Yuchen Cai Week2 Report

Yuchen.cai.uestc@gmail.com

1. 完成任务:

   1) Back Propagation of Neural Network
   2) Back Propagation of Binary Neural Network

2. Future Work:

   1) 代码部分:

      - 国庆期间，看完 util.py 中各个函数的具体写法和作用
      - 国庆期间，学习 Pytorch 0.4(自己之前用的一直是 0.3)

   2) 理论部分:

      - 国庆期间，看 W1 和 W2 额外的四篇论文

3. 未解决部分:

   1) XNOR-NET 论文当中，关于二值化网络的训练部分引用的是 NIPS 2016-Binaryconnect-training-deep-neural-networks-with-binary-weights-during-propagations 这篇论文，其中所给公式为:

$$\frac{\partial C}{\partial W_i} = \frac{\partial C}{\tilde{W}_i}\left(\frac{1}{n} + \frac{\partial sign}{\partial W_i} \times \alpha\right)$$

   但在 github@jiecaoyu 的代码当中，给出了不一样的解释，核心区别在于:

$$\frac{\partial C}{\partial W_i} = \sum_{j=1}^{n}\left(\frac{\partial C}{\partial \tilde{W}_j} \cdot \frac{\partial \tilde{W}_j}{\partial W_i}\right)$$

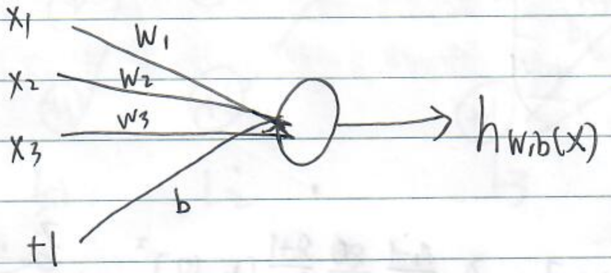
   未能理解哪一个究竟是对的，已提至 issue

No.

Date

Back Propagation of neural network

A fixed training set: $\{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\}$，总共m个 sample



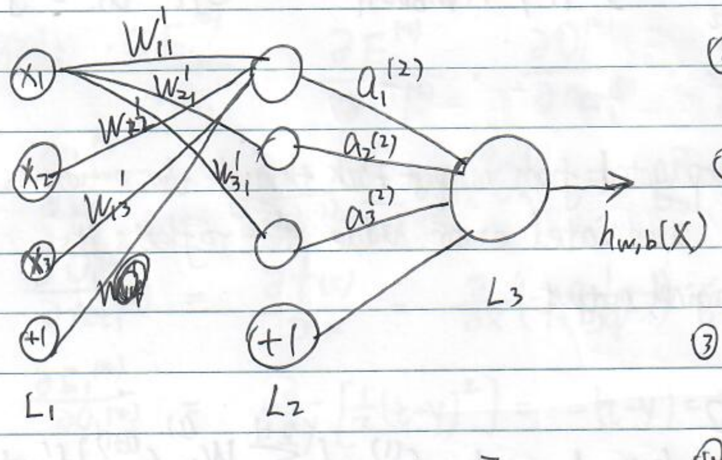
$$h_{W,b}(x) = f(W^T x) = \sum_{i=1}^{3} W_i x_i + b$$

sigmoid: $f(z) = \dfrac{1}{1+\exp(-z)}$ $\quad f'(z) = f(z)(1-f(z))$

tanh: $f(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$

$$a_2^{(2)} = f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)})$$

n_l: number of layers



① $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$

② $W_{ij}^{(l)}$: weight between unit j in layer l and unit i in layer l+1

③ $b_i^{(l)}$: bias associated with unit i in layer l+1.

④ S_l: the number of nodes in layer l (not counting bias unit)

$$W^{(1)}: \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix}$$

⑤ $a_i^{(l)}$: activation of unit i in layer l.

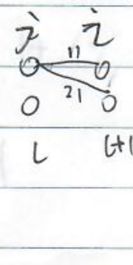
$a_i^{(l)} = f(z_i^{(l)})$ $\quad a_1^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)})$

⑥ $z_i^{(l)}$: total weighted sum of inputs to unit i in layer l.

Compact notation: organizing in matrix

$$\left.\begin{aligned} z^{(2)} &= W^{(1)}x + b^{(1)} \quad (a^{(1)} = x) \\ a^{(2)} &= f(z^{(2)}) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ h_{W,b}(x) &= a^{(3)} = f(z^{(3)}) \end{aligned}\right\}$$ forward-propagation

S.E $$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$



$$\begin{cases} W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \dfrac{\partial}{\partial W_{ij}^{(l)}} J(W,b) \\ b_i^{(l)} = b_i^{(l)} - \alpha \dfrac{\partial}{\partial b_i^{(l)}} J(W,b) \end{cases}$$

如何计算以上2个偏导数? BP 算法:

- ~~$\delta_i^{(l)}$~~ $\delta_i^{(l)}$: error term for each node i in layer l
- set $\delta_i^{(n_l)} = \dfrac{\partial}{\partial z_i^{(n_l)}} \dfrac{1}{2}\|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$

  output layer unit i

The error values are propagated from output back through the network, until each neuron has an associated error value that reflects its contribution to the original output.

- For $l = n_l-1, n_l-2, \cdots 2$

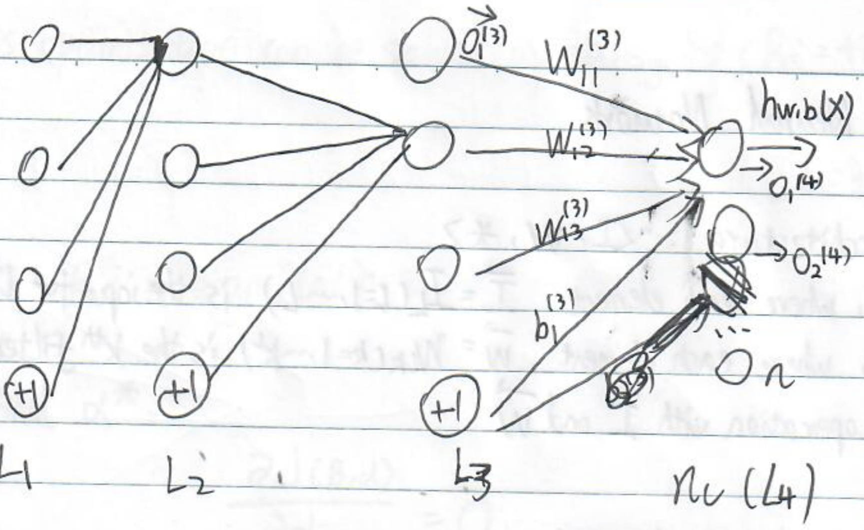
  For each node i in layer l, set $\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)}\right) f'(z_i^{(l)})$
- Desired Partial Derivatives

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)}$$

No.

Date



假设 fully connected，论文中未考虑 bias

Step 1: Forward Propagation，产生了一个输出值 $h_{w,b}(x)$ (predicted)

Step 2: Backward Propagation:

Suppose M.S.E: ~~$E = \frac{1}{2}(h_{w,b}(x) - y)^2$~~ $E = \frac{1}{2}(t-y)^2$

{ j th neuron in L3
{ i th neuron in L4

$$\frac{\partial E}{\partial W_{ij}^{(3)}} = \frac{\partial E}{\partial a_j^{(4)}} \cdot \frac{\partial a_j^{(4)}}{\partial z^{(3)}} \cdot \frac{\partial z_j^{(3)}}{\partial W_{ij}^{(3)}}$$

$a^{(4)} = f(z^{(3)})$

$f(x) = \frac{1}{1+\exp^{-x}}$

$z^{(3)} = W_{11}^{(3)} \cdot x_1^{(3)} + W_{12}^{(3)} \cdot x_2^{(3)} + W_{13}^{(3)} \cdot x_3^{(3)}$

(chain Rule) $$= \frac{\partial E}{\partial O_j^{(4)}} \cdot \frac{\partial O_j^{(4)}}{\partial net_j^{(3)}} \cdot \frac{\partial net_j^{(3)}}{\partial w_{ij}^{(3)}}$$

~~(1) $\frac{\partial net_j^{(3)}}{\partial w_{ij}^{(3)}} = \frac{\partial}{\partial w_{ij}^{(3)}}\left(\sum_{k=1}^{n} W_{kj} \cdot x_k\right) \xrightarrow{\text{only one sum}} \frac{\partial}{\partial W_{ij}^{(3)}} W$~~

具体举例: $$\frac{\partial E_1^{(4)}}{\partial W_{11}^{(3)}} = \frac{\partial E_1^{(4)}}{\partial O_1^{(4)}} \cdot \frac{\partial O_1^{(4)}}{\partial net_1^{(4)}} \cdot \frac{\partial net_1^{(4)}}{\partial W_{11}^{(3)}}$$ (输出层 L → L-1 层)

(1): $$\frac{\partial net_1^{(4)}}{\partial W_{11}^{(3)}} = \frac{\partial}{\partial W_{11}^{(3)}} \sum_{k=1}^{n} W_{1k}^{(3)} O_k^{(3)} = \frac{\partial}{\partial W_{11}^{(3)}} (W_{11}^{(3)} O_1^{(3)}) = O_1^{(3)}$$

(2) $$\frac{\partial O_1^{(4)}}{\partial net_1^{(4)}} = \frac{\partial f(x)}{\partial x} = \frac{\partial}{\partial x}\left(\frac{1}{1+\exp^{-x}}\right) = O_1^{(4)}(1 - O_1^{(4)})$$ $f(x) \rightarrow$ sigmoid

(3) $$\frac{\partial E_1^{(4)}}{\partial O_1^{(4)}} = \frac{\partial}{\partial y}\left[\frac{1}{2}(t-y)^2\right] = -(t-y) = y - t = O_1^{(4)} - t_1^{(4)}$$

So $$\frac{\partial E_1^{(4)}}{\partial W_{11}^{(3)}} = [O_1^{(4)} - t_1^{(4)}] \cdot O_1^{(4)} \cdot (1 - O_1^{(4)}) \cdot O_1^{(3)} = \delta_1^{(3)} \cdot O_1^{(3)}$$ error term

如果是在内层: $$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)}\right) f'(z_i^{(l)})$$

No.

Date

Paper:

① Binary Convolutional Neural Network

- an L-layer CNN Architecture: $\langle I, W, * \rangle$

  I: set of tensors, where each element $\vec{I} = I_l (l=1,\cdots L)$ is the input for l^{th} laye

  W: set of tensors, where each element $\vec{W} = W_{lk} (k=1,\cdots k^l)$ is the k^{th} filter in l

  $*$: A convolutional operation with $\vec{I}$ and $\vec{W}$

BWN: elements of W are binary tensors

XNOR-Network: elements of ~~XNOR-Network~~ both I and W are binary tensors.

引入 Binary Filter $B \in \{+1, -1\}^{c\times w\times h}$, scaling factor α such that $W \approx \alpha B$

$$I * W \approx (I \oplus B)\alpha$$ $\oplus$ indicates a conv without 乘法

BWN: $\langle I, B, A, \oplus \rangle$ $\begin{cases} A: \text{a set of positive scalars} \\ B: \text{a set of binary tensors} \end{cases}$ $(W_{lk} = A_{lk} B_{lk})$

- Estimating binary weights:

optimization $J(B,\alpha) = \|W - \alpha B\|^2$

$$= \alpha^2 B^T B - 2\alpha W^T B + W^T W$$

$$\alpha^*, B^* = \underset{\alpha, B}{\operatorname{argmin}} J(B,\alpha)$$ → 函数J()取得最小值的α,B的集合

$B^T B$ and $W^T W$ is constant.

So $J(B,\alpha) = \alpha^2 n - 2\alpha W^T B + C$

optimization becomes:

$$\Rightarrow \quad B^* = \underset{B}{\operatorname{argmax}} \{W^T B\} \quad \text{s.t. } B \in \{+1, -1\}^n$$

No.

Date

This optimization can be solved by assigning $\begin{cases} B_i = +1 & \cancel{\text{for}} \text{ if } W_i \geq 0 \\ B_i = -1 & \text{if } W_i < 0 \end{cases}$

$\Rightarrow \quad B^* = \text{sign}(W)$

To find α^*:

$$\frac{\partial J(B, \alpha)}{\partial \alpha} = 0$$

$$2\alpha n - 2W^T B = 0$$

$$\alpha n = W^T B$$

$$\alpha = \frac{W^T B}{n}$$

So $\alpha^* = \dfrac{W^T B^*}{n} = \dfrac{W^T \text{sign}(W)}{n} = \dfrac{\Sigma |W_i|}{n}$

Training:

weight update methods: SGD or ADam