EBP Error Back Propagation algorithm

The case with multiple outputs

The weight increment is a superposition of weight modifications from errors on all outputs

$$\Delta w_p = \alpha \sum_{o=1}^{no} \sum_{p=1}^{np} \left[ (d_{op} - o_{op}) F'(z_p) f'(net_p) x_p \right]$$
EBP Error Back Propagation algorithm
The case with large learning rate

EBP Error Back Propagation algorithm
The case with small learning rate

EBP Error Back Propagation algorithm

Heuristic approaches to EBP

• Cumulative versus incremental weight updating (differs for large learning constants)

• Momentum method (valley problem)

\[ \Delta \mathbf{w}_t = -\alpha \nabla E_t + \gamma \Delta \mathbf{w}_{t-1} \]

or by Sejnowski:

\[ \Delta \mathbf{w}_t = -(1 - \gamma) \nabla E_t + \gamma \Delta \mathbf{w}_{t-1} \]
EBP Error Back Propagation algorithm

With momentum

Heuristic approaches to EBP

Variable learning rates
- If error increased by 5% the updated values are ignored and learning constant is reduced.
- If error decreased by more than 5% the learning constant is increased
EBP Error Back Propagation algorithm

Heuristic approaches to EBP

Effect of learning constant
- Large $\Rightarrow$ oscillations
- Small $\Rightarrow$ local minima
- Random values of alpha leads usually to better results.

Network Pruning (chapter 7.10)
- Remove all weights with values close to zero - retrain
- Merge neurons responding the same way, opposite way, or not changing output for all patterns.

EBP Error Back Propagation modifications

Finding minimum along a given direction

The shape of the error function is approximated by parabola and the new point is found at the minimum
EBP Error Back Propagation modifications

Finding minimum along a given direction

\[ y = ax^2 + bx + c \]

0 = a0^2 + b0 + c \rightarrow c = 0

\[ y_1 = a(\Delta x)^2 + b\Delta x \]

\[ y_2 = a(2\Delta x)^2 + b(2\Delta x) \]

\[ a = \frac{y_2 - 2y_1}{2\Delta x^2} \quad b = \frac{4y_1 - y_2}{2\Delta x} \quad c = 0 \]

\[ x_{\text{MIN}} = \Delta x \left( \frac{4y_1 - y_2}{4y_1 - 2y_2} \right) \]
EBP Error Back Propagation modifications

Finding minimum along a given direction

**Algorithm:**
Make weights change with $\alpha$ and $2\alpha$ 
Find $TE(\alpha)$ and $TE(2\alpha)$ 
Then modify weights for

$$
\alpha^* = \alpha \frac{4 \cdot TE(\alpha) - TE(2 \cdot \alpha)}{4 \cdot TE(\alpha) - 2 \cdot TE(2 \cdot \alpha)}
$$

The acceleration is applied only when $y_1$ and $y_2$ are negative 
(means error is getting smaller) 
and 
$2y_1 > y_2$ 
(means positive sign the second derivative).
Activation functions

**unipolar**

\[
o = f(\text{net}) = \frac{1}{1 + \exp(-4k \text{net})}
\]

\[
f'(\text{net}) = \frac{-\exp(-4k \text{net})}{[1 + \exp(-4k \text{net})]^2} (-4k)
\]

\[
f''(\text{net}) = 4k \left( \frac{1}{1 + \exp(-4k \text{net})} \right) \frac{1 + \exp(-4k \text{net}) - 1}{1 + \exp(-4k \text{net})}
\]

\[
f'(o) = 4k o(1 - o)
\]

**bipolar**

\[
o = f(\text{net}) = \frac{2}{1 + \exp(-2k \text{net})} - 1 = \tanh(k \text{net})
\]

\[
o = f(\text{net}) = \frac{1 - \exp(-2k \text{net})}{1 + \exp(-2k \text{net})} = \tanh(k \text{net})
\]

\[
f'(\text{net}) = k \left( \frac{4\exp(-2k \text{net})}{[1 + \exp(-2k \text{net})]^2} \right)
\]

\[
f'(\text{net}) = k \left( 1 - o^2 \right)
\]

\[
f'(\text{net}) = k \left[ 1 - \left( \frac{1 - \exp(-2k \text{net})}{1 + \exp(-2k \text{net})} \right)^2 \right] = k \left( \frac{4\exp(-2k \text{net})}{[1 + \exp(-2k \text{net})]^2} \right)
\]
**Activation functions**

\[ o = f(\text{net}) = \frac{2}{1 + \exp(-2k \text{net})} - 1 \quad \text{Bipolar sigmoidal function tangent hyperbolic} \]

\[ o = \tanh(k \text{net}) \]

\[ o = f(\text{net}) = \frac{1}{1 + \exp(-4k \text{net})} \quad \text{Unipolar sigmoidal function} \]

\[ o = f(\text{net}) = \frac{k \text{net}}{1 + \text{abs}(k \text{net})} \quad \text{Unipolar Elliot function} \]

**Flat spot problem**

A poor convergence of delta algorithm is due to plateaus on the error surface. This problem is also known as “flat spot” problem. The prime reason for the plateau formations is a characteristic shape of the sigmoidal activation functions.
**Flat spot problem**

\[ f''(\text{net}) = k[1 - o^2] \]

\[ f'(\text{net}) = k \left[ 1 - o^2 \left( 1 - \left( \frac{\text{err}}{2} \right)^2 \right) \right] \]

For small errors: \( f'(\text{net}) = k[1 - o^2] \)

For large errors: \( f'(\text{net}) = k \)

Other formulas are possible too

---

**EBP Error Back Propagation modifications**

**Quickprop algorithm by Fahlman** (1)

\[ \Delta w_{ij}(t) = -\alpha S_{ij}(t) + \gamma_{ij} \Delta w_{ij}(t-1) \]

\[ S_{ij}(t) = \frac{\partial E(w(t))}{\partial w_{ij}} + \eta w_{ij}(t) \]

\( \alpha \implies \) learning constant

\( \eta \implies \) memory constant (small 0.0001 range) leads to reduction of weights and limits growth of weights

\( \gamma_{ij} \implies \) momentum term selected individually for each weight

Otherwise

\[ 0.01 < \alpha < 0.6 \quad \text{when} \quad \Delta w_{ij} = 0 \quad \text{or sign of} \quad \Delta w_{ij} \quad \text{changes} : \]

\[ \alpha = 0 \quad \text{otherwise} \]

\[ S_{ij}(t) \Delta w_{ij}(t) > 0 \]
EBP Error Back Propagation modifications

Quickprop algorithm by Fahlman (2)

\[ \Delta w_{ij}(t) = -\alpha S_{ij}(t) + \gamma_{ij} \Delta w_{ij}(t-1) \]

\[ S_{ij}(t) = \frac{\partial E(w(t))}{\partial w_{ij}} + \eta w_{ij}(t) \]

momentum term selected individually for each weight is very important part of this algorithm

\[ \gamma_{ij}(t) = \begin{cases} \gamma_{\text{max}} & \text{when } \beta_{ij}(t) > \gamma_{\text{max}} \text{ or } S_{ij}(t) \Delta w_{ij}(t-1) \beta_{ij}(t) < 0 \\ \beta_{ij}(t) & \text{otherwise} \end{cases} \]

\[ \beta_{ij}(t) = \frac{S_{ij}(t)}{S_{ij}(t-1) - S_{ij}(t)} \]

\[ \gamma_{\text{max}} = 1.75 \]

Quickprop algorithm sometimes reduces computation time a hundreds times

Modified Quickprop algorithm even is simpler often gives better results than the original one.

\[ \gamma_{ij}(t) = \min \left( \beta_{ij}, \gamma_{\text{max}} \right) \]

\[ \beta_{ij}(t) = \frac{S_{ij}(t)}{S_{ij}(t-1) - S_{ij}(t)} \]
RPROP Resilient Error Back Propagation

Very similar to EBP, but weights adjusted without using values of the propagated errors, but only its sign. Learning constants are selected individually to each weight based on the history

\[ \Delta w_{ij}(t) = -\alpha_{ij} \text{sgn} \left( \frac{\partial E(w(t))}{\partial w_{ij}(t)} \right) \]

\[ S_{ij}(t) = \frac{\partial E(w(t))}{\partial w_{ij}} + \eta w_{ij}(t) \]

\[ \alpha_{ij}(t) = \begin{cases} 
\min(a \cdot \alpha_{ij}(t-1), \alpha_{\text{max}}) & \text{for } S_{ij}(t) \cdot S_{ij}(t-1) > 0 \\
\max(b \cdot \alpha_{ij}(t-1), \alpha_{\text{min}}) & \text{for } S_{ij}(t) \cdot S_{ij}(t-1) < 0 \\
\alpha_{ij}(t-1) & \text{otherwise} 
\end{cases} \]

\[ a = 1.2 \quad b = 0.5 \quad \alpha_{\text{min}} = 10^{-6} \quad \alpha_{\text{max}} = 50 \]

Back Percolation

Error is propagated as in EBP and than each neuron is “trained” using and algorithm to train one neuron such as pseudo inversion

Problems:

Pseudo inversion leads to errors

Sometimes propagated errors are larger than 2 for bipolar or larger than 1 for unipolar
**Delta-bar-Delta**

by Jacobs

For each weight the learning coefficient is selected individually. It was developed for quadratic error functions

\[
\Delta \alpha_{ij}(t) = \begin{cases} 
  a & \text{for } S_{ij}(t-1)D_{ij}(t) > 0 \\
  -b \cdot a_{ij}(t-1) & \text{for } S_{ij}(t-1)D_{ij}(t) < 0 \\
  0 & \text{otherwise}
\end{cases}
\]

\[
D_{ij}(t) = \frac{\partial E(t)}{\partial w_{ij}(t)} \quad S_{ij}(t) = (1 - \xi)D_{ij}(t) + \xi S_{ij}(t-1)
\]

\[
0 < \xi < 1 \quad 0 < a < 0.05 \quad 0.1 < b < 0.3
\]