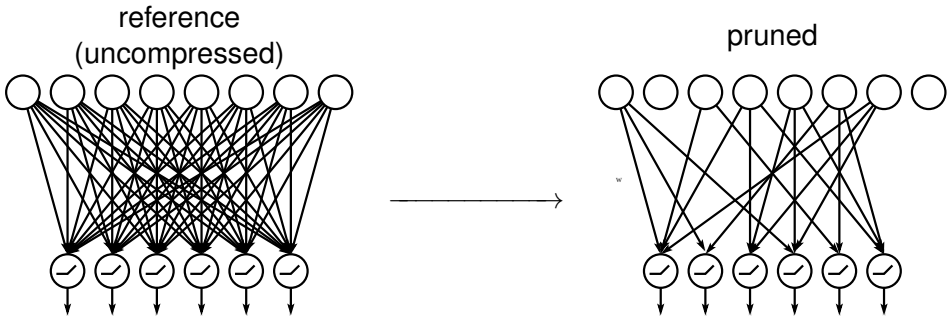
Exploring the effect of ℓ_0/ℓ_2 regularization in neural network pruning using the LC Toolkit

Yerlan Idelbayev and **Miguel Á. Carreira-Perpiñán**
Dept. CSE, University of California, Merced
http://eecs.ucmerced.edu

The code is available at:
https://github.com/UCMerced-ML/LC-model-compression

Introduction: Neural network pruning



Given a network with weights $\mathbf{w}$ and task loss L, most of the NN pruning problems are formulated using a sparsifying penalty $P(\mathbf{w})$ and solve a regularized or constrained formulation given as

$$\min_{\mathbf{w}} L(\mathbf{w}) + \lambda P(\mathbf{w}) \quad \text{or} \quad \min_{\mathbf{w}} L(\mathbf{w}) \text{ s.t. } P(\mathbf{w}) \leq \lambda \tag{1}$$

Introduction: Neural network purning (cont.)

Among those formulations, we are particularly interested in ℓ_0-based pruning of

$$\min_{\mathbf{w}} \quad L(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_0 \leq \kappa, \tag{2}$$

where we can precisely set the number of non-zero (unpruned) weights via parameter κ: the total budget of the remaining parameters.

In principle, for a given level of sparsity the formulation (1) with $P(\mathbf{w}) = \|\mathbf{w}\|_1$ should yield more or less the same networks as the ℓ_0 formulation (2). However, we empirically observe that this is not the case.

Introduction: Some ℓ_0 and ℓ_1 pruning results on LeNet300

Method	Test err	Remaining weights	Remaining neurons
reference	2.01%	100%	784-300-100-10
ℓ_0	2.33%	2.0%	344-210-100-10
ℓ_1	2.44%	2.4%	476-37-84-10

A simple modification to the ℓ_0 pruning problem

We propose modifying the problem (2) by adding a small amount of ℓ_2 weight decay as follows:

$$\min_{\mathbf{w}} \quad L(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{w}\|_0 \leq \kappa, \tag{3}$$

with the motivation that ℓ_2 penalty will nudge the small valued weights towards zero and allow more neurons to be pruned.

We study our modified formulation using the framework of the LC algorithm and demonstrate that: 1) the pruning formulation (3) has a similar effect on the structure of the model weights as the ℓ_1 penalized formulation and 2) the resulting networks achieve better error-pruning tradeoff when compared to pure ℓ_0- and ℓ_1-pruned networks

Optimization with the LC algorithm

To obtain an LC algorithm, we introduce a duplicate variable $\boldsymbol{\theta}$ with an equality constraint of $\mathbf{w} = \boldsymbol{\theta}$ and then apply alternating optimization. Depending on how we introduce the duplicates, we end up with two different versions of the optimization, however, in these slides we only show a single version.

When introducing the variable $\boldsymbol{\theta}$, let us leave $L(\mathbf{w})$ as is, and replace all other $\mathbf{w}$ occurrences with $\boldsymbol{\theta}$ and jointly optimize the following:

$$\min_{\mathbf{w},\boldsymbol{\theta}} \quad L(\mathbf{w}) + \lambda \left\|\boldsymbol{\theta}\right\|_2^2 \quad \text{s.t.} \quad \left\|\boldsymbol{\theta}\right\|_0 \leq \kappa, \quad \mathbf{w} = \boldsymbol{\theta} \tag{4}$$

Optimization with the LC algorithm

Now, to derive an efficient algorithm we apply the Quadratic Penalty to the equality constraints and optimize while driving $\mu \to \infty$:

$$\min_{\mathbf{w},\boldsymbol{\theta}} \quad L(\mathbf{w}) + \lambda \|\boldsymbol{\theta}\|_2^2 + \frac{\mu}{2}\|\mathbf{w} - \boldsymbol{\theta}\|^2 \quad \text{s.t.} \quad \|\boldsymbol{\theta}\|_0 \leq \kappa$$

This reformulation is advantageous as it allows to apply alternating optimization over $\mathbf{w}$ and $\boldsymbol{\theta}$ where steps are given in the forms that can be efficiently solved:

- **L step** of $\min_{\mathbf{w}} L(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{w} - \boldsymbol{\theta}\|^2$.
  This step has the form of the neural network training, but with an additional ℓ_2 penalty. Such training can be handled by any deep learning framework, and does not require any special treatment. Following the LC paper [1] we call this step a learning (L) step.
- **C step** of $\min_{\boldsymbol{\theta}} \frac{\mu}{2}\|\mathbf{w} - \boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|_2^2$ s.t. $\|\boldsymbol{\theta}\|_0 \leq \kappa$
  This step has the form of finding the optimal constrained $\boldsymbol{\theta}$ (with at most κ non zeros) minimizing a convex objective function, and can be solved exactly. This step was called a compression (C) step in [1].

Solution of the C step

To solve the C-step problem, we rewrite the ℓ_0-norm using the index set Ω of size κ: we say $\theta_i = 0$ if and only if $i \notin \Omega$, and collectively refer to non-zero weights as $\boldsymbol{\theta}_\Omega$. Then, the C-step problem for $\mu > 0$ can be written as

$$\min_{\boldsymbol{\theta},\Omega} \quad \frac{\mu}{2} \sum_{i \in \Omega} \left((w_i - \theta_i)^2 + \frac{2\lambda}{\mu} \theta_i^2 \right) + \frac{\mu}{2} \sum_{i \notin \Omega} w_i^2 \tag{5}$$

We note that for a fixed index set Ω, the minimization of (5) wrt $\boldsymbol{\theta}$ is a convex problem with the solution of

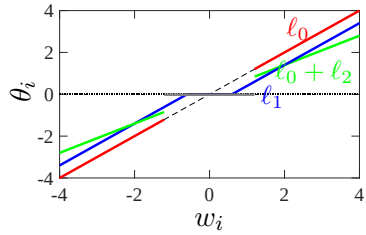
$$\boldsymbol{\theta}^*_\Omega = \frac{\mu}{\mu + 2\lambda} \mathbf{w}_\Omega, \tag{6}$$

Furthermore, in the paper, we show that the optimal Ω^* can be found by:

$$\Omega^* = \{ i : w_i \in \text{Top-}\kappa \text{ largest in magnitude items of } \mathbf{w} \}. \tag{7}$$

C step: pruning properties

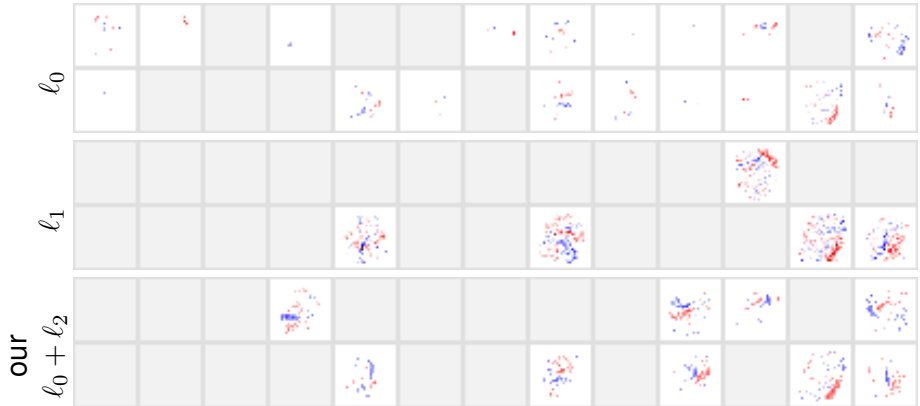
formulation	solution for θ_i
ℓ_0 constrained [2]	$\theta_i = w_i \cdot I(\lvert w_i \rvert \geq \eta_\kappa)$
ℓ_1 constrained [2]	$\theta_i = \left(w_i - \operatorname{sgn}(w_i)\,\frac{\lambda}{\mu}\right) \cdot I(\lvert w_i \rvert \geq \frac{\lambda}{\mu})$
ℓ_0 constrained. + ℓ_2 penalty	$\theta_i = \frac{\mu}{\mu+2\lambda} w_i \cdot I(\lvert w_i \rvert \geq \eta_\kappa)$



C-step solutions of different pruning operators (left) and its graphical representation (right). Here, κ and λ are hyperparameters of pruning operators, μ is the penalty parameter, I is the indicator function, and η_k is κ^{th} largest value in magnitude among all weights. *The pruning operator consisting of only ℓ_0-norm does not impose any shrinkage on the weights, however combining the ℓ_0 with ℓ_1 or ℓ_2^2 penalty will make surviving weights smaller.*

Revisiting LeNet300 experiments

Method	Test err	Remaining weights	Remaining neurons
reference	2.01%	100%	784-300-100-10
ℓ_0	2.33%	2.0%	344-210-100-10
ℓ_1	2.44%	2.4%	476-37-84-10
$\ell_0 + \ell_2$ (ours)	**1.76%**	2.0%	420-70-82-10



Additional results on LeNet300-ReLU

	remaining w.	test-err
reference network	100%	1.87%
variational dropout [3]	1.47%	1.92%
DNS [4]	1.78%	1.99%
DeepHoyer [5]	2.27%	1.60%
our $\ell_0 + \ell_2$, $\lambda = 10^{-4}$	2.00%	**1.60%**
our $\ell_0 + \ell_2$, $\lambda = 10^{-4}$	1.50%	1.99%

Our modified scheme $\ell_0 + \ell_1$ improves over regular ℓ_0 pruning. Right: our pruned LeNet300-ReLU models in comparison to selected results from the literature.

Conclusion

We have studied the behavior of ℓ_0-constrained pruning of the neural networks, which is a non-differentiable and non-convex problem. We observed that by itself, ℓ_0-pruned network results in an architecture with many alive neurons having only a few non-zero weights. To remedy such a behavior we proposed a modification of the ℓ_0 pruning involving a small amount of ℓ_2 penalty and studied its properties using the common algorithmic framework of the LC toolkit.

References

[1] Miguel Á. Carreira-Perpiñán, "Model compression as constrained optimization, with application to neural nets. Part I: General framework," arXiv:1707.01209, July 5 2017.

[2] Miguel Á. Carreira-Perpiñán and Yerlan Idelbayev, ""Learning-compression" algorithms for neural net pruning," in *Proc. of the 2018 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'18)*, Salt Lake City, UT, June 18–22 2018, pp. 8532–8541.

[3] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov, "Variational dropout sparsifies deep neural networks," in *Proc. of the 34th Int. Conf. Machine Learning (ICML 2017)*, Doina Precup and Yee Whye Teh, Eds., Sydney, Australia, Aug. 6–11 2017, pp. 2498–2507.

[4] Yiwen Guo, Anbang Yao, and Yurong Chen, "Dynamic network surgery for efficient DNNs," in *Advances in Neural Information Processing Systems (NIPS)*, D. D. Lee, M. Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and R. Garnett, Eds. 2016, vol. 29, pp. 1379–1387, MIT Press, Cambridge, MA.

[5] Huanrui Yang, Wei Wen, and Hai Li, "DeepHoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures," in *Proc. of the 8th Int. Conf. Learning Representations (ICLR 2020)*, Addis Ababa, Ethiopia, Apr. 26–30 2020.