

Find Derivatives
\[ F_{W_{ij}} = F_{\text{net}_j} * x_i \]
\[ F_{\text{net}_j} = F_x j * s'(\text{net}_j) \]
\[ F_x j = F_{\hat{y}_{j-n}} + \sum_{k>j} F_{\text{net}_k} * W_{jk} = \delta_{x j} \]

Update weights
\[ W'_{ij} = W_{ij} - \eta * F_{W_{ij}} = W_{ij} - \eta \delta_{x j} s'(\text{net}_j)x_i \]
Back propagation

\[ W'_{37} = W_{37} - \eta \delta_{7} \frac{\partial s}{\partial (n_{7})} x_{3} \]
\[ W'_{47} = W_{47} - \eta \delta_{7} \frac{\partial s}{\partial (n_{7})} x_{4} \]
\[ W'_{57} = W_{57} - \eta \delta_{7} \frac{\partial s}{\partial (n_{7})} x_{5} \]

Find Derivatives
\[ F_{W_{ij}} = F_{net_{j}} * x_{i} \]
\[ F_{net_{j}} = F_{x_{j}} * s'(net_{j}) \]
\[ F_{x_{j}} = F_{\hat{y}_{j-n}} + \sum_{k>j} F_{net_{k}} * W_{jk} = \delta_{x_{j}} \]

Update weights
\[ W'_{ij} = W_{ij} - \eta * F_{W_{ij}} = W_{ij} - \eta \delta_{x_{j}} s'(net_{j}) x_{i} \]
Back propagation

\[ W'_{68} = W_{68} - \eta \delta_8 \frac{\partial s}{\partial (n_8)} x_6 \]

\[ W'_{78} = W_{78} - \eta \delta_8 \frac{\partial s}{\partial (n_8)} x_7 \]

Find Derivatives
\[ F_{-W_{ij}} = F_{-\text{net}_j} * x_i \]
\[ F_{-\text{net}_j} = F_{-x_j} * s'(\text{net}_j) \]
\[ F_{-x_j} = F_{-\hat{y}_{j-N}} + \sum_{k>j} F_{-\text{net}_k} * W_{jk} = \delta_{x_j} \]

Update weights
\[ W'_{ij} = W_{ij} - \eta * F_{-W_{ij}} = W_{ij} - \eta \delta_{x_j} s'(\text{net}_j)x_i \]
Back Propagation: Implementation

\[ \delta_3 = W_{36} \delta_6 \frac{\partial s}{\partial (n_3)} + W_{37} \delta_7 \frac{\partial s}{\partial (n_3)} \]
\[ \delta_4 = W_{46} \delta_6 \frac{\partial s}{\partial (n_4)} + W_{47} \delta_7 \frac{\partial s}{\partial (n_4)} \]
\[ \delta_5 = W_{56} \delta_6 \frac{\partial s}{\partial (n_5)} + W_{57} \delta_7 \frac{\partial s}{\partial (n_5)} \]

\[ \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ij}} \]
\[ \frac{\partial \text{net}_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left( \sum_{k=1}^{n} w_{kj} o_k \right) = o_i \]
\[ \frac{\partial o_j}{\partial \text{net}_j} = \frac{\partial}{\partial \text{net}_j} \varphi(\text{net}_j) = \varphi(\text{net}_j)(1 - \varphi(\text{net}_j)) \]
\[ \frac{\partial E}{\partial o_j} = \sum_{l \in L} \left( \frac{\partial E}{\partial \text{net}_l} \frac{\partial \text{net}_l}{\partial o_j} \right) = \sum_{l \in L} \left( \frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial \text{net}_l} w_{jl} \right) \]

\[ w_{\text{net}} = w^T x \]
\[ y = s(\text{net}) \]
Back Propagation: Implementation

\[
\begin{align*}
W'_{36} &= W_{36} - \eta \delta_6 x_3 \\
W'_{46} &= W_{46} - \eta \delta_6 x_4 \\
W'_{56} &= W_{56} - \eta \delta_6 x_5 \\
W'_{37} &= W_{37} - \eta \delta_7 x_3 \\
W'_{47} &= W_{47} - \eta \delta_7 x_4 \\
W'_{57} &= W_{57} - \eta \delta_7 x_5
\end{align*}
\]

\[
\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ij}}
\]

\[
\frac{\partial \text{net}_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left( \sum_{k=1}^{n} w_{kj} o_k \right) = o_i
\]

\[
\frac{\partial o_j}{\partial \text{net}_j} = \frac{\partial}{\partial \text{net}_j} \varphi(\text{net}_j) = \varphi(\text{net}_j)(1 - \varphi(\text{net}_j))
\]

\[
\frac{\partial E}{\partial o_j} = \sum_{l \in L} \left( \frac{\partial E}{\partial \text{net}_l} \frac{\partial \text{net}_l}{\partial o_j} \right) = \sum_{l \in L} \left( \frac{\partial E}{\partial o_l} \frac{\partial \text{net}_l}{\partial w_{jl}} \right)
\]

\[w_{\text{net}} = \mathbf{w}^T \mathbf{x} = s(\text{net})\]

\[\delta_j = \frac{\partial E}{\partial o_i} = \frac{\partial E}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial o_i} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial o_i} = \varphi(\text{net}_j)(1 - \varphi(\text{net}_j)) \frac{\partial \text{net}_j}{\partial w_{ij}} o_i\]
What about $t$?

**Pattern Learning**

- Until satisfied do:
  - For each training data $t=1:T$
    - Compute network output with feed forward algorithm
    - For each output $i=1:n$
      - Compute $F_{\hat{Y}(t)} = \frac{\partial E}{\partial \hat{Y}_j(t)} = \hat{Y}_j(t) - Y_j(t)$
    - For each neurons in hidden layer
      - Compute $F_{W_{ij}}$ with backpropagation
    - For all weights
      - Update $W'_{ij} = W_{ij} - \eta * F_{W_{ij}}$

**Batch Learning**

- Until satisfied do:
  - For each training data $t=1:T$
    - Compute network output with feed forward algorithm
    - For each output $i=1:n$
      - Compute $F_{\hat{Y}(t)} = \frac{\partial E}{\partial \hat{Y}_j(t)} = \hat{Y}_j(t) - Y_j(t)$
    - For each neurons in hidden layer
      - Compute $F_{W_{ij}}$ with backpropagation
    - For all weights
      - Update $W'_{ij} = W_{ij} - \eta * F_{W_{ij}}$

On-line learning => pattern learning/realtime learning
Online vs Batch

• **Time vs Accuracy** (1000 epochs, 3 hidden layers)
  – Online: 632 secs
  – Batch: 87 secs
  – Modern way: mini-batch (mSGD)
    • Divide data into mini-batches

Size: 50000, 
\[ t = 87 \text{ sec} \]

Size: 2000, 
\[ T = 158 \text{ sec} \]

Size: 100, 
\[ T = 223 \text{ sec} \]

Size: 1, 
\[ T = 632 \text{ sec} \]
Online vs Batch

First 100 epochs of training on the Iris Data Set. (Also note over-fitting of training data…)
"The General Inefficiency of Batch Training for Gradient Descent Learning" by D. Wilson and T. Martine, 2003
Back Propagation Through Time

- Use previous “memory” to improve classification
  - Every neuron at time $t$ is allowed to input values from any of the neurons at time $t-1$ and $t-2$
  - Example: speech recognition

$$X(t) = 'ch'$$
Label: \text{speech}

$$\text{'Spee'ch' or Fren'ch'}$$

$$\text{net}_i(t) = \sum_{j=1}^{i-1} W_{ij} x_j(t) + \sum_{j=1}^{N+n} W'_{ij} x_j(t - 1) + \sum_{j=1}^{N+n} W''_{ij} x_j(t - 2).$$
BPTT

\[ Y_t \]
BPTT

\[ F_{x_i}(t) = F_{-\hat{\eta}_{i-N}} + \sum_{j>i} W_{ji} F_{\text{net}_j}(t) \]

\[ \text{net}_i(t) = \sum_{j=1}^{i-1} W_{ij} x_j(t) \]
$F_{x_i(t)} = F_{-\hat{\gamma}_{i-N}} + \sum_{j>i} W_{ji} F_{net_j(t)} + \sum_{j>i} W'_{ji} F_{net_j(t+1)}$

$net_i(t + 1) = \sum_{j=1}^{i-1} W'_{ij} x_j(t + 1) + \sum_{j=1}^{i-1} W_{ij} x_j(t)$
\[ F_{x_i(t)} = F_{-\Phi_{i-N}} + \sum_{j>i} W_{ji} F_{net_j(t)} + \sum_{j>i} W'_{ji} F_{net_j(t+1)} \]

\[ net_i(t + 1) = \sum_{j=1}^{i-1} W'_{ij} x_j(t + 1) + \sum_{j=1}^{i-1} W_{ij} x_j(t) \]
BPTT

\[ F_{x_i}(t) = F_{-\hat{\phi}_{i-N}} + \sum_{j>i} W_{ji} F_{net_j}(t) + \sum_{j>i} W'_{ji} F_{net_j}(t+1) \]

\[ + \sum_{j>i} W_{ji} F_{net_j}(t+2) \]

\[ net_i(t+2) = \sum_{j=1}^{i-1} W''_{ij} x_j(t+2) + \sum_{j=1}^{i-1} W'_{ij} x_j(t+1) + \sum_{j=1}^{i-1} W_{ij} x_j(t) \]
BPTT

\[ F_{x_i}(t) = F_{-\gamma_{i,N}} + \sum_{j>i} W_{ji} F_{\text{net}_j}(t) + \sum_{j>i} W'_{ji} F_{\text{net}_j}(t + 1) + \sum_{j>i} W_{ji} F_{\text{net}_j}(t + 2) \]

\[ net_i(t + 2) = \sum_{j=1}^{i-1} W''_{ij} x_j(t + 2) + \sum_{j=1}^{i-1} W'_{ij} x_j(t + 1) + \sum_{j=1}^{i-1} W_{ij} x_j(t) \]
**BPTT Implementation details**

- Required to store/compute lots of intermediate info
- Use smaller learning rates
- Initialize $W'$, $W''$ to zeros
- We might not care about errors all the time
- Choice of network topology
  - For a brain-like network (each neuron only receive input from a small number of other notes), use Linked-List
- Performance concerns
  - Avoid calculating exponential
  - Other nonlinear functions/Threshold instead

\[
\begin{align*}
s(z) &= 0, & z < 0 \\
s(z) &= z, & 0 < z < 1 \\
s(z) &= 1, & z > 1. \\
\end{align*}
\]
Conclusion

• Contribution
  – A general network topology
  – A general and theoretic analysis of BP/BPTT learning theory
  – Pseudo codes are provided (though he didn’t test them)

• Limitation
  – Few implementation details/know-how are provided
    • Eg. W initialization, learning rate adaption, range checking … etc
      LeCun, Yann A., et al. "Efficient backprop." Neural networks: Tricks of the trade. Springer Berlin Heidelberg, 2012. 9-48.
      Bengio, X. Glorot, Understanding the difficulty of training deep feedforward neural networks, AISTATS 2010
  – Constraints on multi-layer networks with BP/BPTT
    • Computationally expensive
    • The following topics
      – BP is the optimal sol? Local minimum? Crossing plateau…
        » Some conditions must be satisfied… => Minor Issue, see ref:
        LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". Nature 521: 436–444.
Thank You
BPTT