TARGET PROPAGATION VIA REGULARIZED INVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Target Propagation (TP) algorithms compute targets instead of gradients along neural networks and propagate them backward in a way that is similar yet different than gradient back-propagation (BP). The idea was first presented as a perturbative alternative to back-propagation that may achieve greater accuracy in gradient evaluation when training multi-layer neural networks (Le Cun et al., 1989). However, TP has remained more of a template algorithm with many variations than a well-identified algorithm. Revisiting insights of Le Cun et al. (1989) and more recently of Lee et al. (2015), we present a simple version of target propagation based on regularized inversion of network layers, easily implementable in a differentiable programming framework. We compare its computational complexity to the one of BP and delineate the regimes in which TP can be attractive compared to BP. We show how our TP can be used to train recurrent neural networks with long sequences on various sequence modeling problems. The experimental results underscore the importance of regularization in TP in practice.

1 INTRODUCTION

Target propagation algorithms can be seen as perturbative learning alternatives to the gradient back-propagation algorithm, where virtual targets are propagated backwards instead of gradeints (Le Cun, 1986; Le Cun et al., 1989; Rohwer, 1990; Mirowski & LeCun, 2009; Bengio, 2014; Goodfellow et al., 2016). A high-level summary is presented in Fig. 1: while gradient back-propagation considers storing intermediate gradients in a forward pass, target propagation algorithms proceed by computing and storing approximate inverses. The approximate inverses are then passed on backwards along the graph of computations to finally yield a descent direction of the learning objective.

Target propagation aims to take advantage of the availability of approximate inverses to compute better descent directions for the objective at hand. Bengio et al. (2013); Bengio (2020) argued that the approach could be relevant for problems involving multiple compositions such as the training of Recurrent Neural Networks (RNNs), which generally suffer from the phenomenon of exploding or vanishing gradients (Hochreiter, 1998; Bengio et al., 1994). Recently, empirical results indeed showed the superiority of target propagation over classical gradient back-propagation for training RNNs on several tasks (Manchev & Spratling, 2020). These recent investigations remain, however, build on multiple approximations which hinder the analysis of the core idea of TP, i.e., using inverses.

On the theoretical side, difference target propagation, a modern variant of target propagation, was related to an approximate Gauss-Newton method, suggesting interesting venues to explain the benefits of target propagation (Bengio, 2020; Meulemans et al., 2020). Previous works have considered approximating inverses by means of additional reverse layers (Manchev & Spratling, 2020; Meulemans et al., 2020; Bengio, 2020). However, it is unclear whether such reverse layers actually learn inverses throughout the training process. Even if they were, the additional cost in terms of computational complexity of learning approximate inverses should be carefully accounted for.

In this work, we propose a simple target propagation approach, revisiting original insights of Le Cun et al. (1989) on the critical importance of the good conditioning of layer inverses. We define regularized inverses through a variational formulation and we obtain approximate inverses via these regularized inverses. In this spirit, we can also interpret the difference target propagation formula (Lee et al., 2015) as a finite difference approximation of a linearized regularized inverse. We propose a smoother formula that can directly be integrated in a differentiable programming framework.

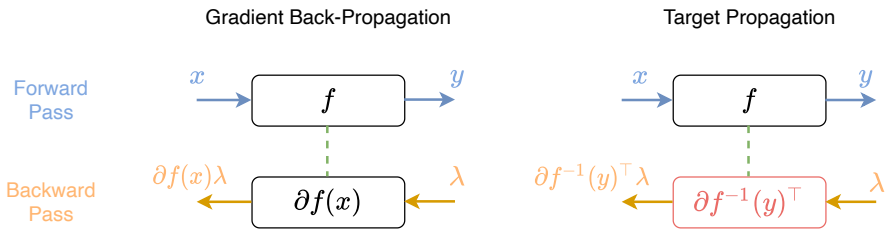


Fig. 1: Our implementation of target propagation uses linearization of gradient inverses instead of gradients in a backward pass akin to gradient back-propagation.

We detail the computational complexity of the proposed target propagation and compare it to the one of gradient back-propagation, showing that the additional cost of computing inverses can be effectively amortized for very long sequences. Following the benchmark of Manchev & Spratling (2020), we observe that the proposed target propagation can perform better and yields smoother behavior than classical gradient-based methods on several tasks involving RNNs.

Related Work. Many variations of back-propagation algorithms have been explored; see Werbos (1994); Goodfellow et al. (2016) for an extensive bibliography. Closer to target propagation, penalized formulations of the training problem have been considered to decouple the optimization of the weights in a distributed way or using an ADMM approach (Carreira-Perpinan & Wang, 2014; Taylor et al., 2016; Gotmare et al., 2018). Rather than modifying the backward operations in the layers, one can also modify the weight updates for deep forward networks by using a regularized inverse (Frerix et al., 2018). Wiseman et al. (2017) recast target propagation as an ADMM-like algorithm for language modeling and reported disappointing experimental results. Recently, in a careful experimental benchmark evaluation, Manchev & Spratling (2020) explored further target propagation to train RNNs mapping a sequence to a single final output, in an attempt to understand the benefits of target propagation to capture long range dependencies, and obtained promising experimental results. Another line of research has considered synthetic gradients that approximate gradients using an additional layer instead of using back-propagated gradients (Jaderberg et al., 2017; Czarnecki et al., 2017) to speed up the training of deep neural networks. Recently, Ahmad et al. (2020); Dalm et al. (2021) considered using analytical inverses to implement target propagation and blend it with what they called a gradient adjusted incremental formula. Yet, an additional orthogonality penalty is critical for their approach to work. Recently, Meulemans et al. (2020) considered using as many reverse layers as forward operations. We focus here on the optimization gains of using target propagation that cannot be obtained by adding a prohibitive number of reverse layers. Finally, we do not discuss the biological plausibility of TP since we are unable to comment on this. We refer the interesting reader to e.g. (Bengio, 2020).

Notations. For $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^d$, we denote partial derivative of f w.r.t. $x \in \mathbb{R}^p$ as $\partial_x f(x, y) = \big(\partial f^j(x, y)/\partial x_i\big)_{i,j} \in \mathbb{R}^{d \times p}$.

2 Target Propagation with Linearized Regularized Inverses

While target propagation was originally developed for multi-layer neural networks, we focus on its implementation for recurrent neural networks, as we shall follow the benchmark of Manchev & Spratling (2020) in the experiments. Recurrent Neural Networks (RNNs) are also a canonical family of neural networks in which interesting phenomena arise in back-propagation algorithms.

Problem setting. A simple RNN parameterized by $\theta = (W_{hh}, W_{xh}, b_h, W_{hy}, b_y)$ maps a sequence of inputs $x_{1:\tau} = (x_1, \dots, x_\tau)$ to an output $\hat{y} = g_\theta(x_{1:\tau})$ by computing hidden states $h_t \in \mathbb{R}^p$ corresponding to the inputs x_t. Formally, the output $\hat{y}$ and the hidden states h_t are computed as an output operation following transition operations defined as

$$\hat{y} = c_\theta(h_\tau) := s(W_{hy} h_\tau + b_y),$$
$$h_t = f_{\theta,t}(h_{t-1}) := a(W_{xh} x_t + W_{hh} h_{t-1} + b_h) \quad \text{for } t \in \{1, \dots, \tau\},$$

where s is e.g. the soft-max function for classification tasks, a is a non-linear operation such as the hyperbolic tangent function and the initial hidden state is generally fixed as $h_0 = 0$. Given samples of sequence-output pairs $(x_{1:\tau}, y)$, the RNN is trained to minimize the error $\ell(y, g_\theta(x_{1:\tau}))$ of predicting $\hat{y} = g_\theta(x_{1:\tau})$ instead of y.

As one considers longer sequences, RNNs face the challenge of exploding/vanishing gradients $\partial g_\theta(x_{1:\tau})/\partial h_t$ (Bengio & Frasconi, 1995); see Appendix A for more discussion. We acknowledge that specific parameterization-based strategies have been proposed to address this issue of exploding/vanishing gradients such as orthonormal parameterizations of the weights (Arjovsky et al., 2016; Helfrich et al., 2018; Lezcano-Casado & Martınez-Rubio, 2019). The focus here is to simplify and understand target propagation as a backpropagation-type algorithm using RNNs as a workbench. Indeed, training RNNs is an optimization problem involving multiple compositions for which approximate inverses can easily be available. The framework could also be potentially applied to *e.g.*, time-series or control models (Roulet et al., 2019).

Given the parameters W_{hh}, W_{xh}, b_h of the transition operations, we can get approximate inverses of $f_{\theta,t}(h_{t-1})$ for all $t \in \{1, \dots, \tau\}$, that yield optimization surrogates that can be better performing than the ones corresponding to regular gradients. We present below a *simple version* of target propagation based on *regularized inverses* and *inverse linearizations*.

Back-propagating targets. The idea of target propagation is to back-propagate virtual targets v_t for $t = \tau, \dots, 1$ such that if the network was able to match the target at time t, i.e., $f_{\theta,t}(h_{t-1}) \approx v_t$, the objective would decrease. The final target v_τ is computed as a gradient step on the loss w.r.t. h_τ. The targets are then back-propagated using an approximate inverse[1] $f_{\theta,t}^{-1}$ of $f_{\theta,t}$ at each time step.

Formally, consider an RNN that computed τ states $h_1, \dots, h_\tau$ from a sequence $x_1, \dots, x_\tau$ with associated output y. For a given stepsize $\gamma_h > 0$, we propose to back-propagate targets by computing

$$v_\tau = h_\tau - \gamma_h \partial_h \ell(y, c_\theta(h_\tau)), \tag{1}$$

$$v_{t-1} = h_{t-1} + \partial_h f_{\theta,t}^{-1}(h_t)^\top (v_t - h_t), \quad \text{for } t \in \{\tau, \dots, 1\}. \tag{2}$$

The update rule (2) blends two idea: i) regularized inversion; ii) linear approximation. We shall describe below that our update (2) allows us tp interpret the magic formula of difference target propagation in Eq. 15 of Lee et al. (2015) as 0-order-th finite difference approximation, while ours is a 1st-order linear approximation. We shall also show that (2) puts in practice an insight from Bengio (2020) suggesting to use the inverse of the gradients in the spirit of a Gauss-Newton method.

Once all targets are computed, the parameters of the transition operations are updated such that the outputs of $f_{\theta,t}$ at each time step move closer to the given target. Formally, the update consists in a gradient step with stepsize γ_θ on the squared error between the targets and the current outputs, i.e., for $\theta_h \in \{W_{hh}, W_{xh}, b_h\}$,

$$\theta_h^{\text{next}} = \theta_h - \gamma_\theta \sum_{t=1}^{\tau} \partial_{\theta_h} \|f_{\theta,t}(h_{t-1}) - v_t\|_2^2/2 \tag{3}$$

As for the parameters $\theta_y = (W_{hy}, b_y)$ of the output operation, they are updated by a simple gradient step on the loss with a stepsize γ_θ.

2.1 Regularized Inversion

To explore further the original idea of Le Cun et al. (1989), we consider using the variational definition of the inverse,

$$f_{\theta,t}^{-1}(v_t) = \operatorname*{argmin}_{v_{t-1} \in \mathbb{R}^p} \|f_{\theta,t}(v_{t-1}) - v_t\|_2^2 = \operatorname*{argmin}_{v_{t-1} \in \mathbb{R}^p} \|a(W_{xh}x_t + W_{hh}v_{t-1} + b_h) - v_t\|_2^2. \tag{4}$$

As long as v_t belongs to the image $f_{\theta,t}(\mathbb{R}^p)$ of $f_{\theta,t}$, this definition recovers exactly the inverse of v_t by $f_{\theta,t}$. More generally, if $v_t \notin f_{\theta,t}(\mathbb{R}^p)$, Eq. (4) computes the *best approximation* of the inverse in the sense of the Euclidean projection. When one considers an activation function a and $\theta_h = (W_{hh}, W_{xh}, b_h)$, the solution of (4) can easily be computed.

[1] In the following, to ease the presentation, we abuse notations and denote approximate inverses by $f_{\theta,t}^{-1}$.

Formally, for the sigmoid, the hyperbolic tangent or the ReLU, their inverse can be obtained analytically for any $v_t \in a(\mathbb{R}^p)$. So for $v_t \in a(\mathbb{R}^p)$ and W_{hh} full rank, we get

$$f_{\theta,t}^{-1}(v_t) = (W_{hh}^\top W_{hh})^{-1} W_{hh}^\top (a^{-1}(v_t) - W_{xh}x_t - b_h).$$

If $v_t \notin a(\mathbb{R}^p)$, the minimizer of (4) is obtained by first projecting v_t onto $a(\mathbb{R}^p)$, before inverting the linear operation. To account for non-invertible matrices W_{hh} we also add a regularization in the computation of the inverse. Overall we consider approximating the inverse of the layer by a regularized inverse of the form

$$f_{\theta,t}^{-1}(v_t) = (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1} W_{hh}^\top (a^{-1}(\pi(v_t)) - W_{xh}x_t - b_h).$$

with $r > 0$ and π a projection onto $a(\mathbb{R}^p)$.

Regularized Inversion vs. Parameterized Inversion. Bengio (2014); Manchev & Spratling (2020) parameterize the inverse as a reverse layer such that

$$f_{\theta,t}^{-1}(v_t) = \psi_{\theta',t}(v_t) := a(W_{xh}x_t + Vv_t + c).$$

and learn the parameters $\theta' = (V, c)$ for this reverse layer to approximate the inverse of the forward computations. The parameterized layer actually needs to be learnt in order to get a good approximation which involves numerically solving an optimization problem for each layer. These optimization problems come with a computational cost which can be better controlled by using regularized inversions presented earlier.

The approach based on parameterized inverses, however, may lack theoretical grounding, as pointed out by Bengio (2020), as we do not know how close is the learnt inverse to the actual inverse throughout the training process. In contrast, the regularized inversion (4) is less *ad hoc* and clearly defined, and, as we shall show in the experiments, leads to smooth curves during the training process and competitive performance on real datasets.

In any case, the analytic formulation of the inverse gives simple insights on an approach with parameterized inverses, were those chosen to be used. Namely, the analytic formula suggests parameterizing the *reverse layer* s.t. (i) the reverse activation is defined as the inverse of the activation and not any activation, (ii) the layer uses a non-linear operation followed by a linear one instead of the usual scheme, i.e., a linear operation followed by a non-linear one.

2.2 Linearized Inversion

Earlier instances of target propagation used directly inverses of the network layers such that the target propagation update formula would read $v_{t-1} = f_{\theta,t}^{-1}(v_t)$ in (2). Yet, we are unaware of successful implementation of TP using directly the inverses. To circumvent this issue, Lee et al. (2015) proposed the *difference target propagation* formula that back-propagates the targets as

$$v_{t-1} = h_{t-1} + f_{\theta,t}^{-1}(v_t) - f_{\theta,t}^{-1}(h_t).$$

If the inverses were exact, the difference target propagation formula would reduce to $v_{t-1} = f_{\theta,t}^{-1}(v_t)$. Lee et al. (2015) introduced the difference target propagation formula to mitigate the approximation error of the inverses by parameterized layers. In practice, difference-type target propagation appear to be the only known successful implementations of target propagation we are aware of. The difference target propagation formula can naturally be interpreted as an approximation of the linearization used in (2), as

$$f_{\theta,t}^{-1}(v_t) - f_{\theta,t}^{-1}(h_t) = \partial_h f_{\theta,t}^{-1}(h_t)^\top (v_t - h_t) + O(\|v_t - h_t\|_2^2)$$

where $\partial_h f_{\theta,t}^{-1}(h_t)^\top$ denotes the Jacobian of the inverse of the layer at h_t.

We show in the Appendix D that the first-order approximation we propose (2) leads to slightly better training curves than the finite-difference approximation. Moreover, our interpretation illuminates the mystery of this formula which appeared to be critical to the success of target propagation.

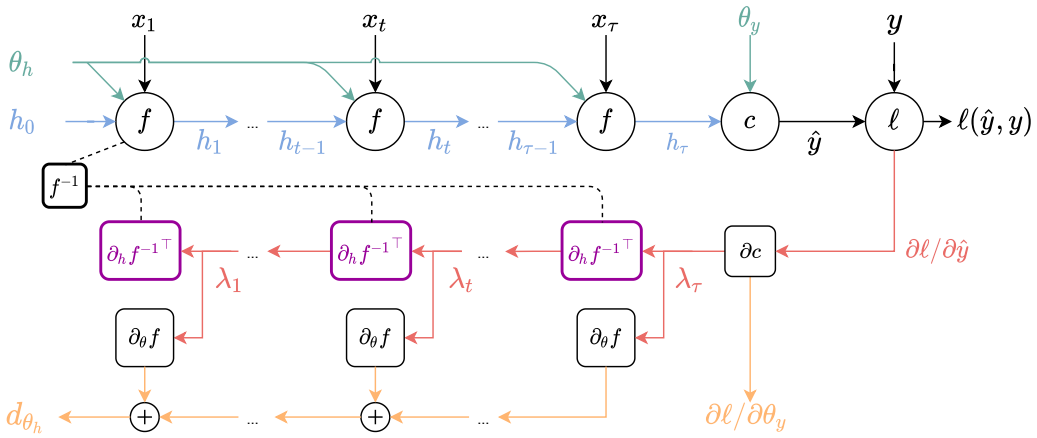


Fig. 2: The graph of computations of target propagation is the same as the one of gradient back-propagation except that f^{-1} needs to be computed and Jacobian of the inverses, $\partial_h f^{-1\top}$ are used instead of gradients $\partial_h f$ in the transition operations.

3 Gradient Back-propagation versus Target Propagation

Graph of computations. Gradient back-propagation and target propagation both compute a descent direction for the objective at hand. The difference lies in the oracles computed and stored in the forward pass, while the graph of computations remains the same. To clarify this view, we reformulate target propagation in terms of displacements $\lambda_t := v_t - h_t$ such that Eq. (1), (2) and (3) read

$$\lambda_\tau = -\gamma_h \partial_h \ell(y, c_\theta(h_\tau)), \qquad \lambda_{t-1} = \partial_h f_{\theta,t}^{-1}(h_t)^\top \lambda_t, \quad \text{for } t \in \{\tau, \dots, 1\},$$

$$d_{\theta_h} = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \lambda_t, \qquad \theta_h^{\text{next}} = \theta_h + \gamma_h d_{\theta_h}.$$

Target propagation amounts then to computing a descent direction d_{θ_h} for the parameters θ_h with a graph of computations, illustrated in Fig. 2, analogous to that of gradient-back-propagation illustrated in Appendix A. The difference lies in the use of the Jacobian of the inverse

$$\partial_h f_{\theta,t}^{-1}(h_t)^\top \quad \text{instead of} \quad \partial_h f_{\theta,t}(h_{t-1}).$$

The implementation of TP with the formula (2) can be done in a differentiable programming framework, where, rather than computing the gradient of the layer, one evaluates the inverse and keep the Jacobian of the inverse. With the precise graph of computation of TP and BP, we can compare explictly their computational complexity and bound the difference of the directions they output.

Arithmetic complexity. Clearly, the space complexities of gradient back-propagation (BP) and our implementation of target propagation (TP) are the same since the Jacobians of the inverse and the original gradients have the same size. In terms of time-complexity, TP appears at first glance to introduce an important overhead since it requires the computation of some inverses. However, a close inspection of the formula of the regularized inverse reveals that a matrix inversion needs to be computed only once for all time steps. Therefore the cost of the inversion may be amortized if the length of the sequence is particularly long.

Formally, the time complexity of the forward-backward pass of gradient back-propagation is essentially driven by matrix vector products, i.e.,

$$\mathcal{T}_{\text{BP}} = \sum_{t=1}^{\tau} \left[\underbrace{\mathcal{T}(f_{\theta,t}) + \mathcal{T}(\partial_h f_{\theta,t}) + \mathcal{T}(\partial_{\theta_h} f_{\theta,t})}_{\text{Forward}} + \underbrace{\mathcal{T}(\partial_h f_{\theta,t}(h_{t-1})) + \mathcal{T}(\partial_\theta f_{\theta,t}(h_{t-1}))}_{\text{Backward}} \right]$$
$$\approx \tau(dp + p^2 + pq) + \tau(p^2 + pq),$$

where, for a function f we denote by $\mathcal{T}(f)$ the time complexity to evaluate f and we consider e.g. $\partial_\theta f_{\theta,t}(h_{t-1}))$ as the linear function $\lambda \to \partial_\theta f_{\theta,t}(h_{t-1}))\lambda$. On the other hand, the time complexity of target propagation is

$$\mathcal{T}_{\text{TP}} = \sum_{t=1}^{\tau} \left[\underbrace{\mathcal{T}(f_{\theta,t}) + \mathcal{T}(f_{\theta,t}^{-1}) + \mathcal{T}\left(\partial_{\theta_h} f_{\theta,t}\right) + \mathcal{T}(\partial_h f_{\theta,t}^{-1})}_{\text{Forward}} + \underbrace{\mathcal{T}(\partial_h f_{\theta,t}^{-1})(h_t)^\top) + \mathcal{T}(\partial_\theta f_{\theta,t}(h_{t-1}))}_{\text{Backward}} \right] + \mathcal{P}(f_{\theta,t}^{-1})$$

where $\mathcal{P}(f_{\theta,t}^{-1})$ is the cost of encoding the inverse, which, in our case, amounts to the cost of encoding $g_\theta : z \to (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1} W_{hh}^\top$, such that our regularized inverse can be computed as $f_{\theta,t}^{-1}(v_t) = g_\theta(a^{-1}(v_t) - W_{xh}x_t + b_h)$. Encoding g comes at the cost of inverting one matrix of size p. Therefore, the time-complexity of target propagation can be estimated as

$$\mathcal{T}_{\text{TP}} \approx p^3 + \tau(dp + p^2 + pq) + \tau(p^2 + pq) \approx \mathcal{T}_{\text{BP}} \quad \text{if } \tau \geq p,$$

i.e., for long sequences whose length is larger than the dimension of the hidden states, the cost of TP with regularized inverses is approximately the same as the cost of BP. If a parameterized inverse was used rather than a regularized inverse, the cost of encoding the inverse would correspond to the cost of updating the reverse layers by e.g. a stochastic gradient descent. This update has a cost similar to BP. However, it is unclear whether these updates get us close to the actual inverses.

Bounding the difference between target propagation and gradient back-propagation. As the computational graphs of BP and TP are the same, we can bound the difference between the oracles returned by both methods. First note that the updates of the parameters of the output functions are the same since in TP, gradients steps of the loss are used to update these parameters. The difference between TP and BP lies in the updates with respect to the parameters of the transition operations. For BP, the updates are computed by chain rule as

$$\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right) = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \frac{\partial h_\tau}{\partial h_t} \partial_h \ell(y, c_\theta(h_\tau)),$$

where the term $\partial h_\tau / \partial h_t$ decomposes along the time steps as $\partial h_\tau / \partial h_t = \prod_{s=t+1}^{\tau} \partial_h f_{\theta,s}(h_{s-1})$. The direction computed by TP has the same structure, namely it can be decomposed for $\gamma_h = 1$ as

$$d_\theta = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \frac{\hat{\partial} h_\tau}{\hat{\partial} h_t} \partial_h \ell(y, c_\theta(h_\tau)),$$

where $\hat{\partial} h_\tau / \hat{\partial} h_t = \prod_{s=t+1}^{\tau} \partial_h f_{\theta,s}^{-1}(h_s)^\top$. We can then bound the difference between the directions given by BP or TP as, for any matrix norm $\|\cdot\|$ as formally stated in the following lemma.

Lemma 3.1. *The difference between the oracle returned by gradient back-propagation $\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right)$ and the oracle returned by target propagation can be bounded as*

$$\|\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right) - d_\theta\| \leq c \sup_{t=1,\dots,\tau} \|\partial_h f_{\theta,t}(h_{t-1}) - \partial_h f_{\theta,t}^{-1}(h_t)^\top\|,$$

where $c = \sum_{t=1}^{\tau} \sum_{s=0}^{t-1} a^s b^{t-1-s}$ with $a = \sup_{t=1,\dots\tau} \|\partial_h f_{\theta,t}(h_{t-1})\|$, $b = \sup_{t=1,\dots\tau} \|\partial_h f_{\theta,t}^{-1}(h_t)^\top\|$.

For regularized inverses, we have, denoting $u_t = W_{xh}x_t + W_{hh}h_{t-1} + b_h$,

$$\|\partial_h f_{\theta,t}(h_{t-1}) - \partial_h f_{\theta,t}^{-1}(h_t)^\top\| \leq \|W_{hh}^\top\| \Big(\|\nabla a(u_t) - \nabla a(u_t)^{-1}\| + \|\,\mathrm{I} - (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1}\| \|\nabla a(u_t)^{-1}\| \Big).$$

For the two oracles to be close we then need the preactivation $u_t = W_{xh}x_t + W_{hh}h_{t-1} + b_h$ to lie in the region of the activation function that is close to be linear s.t. $\nabla a(u_t) \approx \mathrm{I}$. We also need $(W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1}$ to be close to the identity which can be the case if e.g. $r = 0$ and the weight matrices W_h were orthonormal. By initializing the weight matrices as orthonormal matrices, the differences between the two oracles can be closer. However in the long term, target propagation appears to give better oracles as shown in the experiments below.

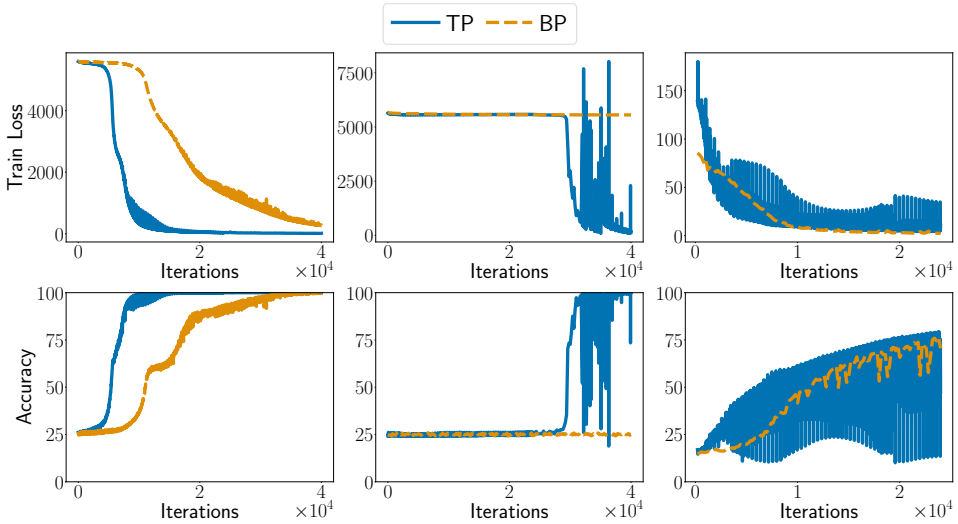


Fig. 3: Temporal order problem $T = 60$, Temporal Problem $T = 120$, Adding problem $T = 30$.

Target propagation as a Gauss-Newton method? Recently target propagation has been interpreted as an approximate Gauss-Newton method, by considering that the difference target propagation formula approximates the linearization of the inverse, which itself is a priori equal to the inverse of the gradients (Bengio, 2020; Meulemans et al., 2020; 2021). Namely, provided that $f_{\theta,t}^{-1}(f_{\theta,t}(h_{t-1})) \approx h_{t-1}$ such that $\partial_h f_{\theta,t}(h_{t-1})\partial_h f_{\theta,t}^{-1}(h_t) \approx \mathrm{I}$, we have

$$\partial_h f_{\theta,t}^{-1}(h_t) \approx \left(\partial_h f_{\theta,t}(h_{t-1})\right)^{-1}.$$

By composing the inverses of the gradients, we get an update similar to the one of Gauss-Newton method except that (i) the gradients with respect to the loss are not inverted, (ii) the gradients with respect to the parameters are not inverted either. By computing the exact inverse without regularization, we can test whether this interpretation may explain the potential success of TP. Namely, if $r = 0$ we have the equality $\partial_h f_{\theta,t}(h_{t-1})\partial_h f_{\theta,t}^{-1}(h_t) = \mathrm{I}$. In our experiments, we have not been able to get a non-diverging sequence of iterates if the regularization was set to 0, which questions the interpretation of target propagation as a Gauss-Newton method, at least to make it work efficiently from an optimization viewpoint.

4 Experiments

In the following, we compare our simple target propagation approach which we shall refer to as **TP**, to gradient Back-Propagation (**BP**). Additional experiments, details on the initialization and the hyper-parameter selection can be found in Appendix D.

Data. We consider two synthetic datasets generated to present training difficulties for RNNs and several real datasets consisting in scanning images pixel by pixel to classify them (Hochreiter & Schmidhuber, 1997; Le et al., 2015; Manchev & Spratling, 2020).

Temporal order problem. A sequence of length T is generated using a set of randomly chosen symbols $\{a, b, c, d\}$. Two additional symbols X and Y are added at positions $t_1 \in [T/10, 2T/10]$ and $t_2 \in [4T/10, 5T/10]$. The network must predict the correct order of appearance of X and Y out of four possible choices $\{XX, XY, YX, YY\}$.

Adding problem. The input consists in two sequences: one is made of randomly chosen numbers from $[0, 1]$ and the other one is a binary sequence full of zeros except at positions $t_1 \in [1, T/10]$ and $t_2 \in [T/10, T/2]$. The second position acts as a marker for the time steps t_1 and t_2. The goal of the network is to output the mean of the two random numbers of the first sequence $(X_{t_1} + X_{t_2})/2$.

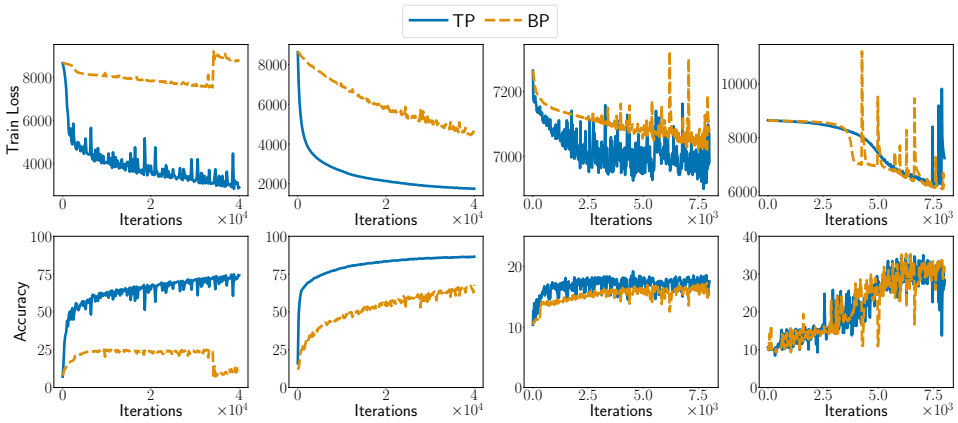


Fig. 4: Image classification pixel by pixel. From left to right: MNIST, MNIST with permuted images, CIFAR10, FashionMNIST with GRU.

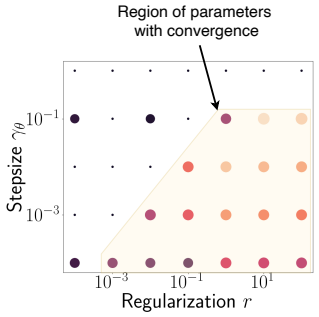
Image classification pixel by pixel. The inputs are images of (i) grayscale handwritten digits given in the database MNIST (LeCun & Cortes, 1998), (ii) colored objects from the database CIFAR10 (Krizhevsky, 2009) or (iii) grayscale images of clothes from the database FashionMNIST (Xiao et al., 2017). The images are scanned pixel by pixel, and channel by channel for CIFAR10, and fed to a sequential network such as a simple RNN or a GRU network (Cho et al., 2014). The inputs are then sequences of $28 \times 28 = 784$ pixels for MNIST or FashionMNIST and $32 \times 32 \times 3 = 3072$ pixels for CIFAR with a very long range dependency problem. We also consider permuting the images of MNIST by a fixed permutation before feeding them into the network which gives potentially longer dependencies in the sequential data.

Model. In both synthetic settings, we consider randomly generated mini-batches of size 20, a simple RNN with hidden states of dimension 100 and hyperbolic tangent activation. For the temporal order problem, the last layer uses a soft-max function on top of a linear operation and the loss is the cross-entropy. For the adding problem the last layer is linear, the loss is the mean-squared error and a sample is considered to be accurately predicted if the mean squared error is less than 0.04 as done by (Manchev & Spratling, 2020).

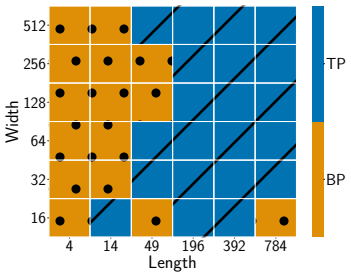
For the classification of images with sequential networks, we consider mini-batches of size 16 and a cross-entropy loss. For MNIST and CIFAR, we consider a simple RNN with hidden states of dimension 100, hyperbolic tangent activation and a softmax output. For FashionMNIST, we consider a GRU network and adapted our implementation of target propagation to that case, while using hidden states of dimension 100 and a softmax output.

Target propagation can tackle long sequences better than gradient back-propagation. In Fig. 3, we observe that TP performs better than BP on the temporal ordering problem: it is able to reach 100% accuracy in fewer iterations than BP for sequences of length 60 and, for sequences of length 120, it is still able to reach 100% accuracy in fewer than 40 000 iterations while BP is not. On the other hand, for the adding problem, TP performs less well than BP. The contrast in performance between the two synthetic tasks was also observed by (Manchev & Spratling, 2020) using difference target propagation with parameterized inverses. The main difference between these tasks is the different nature of the outputs, which are binary for the temporal problem and continuous for the adding problem.

In Fig. 4, we observe that TP performs generally better than BP for image classification tasks. For the MNIST dataset, it reaches around 74% accuracy after $4 \cdot 10^4$ iterations. This phenomenon is also observed with permuted images, where the optimization appears smoother and TP obtains round 86% accuracy after $4 \cdot 10^4$ iterations and is still faster than BP. On the CIFAR dataset, no algorithms appears to reach a significant accuracy, though TP is still faster. On the FashionMNIST dataset, where a GRU network is used, our implementation of TP performs on par with BP, which shows that our approach can be generalized to more complex networks than a simple RNN.



(6a) Conv. w.r.t. stepsize & regularization



(6b) Perf. vs width & length.

Target propagation requires a non-zero regularization term. As mentioned in Sec. 3, by using an analytical formula to compute the inverse of the layers, we can question the interpretation of TP as a Gauss-Newton method, which would amount to TP without regularization. To understand the effect of the regularization term, we computed the area under the training loss curve of TP for 400 iterations on a $\log_{10}$ grid of varying step-sizes γ_θ and regularizations r for a fixed $\gamma_h = 10^{-3}$. The results are presented in Fig. 6a, where the smaller the area, the brighter the point and the absence of dots in the grid mean that the algorithm diverged. Fig. 6a shows that the regularization needs generally to be ten times larger than the stepsize to avoid divergence. Simply using the gradients of the inverse as in a Gauss-Newton method may not directly work for RNNs. Additional modifications of the method could be added to make target propagation closer to Gauss-Newton such as inverting the layers with respect to their parameters as proposed by Bengio (2020). For now, the regularization appears to successfully handle the rationale of target propagation.

Target propagation is adapted for long sequences. In Fig. 6b, we compare the performance of BP and TP in terms of accuracy after 400 iterations on the MNIST problem for various widths determined by the size of the hidden states and various lengths determined by the size of the inputs (i.e., we feed the RNN with k pixels at a time, which gives a length 784/k). Fig 6b shows that TP is generally appropriate for long sequences, while BP remains more efficient for short sequences. TP can then be seen as an interesting alternative for dynamical problem which involve many discretization steps as in RNNs and related architectures.

CONCLUSION

We proposed a simple target propagation approach grounded in two important computational components, regularized inversion and linearize propagation. The proposed approach also sheds light on previous insights and successful rules for target propagation. We will publicly release our code to facilitate to reproduce the results, as we believe that a major impediment to the use of target propagation has been the lack of software as well as a lack of clarity of the updates rules. We have used target propagation within a stochastic gradient outer loop to train neural networks for a fair comparison to stochastic gradient using gradient backpropagation. Developing adaptive stochastic gradient algorithms in the spirit of Adam that lead to boosts in performance when using target propagation instead of gradient backpropagation is an interesting venue for future work. Continuous counterparts of target propagation in a neural ODE spirit is also an interesting venue for future work.

REFERENCES

Nasir Ahmad, Marcel A van Gerven, and Luca Ambrogioni. Gait-prop: A biologically plausible learning rule derived from backpropagation of error. *Advances in Neural Information Processing Systems*, 33, 2020.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Yoshua Bengio. Deriving differential target propagation from iterating approximate inverses. *arXiv preprint arXiv:2007.15139*, 2020.

Yoshua Bengio and Paolo Frasconi. Diffusion of context and credit information in markovian models. *Journal of Artificial Intelligence Research*, 3:249–270, 1995.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Sander Dalm, Nasir Ahmad, Luca Ambrogioni, and Marcel van Gerven. Scaling up learning with gait-prop. *arXiv preprint arXiv:2102.11598*, 2021.

Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020.

Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

Akhilesh Gotmare, Valentin Thomas, Johanni Brea, and Martin Jaggi. Decoupling backpropagation using constrained optimization methods. In *Credit Assignment in Deep Learning and Reinforcement Learning Workshop (ICML 2018 ECA)*, 2018.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Yann Le Cun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*. Springer, 1986.

Yann Le Cun, Conrad C Galland, and Geoffrey E Hinton. GEMINI: gradient estimation through matrix inversion after noise injection. In *Advances in neural information processing systems*, pp. 141–148, 1989.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 1998.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2015.

Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Nikolay Manchev and Michael Spratling. Target propagation in recurrent neural networks. *Journal of Machine Learning Research*, 21(7):1–33, 2020.

Alexander Meulemans, Francesco Carzaniga, Johan Suykens, João Sacramento, and Benjamin F. Grewe. A theoretical framework for target propagation. In *Advances in Neural Information Processing Systems 33*, 2020.

Alexander Meulemans, Matilde Tristany Farinha, Javier García Ordóñez, Pau Vilimelis Aceituno, João Sacramento, and Benjamin F Grewe. Credit assignment in neural networks through deep feedback control. *arXiv preprint arXiv:2106.07887*, 2021.

Piotr Mirowski and Yann LeCun. Dynamic factor graphs for time series modeling. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009.

Yurii Nesterov. *Lectures on convex optimization*. Springer, 2nd edition, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *arXiv preprint arXiv:1211.5063*, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019.

Richard Rohwer. The “moving targets" training algorithm. In *Advances in neural information processing systems 3*, 1990.

Vincent Roulet, Siddhartha Srinivasa, Dmitriy Drusvyatskiy, and Zaid Harchaoui. Iterative linearized control: stable algorithms and complexity guarantees. In *International Conference on Machine Learning*, 2019.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*. MIT Press, Cambridge, MA, USA, 1986.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International conference on machine learning*, 2013.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable ADMM approach. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Paul Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, 1994.

Sam Wiseman, Sumit Chopra, Marc-Aurelio Ranzato, Arthur Szlam, Ruoyu Sun, Soumith Chintala, and Nicolas Vasilache. Training language models using target-propagation. *arXiv preprint arXiv:1702.04770*, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://github.com/zalandoresearch/fashion-mnist.