Summary of Paper: Difference Target Propagation (2015)

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 carrot) that leads to lower loss

$$\mathbf{h}_i = f_i(\mathbf{h}_{i-1}) = s_i(W_i \mathbf{h}_{i-1}), \;\; i = 1, \ldots, M$$

$$L(\mathbf{h}_M(\mathbf{x}; \theta_W^{0,M}), \mathbf{y}) = L(\mathbf{h}_M(\mathbf{h}_i(\mathbf{x}; \theta_W^{0,i}); \theta_W^{i,M}), \mathbf{y})$$

$$L(\mathbf{h}_M(\hat{\mathbf{h}}_i; \theta_W^{i,M}), \mathbf{y}) < L(\mathbf{h}_M(\mathbf{h}_i(\mathbf{x}; \theta_W^{0,i}); \theta_W^{i,M}), \mathbf{y})$$

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

$$L_i(\hat{\mathbf{h}}_i, \mathbf{h}_i) = ||\hat{\mathbf{h}}_i - \mathbf{h}_i(\mathbf{x}; \theta_W^{0,i})||_2^2$$

$$W_i^{(t+1)} = W_i^{(t)} - \eta_{f_i} \frac{\partial L_i(\hat{\mathbf{h}}_i, \mathbf{h}_i)}{\partial W_i} = W_i^{(t)} - \eta_{f_i} \frac{\partial L_i(\hat{\mathbf{h}}_i, \mathbf{h}_i)}{\partial \mathbf{h}_i} \frac{\partial \mathbf{h}_i(\mathbf{x}; \theta_W^{0,i})}{\partial W_i}$$

$$L_i(\hat{\mathbf{h}}_{i+1}, f_i(\hat{\mathbf{h}}_i)) < L_i(\hat{\mathbf{h}}_{i+1}, f_i(\mathbf{h}_i)) \qquad f_i(g_i(\mathbf{h}_i)) \approx \mathbf{h}_i \qquad \hat{\mathbf{h}}_{i-1} = g_i(\hat{\mathbf{h}}_i)$$

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

$$g_i(\mathbf{h}_i) = \bar{s}_i(\mathbf{V}_i \mathbf{h}_i), \quad i = 1, \ldots, M \qquad L_i^{inv} = ||g_i(f_i(\mathbf{h}_{i-1})) - \mathbf{h}_{i-1}||_2^2$$

$$L_i^{inv} = ||g_i(f_i(\mathbf{h}_{i-1} + \epsilon)) - (\mathbf{h}_{i-1} + \epsilon)||_2^2, \quad \epsilon \sim N(0, \sigma)$$

$$0 < \frac{1 + \Delta_1(\hat{\eta})}{\frac{\lambda_{max}}{\lambda_{min}} + \Delta_2(\hat{\eta})} \leq cos(\alpha) \leq 1$$

Difference Target Propagation (2015)

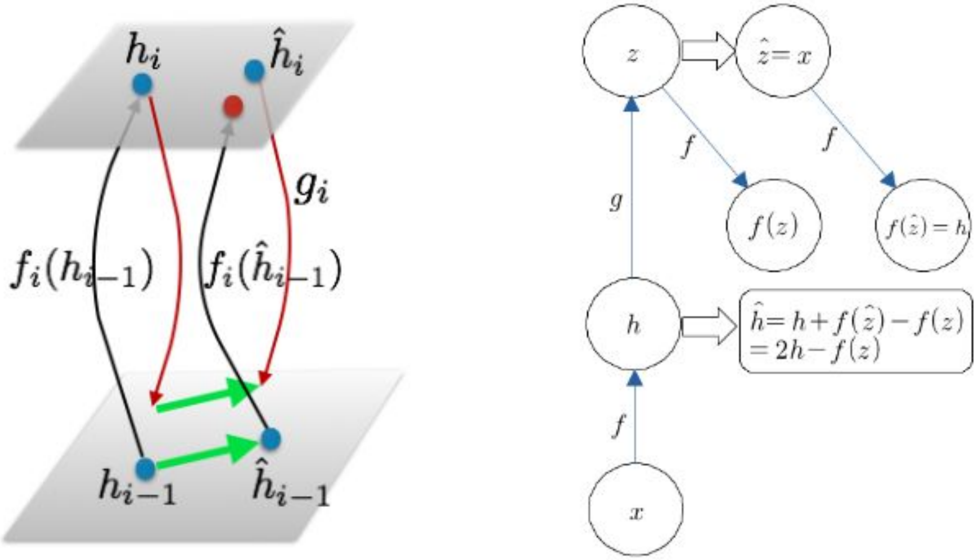


Fig. 1. (left) How to compute a target in the lower layer via difference target propagation. $f_i(\hat{\mathbf{h}}_{i-1})$ should be closer to $\hat{\mathbf{h}}_i$ than $f_i(\mathbf{h}_{i-1})$. (right) Diagram of the back-propagation-free auto-encoder via difference target propagation.

Difference Target Propagation (2015)

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

$$\hat{\mathbf{h}}_{i-1} = \mathbf{h}_{i-1} + g_i(\hat{\mathbf{h}}_i) - g_i(\mathbf{h}_i)$$

$$\mathbf{h}_i = \hat{\mathbf{h}}_i \Rightarrow \mathbf{h}_{i-1} = \hat{\mathbf{h}}_{i-1}$$

$$\mathbf{h}_{i-1} = f_i^{-1}(\mathbf{h}_i) = g_i(\hat{\mathbf{h}}_i) = \hat{\mathbf{h}}_{i-1}$$

Difference Target Propagation (2015)

Algorithm 1 Training deep neural networks via difference target propagation

Compute unit values for all layers:
for $i = 1$ to M **do**
  $\mathbf{h}_i \leftarrow f_i(\mathbf{h}_{i-1})$
end for
Making the first target: $\hat{\mathbf{h}}_{M-1} \leftarrow \mathbf{h}_{M-1} - \hat{\eta}\frac{\partial L}{\partial \mathbf{h}_{M-1}}$, ($L$ is the global loss)
Compute targets for lower layers:
for $i = M - 1$ to 2 **do**
  $\hat{\mathbf{h}}_{i-1} \leftarrow \mathbf{h}_{i-1} - g_i(\mathbf{h}_i) + g_i(\hat{\mathbf{h}}_i)$
end for
Training feedback (inverse) mapping:
for $i = M - 1$ to 2 **do**
  Update parameters for g_i using SGD with following a layer-local loss L_i^{inv}
  $L_i^{inv} = ||g_i(f_i(\mathbf{h}_{i-1} + \epsilon)) - (\mathbf{h}_{i-1} + \epsilon)||_2^2, \quad \epsilon \sim N(0, \sigma)$
end for
Training feedforward mapping:
for $i = 1$ to M **do**
  Update parameters for f_i using SGD with following a layer-local loss L_i
  $L_i = ||f_i(\mathbf{h}_{i-1}) - \hat{\mathbf{h}}_i||_2^2$ if $i < M$, $L_i = L$ (the global loss) if $i = M$.
end for

Difference Target Propagation (2015)

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

$$||\hat{\mathbf{h}}_i - f_i(\hat{\mathbf{h}}_{i-1})||_2^2 < ||\hat{\mathbf{h}}_i - \mathbf{h}_i||_2^2$$

Difference Target Propagation (2015)

- Autoencoders can also be trained using difference target prop

$$\mathbf{h} = f(\mathbf{x}) = sig(W\mathbf{x} + \mathbf{b})$$

$$\mathbf{z} = g(\mathbf{h}) = sig(W^T(\mathbf{h} + \epsilon) + \mathbf{c}), \quad \epsilon \sim N(0, \sigma)$$

$$L = ||\mathbf{z} - \mathbf{x}||_2^2 + ||f(\mathbf{x} + \epsilon) - \mathbf{h}||_2^2, \quad \epsilon \sim N(0, \sigma)$$

$$\hat{\mathbf{h}} = \mathbf{h} + f(\hat{\mathbf{z}}) - f(\mathbf{z}) = 2\mathbf{h} - f(\mathbf{z})$$

Difference Target Propagation (2015)

- Trained deep feedforward net with RMSProp optimization

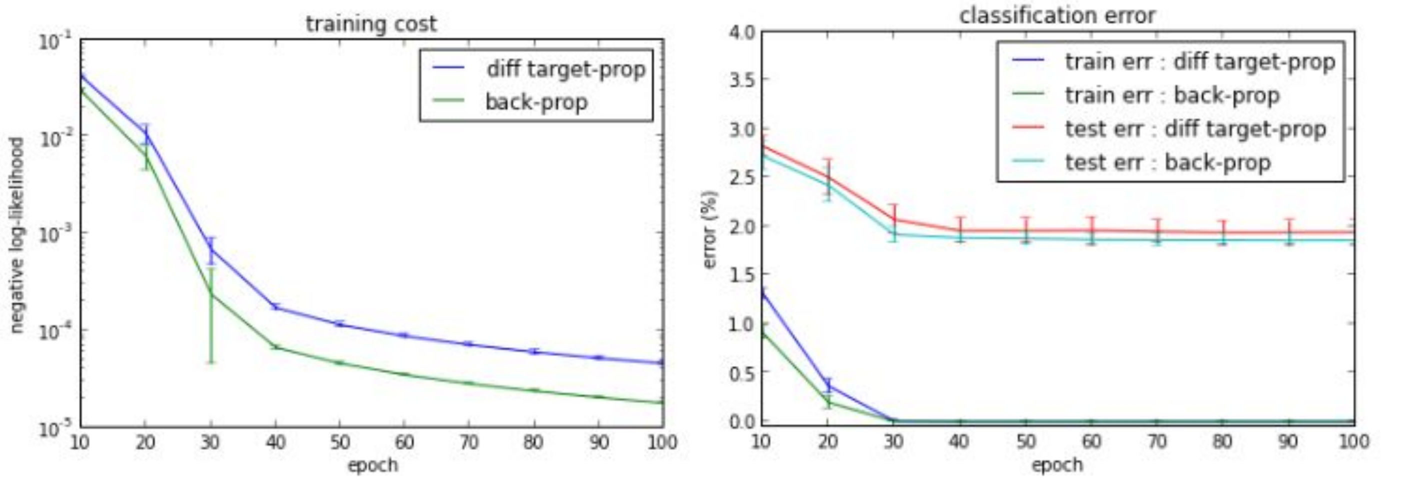


Fig. 2. Mean training cost (left) and train/test classification error (right) with target propagation and back-propagation using continuous deep networks (tanh) on MNIST. Error bars indicate the standard deviation.

Difference Target Propagation (2015)

- Trained discrete networks using target prop

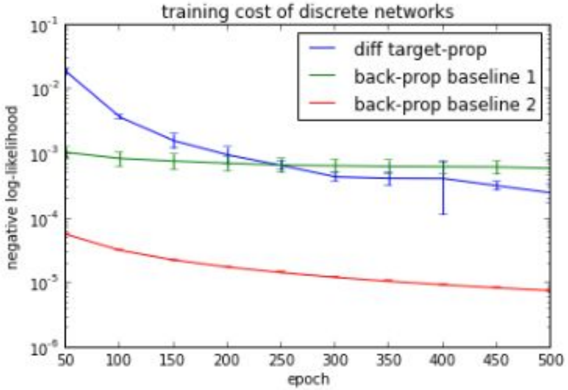
$$\mathbf{h}_1 = f_1(\mathbf{x}) = \tanh(W_1\mathbf{x}) \quad \text{and} \quad \mathbf{h}_2 = f_2(\mathbf{h}_1) = \tanh(W_2 sign(\mathbf{h}_1))$$

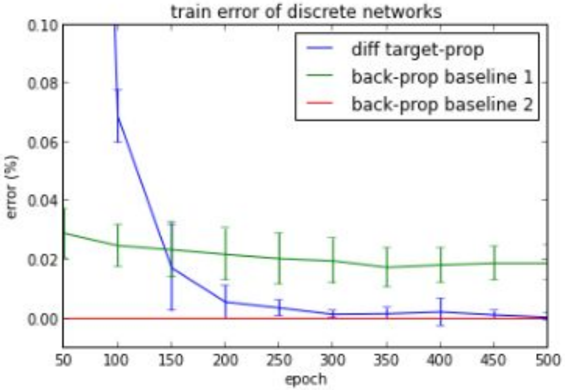
$$p(\mathbf{y}|\mathbf{x}) = f_3(\mathbf{h}_2) = softmax(W_3\mathbf{h}_2)$$

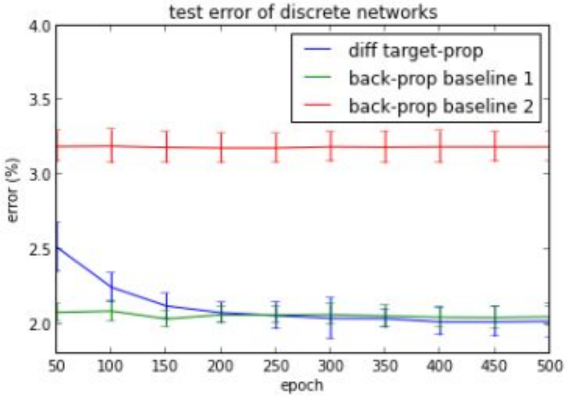
$$g_2(\mathbf{h}_2) = \tanh(V_2 sign(\mathbf{h}_2)) \quad ,$$

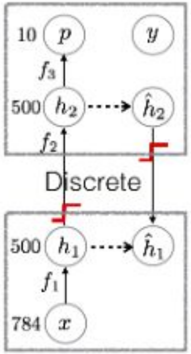
$$L_2^{inv} = ||g_2(f_2(\mathbf{h}_1 + \epsilon)) - (\mathbf{h}_1 + \epsilon)||_2^2, \quad \epsilon \sim N(0, \sigma)$$

Difference Target Propagation (2015)









Difference Target Propagation (2015)

- Also trained stochastic network using difference backprop

$$\mathbf{h}_i^p = P(\mathbf{H}_i = 1|\mathbf{h}_{i-1}) = \sigma(W_i \mathbf{h}_{i-1}), \quad \mathbf{h}_i \sim P(\mathbf{H}_i|\mathbf{h}_{i-1})$$

$$\delta \mathbf{h}_{i-1}^p = \delta \mathbf{h}_i^p \frac{\partial \mathbf{h}_i^p}{\partial \mathbf{h}_{i-1}^p} \approx \sigma'(W_i \mathbf{h}_{i-1}) W_i^T \delta \mathbf{h}_i^p$$

$$L_i^{inv} = ||g_i(f_i(\mathbf{h}_{i-1} + \epsilon)) - (\mathbf{h}_{i-1} + \epsilon)||_2^2, \quad \epsilon \sim N(0, \sigma)$$

Difference Target Propagation (2015)

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

Table 1. Mean test Error on MNIST for stochastoc 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.

Difference Target Propagation (2015)

- 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

Credit Assignment through Time: Alternatives to Backpropagation (1994)

- 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

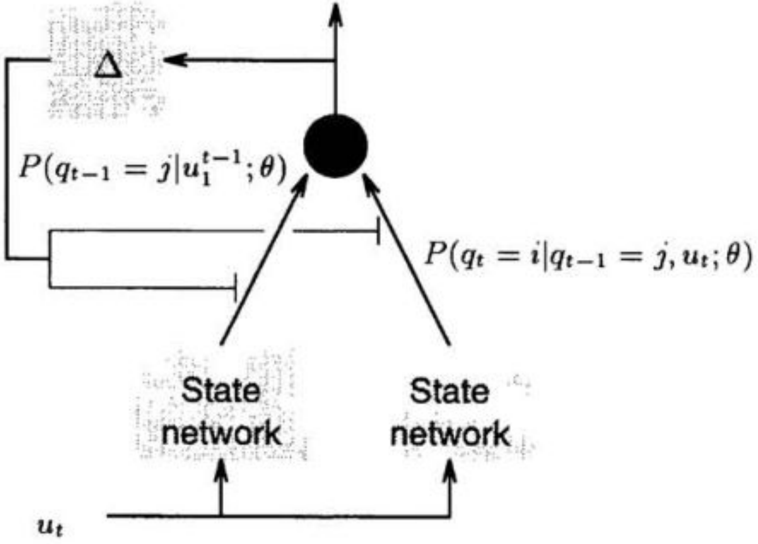
Credit Assignment through Time: Alternatives to Backpropagation (1994)

- 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

$$\Delta w_i(p) = -\sum_t \frac{\eta}{\left|\frac{\partial^2 C(p)}{\partial w_{it}^2}\right| + \mu} \times \frac{\partial C(p)}{\partial w_{it}}$$

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

$$P(q_t = i \mid u_1^t; \theta) = \sum_j P(q_{t-1} = j \mid u_1^{t-1}; \theta) P(q_t = i \mid q_{t-1} = j, u_t; \theta)$$

$$L(\theta) = \prod_p P(q_{T_p} = q^*_{T_p} \mid u_1^{T_p}; \theta) \qquad L_c(\theta) = \prod_p P(q_1^{T_p} \mid u_1^{T_p}; \theta) \qquad Q(\theta, \theta^k) = E[\log L_c(\theta) \mid \theta^k]$$

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

- https://arxiv.org/pdf/1412.7525v5.pdf
- https://papers.nips.cc/paper/724-credit-assignment-through-time-alternatives-to-backpropagation.pdf