Structured Bayesian Pruning via Log-Normal Multiplicative Noise

Kirill Neklyudov k.necludov@gmail.com
Dmitry Molchanov dmolchanov@hse.ru
Arsenii Ashukha aashukha@hse.ru
Dmitry Vetrov vetrovd@yandex.ru

p(B|A)yesgroup.ru Yandex Skoltech Skolkovo Institute of Science and Technology

Key Results

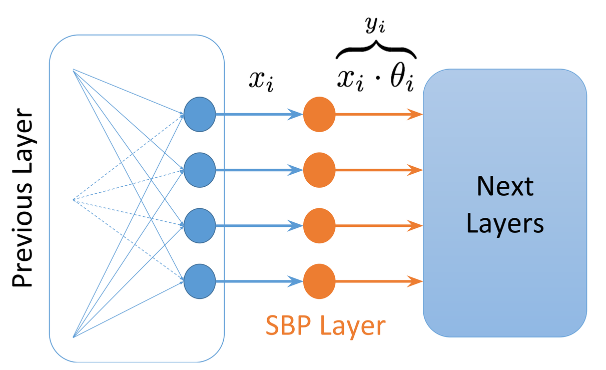
Structured Bayesian Pruning is a new model that provides structured sparsity, e.g. removes neurons and convolutional filters.

Our contributions can be summarized as follows:

- a method of regularization of DNNs that results in structured sparsity
- a proper analog of sparsity-inducing log-uniform prior
- experiments that show that SBP regularizes well and leads to a high level of group sparsity (it removes up to 80% of all units on a VGG-like architecture) and acceleration (up to $4.5\times$ measured speed-up) with small accuracy drop

The method is implemented as a separate dropout-like layer and an additional regularization term. TensorFlow implementation of our method is available.

Stochastic Variational Inference



- Approximation of posterior distribution of θ is $\theta \sim q(\theta \mid \varphi)$
- We put a sparsity-inducing prior over θ_i
- Parameters φ are trained using Stochastic VI

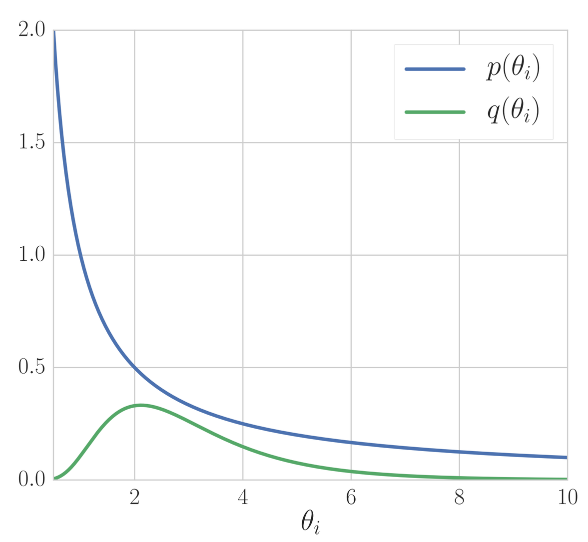
Approximate posterior distribution over θ by Stochastic Variational Inference:

$$L = \underbrace{-\mathbb{E}_{q(\theta\mid\varphi)} \log p(Y \mid X, \theta)}_{\text{Data-term}} + \underbrace{D_{KL}(q(\theta \mid \varphi) \,\|\, p_{prior}(\theta))}_{\text{Regularizer}} \to \min_{\varphi}$$

- The true posterior distribution over θ is approximated by q
$$D_{KL}(q(\theta \mid \varphi) \,\|\, p(\theta \mid X, Y)) \to \min_{\varphi}$$
- Just a slightly different loss function; implementation is basically the same

Structured Bayesian Pruning with Improper Log-Uniform Prior

- The model injects multiplicative noise θ into the output x of the previous layer
$$y_i = x_i \cdot \theta_i \qquad \theta_i \sim p_{noise}(\theta_i)$$
- Log-uniform prior for sparsity:
$$p(\theta_i) = \mathrm{LogU}_{\infty}(\theta_i) \propto \frac{1}{\theta_i} \qquad \theta_i > 0$$
- The approximated posterior is log-normal:
$$\log \theta_i \sim \mathcal{N}(\log \theta_i \mid \varphi_i), \quad \varphi_i = \{\mu_i, \sigma_i^2\}$$



\+ The variational family has no "prior gap"
\+ Log-normal noise does not change the sign of x
\+ The KL-divergence term can be computed analytically
− Due to the improper prior we obtain an ill-posed optimization problem

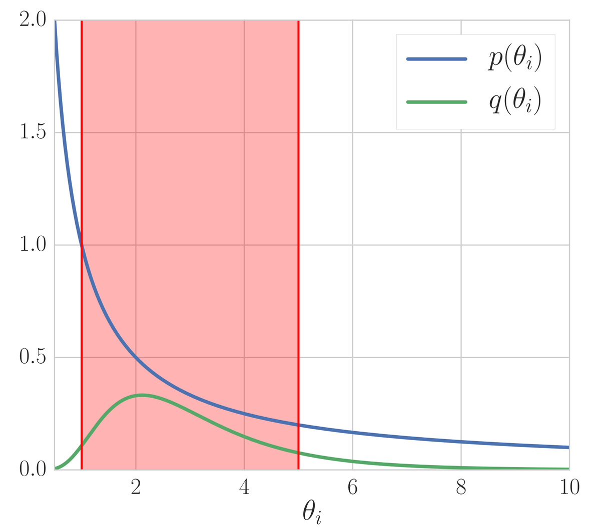
$$\mathrm{KL}\left(\mathrm{LogN}(\theta \mid \mu, \sigma^2) \,\|\, \mathrm{LogU}_{\infty}(\theta)\right) = C - \log\sigma, \;\; C = +\infty$$

Structured Bayesian Pruning with Proper Log-Uniform Prior

In order to obtain a proper probabilistic model, we truncate the prior and the posterior:

- $p(\theta_i) = \mathrm{LogU}_{\infty}(\theta_i) \Rightarrow \mathrm{LogU}_{[a,b]}(\theta_i)$
- $q(\theta_i) = \mathrm{LogN}(\theta_i \mid \varphi_i) \Rightarrow \mathrm{LogN}_{[a,b]}(\theta_i \mid \varphi_i)$

$$\boxed{\mathrm{LogP}_{[a,b]}(\theta_i) \propto \mathrm{LogP}_{\infty}(\theta_i) \cdot \mathrm{I}_{[a,b]}(\log \theta_i)}$$



All necessary statistics can be computed in closed form:

- KL-divergence for training
- Expectation $\mathbb{E}\theta$ for inference during testing
- Signal-to-noise ratio $SNR(\theta) = \mathbb{E}\theta / \sqrt{\mathbb{D}\theta}$ for pruning redundant neurons

Final Algorithm

Our final loss function is negative variational lower bound

$$L = -\mathbb{E}_{q(\theta\mid\mu,\sigma)} \log p(Y \mid X, \theta, W) + \alpha \cdot \mathrm{KL}(q(\theta \mid \mu, \sigma) \,\|\, p(\theta)) \to \min_{\mu,\sigma,W}$$

where W denotes all weights of DNN, q and p are truncated distributions.

Training procedure details:

- All models were pretrained with L2 regularization on parameters W
- Re-weight the KL term by α, proportional to the computational complexity of each specific layer (SPBa procedure).
- Remove neurons with low $SNR(\theta)$ after training; no fine-tuning needed!
- Tricks for numerically stable calculations are presented in the appendix

Experiments: LeNets on MNIST

- In MNIST experiments we compare different structured sparsity-inducing techniques on LeNet-5-Caffe and LeNet-500-300 architectures.
- Our method provides the highest speed-up with the same accuracy.

Network	Method	Error %	Neurons per Layer	CPU	GPU	FLOPs
LeNet 500-300	Original	1.54	$784-500-300-10$	$1.00\times$	$1.00\times$	$1.00\times$
	SparseVD[1]	1.57	$537-217-130-10$	$1.19\times$	$1.03\times$	$3.73\times$
	SSL[2]	1.49	$434-174-78-10$	$2.21\times$	$1.04\times$	$6.06\times$
	StructuredBP	1.55	$245-160-55-10$	$\mathbf{2.33}\times$	$\mathbf{1.08}\times$	$\mathbf{11.23}\times$
LeNet-5	Original	0.80	$20-50-800-500$	$1.00\times$	$1.00\times$	$1.00\times$
	SparseVD[1]	0.75	$17-32-329-75$	$1.48\times$	$1.41\times$	$2.19\times$
	SSL[2]	1.00	$3-12-800-500$	$5.17\times$	$1.80\times$	$3.90\times$
	StructuredBP	0.86	$3-18-284-283$	$\mathbf{5.41}\times$	$\mathbf{1.91}\times$	$\mathbf{10.49}\times$

Table: SSL is based on group lasso regularization, SparseVD induces weight-wise sparsity and can coincidentally remove all weights in filers or neurons, StructuredBP is our model. We report acceleration that was measured on CPU (Intel Xeon E5-2630), GPU (Tesla K40) and in terms of Floating Point Operations (FLOPs).

Experiments: VGG-like on CIFAR-10

- CIFAR-10 experiments were done on a VGG-like architecture[3]. The network consists of 12 convolutional and 2 fully connected layers with Batch Normalization and Binary Dropout
- With small accuracy drop our models provide significant acceleration and high structured sparsity. Presented speed-up was measured on CPU.

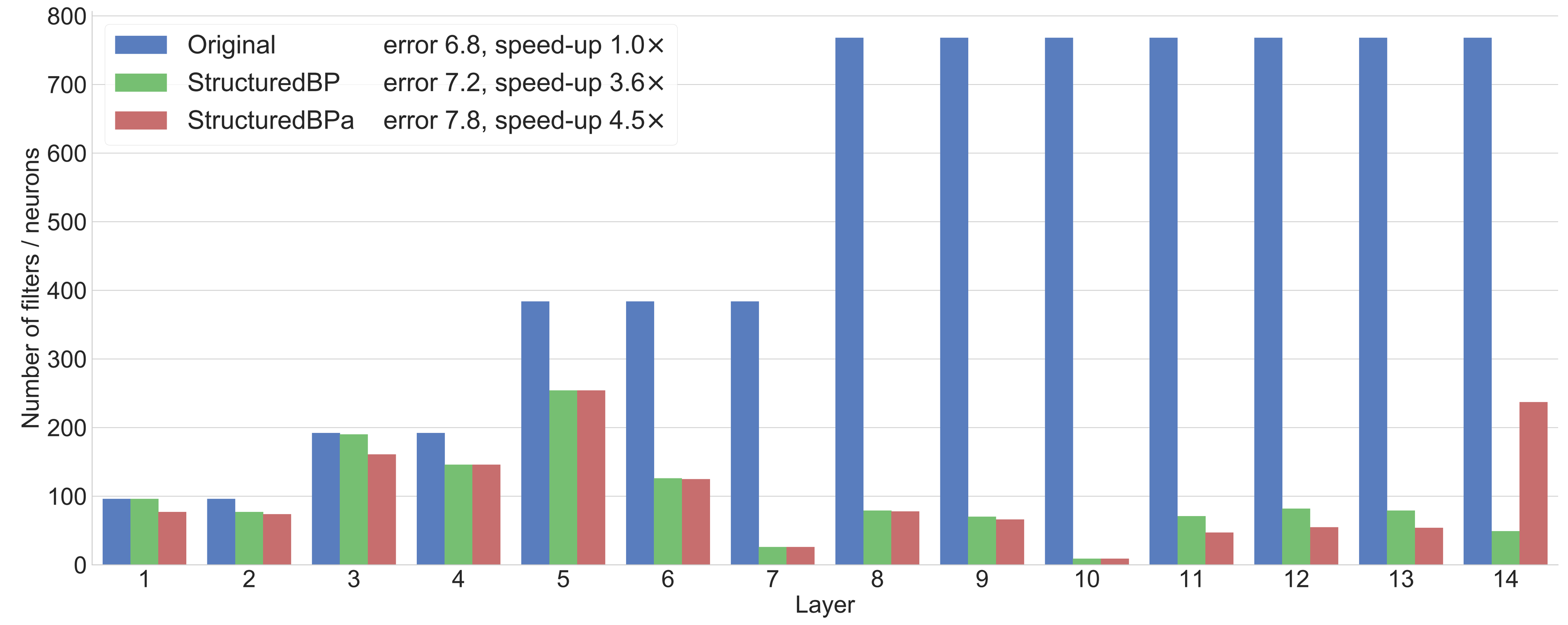


Figure: Original is a dense network, StructuredBP is our model, StructuredBPa is our model with re-weighted KL divergence for the first 6 layers.

Experiments: Random Labels

Dataset	Architecture	Train Acc.	Test Acc.	Sparsity
MNIST	FC + BD	100%	10%	—
MNIST	FC + StructuredBP	10%	10%	100%
CIFAR-10	VGG + BD	100%	10%	—
CIFAR-10	VGG + StructuredBP	10%	10%	100%

Unlike Binary Dropout (BD), Structured BP does not overfit on randomly labeled data and yields an empty network. It is an optimal architecture for this task!

Discussion

- Bayesian Learning framework is well known for providing non-structured sparse solutions. Usually sparsity is caused by Empirical Bayes which adjusts the prior distribution to the data. It can potentially lead to additional overfitting.
- In this work we utilize the Bayesian framework to obtain structured sparsity. We did not adjust the prior distribution, so the risk of overfitting is decreased.

Links and References

[1] Molchanov, D., Ashukha, A. and Vetrov, D. Variational Dropout Sparsifies Deep Neural Networks, ICML 2016
[2] Wen, W., Wu, C., Wang, Y., Chen, Y. and Li, H. Learning structured sparsity in deep neural networks, NIPS 2016
[3] Sergey Zagoruyko. 92.45 on cifar-10 in torch, 2015.

Page: goo.gl/a5CpXk