Foothill Regularizer as a Binary Quantizer

Mouloud Belbahri*, Eyyüb Sari†, Sajad Darabi*, Xinlin Li*, Matthieu Courbariaux †, and Vahid Partovi Nia*

Abstract

Deep neural networks (DNNs) have demonstrated success for many supervised learning tasks, ranging from voice recognition, object detection, to image classification. Binary quantization is an effective approach to compress DNNs in order to meet memory and computation constraints. Here we present foothill regularizer which is flexible enough to deform towards $L_1$ and $L_2$ penalties. We only focus on neural network regularization and quantization, but the application can be extended to regression smoothing and robust estimation. In particular, this regularizer reduces the accuracy gap between deep BNNs and their full-precision counterpart for image classification.

1 Introduction

Inspired by the extensive research literature on regularization in the statistical community we present foothill (Belbahri et al., 2019), a quasiconvex function with attractive properties and potential to be applied in neural network quantization, regularizing neural networks, smoothing, and robust estimation.

Often, binary neural networks (BNNs) are trained with heuristic methods (Rastegari et al., 2016; Hubara et al., 2016). However it is possible to embed the loss function with an appropriate regularization to encourage binary training (Darabi et al., 2018). Let us define the mathematical notation first. Denote univariate variables with lowercase letters, e.g. $x$, vectors with lowercase and bold letters, e.g. $\mathbf{x}$, and matrices with uppercase and bold letters, e.g. $\mathbf{X}$.

Define the foothill regularization function as

$$p_{\alpha,\beta}(x) = \alpha \tanh\left(\frac{\beta x}{2}\right).$$  

(1)

where $\tanh(.)$ is the hyperbolic tangent function, $\alpha > 0$ is a shape parameter and $\beta > 0$ is a scale parameter. The first and the second derivatives (see Figure 1, right panel) of foothill are

$$\frac{dp_{\alpha,\beta}(x)}{dx} = \alpha \tanh\left(\frac{\beta x}{2}\right) + 2 \alpha \beta x \text{sech}^2\left(\frac{\beta x}{2}\right),$$

$$\frac{d^2p_{\alpha,\beta}(x)}{dx^2} = \frac{1}{2} \alpha \beta \text{sech}^2\left(\frac{\beta x}{2}\right) \left\{ 2 - \beta x \tanh\left(\frac{\beta x}{2}\right) \right\},$$

where sech(.) is the hyperbolic secant function.

Figure 1: Left panel: foothill for $\alpha = 1$, $\beta = 1$ (solid line) and $\alpha = 1$, $\beta = 50$ (dashed line). Right panel: foothill’s first (dashed line) and second (solid line) derivatives for $\alpha = 1$ and $\beta = 1$.

The regularization function (1) has several interesting properties. It is infinitely differentiable (except on a set of measure zero) and symmetric about the origin (see Figure 1, left panel), i.e.,

$$p_{\alpha,\beta}(x) = p_{\alpha,\beta}(-x).$$

Also, it is flexible enough to approximate the lasso (Tibshirani, 1996) and Ridge penalties (Hoerl and Kennard, 1970) for particular values of $\alpha$ and $\beta$. These properties suggest that foothill function could be considered as a quasiconvex alternative to the elastic net penalty (Zou and Hastie, 2005).

2 Binary Quantization

The framework of BNN+ (Darabi et al., 2018) introduces modified $L_1$ and $L_2$ regularization functions which encourage the weights to concentrate around $\mu \times \{-1;+1\}$, where $\mu$ is a scaling factor. For all $x \in \mathbb{R}$, $p \in [1;2]$ and a scaling factor $\mu > 0$, a binary quantizer is defined as

$$R_p(x) = |x - \mu \text{sign}(x)|^p. \tag{2}$$

We follow the generalization of Nia and Belbahri (2018) and modify (1) to construct a shifted regularization function $\tilde{p}_{\alpha,\beta}(x)$ as

$$\tilde{p}_{\alpha,\beta}(x) = p_{\alpha,\beta}(x - \mu \text{sign}(x)). \tag{3}$$

The regularization term is added to the loss function,

$$\mathcal{J}(\mathbf{W}, \mathbf{b}) = \mathcal{L}(\mathbf{W}, \mathbf{b}) + \lambda \sum_{h=1}^{H} \tilde{p}_{\alpha,\beta}(\mathbf{W}_h),$$

where $\mathcal{L}(\mathbf{W}, \mathbf{b})$ is the cost function, $\mathbf{W}$ and $\mathbf{b}$ are the matrices of all weights and bias parameters in

\*Huawei Noah’s Ark Lab, †Montreal Institute for Learning Algorithm (MILA).
AlexNet’s accuracy difference ranges in 4.ing while foothill with $\lambda$ difference depending on which $\lambda$. regularized AlexNet’s accuracy can have 34 less sensitive to the choice of $L$ against 3 empirically demonstrate the flexibility of foothill is set to a value in $\lambda$ 10 optimizer with momentum 0 work for 50 epochs using stochastic gradient descent show the robustness of foothill in comparison with foothill regularizer (orange). The validation curves match the scaling factors. Hence, while training, the line to the back-propagation code in order to esti-mate the scaling factors. Adding the regularization function to the objective function of a deep neural networks adds only one line to the back-propagation code in order to estimate the scaling factors. Hence, while training, the regularization function adapts and the weights are encouraged towards $\mu \times \{-1, +1\}$ (see Figure 2). In our experiments, we learn the scaling factors with back-propagation.

### 3 Application

We use AlexNet architecture augmented by batch normalization in order to compare foothill (1) to $L_1$ and $L_2$ regularizers on CIFAR-10. We train the network for 50 epochs using stochastic gradient descent optimizer with momentum 0.9 and a learning rate of $10^{-3}$ that is divided by 10 at epochs 20 and 30.

For each experiment, the regularization constant $\lambda$ is set to a value in $\{10^{-4}, 10^{-3}, 10^{-2}\}$. Figure 3 empirically demonstrate the flexibility of foothill against $L_1$ and $L_2$. Our regularization function is less sensitive to the choice of $\lambda$. For instance, $L_1$-regularized AlexNet’s accuracy can have 34.16% difference depending on which $\lambda$ has been used for training while foothill with $\alpha = 0.5$ and $\beta = 50$ regularized AlexNet’s accuracy difference ranges in 4.96%.

For binary quantization, we use the shifted version from equation (3) in order to quantize a neural network. We quantize AlexNet architecture on ImageNet (Krizhevsky et al., 2012). This dataset consists of ~1.2M training images, 50K validation images and 1000 classes. For pre-processing, images are resized to 256 × 256 and a random crop is applied to obtain 224 × 224 input size. Random horizontal flip is also used as a data augmentation technique. At test time, images are resized to 256 × 256 and a center crop is applied to get 224 × 224 size. For both steps, standardization is applied with mean = $[0.485, 0.465, 0.406]$ and std = $[0.229, 0.224, 0.225]$.

We adopt the architecture described in Darabi et al. (2018) where batch normalization layers are added (Ioffe and Szegedy, 2015). Weights and activations are quantized using the sign function for all convolutional and fully-connected layers except the first and the last ones which are kept in full-precision. We initialize the learning rate with $5 \times 10^{-3}$ and divide it each 10 epochs alternatively, by 5 and by 2. We use $\lambda = 10^{-6} \times \log(t)$ where $t$ is the current epoch and train the networks for 100 epochs. We compare our method to traditional binary networks.

Table 1: Comparison of top-1 and top-5 accuracies of quantized AlexNet to traditional BinaryNet (Hubara et al., 2016) and XNOR-Net (Rastegari et al., 2016) on ImageNet data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>43.0%</td>
<td>67.5%</td>
</tr>
<tr>
<td>$\tilde{p}_{0.5,50}$</td>
<td>44.4%</td>
<td>68.5%</td>
</tr>
<tr>
<td>$R_2$</td>
<td>42.9%</td>
<td>67.5%</td>
</tr>
<tr>
<td>$\tilde{p}_{20,0.1}$</td>
<td>44.5%</td>
<td>68.3%</td>
</tr>
<tr>
<td>BinaryNet</td>
<td>41.2%</td>
<td>65.6%</td>
</tr>
<tr>
<td>XNOR-Net</td>
<td>44.2%</td>
<td>69.2%</td>
</tr>
<tr>
<td>Full-Precision</td>
<td>57.1%</td>
<td>80.3%</td>
</tr>
</tbody>
</table>

In Table 1, we report XNOR-Net performance from the original paper of Rastegari et al. (2016) and the BinaryNet one from the implementation of Lin et al. (2017), which is higher than the one reported in the original paper. We do not report the performance of Darabi et al. (2018) as they make use of a pre-trained model in their experiments, whereas we train the binary neural networks from scratch. We see that quantizing a neural network using foothill function as a regularization that pushes the weights towards binary values gives more accurate results for ImageNet dataset, better than $R_1$ and $R_2$ by more than 1.5%, which is a big gain for BNNs. Furthermore, for AlexNet architecture, our method beats the state of the art BinaryNet and XNOR-Net.
References

Belbahri, M., E. Sari, S. Darabi, and V. P. Nia

Darabi, S., M. Belbahri, M. Courbariaux, and V. P. Nia

Hoerl, A. E. and R. W. Kennard

Hubara, I., M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio

Ioffe, S. and C. Szegedy

Krizhevsky, A., I. Sutskever, and G. E. Hinton

Lin, X., C. Zhao, and W. Pan

Nia, V. P. and M. Belbahri

Rastegari, M., V. Ordonez, J. Redmon, and A. Farhadi

Tibshirani, R.

Zou, H. and T. Hastie