Bayes by Backprop weight pruning

Pavol Vidlička

pavol.vidlicka@gmail.com

May 26, 2018

Abstract

The report presents the report on the tensorflow implementation of a variational bayesian neural network and use of the posterior distribution of weights for weight pruning. We compare pruning based on a signal-to-noise ratio to random pruning and pruning based on absolute value.

I. Introduction

We explore variational bayes model for neural networks proposed in [Blundell et al., 2015] and its training algoritm called 'Bayes by Backprop'. Our implementation closely follows the paper.

Just to recall, a weight from the network w is sampled by $w = \mu + \epsilon \log(1 + \exp(\rho))$, where μ, ρ are trainable parameters and ϵ is a random variable with standard normal distribution.

The loss for the network is the KL divergence of the variational posterior and the true Bayesian posterior of the weight given the data $\mathcal{D}$.

II. Minibatches and loss

Optimization of the KL divergence uses minibatches. We sample the network weight once per minibatch. We use the following loss:

$$\frac{1}{N} KL[q(w|\theta)||P(w)] - \frac{1}{n} \sum_{(x_i,y_i)\in B} P((x_i, y_1)|w))$$

where N is the number of training examples, B is the current batch and $n = |B|$ batch size. The first part of the loss is called the complexity cost and it serves as regularization. The second term is called the likelihood cost. Our loss is closely related to the minibatch loss (8) in [Blundell et al., 2015]. The only difference is that we divide (8) by batch size to obtain a "per example" loss than can be compared across different batch sizes.

This form of the loss also highlight the role of the complexity cost and how its importance changes based on the size of the training set.

The loss depends on on the prior distribution of weights $P(w)$. We use a gaussian mixture prior as suggested in the original paper. The parameters of gaussian mixture are hyperparameters that determine the strength of the regularization.

Notice that the regularization term is implicitly inversely proportional to the train set size.

III. Experiment

i. Implementaton

We implement a bayesian Dense layer in tensorflow. The implementation is based on the source code for the Dense layer (in TensorFlow 1.7.0). It uses the Layer base class to deal with initialization of variables, etc.

Percentage pruned	Random	By absolute value	By signal to noise ratio
25%	23.20%	2.15%	2.15%
50%	57.85%	2.07%	2.13%
75%	60.89%	3.09%	2.13%
90%	81.30%	9.14%	2.21%
95%	84.85%	34.09%	2.21%
98%	95.36%	45.89%	2.28%
99%	88.91%	75.43%	2.35%

The complexity cost is implemented on the layer level similary to the way a L2 regularization would be implemented.

The doubling of the number of parameters when compared to a vanilla dense layer with L2 regularization roughly doubles the pre epoch training time.

ii. Results

We have trained a variational bayesian network on the MNIST dataset with two hidden layers of 1200 units to replicate the weight pruning experiment.

The pixels were preprocessed by dividing the values by 126 (the same treatment as in the original paper).

We used a gaussian mixture prior with $\pi = 0.25, -\ln\sigma_1 = 0, -\ln\sigma_2 = 2$ and the optimization was performed using the Adam optimizer for 100 epochs. We have achieved a 2.16% validation error. This is relatively high and together with the results reported in the [Blundell et al., 2015] it suggests a poor choice of hyperparamters.

Due to slow convergence of Bayes by Backprop and limited computational resources, we did not pursue better hyperparameters, instead we examined the use if posterior variational distribution of weights for weight pruning.

The paper uses the signal to noise ratio $\frac{|\mu|}{\sigma} = \frac{|\mu|}{\log(1+\exp(\rho))}$ to asses weight importance and prunes the network by setting the weights with the smallest signal to noise ratio to zero. We compare this method with randomly setting a percentage of weights to zero and pruning using $|\mu|$. The resulting validation errors after pruning are provided in the table 1.

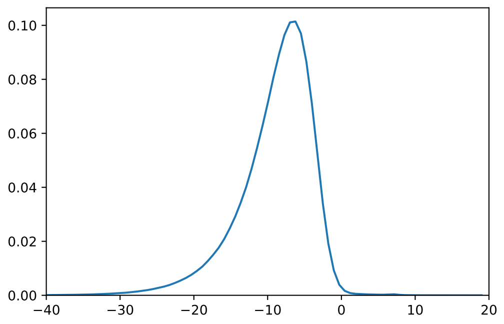

Figure 1: *Density of* $10\log_{10}\frac{|\mu|}{\sigma}$

Our results support the claim that the signal-to-noise metric is related to the importance of weights. Pruning 99% of the weights increases the validation error only by approximately 8%.

Interestingly, when plotting the distribution of the signal-to-noise ratio on a logarithmic scale, we get do not get a multiodal distribution. Nevertheless, the increase of test error due to pruning is similar.

The different distribution of the weights (notice that even the order of magnitude is different) can be a symptom of unsuitable choice of hyperparameters. (The difference in the order of magnitude can be a misinterpretation of the meaning of 'dB', the difference seems to be too high)

IV. Conclusion

We have shown that a dense layer for 'Bayes by Backprop' can be implemented as a straighforward extension of the code for a normal dense layer with regularization. This suggests that the variational bayesian model could be extended to different network architectures.

We provide results that support the use of signal-to-noise ratio for weight pruning.

The need for a large number of epochs results in a high cost of hyperparameter optimization. We propose the following research question: Could a suitable prior distribution for network weights be inferred from the distribution of weights of a vanilla neural network? How much bias would it introduce?

References

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org.