BinaryConnect: Training Deep Neural Networks with binary weights during propagations

2018-11-08

Overview

- 使用binary weights在训练中的forward and backward propagations过程
- 讨论了使用Deterministic / Stochastic Binarization的影响
- 二值化了weight

BinaryConnect: Training Deep Neural Networks with binary weights during propagations

- Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David
- Comments: Accepted at NIPS 2015, 9 pages, 3 figures
- Theano

BinaryConnect

- 将网络中的weights 二值化为 +1 or -1 在propagations过程中使用
- 介绍了Deterministic / Stochastic Binarization

$$w_b = \begin{cases} +1 & \text{if } w \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{1}$$

$$w_b = \begin{cases} +1 & \text{with probability } p = \sigma(w), \\ -1 & \text{with probability } 1 - p. \end{cases} \tag{2}$$

where σ is the *"hard sigmoid"* function:

$$\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2})) \tag{3}$$

- 在SGD的propagations中使用binary weight但在update中不使用（Keeping good precision weights during the updates is necessary for SGD to work at all.）

Algorithm 1 SGD training with BinaryConnect. C is the cost function for minibatch and the functions binarize(w) and clip(w) specify how to binarize and clip weights. L is the number of layers.

Require: a minibatch of (inputs, targets), previous parameters w_{t-1} (weights) and b_{t-1} (biases), and learning rate η.

Ensure: updated parameters w_t and b_t.

1. Forward propagation:

$w_b \leftarrow \text{binarize}(w_{t-1})$

For $k = 1$ to L, compute a_k knowing a_{k-1}, w_b and b_{t-1}

2. Backward propagation:

Initialize output layer's activations gradient $\frac{\partial C}{\partial a_L}$

For $k = L$ to 2, compute $\frac{\partial C}{\partial a_{k-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and w_b

3. Parameter update:

Compute $\frac{\partial C}{\partial w_b}$ and $\frac{\partial C}{\partial b_{t-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and a_{k-1}

$w_t \leftarrow \text{clip}(w_{t-1} - \eta \frac{\partial C}{\partial w_b})$

$b_t \leftarrow b_{t-1} - \eta \frac{\partial C}{\partial b_{t-1}}$

BinaryConnect

- Clipping: we have chosen to clip the real-valued weights within the [-1; 1] interval right after the weight updates因为太大的参数不会影响binary weight的结果

2.5 A few more tricks

Optimization	No learning rate scaling	Learning rate scaling
SGD		11.45%
Nesterov momentum	15.65%	11.30%
ADAM	12.81%	**10.47%**

- What are reasonable ways of using such a trained network, i.e., performing **test-time inference** on new examples?
  - 1. Use the resulting binary weights wb (this makes most sense with the deterministic form of BinaryConnect).（ Deterministic Binarization make sense）
  - 2. Use the real-valued weights w , i.e., the binarization only helps to achieve faster training but not faster test-time performance.（二值加速训练，test-time inference实数计算）
  - 3. In the stochastic case, many different networks can be sampled by sampling a wb for each weight according to Eq. 2. The ensemble output of these networks can then be obtained by averaging the outputs from individual networks.（随机二值化，多模型集成）

Experiments

test-time inference

We use the first method with the deterministic form of BinaryConnect. As for the stochastic form of BinaryConnect, we focused on the training advantage and used the second method in the experiments, i.e., test-time inference using the real-valued weights. This follows the practice of Dropout methods, where at test-time the "noise" is removed.

Method	MNIST	CIFAR-10	SVHN
No regularizer	$1.30 \pm 0.04\%$	10.64%	2.44%
BinaryConnect (det.)	$1.29 \pm 0.08\%$	9.90%	2.30%
BinaryConnect (stoch.)	$1.18 \pm 0.04\%$	**8.27%**	2.15%
50% Dropout	$1.01 \pm 0.04\%$		
Maxout Networks [29]	0.94%	11.68%	2.47%
Deep L2-SVM [30]	**0.87%**		
Network in Network [31]		10.41%	2.35%
DropConnect [21]			1.94%
Deeply-Supervised Nets [32]		9.78%	**1.92%**

Benchmark results (CIFAR-10)

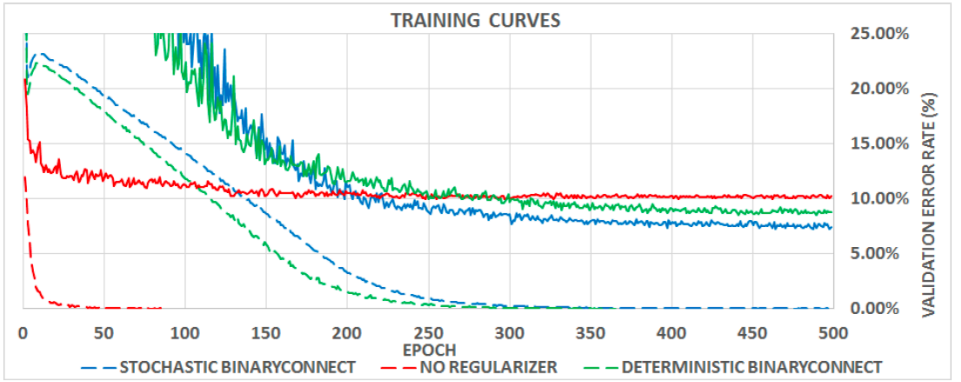

Figure 3: Training curves of a CNN on CIFAR-10 depending on the regularizer. The dotted lines represent the training costs (square hinge losses) and the continuous lines the corresponding validation error rates. Both versions of BinaryConnect significantly augment the training cost, slow down the training and lower the validation error rate, which is what we would expect from a Dropout scheme.