A CONCRETE EXAMPLE: BOOLEAN VARIANT

WEIFAN CHEN

Abstract. A very plain example of how to conduct forward path (FP) and backward path (BP) using boolean variant.

1. Notation

All indices are implicitly counted from 1, because sometimes zero is reserved for other meanings.

$\mathbb{B} = \{T, F\}$: the boolean set.
$\mathbb{N}$: natural number set.

$x^l_{k,i} \in \mathbb{B}$: this variable belongs to the k^{th} sample in the batch. It is the i^{th} input of the l^{th} layer. It is also the output of the $(l-1)^{\text{th}}$ layer. When $l = 1$, this indicates the sample input.

$w^l_{i,j} \in \mathbb{B}$: the weight associated to l^{th} layer input. It is the weight that operates between the i^{th} output of the l^{th} layer and the j^{th} input of the downstream unit. When $i = 0$, this indicates the bias to the j^{th} input of the downstream unit.

$L \in \mathcal{F}(\mathbb{B}, \mathbb{B})$: a boolean-input boolean-valued function. We use XNOR throughout this example. Thus $L = \text{XNOR}$ from now on.

$s^l_{k,j} \in \mathbb{N}$: the pre-activated sum belongs to the l^{th} layer. It is the j^{th} pre-activated value. It is calculated as:

$$s^l_{k,j} = w^l_{0,j} + \sum_{i=1}^{M} L\left(w^l_{i,j}, x^l_{k,i}\right) \tag{1}$$

Note: the bias here doesn't make sense to me as it's really supposed to be a constant, it's just a boolean value? I remember looking at the paper, and I also had the same issue there.

Then the next layer input can be calculated as:

$$x^{l+1}_{k,j} = \sigma_\tau\left(s^l_{k,j}\right) \tag{2}$$

in which $\sigma_\tau \in \mathcal{F}(\mathbb{N}, \mathbb{B})$ is a natural-input boolean-value activation function defined as:

$$\sigma_\tau(x) = \begin{cases} T & \text{if } x \geq \tau \\ F & \text{if } x < \tau \end{cases} \tag{3}$$

τ is taken to be constant in this example. It can be half of the number of next layer inputs.

$\text{loss}_k \in \mathcal{F}(\mathbb{B}, \mathbb{B})$: the boolean-input boolean-value function that maps to the loss for

the k^{th} sample in the batch. Throughout this writing, it is XOR for indicating the mismatch between the prediction (x_k), and the ground truth label (y_k). Formally:

$$\text{loss}_k(x_k) = \mathbf{xor}(y_k, x_k) \tag{4}$$

$\mathcal{L}$: the accumulated loss function defined as:

$$\mathcal{L} = \sum_{i=1}^{K} \text{loss}_k(x_k) \tag{5}$$

in which K is the number of samples in the batch.

2. Important Concepts

2.1. **Variant and derivative.**

The notation $f'(x)$ is commonly used for real-number input function. To operate on a boolean DNN, the notation has to adapt for natural-number input functions and boolean-input functions. To do so, first define *variant* δ and its related operations.

Definition: variant

For **any** value a and b, the variant from a to b is:

$$\delta(a \to b) = \begin{cases} T & \text{if } \ b > a \\ 0 & \text{if } \ a = b \\ F & \text{if } \ a > b \end{cases} \tag{6}$$

The value type of δ is three-value logic $\mathbb{M} = \{T, F, 0\}$.

It is useful to define a shorthand:

$$\delta f(a \to b) = \delta(f(a) \to f(b)) \tag{7}$$

For $x \in \mathbb{N}$, define

$$f'(x) = \delta f(x \to x + 1) \tag{8}$$

For $x \in \mathbb{B}$, define (one of the most elegant contributions of the paper)

$$f'(x) = \mathbf{xnor}(\delta(x \to \neg x), \delta f(x \to \neg x)) \tag{9}$$

If any of the inputs of XNOR are 0, then XNOR evaluates to 0. The interpretation of Equation 9 can be understood as the following example. Consider $f'(F) = T$. The truth on the right-hand-side indicates the value of the function $f(x)$ evaluated at $x = F$ will vary to the same direction as x varies. I.e. when x varies from F to T (x is increasing), the value of $f(x)$ will also increase. That is to say, $f(T) > f(F)$.

Here is another example, consider $f'(T) = F$. This indicates the variant of the function value will go to the opposite direction as x varies. Thus as the value of x going down ($i.e. T \to F$), the value of $f(x)$ will go up. I.e. $f(F) > f(T)$.

Here is another example, consider $f'(T) = 0$. This indicates changing x from T to F will not affect value of $f(x)$, i.e. $f(T) = f(F)$.

2.2. **Commonly use results.**

The following useful shorthand will be very helpful while conducting chain rule.

$$f'(x) = \frac{\delta f(x)}{\delta x} \tag{10}$$

State without proof the following properties:

$$\forall x \in \mathbb{B}, \frac{\delta \mathbf{xnor}(a, x)}{\delta x} = a \tag{11}$$

$$\forall x \in \mathbb{B}, \frac{\delta \mathbf{xor}(a, x)}{\delta x} = \neg a \tag{12}$$

The chain rule is calculated (another one of the most elegant contributions of the paper) as:

$$\begin{aligned} \forall x \in \mathbb{B}, (g \circ f)'(x) &= g(f(x))' \\ &= \mathbf{xnor}(g'(f(x)), f'(x)) \end{aligned} \tag{13}$$

Eq.Equation 13 answers the questions: how will the loss change if a boolean weight is flipped.

Lastly, state without proof, the derivative on the cutoff activation function Equation 3

$$\sigma'_\tau(x) = \frac{\delta \sigma_\tau(x)}{\delta x} = \begin{cases} T & \text{if } \ x = \tau - 1 \\ 0 & \text{else} \end{cases} \tag{14}$$

Notice, the equation above answers how the function value will change if x is increased by 1. This is because σ is a natural-number-input function, not a boolean-input function. It only increases as x increases from $\tau - 1$ to τ. Now we are ready to present a complete example.

3. A Concrete Example

First state the configuration. Each sample is a three-dimensional boolean vector, i.e. $x_k^1 \in \mathbb{B}^3$. As an example, $x_{k,1}^1 \in \mathbb{B}$ is first input of the k^{th} sample in the batch (remember we index from 1). It is also the first input of the first layer of the k^{th} sample. The first layer is 3×2. The first layer not only do the weight multiplication, it applies the activation functions as well. The second layer is 2×1. The second layer's output is the final prediction. This prediction will compare with the k^{th} ground truth label y_k. Recall the loss for the k^{th} sample is $\text{loss}_k = \mathbf{xor}\Big(y_k, x_{k,1}^2\Big)$.

3.1. Forward Path (FP).

Only consider the k^{th} sample in the batch. The first layer operation is (1) applying weight multiplication, and (2) applying the activation. Recall L is **xnor**. The weight multiplication is the following:

$$\begin{aligned} s_{k,1}^1 = w_{0,1}^1 &+ L\Big(w_{1,1}^1, x_{k,1}^1\Big) + \\ &L\Big(w_{2,1}^1, x_{k,2}^1\Big) + L\Big(w_{3,1}^1, x_{k,3}^1\Big) \end{aligned} \tag{15}$$

$$\begin{aligned} s_{k,2}^1 = w_{0,2}^1 &+ L\Big(w_{1,2}^1, x_{k,1}^1\Big) + \\ &L\Big(w_{2,2}^1, x_{k,2}^1\Big) + L\Big(w_{3,2}^1, x_{k,3}^1\Big) \end{aligned} \tag{16}$$

Now calculate the activation to get the next layer input:

$$x_{k,1}^2 = \sigma_\tau^1\Big(s_{k,1}^1\Big) \tag{17}$$

$$x_{k,2}^2 = \sigma_\tau^1\Big(s_{k,2}^1\Big) \tag{18}$$

Recall that τ is the number of input for the input layer, thus in this case we have $\tau = \lceil \frac{3}{2} \rceil = 2$ for σ_τ^1. Now we are ready to calculate the second layer.

The weight multiplication is:

$$s_{k,1}^2 = w_{0,1}^2 + L\Big(w_{1,1}^2, x_{k,1}^2\Big) + L\Big(w_{2,1}^2, x_{k,2}^2\Big) \tag{19}$$

The activation is:

$$x_{k,1}^3 = \sigma_\tau^2\Big(s_{k,1}^2\Big) \tag{20}$$

$x_{k,1}^3$ is also the model prediction. Finally, we can calculate the sample-wise loss

$$\text{loss}_k = \mathbf{xor}\Big(y_k, x_{k,1}^3\Big) \tag{21}$$

The total loss is:

$$\mathcal{L} = \sum_{k=1}^{K} \text{loss}_k\Big(y_k, x_{k,1}^3\Big) \tag{22}$$

Where K is the total number of samples in a batch.

3.2. **Backward Path (BP).**

We first consider the BP for a single sample, then aggregate to the batch. Now assume we want to answer the question: if $w_{1,1}^2$ is flipped, will loss_k increase or decrease? First, calculate its BP.

$$\begin{aligned}
\frac{\delta \text{loss}_k}{\delta w_{1,1}^2} &= \frac{\delta}{\delta w_{1,1}^2}\left(\text{loss}_k\Big(y_k, x_{k,1}^3\Big)\right) \\
&= \mathbf{xnor}\left(\frac{\delta \text{loss}_k\Big(y_k, x_{k,1}^3\Big)}{\delta x_{k,1}^3}, \frac{\delta x_{k,1}^3}{\delta w_{1,1}^2}\right) \\
&= \mathbf{xnor}\left(\neg y_k, \frac{\delta x_{k,1}^3}{\delta w_{1,1}^2}\right) \\
&= \mathbf{xnor}\left(\neg y_k, \mathbf{xnor}\left(\frac{\delta x_{k,1}^3}{\delta s_{k,1}^2}, \frac{\delta s_{k,1}^2}{\delta w_{1,1}^2}\right)\right) \\
&= \mathbf{xnor}\Big(\neg y_k, \mathbf{xnor}\Big(\sigma_\tau^{2'}\Big(s_{k,1}^2\Big), x_{k,1}^2\Big)\Big)
\end{aligned} \tag{23}$$

The expression above can be evaluated already, because $s_{k,1}^2$ and $x_{k,1}^2$ have been calculated in FP.

Now let's calculate the variant on weights in the first layer.

$$\begin{aligned}\frac{\delta \text{loss}_k}{\delta w^1_{3,2}} &= \mathbf{xnor}\left(\frac{\delta \text{loss}_k\left(y_k, x^3_{k,1}\right)}{\delta x^3_{k,1}}, \frac{\delta x^3_{k,1}}{\delta w^1_{3,2}}\right) \\ &= \mathbf{xnor}\left(\neg y_k, \mathbf{xnor}\left(\frac{\delta x^3_{k,1}}{\delta s^2_{k,1}}, \frac{\delta s^2_{k,1}}{\delta w^1_{3,2}}\right)\right) \\ &= \mathbf{xnor}\left(\neg y_k, \mathbf{xnor}\left(\sigma'^2_\tau\left(s^2_{k,1}\right), \mathbf{xnor}\left(\frac{\delta s^2_{k,1}}{\delta x^2_{k,2}}, \frac{\delta x^2_{k,2}}{\delta w^1_{2,3}}\right)\right)\right)\end{aligned} \tag{24}$$

The equation is awfully long. Now only computer the component

$$\begin{aligned}\mathbf{xnor}\left(\frac{\delta s^2_{k,1}}{\delta x^2_{k,2}}, \frac{\delta x^2_{k,2}}{\delta w^1_{2,3}}\right) &= \mathbf{xnor}\left(w^2_{2,1}, \mathbf{xnor}\left(\frac{\delta x^2_{k,2}}{\delta s^1_{k,2}}, \frac{\delta s^1_{k,2}}{\delta w^1_{2,3}}\right)\right) \\ &= \mathbf{xnor}\left(w^2_{2,1}, \mathbf{xnor}\left(\sigma^{1'}_\tau\left(s^1_{k,2}\right), x^1_{k,3}\right)\right)\end{aligned} \tag{25}$$

At this point, the value can be fully evaluated.

Let's go through a final example to illustrate how to handle the variant of summations. Consider a three-layer network. $1 \times 2, 2 \times 2, 2 \times 1$. For simplicity, we drop the subscript indicating the sample index. The final output x^4_1. The loss is $x^4_1 \oplus y$ ($\oplus$ is the notation for XOR). When conducting backward propagation, we run into this term

$$s^3_1 = w^3_0 + \mathbf{xnor}\left(w^3_{1,1}, x^3_1\right) + \mathbf{xnor}\left(w^3_{2,1}, x^3_2\right) \tag{26}$$

This express has one key difference compared with the previous examples, because if we want to calculate $\frac{\delta \text{loss}}{\delta w^1_{1,2}}$), the expression will have two terms remaining. In all previous examples, only one term will remain. Explicitly

$$\frac{\delta s^3_1}{\delta w^1_{1,2}} = \frac{\delta \mathbf{xnor}\left(w^3_{1,1}, x^3_1\right)}{\delta w^1_{1,2}} + \frac{\delta \mathbf{xnor}\left(w^3_{2,1}, x^3_2\right)}{\delta w^1_{1,2}} \tag{27}$$

in which both x^3_1 and x^3_2 depend on $w^1_{1,2}$. Appendix Proposition A.2 of the paper [1] indicates the summation can be conducted by type casting the three-value logic into integer. The rule is:

$$e(a) = \begin{cases} +1 & \text{if } a = T \\ 0 & \text{if } a = 0 \\ -1 & \text{if } a = F \end{cases} \tag{28}$$

3.3. **Optimization.**

Finally we are ready to update the weights on a per-batch basis. Let's say we want to decide whether $w^1_{1,2}$ should be flipped. The procedure is the following: first calculate

$$q^1_{1,2,k} = \frac{\delta \text{loss}_k}{\delta w^1_{1,2}} \tag{29}$$

Now we aggregate each sample in the batch. For the k^{th} sample, if $q^1_{1,2,k} = T$, this indicates the loss will vary to to same direction as $w^1_{1,2}$ varies. 0 means it will not vary as the weight varies. False indicates it will vary the opposite direction. Thus

we need to count the number of True and False in the batch, so that we can tell the net result of flipping the weight. The paper has an elegant formulation on this. First we calculate the aggregated result

$$q^1_{1,2} = (\text{num. of True samples}) - (\text{num. of False samples}) \tag{30}$$

Then the net decision is simply

$$w^1_{1,2} = \neg w^1_{1,2} \text{ if } \mathbf{xnor}\big(q^1_{1,2}, w^1_{1,2}\big) = T \tag{31}$$

Now, the BP can be conducted fully (hopefully). The paper has more contents regarding how to handle integer mix type and some other optimizers.

3.4. **Boolean Function Differentiation.**

Let's take another approach to calculus of variations, but this time purely based on booleans and boolean functions. Instead of using three-valued logic to indicate direction of change, we can use a single boolean to indicate whether a function's output would change when its input is flipped.

3.4.1. *Definition.*

For any boolean function $f : \mathbb{B} \to \mathbb{B}$, we define its boolean derivative at point x as:

$$\frac{\delta f}{\delta x} = \text{xor}(f(x), f(\neg x)) \tag{32}$$

This derivative has a simple interpretation:

- If $\frac{\delta f}{\delta x} = T$, then flipping x will change the output of f
- If $\frac{\delta f}{\delta x} = F$, then flipping x will not change the output of f

3.4.2. *Properties.*

The boolean derivative has several useful properties:

1. **Symmetry**: The derivative is symmetric around the flip point
$$\frac{\delta f}{\delta x} = \frac{\delta f}{\delta \neg x} \tag{33}$$
2. **Chain Rule**: For composed functions $g \circ f$, the derivative follows:
$$\frac{\delta g \circ f}{\delta x} = \text{AND}\left(\frac{\delta f}{\delta x}, \frac{\delta g}{\delta f(x)}\right) \tag{34}$$
To verify this, consider $f(x) = \text{not } x$ and $g(x) = \text{and}(T, x)$. Then:
$$(g \circ f)(x) = \text{and}(T, \text{not } x) = \text{not } x \tag{35}$$
We can verify the chain rule:
   - Left side: $\frac{\delta g \circ f}{\delta x} = \text{xor}(\text{not } x, \text{not}(\text{not } x)) = T$
   - Right side: $\text{xnor}\Big(\frac{\delta f}{\delta x}, \frac{\delta g}{\delta f(x)}\Big) = \text{xnor}\Big(T, \frac{\delta g}{\delta \text{not } x}\Big) = \text{xnor}(T, \text{xor}(\text{and}(T, \text{not } x), \text{and}(T, \text{not}(\text{not } x)))) = \text{xnor}(T, \text{xor}(\text{not } x, x)) = \text{xnor}(T, T) = T$
3. **Basic Function Derivatives**:
   - For constant functions: $\frac{\delta c}{\delta x} = F$
   - For identity function: $\frac{\delta \text{id}}{\delta x} = T$
   - For NOT function: $\frac{\delta \text{not}}{\delta x} = T$
   - For AND function: $\frac{\delta \text{and}(a,x)}{\delta x} = a$

- For OR function: $\frac{\delta \text{or}(a,x)}{\delta x} = \neg a$
- For XOR function: $\frac{\delta \text{xor}(a,x)}{\delta x} = T$
- For XNOR function: $\frac{\delta \text{xnor}(a,x)}{\delta x} = T$

3.4.3. *Simple Example.*

Let's consider a simple boolean circuit with two weights w_1, w_2 and one input x:

$$f(x, w_1, w_2) = \text{or}(w_1, \text{and}(w_2, x)) \tag{36}$$

To find how the output would change if we flip each weight, we calculate:

1. For w_1 (with fixed w_2 and x):

$$\frac{\delta f}{\delta w_1} = \text{xor}(f(w_1, w_2, x), f(\neg w_1, w_2, x)) \tag{37}$$

Let's evaluate this for $x = T, w_2 = T$:

$$\frac{\delta f}{\delta w_1} = \text{xor}(\text{or}(w_1, \text{and}(T, T)), \text{or}(\neg w_1, \text{and}(T, T))) \tag{38}$$

$$= \text{xor}(\text{or}(w_1, T), \text{or}(\neg w_1, T)) \tag{39}$$

$$= \text{xor}(T, T) = F \tag{40}$$

This makes sense - when $x = T$ and $w_2 = T$, the output is always T regardless of w_1.

2. For w_2 (with fixed w_1 and x):

$$\frac{\delta f}{\delta w_2} = \text{xor}(f(w_1, w_2, x), f(w_1, \neg w_2, x)) \tag{41}$$

Let's evaluate this for $x = T, w_1 = F$:

$$\frac{\delta f}{\delta w_2} = \text{xor}(\text{or}(F, \text{and}(w_2, T)), \text{or}(F, \text{and}(\neg w_2, T))) \tag{42}$$

$$= \text{xor}(w_2, \neg w_2) = T \tag{43}$$

This makes sense - when $x = T$ and $w_1 = F$, flipping w_2 will always change the output.

Let's verify our results using the chain rule:

1. For w_1 with $x = T, w_2 = T$:

$$\frac{\delta f}{\delta w_1} = \frac{\delta \text{or}(w_1, \text{and}(w_2, x))}{\delta w_1} \tag{44}$$

Since this is a direct OR with w_1, we can compute:

$$= \frac{\delta \text{or}(w_1, T)}{\delta w_1} \tag{45}$$

$$= \begin{cases} \neg T & \text{if } w_1 = F \\ T & \text{if } w_1 = T \end{cases} \tag{46}$$

$$= F \tag{47}$$

2. For w_2 with $x = T, w_1 = F$: Here we need the chain rule since w_2 appears inside the AND:

$$\frac{\delta f}{\delta w_2} = \text{and}\left(\frac{\delta \text{and}(w_2, x)}{\delta w_2}, \frac{\delta \text{or}(w_1, \_)}{\delta \text{and}(w_2, x)}\right) \tag{48}$$

With $x = T, w_1 = F$:

$$= \text{and}\left(\frac{\delta \text{and}(w_2, T)}{\delta w_2}, \frac{\delta \text{or}(F, \_)}{\delta \text{and}(w_2, T)}\right) \tag{49}$$

Let's compute each part:

- First part: $\frac{\delta \text{and}(w_2, T)}{\delta w_2} = \begin{cases} T & \text{if } w_2 = F \\ \neg T & \text{if } w_2 = T \end{cases} = T$
- Second part: $\frac{\delta \text{or}(F, y)}{\delta y} \mid_{y=\text{and}(w_2, T)} = \begin{cases} \neg F & \text{if } y = F \\ F & \text{if } y = T \end{cases} = T$

Therefore:

$$= \text{and}(T, T) = T \tag{50}$$

Both methods give us the same results, confirming our calculations.

References

1. Nguyen, V. M., Ocampo, C., Askri, A., Tran, B.-H.: Boolean Logic for Low-Energy Deep Learning. In: 2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024) (2024)

Department of Computer Science, Boston University, Boston, U.S.A.
Email address: wfchen@bu.edu