Sparsifying Networks via Subdifferential Inclusion

S. Verma[1] J.-C. Pesquet[1]

[1]Université Paris-Saclay, CentraleSupélec, Inria, Centre de Vision Numérique

International Conference on Machine Learning(ICML) 2021

Table of Contents

Introduction

Generating sparse weights for a pretrained network.

- Properties of activation functions.
- Approximate subdifferential inclusion.
- Iterative optimization.
- Mini-batch approach, partial training data.
- Vision, NLP, speech, and time-series.

Table of Contents

Related Work

Inducing Sparsity Post Training

- After training.
- Several pruning and fine-tuning cycles [Park et al., 2020].
- Net-Trim algorithm: convex programming [Aghasi et al., 2017].
- Lower rank paramtere tensors [Lu et al., 2016].
- Removing channels and filter, group sparsity [Yu et al., 2019].

Related Work

Inducing Sparsity During Training

- Loss function modified [Ullrich et al., 2017, Neklyudov et al., 2017].
- Extra parameters per neuron
  - Bayesian compression [Louizos et al., 2017].
  - L_0, L_1 regularization [Louizos et al., 2018].
  - $2\times$ memory and $4\times$ compute.

Related Work

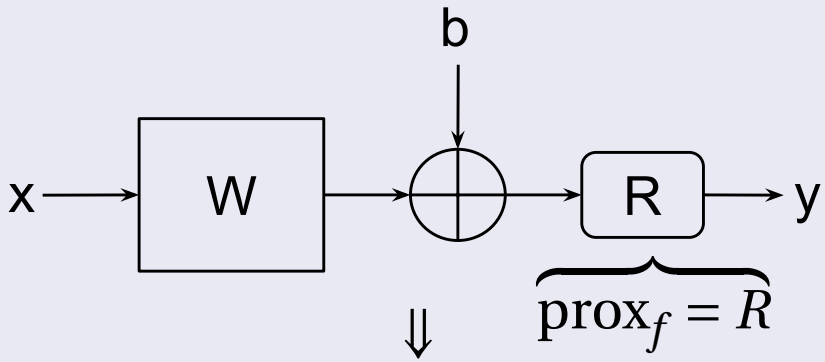
Training Sparsely Initialized Networks

- SNIP [Lee et al., 2019].
- GraSP [Wang et al., 2020].
- SynFlow [Tanaka et al., 2020].
- FORCE [Jorge et al., 2020].

Table of Contents

Proposed Method

Variational Principles



$$\Downarrow$$

$$Wx + b - y \in \partial f(y) \tag{1}$$

where $\partial f(y)$ is the Moreau subdifferential of f at y defined as

$$\partial f(y) = \left\{ t \in \mathbb{R}^N \mid (\forall z \in \mathbb{R}^N) f(z) \geqslant f(y) + \left\langle t \mid z - y \right\rangle \right\}. \tag{2}$$

The subdifferential inclusion in equation (1) is equivalent to

$$d_{\partial f(y)}(Wx + b - y) = 0. \tag{3}$$

Proposed Method

Optimization Problem

$$\underset{(W,b)\in C}{\text{minimize}}\ g(W,b) \tag{4}$$

with

$$C = \Big\{(W,b) \in \mathbb{R}^{N\times M} \times \mathbb{R}^{N} \mid (\forall j \in \{1,\dots,J\}) \sum_{t=1}^{T} d^2_{\partial f(y_{j,t})}(Wx_{j,t} + b - y_{j,t}) \leqslant T\eta\Big\}, \tag{5}$$

where g is a sparsity measure defined on $\mathbb{R}^{N\times M} \times \mathbb{R}^{N}$ and $\eta \in [0, +\infty[$ is some accuracy tolerance.
j is a mini-batch out of J batches.
t is a sample out of T samples in a mini-batch.

Algorithm 1: Douglas-Rachford algorithm for network compression

Initialize: $\widehat{W}_0 \in \mathbb{R}^{N\times M}$, $b_0 \in \mathbb{R}^N$, $\gamma \in \,]0,+\infty[$ and $(\lambda_n)_{n\in\mathbb{N}} \in]0,2[$

for $n = 0, 1, \ldots$ **do**

$W_n = \mathrm{prox}_{\gamma h}(\widehat{W}_n)$ ▷ Sparsification step

$(\widetilde{W}_n, \widetilde{b}_n) = \mathrm{proj}_C(2W_n - \widehat{W}_n, b_n)$ ▷ Maintaining desired accuracy

$\widehat{W}_{n+1} = \widehat{W}_n + \lambda_n(\widetilde{W}_n - W_n)$ ▷ Update rule for weight

$b_{n+1} = b_n + \lambda_n(\widetilde{b}_n - b_n)$. ▷ Update rule for bias

Computing proj_C is intractable.

Proposed Method

Subgradient Projection

For every mini-batch index $j \in \{1, \ldots, J\}$, we define the following convex function:

$$(\forall (W, b) \in \mathbb{R}^{N \times M} \times \mathbb{R}^{N})$$

$$c_j(W, b) = \sum_{t=1}^{T} d^2_{\partial f(y_{j,t})}(Wx_{j,t} + b - y_{j,t}) - T\eta. \quad (6)$$

Proposed Method

Subgradient Projection

$$\nabla_{\mathsf{W}} c_j(W,b) = 2\sum_{t=1}^{T} e_{j,t} x_{j,t}^{\top}, \quad \nabla_{\mathsf{b}} c_j(W,b) = 2\sum_{t=1}^{T} e_{j,t} \tag{7}$$

with, for every $t \in \{1,\dots,T\}$,

$$e_{j,t} = Wx_{j,t} + b - y_{j,t} - \mathrm{proj}_{\partial f(y_{j,t})}(Wx_{j,t} + b - y_{j,t}). \tag{8}$$

The subgradient projection algorithm described in the paper uses the above expressions to compute proj_C.

Proposed Method

Proximal Operator of Activation Functions

Name	$\rho(\zeta)$	$\mathrm{proj}_{\partial\varphi(v)}(\zeta)$
Sigmoid	$(1+e^{-\zeta})^{-1}-\frac{1}{2}$	$\ln(v+1/2)-\ln(v-1/2)-v$
ReLU	$\max\{\zeta,0\}$	$\begin{cases} 0 & \text{if } v>0 \text{ or } \zeta \geqslant 0 \\ \zeta & \text{otherwise} \end{cases}$
ELU	$\begin{cases} \zeta & \text{if } \zeta \geqslant 0 \\ \alpha\big(\exp(\zeta)-1\big) & \text{otherwise} \end{cases}$	$\begin{cases} 0 & \text{if } v>0 \\ \ln\left(\frac{v+\alpha}{\alpha}\right)-v & \text{otherwise} \end{cases}$
QuadReLU	$\dfrac{(\zeta+\alpha)\mathrm{ReLU}_{2\alpha}(\zeta+\alpha)}{4\alpha}$	$\begin{cases} v & \text{if } v=0 \text{ and } \zeta \leqslant -\alpha \\ -v+2\sqrt{\alpha v}-\alpha & \text{if } v\in]0,\alpha] \\ & \text{or } (v=0 \text{ and } \zeta>-\alpha) \\ v-\alpha & \text{otherwise} \end{cases}$

More activation functions in the paper!

Table of Contents

Experiments

ResNet50 on ImageNet

Sparsity	80%			96.5%		
	Train/Prune FLOPs (×e16)	Top-1 Acc(%)	Infer FLOPs	Train/Prune FLOPs (×e16)	Top-1 Acc(%)	Infer FLOPs
SNIP	0.696	72.34	941M	0.502	59.16	292M
GraSP	0.650	74.21	786M	0.501	69.55	290M
SynFlow	0.665	74.14	776M	0.500	70.10	288M
FORCE	0.619	73.42	685M	0.497	72.04	276M
SparseVD	1.737	74.68	811M	1.685	67.13	286M
BC-GHS.	1.737	74.15	813M	1.684	68.54	282M
$L_{0_{hc}}, \lambda = e-5$	1.736	76.67	802M	1.684	68.61	276M
STR	0.625	76.11	704M	0.449	71.87	117M
NetTrim	0.866	72.88	842M	0.283	62.01	281M
SIS (Ours)	**0.435**	**76.96**	**647M**	**0.102**	**73.11**	**101M**

Experiments

More Experiments

We also perform experiments on

- VGG19 and ResNet50 on CIFAR-10 and CIFAR-100
- MobileNet-V1/V2/V3 on ImageNet
- Jasper on LibriSpeech
- Transformer-XL on WikiText-103
- N-BEATS on M4

Experiments

η (Sparsity vs Accuracy)

How to choose η to control sparsity and accuracy?

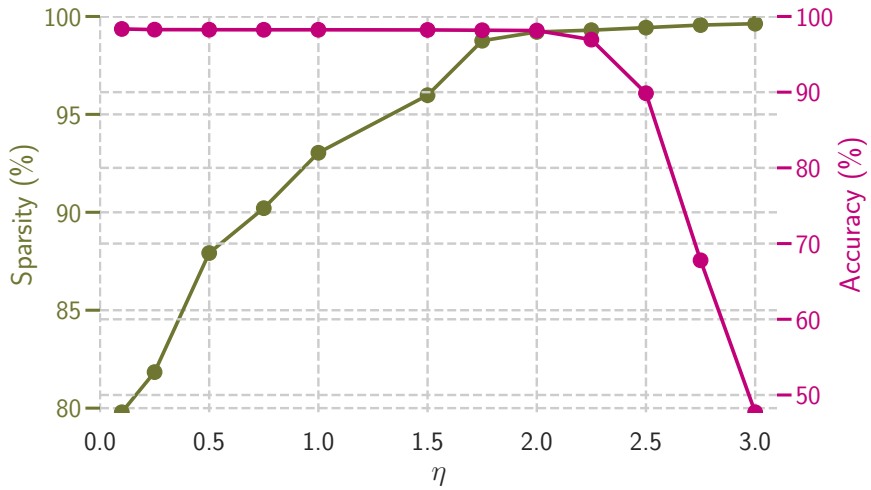


Figure: Effect of η on LeNet-FCN

Table of Contents

Conclusion

- Reliable in terms of iteration convergence guarantees,
- Applicable to a wide range of activation operators,
- Able to deal with large datasets split into mini-batches.

Thank You!

Code and weights available at
https://sagarverma.github.io/compression.html