
Muthu Chidambaram

Department of Computer Science, University of Virginia

https://qdata.github.io/deep2Read/
Difference Target Propagation (2015)

- Authors: Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, Yoshua Bengio
- Proposes target propagation as alternative to back prop for handling deeply non-linear (i.e. discrete) functions
- Propagate targets for each layer instead of gradients
Difference Target Propagation (2015)

- Vanishing/exploding gradient problem in back prop
  - Extreme case: discrete functions
- Biological implausibility of back prop
- Target prop: each feedforward unit’s activation value is associated with a target value
- Layer-local training criterion for updating each layer separately
Difference Target Propagation (2015)

- Instead of propagating error signals, assign each hidden layer a nearby value ($h_i$) that leads to lower loss.

\[ h_i = f_i(h_{i-1}) = s_i(W_i h_{i-1}), \quad i = 1, \ldots, M \]

\[ L(h_M(x; \theta^0_W), y) = L(h_M(h_i(x; \theta^0_i; \theta^i_M), y) \]

\[ L(h_M(\hat{h}_i; \theta^i_M), y) < L(h_M(h_i(x; \theta^0_i; \theta^i_M), y) \]
Difference Target Propagation (2015)

- Define layer-local target loss and update using SGD
  - Avoids vanishing/exploding gradient by computing derivatives for only a single layer
- Use “approximate inverse” to define intermediate targets

\[
L_i(\hat{h}_i, h_i) = \|\hat{h}_i - h_i(x; \theta_{W}^{0;i})\|_2^2
\]

\[
W_i^{(t+1)} = W_i^{(t)} - \eta f_i \frac{\partial L_i(\hat{h}_i, h_i)}{\partial W_i} = W_i^{(t)} - \eta f_i \frac{\partial L_i(\hat{h}_i, h_i)}{\partial h_i} \frac{\partial h_i(x; \theta_{W}^{0;i})}{\partial W_i}
\]

\[
L_i(\hat{h}_{i+1}, f_i(\hat{h}_i)) < L_i(\hat{h}_{i+1}, f_i(h_i)) \quad f_i(g_i(h_i)) \approx h_i \quad \hat{h}_{i-1} = g_i(\hat{h}_i)
\]
Difference Target Propagation (2015)

- Learn approximate inverse using auto-encoder
- Train inverse mapping via additional autoencoder loss
  - Modify loss with noise injection so inverse corresponds to neighborhood
- Update direction of target prop does not deviate by more than 90 degrees from gradient direction

\[
g_i(h_i) = \tilde{s}_i(V_i h_i), \quad i = 1, \ldots, M \quad L_i^{inv} = \|g_i(f_i(h_{i-1})) - h_{i-1}\|_2^2
\]

\[
L_i^{inv} = \|g_i(f_i(h_{i-1} + \epsilon)) - (h_{i-1} + \epsilon)\|_2^2, \quad \epsilon \sim N(0, \sigma)
\]

\[
0 < \frac{1 + \Delta_1(\hat{\eta})}{\frac{\lambda_{max}}{\lambda_{min}} + \Delta_2(\hat{\eta})} \leq \cos(\alpha) \leq 1
\]
Difference Target Propagation (2015)

Fig. 1. (left) How to compute a target in the lower layer via difference target propagation. $f_i(\hat{h}_{i-1})$ should be closer to $\hat{h}_i$ than $f_i(h_{i-1})$. (right) Diagram of the back-propagation-free auto-encoder via difference target propagation.
Difference Target Propagation (2015)

- Using just the inverse function leads to optimization problems
  - Proposes linear correction to target prop

\[
\hat{h}_{i-1} = h_{i-1} + g_i(\hat{h}_i) - g_i(h_i)
\]

\[
h_i = \hat{h}_i \Rightarrow h_{i-1} = \hat{h}_{i-1}
\]

\[
h_{i-1} = f_i^{-1}(h_i) = g_i(\hat{h}_i) = \hat{h}_{i-1}
\]
Algorithm 1: Training deep neural networks via difference target propagation

Compute unit values for all layers:

\begin{align*}
&\text{for } i = 1 \text{ to } M \text{ do} \\
&\quad h_i \leftarrow f_i(h_{i-1}) \\
&\text{end for}
\end{align*}

Making the first target: \( \hat{h}_{M-1} \leftarrow h_{M-1} - \hat{\eta} \frac{\partial L}{\partial h_{M-1}}, \) \( (L \text{ is the global loss}) \)

Compute targets for lower layers:

\begin{align*}
&\text{for } i = M - 1 \text{ to } 2 \text{ do} \\
&\quad \hat{h}_{i-1} \leftarrow h_{i-1} - g_i(h_i) + g_i(\hat{h}_i) \\
&\text{end for}
\end{align*}

Training feedback (inverse) mapping:

\begin{align*}
&\text{for } i = M - 1 \text{ to } 2 \text{ do} \\
&\quad \text{Update parameters for } g_i \text{ using SGD with following a layer-local loss } L_i^{inv} \\
&\quad L_i^{inv} = ||g_i(f_i(h_{i-1} + \epsilon)) - (h_{i-1} + \epsilon)||^2_2, \quad \epsilon \sim N(0, \sigma) \\
&\text{end for}
\end{align*}

Training feedforward mapping:

\begin{align*}
&\text{for } i = 1 \text{ to } M \text{ do} \\
&\quad \text{Update parameters for } f_i \text{ using SGD with following a layer-local loss } L_i \\
&\quad L_i = ||f_i(h_{i-1}) - \hat{h}_i||^2_2 \quad \text{if } i < M, \quad L_i = L \text{ (the global loss)} \quad \text{if } i = M. \\
&\text{end for}
\end{align*}
Difference Target Propagation (2015)

- If input of ith layer becomes target, output of ith layer also gets closer to target

\[ \| \hat{h}_i - f_i(\hat{h}_{i-1}) \|_2^2 < \| \hat{h}_i - h_i \|_2^2 \]
Difference Target Propagation (2015)

- Autoencoders can also be trained using difference target prop

\[
\begin{align*}
    \mathbf{h} &= f(\mathbf{x}) = \text{sig}(\mathbf{Wx} + \mathbf{b}) \\
    \mathbf{z} &= g(\mathbf{h}) = \text{sig}(\mathbf{WT}(\mathbf{h} + \epsilon) + \mathbf{c}), \quad \epsilon \sim N(0, \sigma) \\
    \mathbf{L} &= ||\mathbf{z} - \mathbf{x}||_2^2 + ||f(\mathbf{x} + \epsilon) - \mathbf{h}||_2^2, \quad \epsilon \sim N(0, \sigma) \\
    \hat{\mathbf{h}} &= \mathbf{h} + f(\hat{\mathbf{z}}) - f(\mathbf{z}) = 2\mathbf{h} - f(\mathbf{z})
\end{align*}
\]
Difference Target Propagation (2015)

- Trained deep feedforward net with RMSProp optimization

![Graphs showing training cost and classification error](image)

**Fig. 2.** Mean training cost (left) and train/test classification error (right) with target propagation and back-propagation using continuous deep networks (tanh) on MNIST. Error bars indicate the standard deviation.
Difference Target Propagation (2015)

- Trained discrete networks using target prop

\[
\begin{align*}
  h_1 &= f_1(x) = \tanh(W_1 x) \quad \text{and} \quad h_2 = f_2(h_1) = \tanh(W_2 \text{sign}(h_1)) \\
  p(y|x) &= f_3(h_2) = \text{softmax}(W_3 h_2) \\
  g_2(h_2) &= \tanh(V_2 \text{sign}(h_2)) \\
  L_2^{inv} &= \|g_2(f_2(h_1 + \epsilon)) - (h_1 + \epsilon)\|_2^2, \quad \epsilon \sim N(0, \sigma)
\end{align*}
\]
Difference Target Propagation (2015)
Difference Target Propagation (2015)

• Also trained stochastic network using difference backprop

\[
\begin{align*}
    h^p_i &= P(H_i = 1|h_{i-1}) = \sigma(W_i h_{i-1}), \quad h_i \sim P(H_i|h_{i-1}) \\
    \delta h^p_{i-1} &= \delta h^p_i \frac{\partial h^p_i}{\partial h^p_{i-1}} \approx \sigma'(W_i h_{i-1}) W^T_i \delta h^p_i \\
    L^\text{inv}_i &= ||g_i(f_i(h_{i-1} + \epsilon)) - (h_{i-1} + \epsilon)||^2_2, \quad \epsilon \sim N(0, \sigma)
\end{align*}
\]
### Difference Target Propagation (2015)

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference Target-Propagation, M=1</td>
<td>1.54%</td>
</tr>
<tr>
<td>Straight-through gradient estimator [5] + backprop, M=1</td>
<td>1.71%</td>
</tr>
<tr>
<td>as reported in Raiko et al. [17]</td>
<td></td>
</tr>
<tr>
<td>as reported in Tang and Salakhutdinov [20], M=20</td>
<td>3.99%</td>
</tr>
<tr>
<td>as reported in Raiko et al. [17], M=20</td>
<td>1.63%</td>
</tr>
</tbody>
</table>

**Table 1.** Mean test Error on MNIST for stochastic networks. The first row shows the results of our experiments averaged over 10 trials. The second row shows the results reported in [17]. M corresponds to the number of samples used for computing output probabilities. We used M=1 during training and M=100 for the test set.
• Target prop can be used to train autoencoders to achieve a good initial representation (pre-training)
• Conclusion: Target prop performs comparably to backprop with deep feedforward nets and autoencoders, and performs state-of-the-art with stochastic networks
Credit Assignment through Time: Alternatives to Backpropagation (1994)

- Authors: Yoshua Bengio, Paolo Frasconi
- Considers the use of alternative algorithms for training recurrent neural networks to avoid vanishing/exploding gradient
- Gradient descent effectiveness decreases as duration of dependencies increases
Latch: store bits of information for arbitrary durations
M: nonlinear map with (potentially) tunable parameters
A_t: system state, u_t: external input
\[ a_t = M(a_{t-1}) + u_t \]
Latching information can be done by restricting A_t to a subset of its domain
  - This region is a basin of attraction; information can be unlatched by being pushed out of the basin

Either system will be very sensitive to noise, or derivatives will converge exponentially to 0 as t increases
Credit Assignment through Time: Alternatives to Backpropagation (1994)

- Simulated annealing: perform cycle of random moves
- Multi-grid random search: search around fixed hyper-rectangle of points
- Time weighted pseudo-newton: uses second-order derivatives of the cost function wrt weights at different time steps

\[
\Delta w_i(p) = - \sum_t \frac{\eta}{|\partial^2 C(p) / \partial w^2_{it}| + \mu} \times \frac{\partial C(p)}{\partial w_{it}}
\]
Credit Assignment through Time: Alternatives to Backpropagation (1994)

- Discrete error propagation: Error propagation rules for simple discrete units (i.e. thresholds)
- EM Approach to target prop
  - Finite-state learning system
  - State $q_t$ takes on one of $n$ vals
  - Learning is max likelihood
Credit Assignment through Time: Alternatives to Backpropagation (1994)

- Propose feedforward subnetwork for each state that outputs probability using softmax
- Distribution over states at time $t$ is linear combination of outputs of subnetworks (using Markovian assumption)
- Training finds parameters $\theta$ to maximize likelihood of correct state at end of sequence

\[
P(q_t = i \mid u_t^t, \theta) = \sum_j P(q_{t-1} = j \mid u_{1}^{t-1}; \theta) P(q_t = i \mid q_{t-1} = j, u_t; \theta)
\]

\[
L(\theta) = \prod_p P(q_{T_p} = q_{T_p}^* \mid u_{1}^{T_p}; \theta) \quad L_c(\theta) = \prod_p P(q_{1}^{T_p} \mid u_{1}^{T_p}; \theta) \quad Q(\theta, \theta^k) = E[\log L_c(\theta) \mid \theta^k]
\]
Credit Assignment through Time: Alternatives to Backpropagation (1994)

---

- Experiments: classify sequences/compute parity of sequence
  - EM target prop performed the best
- Conclusion: BP can be outperformed by alternative approaches in sequence classification tasks, but generalizations
References