NEURAL NETWORKS (ELEC 5240 and ELEC 6240) modifications of EBP

Bodgan M. Wilamowski

EBP Error Back Propagation algorithm

The case with multiple outputs

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

\[ \Delta w_p = \alpha \sum_{o=1}^{m} \sum_{p=1}^{m} \left[ (d_{op} - o_{op}) f'(z_p) f'(net_p) x_p \right] \]

EBP Error Back Propagation algorithm

The case with large learning rate

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)
  \[ \Delta w_i = -\alpha \nabla E_i + \gamma \Delta w_{i-1} \]
  or by Sejnowski:
  \[ \Delta w_i = -(1 - \gamma) \nabla E_i + \gamma \Delta w_{i-1} \]

EBP Error Back Propagation algorithm

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

\[ y = ax^2 + bx + c \]
\[ 0 = a0^2 + b0 + c \rightarrow c = 0 \]
\[ y_1 = a(\Delta x)^2 + b\Delta x \]
\[ y_2 = a(2\Delta x)^2 + b(2\Delta x) \]
\[ a = \frac{y_2 - 2y_1}{2\Delta x^2} \]
\[ b = \frac{4y_1 - y_2}{2\Delta x} \]
EBP Error Back Propagation modifications
Finding minimum along a given direction

\[ y = ax^2 + bx + c \quad c = 0 \]
\[ a = \frac{y_2 - 2y_1}{2Ax^2} \quad b = \frac{4y_1 - y_2}{2A} \]
\[ \frac{dy}{dx} = 2ax + b = 0 \]
\[ x = \frac{-b}{2a} = \Delta x \left( \frac{4y_1 - y_2}{4y_1 - 2y_2} \right) \]

Activation functions

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

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

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.

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

EBP Error Back Propagation modifications

Quickprop algorithm by Fahlman
\[ \Delta w_j(t) = -\alpha S_k(t) + \gamma \Delta w_j(t-1) \]
\[ S_k(t) = \frac{\partial E(w_j) \partial w_j(t)}{\partial w_j} + \eta \Delta w_j(t) \]
\[ \alpha \Rightarrow \text{learning constant} \]
\[ \eta \Rightarrow \text{memory constant (small 0.0001 range) leads to reduction of weights and limits growth of weights} \]
\[ \gamma \Rightarrow \text{momentum term selected individually for each weight} \]
Otherwise
\[ 0.01 < \alpha < 0.6 \text{ when } \Delta w_j = 0 \text{ or sign of } \Delta w_j \text{ changes:} \]
\[ \alpha = 0 \text{ otherwise} \]
\[ S_k(t) \Delta w_j(t) > 0 \]
EBP Error Back Propagation modifications

Quickprop algorithm by Fahlman (2)

\[ \Delta w_j(t) = -\alpha S_j(t) + \gamma_j \Delta w_j(t-1) \]

\[ S_j(t) = \frac{\partial E(w(t))}{\partial w_j} + \eta w_j(t) \]

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

\[ \gamma_j(t) = \begin{cases} \gamma_{\max} \text{ when } \beta_j(t) > \gamma_{\max} \text{ or } S_j(t) \Delta w_j(t-1) \beta_j(t) < 0 \\ \beta_j(t) \text{ otherwise} \end{cases} \]

\[ \beta_j(t) = \frac{S_j(t)}{S_j(t-1) - S_j(t)} \]

Quickprop algorithm sometimes reduces computation time a hundreds times

EBP Error Back Propagation modifications

Quickprop algorithm by Fahlman (modified) (3)

Later this algorithm was simplified: \( \gamma_{\max} = 1.75 \)

\[ \Delta w_j(t) = \begin{cases} \gamma_j(t) \Delta w_j(t-1) \text{ when } \Delta w_j(t-1) = 0 \\ -\alpha \frac{\partial E(w(t))}{\partial w_j} \text{ otherwise} \end{cases} \]

\[ S_j(t) = \frac{\partial E(w(t))}{\partial w_j} + \eta w_j(t) \]

\[ \gamma_j(t) = \min\{\beta_j(t), \gamma_{\max}\} \]

\[ \beta_j(t) = \frac{S_j(t)}{S_j(t-1) - S_j(t)} \]

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

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 then 1 for unipolar

Delta-bar-Delta

by Jacobs

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

\[ \Delta a_j(t) = \begin{cases} a \text{ for } S_j(t-1) D_j(t) > 0 \\ -b \cdot a_j(t-1) \text{ for } S_j(t-1) D_j(t) < 0 \end{cases} \]

\[ D_j(t) = \frac{\partial E(t)}{\partial w_j(t)} \]

\[ S_j(t) = (1 - \xi) D_j(t) + \xi S_j(t-1) \]

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

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

\[ \Delta w_j(t) = -\alpha_j \sgn \left( \frac{\partial E(w(t))}{\partial w_j(t)} \right) \]

\[ S_j(t) = \frac{\partial E(w(t))}{\partial w_j(t)} + \eta w_j(t) \]

\[ \alpha_j(t) = \begin{cases} \min\{a \cdot \alpha_j(t-1) \alpha_{\max}\} \text{ for } S_j(t) S_j(t-1) > 0 \\ \max\{b \cdot a_j(t-1) \alpha_{\max}\} \text{ for } S_j(t) S_j(t-1) < 0 \end{cases} \]

\[ a = 1.2 \quad b = 0.5 \quad \alpha_{\max} = 10^{-4} \quad \alpha_{\max} = 50 \]