LocoProp

Richard Cepka
Comenius University in Bratislava

Abstract

In this project, we review LocoProp ideas and experimentally validate them.

1 Notation

We adopt (almost) same notation as in the origanal LocoProp paper [1]. We use (x, y) to denote the input instance and target label pair. For an M-layer neural network and at a given layer $m \in [M]$, we use $\hat{a}_m$ (respectively, $\hat{y}_m$) and a_m (respectively, $\hat{y}_m$) for the predicted and target pre (post)-activations, respectively. Note that $\hat{a}_m = W_m \hat{y}_{m-1}$ and $\hat{y}_m = \sigma_m(\hat{a}_m)$ where W_m is the weight matrix and σ_m is an elementwise activation function. We denote the loss of the network in the final layer as $Loss$.

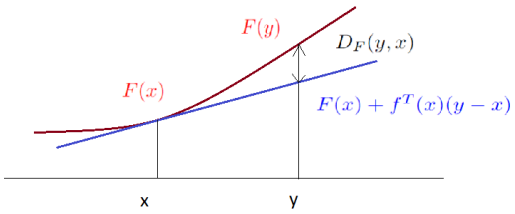

Figure 1: The Bregman divergence induced by a F is difference between the value of F at point x and the value of the first-order Taylor expansion of F around point y evaluated at point x.

2 LocoProp

Standard Neural network optimization methods update weights in direction of negative gradient $\nabla_W Loss$ via *Backpropagation*. Let's try to look at it from a different perspective. In first step, imagine that some oracle give us optimal activation of each layer a_m. That means, if layer m produce activation a_m than $Loss$ will be the smallest. In this perspective we can try to choose new weights $\overline{W}_m$ of layer m, that produces a_m. This leads to the following optimization problem:

$$\min_{\overline{W}_m} L(a_m, \overline{W}_m \hat{y}_{m-1}), \tag{1}$$

where L denotes some *matching loss*. Lets concretely define *matching loss* as:

$$L_f(a, \hat{a}) \triangleq \int_a^{\hat{a}} (f(z) - f(a))^T \, dz. \tag{2}$$

Then the *matching loss* is equal to *Bregman divergence* induced by a continuously-differentiable and convex function $F : \mathbb{R}^d \to \mathbb{R}$ such that $f = \nabla F$:

$$L_f(a, \hat{a}) = [F(z) - f(a)^T z]_a^{\hat{a}} = F(\hat{a}) - F(a) - f(a)^T \hat{a} + f(a)^T a \triangleq D_F(\hat{a}, a).$$

Bregman divergences are non-negative measure of difference between two points that satisfy many desirable properties, e.g., **(I) Convexity:** $D_F(\hat{a}, a)$ convex in first argument. **(II) Positivity:** When F is stricli convex, $D_F(\hat{a}, a) = 0$ iff $\hat{a} = a$.

From the property **(I)** we know that optimization problem is convex in $\overline{W}_m$, because is linear in $\hat{a}_m$. And from the property **(II)**, we know that when f is strictly increasing ($\nabla^2 F = \nabla f > 0$), $L_f(a, \hat{a}) = 0$ iff $\hat{a} = a$. That means that for *matching loss* defined as (2) has optimization problem (1) desirable properties. It is also common add regularization term $R(\overline{W}_m, W_m)$ to optimization problem (1), for convexity preservation of (1) we define regularization term as $R(\overline{W}_m, W_m) \triangleq \frac{\lambda_r}{2}||\overline{W}_m - W_m||_2^2$. Where $\lambda_r > 0$ controls the trade-off between minimizing the loss and the regularizer. Then we end up with:

$$W_m^{new} = \underset{\overline{W}_m}{\operatorname{argmin}} D_{F_m}(\hat{a}_m, a_m) + \frac{\lambda_r}{2}||\overline{W}_m - W_m||_2^2.$$

From convexity we know that setting the derivative of the objective to zero is necessary and sufficient condition for optimum:

$$W^{new} = W_m - \frac{1}{\lambda_r}[f_m(W_m^{new}\hat{y}_{m-1}) - f_m(a_m)]\hat{y}_{m-1}^T. \tag{3}$$

It is implicit equation witch can be numerically solved, e.g. by *fixed-point iteration* (FPI), *Anderson acceleration* [2] and many others. For some desired function f_m equation (3) has exact solution.

3 LocoProp and others

In previous section we review LocoProp general setting, by choosing function f_m and estimating oracle a_m we can derived other methods as special case of LocoProp.

3.1 BackProp

If we estimate activation of oracle as $f_m(a_m) \approx f_m(\hat{a}_m) - \lambda_a \nabla_{\hat{a}_m} Loss$ and do one step of FPI on (3) with starting point ${}_0W_m^{new} = W_m$, we get:

$$\begin{aligned}
W_m^{new} &\approx W_m - \frac{1}{\lambda_r}[f_m(W_m\hat{y}_{m-1}) - f_m(a_m)]\hat{y}_{m-1}^T \\
&= W_m - \frac{1}{\lambda_r}[f_m(\hat{a}_m) - f(\hat{a}_m) + \lambda_a \nabla_{\hat{a}_m} Loss]\hat{y}_{m-1}^T \\
&= W_m - \frac{\lambda_a}{\lambda_r}\nabla_{\hat{a}_m} Loss\, \hat{y}_{m-1}^T \\
&= W_m - \frac{\lambda_a}{\lambda_r}\frac{\partial Loss}{\partial \hat{a}_m}\frac{\partial \hat{a}_m}{\partial W_m}.
\end{aligned}$$

Which is *Backpropagation* with learning rate $\frac{\lambda_a}{\lambda_r}$. Natural choose of funtion f_m is activation function of layer m, $f_m \equiv \sigma_m$.[1]

3.2 FOOF

If we choose $f_m \equiv identity$ ($F(x) = \frac{1}{2}||x||_2^2$), than *Bregman divergences* is Euclidian distance, and hence:

$$W_m^{new} = \underset{\overline{W}_m}{\operatorname{argmin}} \frac{1}{2}||\overline{W}_m\hat{y}_{m-1} - a_m||_2^2 + \frac{\lambda_r}{2}||\overline{W}_m - W_m||_2^2.$$

[1] Activation function σ_m should satisfy property **(II)**.

With the same estimation of oracle as in previes subsection, $a_m \approx \hat{a}_m - \lambda_a \nabla_{\hat{a}_m} Loss$, we get formulation of *FOOF*:

$$
\begin{aligned}
W_m^{new} &= \underset{\overline{W}_m}{\operatorname{argmin}} \frac{1}{2} ||\overline{W}_m \hat{y}_{m-1} - \hat{a}_m + \lambda_a \nabla_{\hat{a}_m} Loss||_2^2 + \frac{\lambda_r}{2} ||\overline{W}_m - W_m||_2^2 \\
&= \underset{\overline{W}_m}{\operatorname{argmin}} \frac{1}{2} ||(\overline{W}_m - W_m) \hat{y}_{m-1} + \lambda_a \nabla_{\hat{a}_m} Loss||_2^2 + \frac{\lambda_r}{2} ||\overline{W}_m - W_m||_2^2 \\
&= W_m + \underset{\triangle W_m}{\operatorname{argmin}} \frac{1}{2} ||\triangle W_m \hat{y}_{m-1} + \lambda_a \nabla_{\hat{a}_m} Loss||_2^2 + \frac{\lambda_r}{2} ||\triangle W_m||_2^2.
\end{aligned} \tag{4}
$$

We get the same necessary and sufficient condition as (3), but now has explicit solution:

$$
W_m^{new} = W_m - \frac{\lambda_a}{\lambda_r} \nabla_{W_m} Loss \, (I + \frac{1}{\lambda_r} \hat{y}_{m-1} \hat{y}_{m-1}^T)^{-1}. \tag{5}
$$

Which is K-FAC with only right precondioner, with learning rate λ_a and damping factor λ_r. In the paper [3] they show that update rule (5) perform comparable with K-FAC.

4 Experiments

We experimentally validate previously reviewed ides on Fashion-MNIST with bath size 1028, and on a fully connected network (5 hidden layers, with 512 hidden dimension and with $\sigma_m \equiv \tanh$ nonlinearity). We report each method's best result after a hyper-parameter search.[2]

5 Results

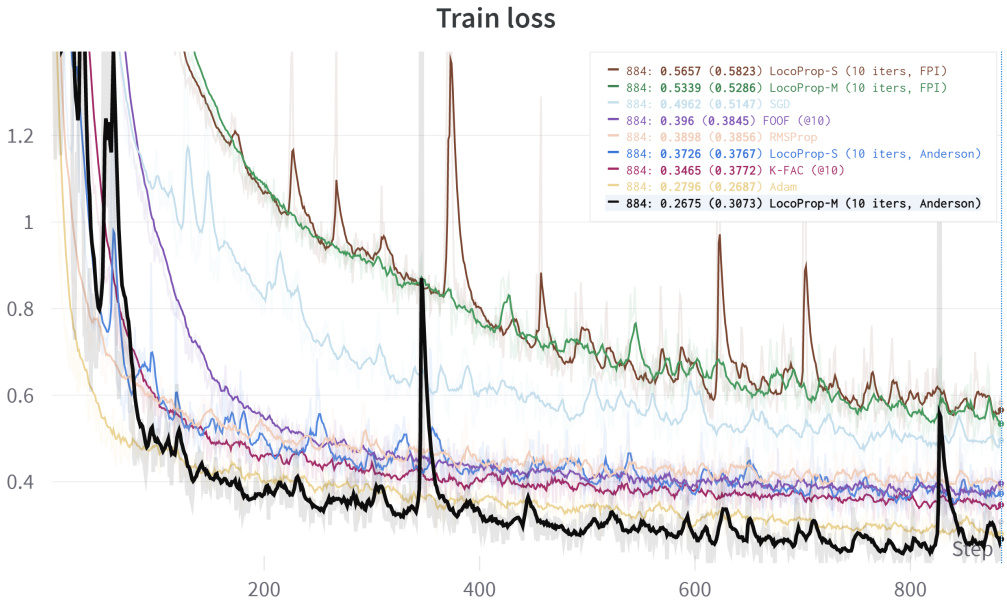

Figure 2: Results are smoothed by Exponential Moving Average with 0.7.

LocoProp-M ended up always better than LocoPopS, which means that taking into account also activation can help. Also, an interesting observation is that LocoProp-S with a better solver ended up the same as *FOOF*. That makes sense because LocoProp-S is the iterative approximation of *FOOF*.

[2]Due to computational constraints, we run only a few rounds of Bayesian hyper-parameter search.

6 Conclusion

LocoProp is an interesting optimization paradigm to train Neural networks. It can compete with SOTA optimizers like Adam and K-FAC. Future work involves scaling the method up to much larger and broader architectures (e.g. Transformer, ResNet). The next step can also be to investigate different oracle estimations (e.g. Adam update), and investigate stronger optimizers for *matching loss*.

References

[1] Ehsan Amid, Rohan Anil, and Manfred K. Warmuth. Locoprop: Enhancing backprop via local loss optimization. 2021.

[2] Donald G. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, oct 1965.

[3] Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization, 2022.

Appendix

MISSING DERIVATIONS

Proposition 1. *Proposition 1 Derivation of* $G \triangleq D_{F_m}(\hat{a}_m, a_m) + \frac{\lambda_r}{2}||\overline{W}_m - W_m||_2^2$*, respect to* $\overline{W}_m$ *is:*

$$\nabla_{\overline{W}_m} G = \lambda_r(\overline{W}_m - W_m) + [f_m(\overline{W}_m \hat{y}_{m-1}) - f_m(a_m)]\hat{y}_{m-1}^T.$$

Proof.

$$\begin{aligned}
\nabla_{\overline{W}_m} G &= \nabla_{\overline{W}_m} D_{F_m}(\hat{a}_m, a_m) + \nabla_{\overline{W}_m} \frac{\lambda_r}{2}||\overline{W}_m - W_m||_2^2 \\
&= \nabla F(\overline{W}_m \hat{y}_{m-1})\hat{y}_{m-1}^T - f(a_m)\hat{y}_{m-1}^T + \lambda_r(\overline{W}_m - W_m) \\
&= \lambda_r(\overline{W}_m - W_m) + [f(\overline{W}_m \hat{y}_{m-1}) - f(a_m)]\hat{y}_{m-1}^T.
\end{aligned}$$

□

Proposition 2. *Close form solution of:*

$$W_m^{new} = W_m - \frac{1}{\lambda_r}[W_m^{new}\hat{y}_{m-1} - (\hat{a}_m - \lambda_a \nabla_{\hat{a}_m} Loss)]\hat{y}_{m-1}^T,$$

is: $W_m^{new} = W_m - \lambda_a \nabla_{W_m} Loss\,(\lambda_r I + \hat{y}_{m-1}\hat{y}_{m-1}^T)^{-1}$.

Proof.

$$\begin{aligned}
W_m^{new} &= W_m - \frac{1}{\lambda_r}[W_m^{new}\hat{y}_{m-1} - (\hat{a}_m - \lambda_a \nabla_{\hat{a}_m} Loss)]\hat{y}_{m-1}^T \\
\Delta W_m &= -\frac{1}{\lambda_r}[\triangle W_m \hat{y}_{m-1} + \lambda_a \nabla_{\hat{a}_m} Loss]\hat{y}_{m-1}^T \\
\Delta W_m + \frac{1}{\lambda_r}\Delta W_m \hat{y}_{m-1}\hat{y}_{m-1}^T &= -\frac{\lambda_a}{\lambda_r}\nabla_{\hat{a}_m} Loss\,\hat{y}_{m-1}^T \\
\Delta W_m(I + \frac{1}{\lambda_r}\hat{y}_{m-1}\hat{y}_{m-1}^T) &= -\frac{\lambda_a}{\lambda_r}\nabla_{W_m} Loss \\
\Delta W_m &= -\frac{\lambda_a}{\lambda_r}\nabla_{W_m} Loss(I + \frac{1}{\lambda_r}\hat{y}_{m-1}\hat{y}_{m-1}^T)^{-1} \\
W_m^{new} &= W_m - \lambda_a \nabla_{W_m} Loss\,(\lambda_r I + \hat{y}_{m-1}\hat{y}_{m-1}^T)^{-1}
\end{aligned}$$

□

Remark.

$$\nabla_{\hat{a}_m} Loss\,\hat{y}_{m-1}^T = \nabla_{W_m} Loss,$$

$$\Delta W_m \triangleq W_m^{new} - W_m.$$

LocoProp ?

In the LocoProp paper, they propose updating rule as following:

Algorithm 1 LocoProp-S: W-step

${}_0W_m^{new} \leftarrow W_m$
for $k = 0, \ldots, T-1$ **do**
  ${}_{k+1}W_m^{new} \leftarrow_k W_m^{new} - \eta({}_kW_m^{new}\hat{y}_{m-1} - a_m)\hat{y}_{m-1}^T$

but if we do gradient descent (same as FPI, $\eta = \frac{1}{\lambda_r}$) on (4), the update rule will be diferent.

Algorithm 2 LocoProp-S (Fixed): W-step

${}_0W_m^{new} \leftarrow W_m$
for $k = 0, \ldots, T-1$ **do**
  ${}_{k+1}W_m^{new} \leftarrow_k W_m^{new} - \eta\lambda_r({}_kW_m^{new} - W_m) - \eta({}_kW_m^{new}\hat{y}_{m-1} - a_m)\hat{y}_{m-1}^T$

The first step will indeed be the same, due to initialisation, but the LocoProp-S following steps entirely omit the regularization term.