Backpropagation and Higher-order Gradients

Kazuki Yoshiyama

July 2020

Abstract

This is the mathematical notes for computing the first-order gradients of functions and higher-order gradients in terms of mathematical view.

1 Introduction

This is the mathematical notes for computing the first-order gradients of functions and higher-order gradients. We can universally support the n-th order gradients, i.e., the infinite order gradients by definition of the first-order gradients and construction of the implementation of a computational graph. Note that the latter is out-of-scope in this technical notes.

The first-order gradients of many functions can be computed by the combination of other functions, and also there sometimes exists a periodic pattern among n-th order gradients of a function (e.g., *sin* and *cos* functions). Thus, we do not need to consider really messy derivation for n-th order gradients for almost all functions.

The function inputs are generally denoted by x, w for trainable parameters, and y for outputs. The first-order gradient operator is denoted by d, or $\frac{\partial L}{\partial \cdot}$ where L is the objective in an optimization problem and $\cdot$ is of the w.r.t.. The n-th gradient operator is denoted by $d_n := \frac{\partial L_n}{\partial \cdot}$. We generally use the gradient operator up to d_2 mainly because of the periodicity. In case of the first-order gradients, we generally omit the suffix for simpler notations.

From the following section, a section shows a group of function, and subsection is for a specific function.

2 Neural Network Layer

2.1 Affine (Linear)

Forward pass:

$$y = Wx + b \tag{1}$$

Backward pass:

$$dx = W^T dy \tag{2}$$
$$dW = dy x^T \tag{3}$$
$$db = dy. \tag{4}$$

If we take a gradient further in case of the backward pass,

$$d_2 x = (d_2 d_1 W)^T d_1 y \tag{5}$$
$$d_2 W = d_1 y (d_2 d_1 x)^T \tag{6}$$
$$d_2 d_1 y = W(d_2 d_1 x) + (d_2 d_1 W)x + d_2 d_1 b. \tag{7}$$

Interestingly, the first equation is same as (2) as a function, the second equation is same as (3) also as a function, and finally the third equation is same as the forward pass definition of the affine, all are same as a function but with the different inputs. Thus, we can reuse the existing implementation, and also there are a cycle among the n-th order gradients.

This interesting pattern can also be seen in other functions, we sometimes note that in each function.

2.2 Convolution

The convolution is the same operation as affine's but within a receptive field. Thus, we can apply the same argument of the affine for the backward and higher-order gradients.

2.3 DepthwiseConvolution

The depthwise convolution is the same operation of the convolution in the forward and backward except that the convolution does not happens over the feature dimension. Thus, we can apply the same argument of the affine for the backward and higher-order gradients.

2.4 Deconvolution

The deconvolution is the opposite operation of the convolution in the forward and backward pass. Thus, we can apply the same argument of the affine for the backward and higher-order gradients. The only exception is the padding case. For some padding patterns, we can not use the directly opposite operations.

2.5 DepthwiseDeconvolution

The depthwise convolution is the same operation of the deconvolution in the forward and backward except that the convolution does not happens over the feature dimension. Thus, we can apply the same argument of the affine for the backward and higher-order gradients.

2.6 RNN

Forward pass:

$$TODO. \tag{8}$$

This is a composite function of affines and non-linearity. Thus, we can reuse those functions.

2.7 LSTM

Forward pass:

$$TODO. \tag{9}$$

This is a composite function of affines and non-linearity. Thus, we can reuse those functions.

2.8 GRU

Forward pass:

$$TODO. \tag{10}$$

This is a composite function of affines and non-linearity. Thus, we can reuse those functions.

3 Pooling

3.1 MaxPooling

Forward pass:

$$y_{i_1,i_2} = \max_{k_1,k_2 \in K} (x_{i_1+k_1,i_2+k_2}). \tag{11}$$

Backward pass:

$$dx_{i_1+k_1,i_2+k_2} = dy_{i_1,i_2}, \tag{12}$$

$$k_1, k_2 = \arg\max_{k_1,k_2 \in K} (x_{i_1+k_1,i_2+k_2}). \tag{13}$$

The second order gradient is the maxpooling of $d_2 d_1 x_{i_1+k_1,i_2+k_2}$ again. Thus, we have a periodic relationship the gradient operators: $d_0(\cdot) = d_2(\cdot), d_1(\cdot) = d_3(\cdot), \ldots$.

3.2 AveragePooling

Forward pass:

$$y_{i_1,i_2} = \frac{1}{K_1 K_2} \sum_{k1} \sum_{k2} x_{i_1+k_1,i_2+k_2} \tag{14}$$

Backward pass:

$$dx_{i_1+k_1,i_2+k_2} = dy_{i_1,i_2} \times \frac{1}{K_1 K_2}. \tag{15}$$

Note that the backward of the average pooling is simply the broadcast in the receptive field and multiplication.

The second order gradient is the average pooling of $d_2 d_1 x_{i_1+k_1,i_2+k_2}$ again. Thus, we have a periodic relationship of the gradient operators: $d_0(\cdot) = d_2(\cdot), d_1(\cdot) = d_3(\cdot), \ldots$.

3.3 GlobalAveragePooling

Forward pass:

$$y_{i_1,i_2} = \frac{1}{K_1 K_2} \sum_{k1} \sum_{k2} x_{i_1+k_1,i_2+k_2} \tag{16}$$

Backward pass:

$$dx_{i_1+k_1,i_2+k_2} = dy_{i_1,i_2} \times \frac{1}{K_1 K_2}. \tag{17}$$

Note that the backward of the global average pooling is simply the broadcast in the receptive field (over input feature map) and multiplication. The second order gradient is the global average pooling of $d_2 d_1 x_{i_1+k_1,i_2+k_2}$ again. Thus, we have a periodic relationship the gradient operators: $d_0(\cdot) = d_2(\cdot), d_1(\cdot) = d_3(\cdot), \ldots$.

3.4 SumPooling

Forward pass:

$$y_{i_1,i_2} = \sum_{k1} \sum_{k2} x_{i_1+k_1,i_2+k_2}. \tag{18}$$

Backward pass:

$$dx_{i_1+k_1,i_2+k_2} = dy_{i_1,i_2}. \tag{19}$$

Note that the backward of the sum pooling is simply the broadcast in the receptive field.

The second order gradient is the sum pooling of $d_2 d_1 x_{i_1+k_1,i_2+k_2}$ again. Thus, we have a periodic relationship the gradient operators: $d_0(\cdot) = d_2(\cdot), d_1(\cdot) = d_3(\cdot), \ldots$.

3.5 Unpooling

Unpooling is the broadcast in the receptive field.

Forward pass:

$$y_{k_1 i_1 + j_1, k_2 i_2 + j_2} = x_{i_1, i_2} \tag{20}$$

Backward pass:

$$dx_{i_1, i_2} = \sum_{k1} \sum_{k2} dy_{k_1 i_1 + j_1, k_2 i_2 + j_2} \tag{21}$$

The second order gradient is the unpooling of $d_2 d_1 x_{i_1, i_2}$ again. Thus, we have a periodic relationship the gradient operators: $d_0(\cdot) = d_2(\cdot), d_1(\cdot) = d_3(\cdot), \ldots$.

3.6 Embed

Embed is the affine function but with a fixed look-up indices.

$$y = Wx. \tag{22}$$

Backward pass:

$$dW = dy x^T. \tag{23}$$

If we take a gradient further in case of the backward pass,

$$d_2 d_1 y = (d_2 d_1 W) x. \tag{24}$$

4 Activation Function

4.1 Sigmoid

Forward pass:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{25}$$

Backward pass:

$$dx = dy \times \sigma(x)(1 - \sigma(x)). \tag{26}$$

4.2 Swish

Forward pass:

$$y_i = \frac{x_i}{1 + \exp(-x_i)} \tag{27}$$

$$= x_i \sigma(x_i). \tag{28}$$

Backward pass:

$$dx_i = dy_i \times \sigma(x_i)(1 + x_i(1 - \sigma(x_i))). \tag{29}$$

4.3 Tanh

Forward pass:

$$y_i = \tanh(x_i). \tag{30}$$

Backward pass:

$$dx_i = dy_i(1 - \tanh^2(x_i)). \tag{31}$$

4.4 ReLU

Forward pass:

$$y_i = \max(0, x_i). \tag{32}$$

Backward pass:

$$dx_i = dy_i \begin{cases} 1 & (x > 0) \\ 0 & (\text{otherwise}) \end{cases}. \tag{33}$$

4.5 LeakyReLU

Forward pass:

$$y_i = \alpha * \min(0, x_i) + \max(0, x_i). \tag{34}$$

Backward pass:

$$dx_i = dy_i \begin{cases} 1 & (x > 0) \\ \alpha & (\text{otherwise}) \end{cases}. \tag{35}$$

4.6 Softmax

Forward pass:

$$\sigma(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{36}$$

Backward pass:

$$\begin{aligned} dx_j &= \sum_i dy_i \sigma(x_i)(\delta_{ij} - \sigma(x_j)) && (37) \\ &= dy_j \sigma(x_j) - \sigma(x_j) \sum_i dy_i \sigma(x_i). && (38) \end{aligned}$$

4.7 LogSoftmax

Forward pass:

$$y_i = \log \frac{\exp(x_i)}{\sum_j \exp(x_j)}. \tag{39}$$

Backward pass:

$$\begin{aligned} dx_j &= \sum_i dy_i (\delta_{ij} - \sigma(x_j)) && (40) \\ &= dy_j - \sigma(x_j) \sum_i dy_i. && (41) \end{aligned}$$

4.8 ELU

Forward pass:

$$y_i = \begin{cases} x_i & (x > 0) \\ \alpha(\exp(x_i) - 1) & (x \leq 0) \end{cases}. \tag{42}$$

Backward pass:

$$dx_i = dy_i \begin{cases} 1 & (x > 0) \\ \alpha \exp(x_i) & (x \leq 0) \end{cases}. \tag{43}$$

4.9 SELU

Forward pass:

$$y_i = \lambda \begin{cases} x_i & (x > 0) \\ \alpha(\exp(x_i) - 1) & (x \leq 0) \end{cases}. \tag{44}$$

Backward pass:

$$dx_i = dy_i \lambda \begin{cases} 1 & (x > 0) \\ \alpha \exp(x_i) & (x \leq 0) \end{cases}. \tag{45}$$

4.10 CReLU

Forward pass:

$$y = [\max(0, x), \max(0, -x)]. \tag{46}$$

Backward pass:

$$dx = dy_0 \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases} - dy_1 \begin{cases} 1 & (-x > 0) \\ 0 & (-x \leq 0) \end{cases}, \tag{47}$$

where dy_0 and dy_1 are corresponding parts by the concatenation.

4.11 CELU

Forward pass:

$$y_i = [elu(x_i), elu(-x_i)]. \tag{48}$$

Backward pass:

$$dx = dy_0 \begin{cases} 1 & (x > 0) \\ \alpha \exp(x_i) & (x \leq 0) \end{cases} - dy_1 \begin{cases} 1 & (-x > 0) \\ \alpha \exp(-x_i) & (-x \leq 0) \end{cases}, \tag{49}$$

where dy_0 and dy_1 are corresponding parts by the concatenation.

4.12 PReLU

Forward pass:

$$y_i = \max(0, x_i) + w_i \min(0, x_i). \tag{50}$$

Backward pass:

$$dx_i = dy_i \begin{cases} 1 & (x > 0) \\ w_i & (x \leq 0) \end{cases}, \tag{51}$$

$$w_i = dy_i \begin{cases} 0 & (x > 0) \\ x_i & (x \leq 0) \end{cases} \tag{52}$$

Take care that we have to take the summation over the other dimensions other than i. In case of the shared w,

$$dx_i = dy_i \begin{cases} 1 & (x > 0) \\ w_i & (x \leq 0) \end{cases}, \tag{53}$$

$$w = \sum_i dy_i \begin{cases} 0 & (x > 0) \\ x_i & (x \leq 0) \end{cases} \tag{54}$$

Take care that we also have to take the summation over the other dimensions other than i.

4.13 GELU

Forward pass:

$$GELU(x) = xP(X \leq x) = x\Phi(x) \tag{55}$$

$$\approx 0.5x \times (1 + \tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\right)) \tag{56}$$

Backward pass:

$$dx = dy\left(0.5u + 0.5x(1 - \tanh^2(v))\sqrt{2/\pi}(1 + 0.134145x^2)\right) \tag{57}$$

$$u = 1 + \tanh(v) \tag{58}$$

$$v = \sqrt{2/\pi}(x + 0.044715x^3). \tag{59}$$

4.14 ReLU6

Forward pass:

$$y = \min(\max(0, x), 6). \tag{60}$$

Backward pass:

$$dx_i = dy_i \begin{cases} 0 & (x >= 6) \\ 1 & (\text{otherwise}) \\ 0 & (x <= 0) \end{cases}. \tag{61}$$

4.15 HardSigmoid

Forward pass:

$$y = \begin{cases} 1 & (x >= 2.5) \\ 0.2x + 0.5 & (\text{otherwise}) \\ 0 & (x <= -2.5) \end{cases}. \tag{62}$$

Backward pass:

$$dx = dy \begin{cases} 0 & (x > 2.5) \\ 0.2 & (\text{otherwise}) \\ 0 & (x < -2.5) \end{cases}. \tag{63}$$

4.16 HardTanh

Forward pass:

$$y = x \begin{cases} 1 & (x > 1) \\ x & (\text{otherwise}) \\ -1 & (x < -1) \end{cases}. \tag{64}$$

Backward pass:

$$dx = dy \begin{cases} 0 & (x >= 1) \\ 1 & (\text{otherwise}) \\ 0 & (x <= -1) \end{cases}. \tag{65}$$

4.17 LogSigmoid

Forward pass:

$$\begin{aligned} y_i &= \log(1/(1 + \exp(-x_i))) && (66) \\ &= \log(\sigma(x_i)). && (67) \end{aligned}$$

Backward pass:

$$dx_i = dy_i(1 - \sigma(x_i)). \tag{68}$$

4.18 SoftPlus

Forward pass:

$$y_i = \log(1 + \exp(x_i)). \tag{69}$$

Backward pass:

$$dx_i = dy_i \frac{\exp(x_i)}{1 + \exp(x_i)}. \tag{70}$$

4.19 SoftSign

Forward pass:

$$y = x/(1 + |x|). \tag{71}$$

Backward pass:

$$dx = dy \frac{1}{(1 + |x)|)^2}. \tag{72}$$

4.20 TanhShrink

Forward pass:

$$y = x - \tanh(x). \tag{73}$$

Backward pass:

$$dx = dy \left(1 - (1 - \tanh^2(x))\right). \tag{74}$$

4.21 Sinc

Forward pass:

$$y = \begin{cases} \frac{\sin(x)}{x} & (x \neq 0) \\ 1 & (\text{otherwise}) \end{cases}. \tag{75}$$

Backward pass:

$$dx = dy \begin{cases} \frac{\cos(x) - sinc(x)}{x} & (x \neq 0) \\ 0 & (\text{otherwise}) \end{cases}. \tag{76}$$

5 Normalization

5.1 BatchNormalization

Forward pass (Training):

$$\begin{aligned} \mu &= \frac{1}{M} \sum x_i && (77) \\ \sigma^2 &= \frac{1}{M} \left(\sum x_i - \mu \right)^2 && (78) \\ \hat{x}_i &= \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} && (79) \\ y_i &= \hat{x}_i \gamma + \beta. && (80) \end{aligned}$$

Forward pass (Test):

$$y_i = \gamma \frac{x_i - \mu_r}{\sqrt{\sigma_r^2 + \epsilon}} + \beta \tag{81}$$

Backward pass (Training):

$$d\hat{x}_i = dy_i \cdot \gamma, \tag{82}$$

$$d\sigma^2 = \sum_{i=1}^{m} d\hat{x}_i \cdot (x_i - \mu) \cdot \frac{-1}{2}(\sigma^2 + \epsilon)^{-3/2}, \tag{83}$$

$$d\mu = \frac{-1}{\sqrt{\sigma^2 + \epsilon}} \sum_{i=1}^{m} d\hat{x}_i, \tag{84}$$

$$dx_i = d\hat{x}_i \frac{1}{\sqrt{\sigma^2 + \epsilon}} + d\sigma^2 \frac{2(x_i - \mu)}{m} + d\mu \cdot \frac{1}{m}, \tag{85}$$

$$d\gamma = \sum_{i=1}^{m} dy_i \cdot \hat{x}_i, \tag{86}$$

$$d\beta = \sum_{i=1}^{m} dy_i. \tag{87}$$

Note the original batch normalization paper includes the redundant term in the gradient w.r.t. the mean input. It is explicitly excluded in the above gradient derivation.

Backward pass (Test):

$$dx_i = dy_i \frac{\gamma}{\sqrt{\sigma_r^2 + \epsilon}} \tag{88}$$

$$d\gamma = \sum_i dy_i \frac{x_i - \mu_r}{\sqrt{\sigma_r^2 + \epsilon}} \tag{89}$$

$$d\beta = \sum_i dy_i. \tag{90}$$

5.2 SyncBatchNormalization

Formulation is same as the batch normalization, but the running mean/variance are computed over multiple devices.

5.3 WeightNormalization

Forward pass (for one filter):

$$\hat{v} = g \odot v \odot s^{-1/2}, \tag{91}$$

$$s = \sum_i v_i^2 + \epsilon \tag{92}$$

$$g \in \mathbb{R}, v \in \mathbb{R}^D \tag{93}$$

Backward pass (for one filter):

$$dv_j = d\hat{v}_j \odot g \odot s^{-1/2} - \left(\sum_i d\hat{v}_i v_i\right) \odot g \odot s^{-3/2} \odot v_j, \quad (94)$$

$$dg = \sum_i d\hat{v}_i v_i \odot s^{-1/2}. \quad (95)$$

5.4 Norm

Forward pass:

$$y = \|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}} \quad (96)$$

Backward pass:

$$dx_i = dy \times \left(\sum_i |x_i|^p\right)^{\frac{1}{p}-1} \times \begin{cases} |x_i|^{p-1} & (x_i >= 0) \\ -|x_i|^{p-1} & (\text{otherwise}) \end{cases}. \quad (97)$$

5.5 NormNormalize

Forward pass:

$$y = x_i \times \left(\sum_i |x_i|^p\right)^{-\frac{1}{p}} \quad (98)$$

$$= \frac{x_i}{\|x\|_p} \quad (99)$$

Backward pass:

$$dx_i = dy \times \left(\sum_i |x_i|^p\right)^{-\frac{1}{p}} + dy \times \left(\sum_i |x_i|^p\right)^{-\frac{1}{p}-1} \times \begin{cases} x_i^2 & (x_i >= 0) \\ -x_i^2 & (\text{otherwise}) \end{cases}. \quad (100)$$

6 Reduction

6.1 Sum

Forward pass:

$$y = \sum_i x_i. \quad (101)$$

Backward pass (broadcast):

$$dx_i = dy. \quad (102)$$

6.2 Mean

Forward pass:

$$y = \sum_{i}^{N} x_i / N. \tag{103}$$

Backward pass:

$$dx_i = dy/N. \tag{104}$$

6.3 Max

Forward pass:

$$y = \max_i x_i. \tag{105}$$

Backward pass:

$$dx_i = dy \begin{cases} 1 & (i = \arg\max_i(x_i) \\ 0 & (\text{otherwise}) \end{cases}. \tag{106}$$

6.4 Min

Forward pass:

$$y = \min_i x_i. \tag{107}$$

Backward pass:

$$dx_i = dy \begin{cases} 1 & (i = \arg\min_i(x_i) \\ 0 & (\text{otherwise}) \end{cases}. \tag{108}$$

6.5 Prod

Forward pass:

$$y = \Pi_i x_i. \tag{109}$$

Backward pass:

$$dx_i = dy \Pi_{j \neq i} x_j. \tag{110}$$

6.6 Cumsum

Forward pass:

$$y_i = \sum_{j<i} x_j. \tag{111}$$

Backward pass (reverse cumsum):

$$dx_k = \sum_{i>k} dy_i. \tag{112}$$

6.7 Cumprod

Forward pass:

$$y_i = \Pi_{j<i} x_j \tag{113}$$

Backward pass:

$$dx_k = \sum_{i>k} (dy_i \Pi_{j<i, j\neq k} x_j). \tag{114}$$

7 Arithmetic

7.1 Add2

Forward pass:

$$y_i = x_i^{(0)} + x_i^{(1)}. \tag{115}$$

Backward pass:

$$dx_i^{(0)} = dy_i, \tag{116}$$
$$dx_i^{(1)} = dy_i. \tag{117}$$

7.2 AddN

Forward pass:

$$y_i = x_i^{(0)} + ... + x_i^{(n-1)}. \tag{118}$$

Backward pass:

$$dx_i^{(m)} = dy_i \tag{119}$$

7.3 Sub2

Forward pass:

$$y_i = x_i^{(0)} - x_i^{(1)}. \tag{120}$$

Backward pass:

$$dx_i^{(0)} = dy_i, \tag{121}$$
$$dx_i^{(1)} = -dy_i. \tag{122}$$

7.4 Mul2

Forward pass:

$$y_i = x_i^{(0)} x_i^{(1)}. \tag{123}$$

Backward pass:

$$dx_i^{(0)} = dy_i \times x_i^{(1)}, \tag{124}$$
$$dx_i^{(1)} = dy_i \times x_i^{(0)}. \tag{125}$$

7.5 MulN

Forward pass:

$$y_i = x_i^{(0)} ... x_i^{(n-1)}. \tag{126}$$

Backward pass:

$$dx_i^{(m)} = dy_i \Pi_{n \neq m} x_i^{(n)}. \tag{127}$$

7.6 Div2

Forward pass:

$$y_i = \frac{x_i^{(0)}}{x_i^{(1)}}. \tag{128}$$

Backward pass:

$$dx_i^{(0)} = dy_i x_i^{(1)}, \tag{129}$$
$$dx_i^{(1)} = -dy_i \frac{x_i^{(0)}}{(x_i^{(1)})^2}. \tag{130}$$

7.7 Pow2

Forward pass:

$$y_i = (x_i^{(0)})^{x_i^{(1)}}. \tag{131}$$

Backward pass:

$$dx_i^{(0)} = dy_i \times x_i^{(1)} \times (x_i^{(0)})^{x_i^{(1)}-1}, \tag{132}$$

$$dx_i^{(1)} = dy_i \times (x_i^{(0)})^{x_i^{(1)}} \times \log x_i^{(0)}. \tag{133}$$

7.8 AddScalar

Forward pass:

$$y_i = x_i + v. \tag{134}$$

Backward pass:

$$dx_i = dy_i. \tag{135}$$

7.9 MulScalar

Forward pass:

$$y_i = vx_i. \tag{136}$$

Backward pass:

$$dx_i = dy_i v. \tag{137}$$

7.10 PowScalar

Forward pass:

$$y_i = (x_i)^v. \tag{138}$$

Backward pass:

$$dx_i = dy_i \times v \times (x_i)^{v-1}. \tag{139}$$

7.11 RSubScalar

Forward pass:

$$y_i = v - x_i. \tag{140}$$

Backward pass:

$$dx_i = -dy_i \tag{141}$$

7.12 RDivScalar

Forward pass:

$$y_i = \frac{v}{x_i}. \tag{142}$$

Backward pass:

$$dx_i = -dy_i \frac{v}{x_i^2}. \tag{143}$$

7.13 RPowScalar

Forward pass:

$$y_i = v^{x_i}. \tag{144}$$

Backward pass:

$$dx_i = dy_i \times v^{x_i} \times \log v. \tag{145}$$

8 Logical

8.1 Sign

Forward pass:

$$y = \begin{cases} 1 & (x > 0) \\ -1 & (x < 0) \end{cases}. \tag{146}$$

Backward pass (Zero sub-gradient):

$$dx = 0. \tag{147}$$

Backward pass (Straight-Through-Estimator):

$$dx = dy. \tag{148}$$

8.2 Minimum2

Forward pass:

$$y_i = \min(x_i^{(0)}, x_i^{(1)}). \tag{149}$$

Backward pass:

$$dx_i^{(0)} = dy_i \begin{cases} 1 & (x_i^{(0)} < x_i^{(1)}) \\ 0 & (\text{otherwise}) \end{cases} \tag{150}$$

$$dx_i^{(1)} = dy_i \begin{cases} 1 & (x_i^{(1)} < x_i^{(0)}) \\ 0 & (\text{otherwise}) \end{cases} \tag{151}$$

8.3 Maximum2

Forward pass:

$$y_i = \max(x_i^{(0)}, x_i^{(1)}). \tag{152}$$

Backward pass:

$$dx_i^{(0)} = dy_i \begin{cases} 1 & (x_i^{(0)} > x_i^{(1)}) \\ 0 & (\text{otherwise}) \end{cases} \tag{153}$$

$$dx_i^{(1)} = dy_i \begin{cases} 1 & (x_i^{(1)} > x_i^{(0)}) \\ 0 & (\text{otherwise}) \end{cases} \tag{154}$$

8.4 MinimumScalar

Forward pass:

$$y_i = \min(x_i, v). \tag{155}$$

Backward pass:

$$dx_i^{(0)} = dy_i \begin{cases} 1 & (x_i^{(0)} < v) \\ 0 & (\text{otherwise}) \end{cases} \tag{156}$$

$$\tag{157}$$

8.5 MaximumScalar

Forward pass:

$$y_i = \max(x_i, v). \tag{158}$$

Backward pass:

$$dx_i^{(0)} = dy_i \begin{cases} 1 & (x_i^{(0)} > v) \\ 0 & (\text{otherwise}) \end{cases} \tag{159}$$

$$\tag{160}$$

8.6 LogicalAnd

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} \neq 0 \ \& \ x_i^{(1)} \neq 0) \\ 0 & \text{otherwise} \end{cases}. \tag{161}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{162}$$

8.7 LogicalOr

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 0 & (x_i^{(0)} = 0 \ \& \ x_i^{(1)} = 0) \\ 1 & \text{otherwise} \end{cases}. \tag{163}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{164}$$

8.8 LogicalXor

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} = 0 \ \& \ x_i^{(1)} = 0) \\ 1 & (x_i^{(0)} \neq 0 \ \& \ x_i^{(1)} \neq 0) \\ 0 & \text{otherwise} \end{cases}. \tag{165}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{166}$$

8.9 Equal

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} = x_i^{(1)}) \\ 0 & \text{otherwise} \end{cases}. \tag{167}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{168}$$

8.10 NotEqual

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 0 & (x_i^{(0)} = x_i^{(1)}) \\ 1 & \text{otherwise} \end{cases}. \tag{169}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{170}$$

8.11 GreaterEqual

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} \geq x_i^{(1)}) \\ 0 & (x_i^{(0)} < x_i^{(1)}) \end{cases}. \tag{171}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{172}$$

8.12 Greater

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} > x_i^{(1)}) \\ 0 & (x_i^{(0)} \leq x_i^{(1)}) \end{cases}. \tag{173}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{174}$$

8.13 LessEqual

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} \leq x_i^{(1)}) \\ 0 & (x_i^{(0)} > x_i^{(1)}) \end{cases}. \tag{175}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{176}$$

8.14 Less

Forward pass:

$$f(x_i^{(0)}, x_i^{(1)}) = \begin{cases} 1 & (x_i^{(0)} < x_i^{(1)}) \\ 0 & (x_i^{(0)} \geq x_i^{(1)}) \end{cases}. \tag{177}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{178}$$

8.15 LogicalAndScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i \neq 0 \ \& \ v \neq 0) \\ 0 & \text{otherwise} \end{cases}. \tag{179}$$

Backward pass:

$$dx = 0. \tag{180}$$

8.16 LogicalOrScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 0 & (x_i = 0 \ \& \ v = 0) \\ 1 & \text{otherwise} \end{cases}. \tag{181}$$

Backward pass:

$$dx = 0. \tag{182}$$

8.17 LogicalXorScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i = 0 \ \& \ v = 0) \\ 1 & (x_i \neq 0 \ \& \ v \neq 0) \\ 0 & \text{otherwise} \end{cases}. \tag{183}$$

Backward pass:

$$dx = 0. \tag{184}$$

8.18 EqualScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i = v) \\ 0 & \text{otherwise} \end{cases}. \tag{185}$$

Backward pass:

$$dx = 0. \tag{186}$$

8.19 NotEqualScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 0 & (x_i = v) \\ 1 & \text{otherwise} \end{cases}. \tag{187}$$

Backward pass:

$$dx = 0. \tag{188}$$

8.20 GreaterEqualScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i \geq v \\ 0 & (x_i < v \end{cases}. \tag{189}$$

Backward pass:

$$dx = 0. \tag{190}$$

8.21 GreaterScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i > v \\ 0 & (x_i \leq v \end{cases}. \tag{191}$$

Backward pass:

$$dx = 0. \tag{192}$$

8.22 LessEqualScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i \leq v) \\ 0 & (x_i > v) \end{cases}. \tag{193}$$

Backward pass:

$$dx = 0. \tag{194}$$

8.23 LessScalar

Forward pass:

$$f(x_i, v) = \begin{cases} 1 & (x_i < v) \\ 0 & (x_i \geq v) \end{cases}. \tag{195}$$

Backward pass:

$$dx = 0. \tag{196}$$

8.24 LogicalNot

Forward pass:

$$f(x_i) = \begin{cases} 1 & (x_i = 0) \\ 0 & \text{otherwise} \end{cases}. \tag{197}$$

Backward pass:

$$dx_i^{(\cdot)} = 0. \tag{198}$$

8.25 Where

Forward pass:

$$y = \begin{cases} x_{true} & (cond) \\ x_{false} & (not\ cond) \end{cases}. \tag{199}$$

Backward pass:

$$dx_{true} = dy \begin{cases} 1 & (cond) \\ 0 & (not\ cond) \end{cases}, \tag{200}$$

$$dx_{false} = dy \begin{cases} 0 & (cond) \\ 1 & (not\ cond) \end{cases}. \tag{201}$$

9 Math

9.1 Constant

Forward pass:

$$y = const. \tag{202}$$

Backward pass:

$$dx^{(\cdot)} = 0. \tag{203}$$

9.2 Arange

Forward pass:

$$y = arange(start, stop, step) \tag{204}$$

Backward pass:

$$dx^{(\cdot)} = 0. \tag{205}$$

9.3 Abs

Forward pass:

$$y_i = |x_i| \tag{206}$$

Backward pass:

$$dx_i = dy_i \begin{cases} 1 & (x \geq 0) \\ -1 & (\text{otherwise}) \end{cases}. \tag{207}$$

9.4 Exp

Forward pass:

$$y_i = \exp(x_i). \tag{208}$$

Backward pass:

$$dx_i = dy_i \exp(x_i). \tag{209}$$

9.5 Log

Forward pass:

$$y_i = \ln(x_i). \tag{210}$$

Backward pass:

$$dx_i = dy_i \frac{1}{x_i}. \tag{211}$$

9.6 Identity

Forward pass:

$$y = x \tag{212}$$

Backward pass:

$$dx = dy. \tag{213}$$

9.7 BatchMatmul

Forward pass:

$$R = PQ. \tag{214}$$

Backward pass:

$$dP = dR\ Q^T, \tag{215}$$
$$dQ = P^T\ dR. \tag{216}$$

This is the case where no transpose occurs. If there is a transpose operation, we have to consider the order of the matrix multiplication and the transpose of the operand matrices.

Similar to the affine in the case of n-th order gradients, there is also periodic pattern. Thus, we can reuse the forward pass of the batch matmul for computing n-th order gradients.

9.8 Round

Forward pass:

$$y_i = round(x_i). \tag{217}$$

Backward pass (Zero sub-gradient):

$$dx_i = 0. \tag{218}$$

Backward pass (Straight-Through-Estimator):

$$dx_i = dy_i. \tag{219}$$

9.9 Ceil

Forward pass:

$$y_i = ceil(x_i). \tag{220}$$

Backward pass (Zero sub-gradient):

$$dx_i = 0. \tag{221}$$

Backward pass (Straight-Through-Estimator):

$$dx_i = dy_i. \tag{222}$$

9.10 Floor

Forward pass:

$$y_i = floor(x_i). \tag{223}$$

Backward pass (Zero sub-gradient):

$$dx_i = 0. \tag{224}$$

Backward pass (Straight-Through-Estimator):

$$dx_i = dy_i. \tag{225}$$

9.11 Sin

Forward pass:

$$y_i = \sin(x_i). \tag{226}$$

Backward pass:

$$dx_i = dy_i \cos(x_i). \tag{227}$$

9.12 Cos

Forward pass:

$$y_i = \cos(x_i). \tag{228}$$

Backward pass:

$$dx_i = -dy_i \sin(x_i). \tag{229}$$

9.13 Tan

Forward pass:

$$y_i = \tan(x_i). \tag{230}$$

Backward pass:

$$dx_i = dy_i \frac{1}{\cos(x_i)^2}. \tag{231}$$

9.14 Sinh

Forward pass:

$$y_i = \sinh(x_i). \tag{232}$$

Backward pass:

$$dx_i = dy_i \cosh(x_i). \tag{233}$$

9.15 Cosh

Forward pass:

$$y_i = \cosh(x_i). \tag{234}$$

Backward pass:

$$dx_i = dy_i \sinh(x_i). \tag{235}$$

9.16 ASin

Forward pass:

$$y_i = asin(x_i). \tag{236}$$

Backward pass:

$$dx_i = dy_i \frac{1}{\sqrt{1 - x^2}}. \tag{237}$$

9.17 ACos

Forward pass:

$$y_i = acos(x_i). \tag{238}$$

Backward pass:

$$dx_i = dy_i \frac{-1}{\sqrt{1 - x^2}}. \tag{239}$$

9.18 ATan

Forward pass:

$$y_i = atan(x_i). \tag{240}$$

Backward pass:

$$dx_i = dy_i \frac{1}{1 + x^2}. \tag{241}$$

9.19 ATan2

Forward pass:

$$z = atan2(y, x) = 2 \times arctan(\frac{y}{\sqrt{x^2 + y^2} + x}) \tag{242}$$

Backward pass:

$$dy = dz \frac{x}{x^2 + y^2}, \tag{243}$$

$$dx = dz \frac{-y}{x^2 + y^2} \tag{244}$$

9.20 ASinh

Forward pass:

$$y_i = asinh(x_i). \tag{245}$$

Backward pass:

$$dx_i = dy_i \frac{1}{\sqrt{x^2 + 1}}. \tag{246}$$

9.21 ACosh

Forward pass:

$$y_i = acosh(x_i). \tag{247}$$

Backward pass:

$$dx_i = dy_i \frac{1}{\sqrt{x^2 - 1}}. \tag{248}$$

9.22 ATanh

Forward pass:

$$y_i = atanh(x_i). \tag{249}$$

Backward pass:

$$dx_i = dy_i \frac{1}{1 - x^2}. \tag{250}$$

10 Array Manipulation

The backward of array manipulations is simply that the in-coming gradients go back to the location where the data comes from in the forward pass. Sometimes, there are the opposite relationship like the backward pass of the concatenation is the forward pass of the split.

10.1 Concatenate

Forward pass:

$$y = [x_1, \ldots, x_n]. \tag{251}$$

Backward pass:

$$[dx_1, \ldots, dx_n] = dy. \tag{252}$$

10.2 Split

Forward pass:

$$[y_1, \ldots, y_n] = x. \tag{253}$$

Backward pass:

$$dx = [dy_1, \ldots, dy_n]. \tag{254}$$

10.3 Stack

Forward pass:

$$y = [x_1, \ldots, x_n]. \tag{255}$$

Backward pass:

$$[dx_1, \ldots, dx_n] = dy. \tag{256}$$

10.4 Slice

Forward pass:

$$y = x[start_1 : stop_1 : step_1, ..., start_n : stop_n : step_n] \tag{257}$$

Backward pass:

$$dx[start_1 : stop_1 : step_1, ..., start_n : stop_n : step_n] = dy. \tag{258}$$

10.5 Pad

Forward pass (constant):

$$v(x) = \begin{cases} c & (x < 0) \\ u(x) & (\text{otherwise}) \\ c & (x > W - 1) \end{cases}. \tag{259}$$

Backward pass (constant):

$$du(x) = dv(x) \begin{cases} 0 & (x < 0) \\ 1 & (\text{otherwise}) \\ 0 & (x > W - 1) \end{cases}. \tag{260}$$

10.6 Transpose

Forward pass:

$$y[i_1 \dots i_N] = x[T(i_1), \dots T(i_N)]. \tag{261}$$

Backward pass:

$$dx[T(i_1), \dots T(i_N)] = dy[i_1, \dots i_N]. \tag{262}$$

10.7 Broadcast

Forward pass:

$$y_i = x. \tag{263}$$

Backward pass (sum):

$$dx = \sum_i dy_i. \tag{264}$$

10.8 Tile

Forward pass:

$$y = [x_1, \dots, x_n, x_{n+1}, \dots, x_{2n}, \dots, x_{K\times(n-1)+1}, \dots, x_{K\times n}] \tag{265}$$

Backward pass:

$$dx_i = \sum_{j, mod(j,n)=i} dy_j. \tag{266}$$

10.9 OneHot

Forward pass:

$$TODO. \tag{267}$$

Backward pass:

$$TODO. \tag{268}$$

10.10 Flip

Forward pass:

$$TODO. \tag{269}$$

Backward pass:

$$TODO. \tag{270}$$

10.11 Shift

Forward pass:

$$TODO. \tag{271}$$

Backward pass:

$$TODO. \tag{272}$$

10.12 Sort

Forward pass:

$$y = sort(x) \tag{273}$$

Backward pass:

$$dx = dy[index\ of\ sort(x)]. \tag{274}$$

10.13 Reshape

Forward pass:

Shape of an NdArray changes and strides consequently.

Backward pass:

$$dy_i = dx_i, \tag{275}$$

Shape of an NdArray changes and strides consequently.

10.14 MatrixDiag

Forward pass:

$$\begin{bmatrix} y_1 & & \\ & \ddots & \\ & & y_n \end{bmatrix} = [x_1, \ldots, x_n]. \tag{276}$$

Backward pass:

$$[dx_1, \ldots, dx_n] = \begin{bmatrix} dy_1 & & \\ & \ddots & \\ & & dy_n \end{bmatrix} \tag{277}$$

10.15 MatrixDiagPart

Forward pass:

$$[y_1, \ldots, y_n] = \begin{bmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{bmatrix} \tag{278}$$

Backward pass:

$$\begin{bmatrix} dx_1 & & \\ & \ddots & \\ & & dx_n \end{bmatrix} = [dy_1, \ldots, dy_n]. \tag{279}$$

10.16 BatchInv

Forward pass:

$$Y = X^{-1}. \tag{280}$$

Backward pass:

$$dX = -(X^{-1})^T dY (X^{-1})^T \tag{281}$$

10.17 BatchDet

Forward pass:

$$Y = det(X). \tag{282}$$

Backward pass:

$$dX = dY \odot det(X) \odot (X^{-1})^T. \tag{283}$$

10.18 BatchLogdet

Forward pass:

$$Y = \log(|\det(X)|). \tag{284}$$

Backward pass:

Composite backward of $BatchDet \rightarrow Abs \rightarrow Log$.

10.19 GatherNd

Forward pass:

$$U = GatherNd(O, I). \tag{285}$$

Backward pass:

$$dO = ScatterNd(dU, I). \tag{286}$$

10.20 ScatterNd

Forward pass:

$$O = ScatterNd(U, I). \tag{287}$$

Backward pass:

$$dU = GatherNd(dO, I). \tag{288}$$

11 Signal Processing

11.1 Linear interpolate

Notation of the linear interpolation function s slightly different from the others. Given the real-number query points x, y on the 2D-spatial feature u, the interpolation coefficients are computed as

$$\begin{aligned} p_1 &= x - \lfloor x \rfloor & (289)\\ p_0 &= 1 - p_1 & (290)\\ q_1 &= y - \lfloor y \rfloor & (291)\\ q_0 &= 1 - q_1. & (292) \end{aligned}$$

Where-to-look, or where-to-fetch on the grid feature u are denoted as

$$\begin{aligned} x_0 &= \lfloor x \rfloor & (293)\\ x_1 &= 1 - x_0 & (294)\\ y_0 &= \lfloor y \rfloor & (295)\\ y_1 &= 1 - y_0. & (296) \end{aligned}$$

See the bilinear interpolation for visual understanding. Then,

Forward pass:

$$\begin{aligned} v &= p_0 * q_0 * u(x_0, y_0) && (297)\\ &+ p_0 * q_1 * u(x_0, y_1) && (298)\\ &+ p_1 * q_0 * u(x_1, y_0) && (299)\\ &+ p_1 * q_1 * u(x_1, y_1). && (300) \end{aligned}$$

Backward pass:

$$\begin{aligned} du(x_0, y_0) &= dv * p_0 * q_0 && (301)\\ du(x_0, y_1) &= dv * p_0 * q_1 && (302)\\ du(x_1, y_0) &= dv * p_1 * q_0 && (303)\\ du(x_1, y_1) &= dv * p_1 * q_1. && (304) \end{aligned}$$

The second order gradient is the interpolation of $d_2 d_1 u(\cdot, \cdot)$ again. Thus, we have a periodic relationship for the gradient operators: $d_0(\cdot) = d_2(\cdot), d_1(\cdot) = d_3(\cdot), \ldots$.

This 2D linear interpolation can be generalized to the N-d case.

$$v = \sum_{i_1, i_2, i_n \in \{0,1\}} p_{i_1} \ldots p_{i_n} u(x_{i_1}, \ldots x_{i_n}). \quad (305)$$

11.2 FFT

Forward pass:

$$y = W_{FFT} x =: FFT(x). \quad (306)$$

Backward pass:

$$dx = IFFT(dy). \quad (307)$$

Take care of the normalization coefficient. In the normalized FFT, the backward does not need to deal with the normalized coefficient. In the unnormailzed FFT, the backward needs to address the normalization coefficient.

11.3 IFFT

Forward pass:

$$y = W_{IFFT} x =: IFFT(x). \quad (308)$$

Backward pass:

$$dx = FFT(dy). \tag{309}$$

Take care of the normalization coefficient. In the normalized FFT, the backward does not need to deal with the normalized coefficient. In the unnormailzed FFT, the backward needs to address the normalization coefficient.

12 Stochasticity

12.1 Dropout

Forward pass:

$$y = \begin{cases} \frac{x}{1-p} & (u > p) \\ 0 & (\text{otherwise}) \end{cases} \tag{310}$$

Backward pass:

$$dx = dy \begin{cases} \frac{1}{1-p} & (u > p) \\ 0 & (\text{otherwise}) \end{cases} \tag{311}$$

12.2 TopKData

Forward pass:

$$y = TopKData(x). \tag{312}$$

Backward pass:

$$dx = dy[TopKDataIndex] \tag{313}$$

12.3 TopKGrad

Forward pass:

$$y = TopKGrad(x). \tag{314}$$

Backward pass:

$$dx = dy[TopKGradIndex]. \tag{315}$$

13 Loss Functions

13.1 SigmoidCrossEntropy

Forward pass:

$$y_i = -\left(x_i^{(1)} \ln\left(\sigma\left(x_i^{(0)}\right)\right) + \left(1 - x_i^{(1)}\right) \ln\left(1 - \sigma\left(x_i^{(0)}\right)\right)\right) \tag{316}$$

Backward pass:

$$dx_i^{(0)} = dy_i(\sigma_i - x_i^{(1)}) \tag{317}$$

13.2 BinaryCrossEntropy

Forward pass:

$$y_i = -\left(x_i^{(1)} * \ln\left(x_i^{(0)}\right) + \left(1 - \ x_i^{(1)}\right) * \ln\left(1 - x_i^{(0)}\right)\right). \tag{318}$$

Backward pass:

$$dx_i^{(0)} = dy_i \frac{x_i^{(0)} - x_i^{(1)}}{x_i^{(0}(1 - x_i^{(0)})} \tag{319}$$

13.3 SoftmaxCrossEntropy

Forward pass:

$$y_i \quad = -\ln \sum_i t_i \sigma(x_i) \tag{320}$$

Backward pass:

$$dx_j \quad = dy(\sigma(x_j) - t_j). \tag{321}$$

where the $\sigma(x)$ is the softmax.

13.4 CategoricalCrossEntropy

Forward pass:

$$y_i \quad = -\ln \sum_i t_i x_i \tag{322}$$

Backward pass:

$$dx_j \quad = -dy \frac{t_j}{x_j}. \tag{323}$$

13.5 SquaredError

Forward pass:

$$y_i = \left(x_i^{(0)} - x_i^{(1)}\right)^2. \tag{324}$$

Backward pass:

$$dx_i^{(0)} = dy_i \times 2 \times \left(x_i^{(0)} - x_i^{(1)}\right) \tag{325}$$

$$dx_i^{(1)} = -dy_i \times 2 \times \left(x_i^{(0)} - x_i^{(1)}\right). \tag{326}$$

13.6 AbsoluteError

Forward pass:

$$y_i = |x_i^{(0)} - x_i^{(1)}|. \tag{327}$$

Backward pass:

$$dx_i^{(0)} = \begin{cases} dy_i & (x_i^{(0)} > x^{(1)}) \\ -dy_i & (\text{otherwise}) \end{cases} \tag{328}$$

$$dx_i^{(1)} = \begin{cases} -dy_i & (x_i^{(0)} > x^{(1)}) \\ dy_i & (\text{otherwise}) \end{cases} \tag{329}$$

13.7 HuberLoss

Forward pass:

$$y_i = \begin{cases} (x_i^{(0)} - x_i^{(1)})^2 & (|x_i^{(0)} - x_i^{(1)}| < \delta) \\ \delta(2|x_i^{(0)} - x_i^{(1)}| - \delta) & (\text{otherwise}) \end{cases} \tag{330}$$

Backward pass:

$$dx_i^{(0)} = dy_i \begin{cases} 2(x_i^{(0)} - x_i^{(1)}) & (|x_i^{(0)} - x_i^{(1)}| < \delta) \\ 2\delta & (|x_i^{(0)} - x_i^{(1)}| \geq \delta \ and \ x_i^{(0)} \geq x_i^{(1)}) \\ -2\delta & (|x_i^{(0)} - x_i^{(1)}| \geq \delta \ and \ x_i^{(0)} < x_i^{(1)}) \end{cases} \tag{331}$$

$$dx_i^{(1)} = dy_i \begin{cases} -2(x_i^{(0)} - x_i^{(1)}) & (|x_i^{(0)} - x_i^{(1)}| < \delta) \\ -2\delta & (|x_i^{(0)} - x_i^{(1)}| \geq \delta \ and \ x_i^{(0)} \geq x_i^{(1)}) \\ 2\delta & (|x_i^{(0)} - x_i^{(1)}| \geq \delta \ and \ x_i^{(0)} < x_i^{(1)}) \end{cases} \tag{332}$$

13.8 EpsilonInsensitiveLoss

Forward pass:

$$y_i = \begin{cases} |x_i^{(0)} - x_i^{(1)}| - \epsilon & (|x_i^{(0)} - x_i^{(1)}| > \epsilon) \\ 0 & (\text{otherwise}) \end{cases} \tag{333}$$

Backward pass:

$$dx_i^{(0)} = dy_i \begin{cases} 1 & (|x_i^{(0)} - x_i^{(1)}| > \epsilon \; and \; x_i^{(0)} >= x_i^{(1)}) \\ -1 & (|x_i^{(0)} - x_i^{(1)}| > \epsilon \; and \; x_i^{(0)} < x_i^{(1)}) \\ 0 & \text{otherwise} \end{cases} \tag{334}$$

$$dx_i^{(1)} = dy_i \begin{cases} -1 & (|x_i^{(0)} - x_i^{(1)}| > \epsilon \; and \; x_i^{(1)} >= x_i^{(0)}) \\ 1 & (|x_i^{(0)} - x_i^{(1)}| > \epsilon \; and \; x_i^{(1)} < x_i^{(0)}) \\ 0 & \text{otherwise} \end{cases} \tag{335}$$

13.9 KLMultinomial

Forward pass:

$$D = \sum_i p_i \log\left(\frac{p_i}{q_i}\right). \tag{336}$$

Backward pass:

$$dp_i = dD\left(\log(\frac{p_i}{q_i}) + 1\right), \tag{337}$$

$$dq_i = dD\frac{-p_i}{q_i}. \tag{338}$$

14 Quantization Neural Network Layers

Quantization layers mainly relies on Straight-Through-Estimator (STE) where we use the proxy of a quantized function for deriving the backward pass of the quantized function to simulate smooth gradients. In this sence, the proxy should be closer to the quantized function and smooth one, but we can replace the gradient of a quantized function with much simplier one like 1, meaning just pass the in-coming gradients. This sounds a bit cheating, but STE works fine in practice.

14.1 BinarySigmoid

Forward pass:

$$y = \begin{cases} 1 & (x > 0) \\ 0 & (\text{otherwise}) \end{cases}, \tag{339}$$

Backward pass:

$$dx = 0.5 * dy. \tag{340}$$

14.2 BinaryTanh

Forward pass:

$$f(x) = \begin{cases} 1 & (x > 0) \\ -1 & (\text{otherwise}) \end{cases}, \tag{341}$$

Backward pass:

$$f(x) = \begin{cases} 1 & (x > 0) \\ -1 & (\text{otherwise}) \end{cases}, \tag{342}$$

14.3 Prune

Forward pass:

$$q_i = \begin{cases} 0 & abs(x_i) < threshold \\ x_i & \text{otherwise} \end{cases} \tag{343}$$

Backward pass:

$$dx_i = dq_i. \tag{344}$$

14.4 QuantizeLinear

Forward pass:

$$y = saturate(round(x/s) + z). \tag{345}$$

The *saturate* is, for example in case of int8, $clip(x, -128, 127)$. The *round* depends on implementation, but *round-to-even* or *round-away-from-zero* are often used. Zero-point z is sometimes omitted for computational efficiency.

Backward pass:

$$dx \quad = \begin{cases} dy/s & (saturate(round(x/s)) \\ 0 & (otherwise). \end{cases} \tag{346}$$

14.5 DequantizeLinear

Forward pass:

$$y = (x - z) * s. \tag{347}$$

Backward pass:

$$dx = dy * s \tag{348}$$