Improving the Convergence of Back-Propogation Learning with Second Order Methods

Sue Becker and Yann le Cun, Sept 1988

Kasey Bray, October 2017
Table of Contents

1 The Problem with Back-Propagation

2 Improving the Convergence of BP

3 A Computationally Feasible Approximation to Newton’s Method

4 Analysis of the Diagonal Approximation

5 Learning Experiments with the Pseudo-Newton Algorithm
# Table of Contents

1. The Problem with Back-Propagation

2. Improving the Convergence of BP

3. A Computationally Feasible Approximation to Newton’s Method

4. Analysis of the Diagonal Approximation

5. Learning Experiments with the Pseudo-Newton Algorithm
Back-Propagation

The back-propagation algorithm performs a gradient descent search in weight space to determine the weights $w_{ij}$ that minimize some cost function $L$.

$$\Delta w_{ij}(t) = -\epsilon \frac{\partial L}{\partial w_{ij}(t)},$$

where $\epsilon$ is a learning constant.
Back-Propagation

The back-propagation algorithm performs a gradient descent search in weight space to determine the weights $w_{ij}$ that minimize some cost function $L$.

$$\Delta w_{ij}(t) = -\epsilon \frac{\partial L}{\partial w_{ij}(t)},$$

where $\epsilon$ is a learning constant.

- Gradient-based numerical optimization methods are slow to converge.
The back-propagation algorithm performs a gradient descent search in weight space to determine the weights $w_{ij}$ that minimize some cost function $L$.

$$\Delta w_{ij}(t) = -\epsilon \frac{\partial L}{\partial w_{ij}(t)},$$

where $\epsilon$ is a learning constant.

- Gradient-based numerical optimization methods are slow to converge.

- Can we apply second derivative-based methods to improve the convergence of the back-propagation algorithm?
Table of Contents

1 The Problem with Back-Propagation

2 Improving the Convergence of BP

3 A Computationally Feasible Approximation to Newton’s Method

4 Analysis of the Diagonal Approximation

5 Learning Experiments with the Pseudo-Newton Algorithm
Acceleration Methods

- **Back-Propagation + Momentum**
  - Add a fixed proportion of the previous weight change to the current weight change.

\[
\Delta w_{ij}(t) = -\epsilon \frac{\partial L}{\partial w_{ij}(t)} + \alpha \Delta w_{ij}(t - 1)
\]

where \( \alpha \) is a momentum rate between 0 and 1.

- Accelerates learning once a stable descent direction has been found.
Acceleration Methods

- **Conjugate Gradient Method**
  - Generates a series of mutually conjugate search directions, and at each step the function is minimized along one of these conjugate directions.
  
  \[
  \Delta w_{ij}(t) = -\epsilon \frac{\partial L}{\partial w_{ij}(t)} + \alpha_n \Delta w_{ij}(t - 1).
  \]
  
  - Good for large non-sparse optimization problems—uses only gradient information, and doesn’t require storage of a matrix.
Improving Convergence of Back-Prop

The Problem
Existing Solutions
Pseudo-Newton
Diagonal Approximation
Experiments

Acceleration Methods

- Conjugate Gradient Method
  - Generates a series of mutually conjugate search directions, and at each step the function is minimized along one of these conjugate directions.
  
  \[ \Delta w_{ij}(t) = -\epsilon \frac{\partial L}{\partial w_{ij}(t)} + \alpha_n \Delta w_{ij}(t - 1). \]

  - Good for large non-sparse optimization problems—uses only gradient information, and doesn’t require storage of a matrix.

  - Converges linearly in \( n \) (where \( n \) is the number of weights) – prohibitively slow for large problems.
Second Derivative Methods

- Requires the Hessian matrix; \( \frac{\partial^2 L}{\partial w_{ij}^2} \).
Second Derivative Methods

- Requires the Hessian matrix; \( \frac{\partial^2 L}{\partial w_{ij}^2} \).
- Newton’s Method

\[
\Delta w_{ij}(t) = -\frac{\partial L/\partial w_{ij}(t)}{\partial^2 L/\partial w_{ij}^2(t)}.
\]

- Convergence is superlinear (quadratic under certain conditions).
Second Derivative Methods

- Requires the Hessian matrix; \( \frac{\partial^2 L}{\partial w_{ij}^2} \).

- Newton’s Method

\[
\Delta w_{ij}(t) = -\frac{\partial L/\partial w_{ij}(t)}{\partial^2 L/\partial w_{ij}^2(t)}.
\]

- Convergence is superlinear (quadratic under certain conditions).

- Requires a good starting point.

- Expensive in both storage and computation (\( O(n^3) \) ops to invert the Hessian).

- If \( L \) is non-convex, can converge to a local maximum, saddle point, or minimum.
Second Derivative Methods

- Broyden-Fletcher-Goldfarb-Shanno Algorithm

**Input** $x_0$ and $B_0$.

For $k = 1, 2, ...$

1. Solve $B_k p_k = -\nabla f(x_k)$ for direction $p_k$.
2. Perform line search to obtain step size $\alpha_k$.
   
   $\alpha_k = \text{argmin } f(x_k + \alpha p_k)$.
3. $s_k = \alpha_k p_k$.
   Update $x_{k+1} = x_k + s_k$.
4. $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
5. $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$.

End.
Second Derivative Methods

- BFGS Algorithm
  - Good convergence properties.
  - Invariant under linear transformations of the parameter space.
  - Approximates the inverse Hessian in $\mathcal{O}(n^2)$ operations.
  - The Hessian update is SPD, hence the algorithm is numerically stable.
Second Derivative Methods

- **BFGS Algorithm**
  - Good convergence properties.
  - Invariant under linear transformations of the parameter space.
  - Approximates the inverse Hessian in $\mathcal{O}(n^2)$ operations.
  - The Hessian update is SPD, hence the algorithm is numerically stable.
  - Each iteration requires $\mathcal{O}(n^2)$ operations, thus the computation advantage probably only holds for "small to moderately sized problems."
  - Does not have locally computable weight update terms.
Table of Contents

1. The Problem with Back-Propagation
2. Improving the Convergence of BP
3. A Computationally Feasible Approximation to Newton’s Method
4. Analysis of the Diagonal Approximation
5. Learning Experiments with the Pseudo-Newton Algorithm
The Pseudo-Newton Step

Approximate the Hessian with a diagonal matrix.

\[ \Delta w_{ij}(t) = - \frac{\partial L/\partial w_{ij}(t)}{|\partial^2 L/\partial w_{ij}^2(t)| + \mu} \]
The Pseudo-Newton Step

Approximate the Hessian with a diagonal matrix.

\[ \Delta w_{ij}(t) = -\frac{\partial L/\partial w_{ij}(t)}{|\partial^2 L/\partial w_{ij}^2(t)| + \mu} \]

- Requires only \( O(n) \) operations.
- Diagonal Hessian terms can be computed and stored locally.
- Trivial to invert.
- Can deal with negative curvature—no positive definite requirement.
The Pseudo-Newton Step

Approximate the Hessian with a diagonal matrix.

\[ \Delta w_{ij}(t) = -\frac{\partial L / \partial w_{ij}(t)}{|\partial^2 L / \partial w_{ij}^2(t)| + \mu} \]

- Requires only \( O(n) \) operations.
- Diagonal Hessian terms can be computed and stored locally.
- Trivial to invert.
- Can deal with negative curvature—no positive definite requirement.

Do the diagonal elements of the Hessian well approximate the true Hessian?
Table of Contents

1. The Problem with Back-Propagation
2. Improving the Convergence of BP
3. A Computationally Feasible Approximation to Newton’s Method
4. Analysis of the Diagonal Approximation
5. Learning Experiments with the Pseudo-Newton Algorithm
Three Small Problems

Compare $L_1$ norms and eigenvalues of the diagonal approx., $D$, and full Hessian, $H$, for three small problems:

1. **A Tiny Problem**: Only three units (1 hidden), and one input pattern with no thresholds.
   - Input unit always on, desired output is 1.
   - Two weights

2. **A 4:2:4 Encoder**: Four input patterns, 22 weights.
   - Input vector has one bit on (the rest off), and the desired output pattern identical to the input.

3. **An 8:4:8 Encoder**: Eight input patterns, 76 weights.
   - Input vector has one bit on (the rest off), and the desired output pattern identical to the input.
Three Small Problems

Compare $L_1$ norms and eigenvalues of the diagonal approx., $D$, and full Hessian, $H$, for three small problems:

1. **A Tiny Problem**: Only three units (1 hidden), and one input pattern with no thresholds.
   - Input unit always on, desired output is 1.
   - Two weights

2. **A 4:2:4 Encoder**: Four input patterns.
   - 22 weights.
   - Input vector has one bit on (the rest off), and the desired output pattern identical to the input.
Three Small Problems

Compare $L_1$ norms and eigenvalues of the diagonal approx., $D$, and full Hessian, $H$, for three small problems:

1. **A Tiny Problem**: Only three units (1 hidden), and one input pattern with no thresholds.
   - Input unit always on, desired output is 1.
   - Two weights

2. **A 4:2:4 Encoder**: Four input patterns.
   - 22 weights.
   - Input vector has one bit on (the rest off), and the desired output pattern identical to the input.

3. **An 8:4:8 Encoder**: Eight input patterns.
   - 76 weights.
   - Input vector has one bit on (the rest off), and the desired output pattern identical to the input.
### Improving Convergence of Back-Prop

#### The Problem

#### Existing Solutions

<table>
<thead>
<tr>
<th>Pseudo-Newton</th>
<th>Diagonal Approximation</th>
</tr>
</thead>
</table>

#### Experiments

Comparing Norms

#### Table 1: Norms of Hessians, Diagonal Hessians, And Gradients For Three Networks

<table>
<thead>
<tr>
<th>Net</th>
<th>Weights</th>
<th>$|H|$</th>
<th>$|D|$</th>
<th>$\frac{|D|}{|H|}$</th>
<th>$|\nabla C|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>tiny</td>
<td>Random</td>
<td>1.87</td>
<td>1.17</td>
<td>0.63</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>Solution</td>
<td>0.77</td>
<td>0.43</td>
<td>0.56</td>
<td>0.098</td>
</tr>
<tr>
<td>4:2:4</td>
<td>Random</td>
<td>14.7</td>
<td>5.35</td>
<td>0.36</td>
<td>2.87</td>
</tr>
<tr>
<td></td>
<td>Solution</td>
<td>24.6</td>
<td>4.76</td>
<td>0.19</td>
<td>0.042</td>
</tr>
<tr>
<td>8:4:8</td>
<td>Random</td>
<td>99.7</td>
<td>10.88</td>
<td>0.11</td>
<td>8.69</td>
</tr>
<tr>
<td></td>
<td>Solution</td>
<td>164.8</td>
<td>16.07</td>
<td>0.10</td>
<td>0.048</td>
</tr>
</tbody>
</table>
Examining Eigenvalues

- Eigenvalues of the Hessian yield information about the curvature of the error curve $L$. 
Examining Eigenvalues

- Eigenvalues of the Hessian yield information about the curvature of the error curve $L$.

- The spread between the largest and small eigenvalue indicates the eccentricity of $L$ (or how badly the problem is conditioned).
Examining Eigenvalues

- Eigenvalues of the Hessian yield information about the curvature of the error curve $L$.

- The spread between the largest and small eigenvalue indicates the eccentricity of $L$ (or how badly the problem is conditioned).

- The clustering of the eigenvalues of $D$ compared with those of $H$, can tell us to what extent the principle curvature of the error surface is captured in the diagonal terms.
Examining Eigenvalues

- Eigenvalues of the Hessian yield information about the curvature of the error curve $L$.

- The spread between the largest and small eigenvalue indicates the eccentricity of $L$ (or how badly the problem is conditioned).

- The clustering of the eigenvalues of $D$ compared with those of $H$, can tell us to what extent the principal curvature of the error surface is captured in the diagonal terms.
Examining Eigenvalues

Figure: Eigenvalue histogram for 8:4:8 encoder. White bars for H, black bars for D. Top: Random weight. Bottom: Solution point.
Table of Contents

1. The Problem with Back-Propagation
2. Improving the Convergence of BP
3. A Computationally Feasible Approximation to Newton’s Method
4. Analysis of the Diagonal Approximation
5. Learning Experiments with the Pseudo-Newton Algorithm
Experiment Network Set-up

Create a problem which would be difficult for back-propagation to solve.

- Generate 64 patterns consisting of 32 bits randomly set to $\pm 1$, and each pattern is assigned to one of four classes.
- The network has 32 input units and 4 output units.
- Activation functions $g(x) = 1.7159 \tanh \left( \frac{2}{3} x \right)$, so $g(1) = 1$ and $g(-1) = -1$.
- Cost function is the $L_2$ norm.
- Desired outputs are $+1$ and $-1$.
- Relative learning rate set for each layer is $\frac{\epsilon}{\sqrt{i}}$ ($i$ is number of inputs to each unit in that layer).
Improving Convergence of Back-Prop

The Problem

Existing Solutions

Pseudo-Newton Method
- $\epsilon = \frac{1}{\sqrt{i}}$, and $\mu = 1$.

Online Back-propagation
- Weights adjusted after each pattern presentation.
  - $\epsilon = .01 \sqrt{\frac{1}{i}}$.

Batch Back-propagation
- Gradient is accumulated over the whole training set, and then the weights are updated.
  - $\epsilon = .04 \sqrt{\frac{1}{i}}$

Experiments
- Learning trials repeated from 100 different random initial weight settings.
Improving Convergence of Back-Prop

The Problem

Existing Solutions

Pseudo-Newton

Diagonal Approximation

Experiments

Compare with BP

Table 2: Mean Trials to Reach 100% Correctness

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Back-propagation</td>
<td>1571.20</td>
<td>383.93</td>
</tr>
<tr>
<td>Batch Back-propagation</td>
<td>949.12</td>
<td>148.10</td>
</tr>
<tr>
<td>Pseudo-Newton</td>
<td>603.52</td>
<td>101.00</td>
</tr>
</tbody>
</table>

Figure: The mean and standard deviation of the error plotted over leaning trials, for 100 repetitions and 1280 pattern presentations.
Discussion

- With momentum and optimally tuned learning parameters, the algorithm is successful:
  - Number of iterations required to learn to 100% correctness reduced by a factor of $\sim 1.5$ compared to batch BP and $\sim 2.5$ compared to online BP.

- Under certain conditions the algorithm can fail:
  - If initial weights are very large or very small.
  - Without momentum, performs about the same as back-propagation.
  - The values of $\mu$ and $\epsilon$ are critical in getting reasonable behavior with the pseudo-Newton algorithm.