Deep Learning HW1

Yuanhe Guo

February 2024

1 Theory

1.1 Two-Layer Neural Nets

1.1.1 Regression Task

(a) 5 Steps:

$\tilde{\text{y}}$ = model(x) ; `// generate a prediction`
L = F = C($\tilde{\text{y}}$, y) ; `// compute the loss`
optimiser.zero_grad() ; `// zero ∇params`
L.backward() ; `// compute&accumulate ∇params`
optimizer.step() ; `// step in towards -∇params`

(b)
- layer1: *input* $= \boldsymbol{x}$
  output $= f(\boldsymbol{W^{(1)}}\boldsymbol{x} + \boldsymbol{b^{(1)}}) = 3\texttt{ReLU}(\boldsymbol{W^{(1)}}\boldsymbol{x} + \boldsymbol{b^{(1)}})$
- layer2: *input* $= 3\texttt{ReLU}(\boldsymbol{W^{(1)}}\boldsymbol{x} + \boldsymbol{b^{(1)}})$
  output $= g(\boldsymbol{W^{(2)}}3\texttt{ReLU}(\boldsymbol{W^{(1)}}\boldsymbol{x}+\boldsymbol{b^{(1)}})+\boldsymbol{b^{(2)}}) = \boldsymbol{W^{(2)}}3\texttt{ReLU}(\boldsymbol{W^{(1)}}\boldsymbol{x}+\boldsymbol{b^{(1)}}) + \boldsymbol{b^{(2)}}$

(c)
- $\frac{\partial C}{\partial \boldsymbol{W^{(2)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\frac{\partial \boldsymbol{s_2}}{\partial \boldsymbol{W^{(2)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\boldsymbol{a_1} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}3\texttt{ReLU}(\boldsymbol{W^{(1)}}\boldsymbol{x} + \boldsymbol{b^{(1)}})$
- $\frac{\partial C}{\partial \boldsymbol{b^{(2)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\frac{\partial \boldsymbol{s_2}}{\partial \boldsymbol{b^{(2)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}$
- $\frac{\partial C}{\partial \boldsymbol{W^{(1)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\frac{\partial \boldsymbol{s_2}}{\partial \boldsymbol{a_1}}\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{s_1}}\frac{\partial \boldsymbol{s_1}}{\partial \boldsymbol{W^{(1)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\boldsymbol{W^{(2)}}\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{s_1}}\boldsymbol{x}$
- $\frac{\partial C}{\partial \boldsymbol{b^{(1)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\frac{\partial \boldsymbol{s_2}}{\partial \boldsymbol{a_1}}\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{s_1}}\frac{\partial \boldsymbol{s_1}}{\partial \boldsymbol{b^{(1)}}} = \frac{\partial C}{\partial \boldsymbol{\tilde{y}}}\frac{\partial \boldsymbol{\tilde{y}}}{\partial \boldsymbol{s_2}}\boldsymbol{W^{(2)}}\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{s_1}}$

(d)
- $\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{s_1}} = \begin{bmatrix} \frac{\partial \boldsymbol{a^1_{11}}}{\partial \boldsymbol{s^1_{11}}} & \frac{\partial \boldsymbol{a^1_{12}}}{\partial \boldsymbol{s^1_{12}}} & \cdots & \frac{\partial \boldsymbol{a^1_{1j}}}{\partial \boldsymbol{s^1_{1j}}} \\ \frac{\partial \boldsymbol{a^1_{21}}}{\partial \boldsymbol{s^1_{21}}} & \frac{\partial \boldsymbol{a^1_{22}}}{\partial \boldsymbol{s^1_{22}}} & \cdots & \frac{\partial \boldsymbol{a^1_{2j}}}{\partial \boldsymbol{s^1_{2j}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{a^1_{i1}}}{\partial \boldsymbol{s^1_{i1}}} & \frac{\partial \boldsymbol{a^1_{i2}}}{\partial \boldsymbol{s^1_{i2}}} & \cdots & \frac{\partial \boldsymbol{a^1_{ij}}}{\partial \boldsymbol{s^1_{ij}}} \end{bmatrix}$

$$a_{ij}^1 = \begin{cases} 3 & \text{if } s_{ij}^1 > 0 \\ 0 & \text{if } s_{ij}^1 < 0 \end{cases}$$

- $\frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{s_2}} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$
- $\frac{\partial C}{\partial \tilde{\boldsymbol{y}}} = (2\tilde{\boldsymbol{y}} - 2\boldsymbol{y})^T$

1.1.2 Classification Task

(a)
- For (b), the equations will substitute f and g with other functions, resulting in layer1 output and layer2 input to be $\texttt{tanh}(\boldsymbol{W^{(1)}x}+\boldsymbol{b^{(1)}})$, layer2 output to be $(1+\texttt{exp}(-(\boldsymbol{W^{(2)}}\texttt{tanh}(\boldsymbol{W^{(1)}x}+\boldsymbol{b^{(1)}})+\boldsymbol{b^{(2)}})))^{-1}$
- For (c), we don't need to change any thing, but the value inside parameters may change accordingly.
- For (d),

$$\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{s_1}} = \begin{bmatrix} 1-tanh^2(\boldsymbol{s_{11}^1}) & 1-tanh^2(\boldsymbol{s_{12}^1}) & \dots & 1-tanh^2(\boldsymbol{s_{1j}^1}) \\ 1-tanh^2(\boldsymbol{s_{21}^1}) & 1-tanh^2(\boldsymbol{s_{22}^1}) & \dots & 1-tanh^2(\boldsymbol{s_{2j}^1}) \\ \vdots & \vdots & \ddots & \vdots \\ 1-tanh^2(\boldsymbol{s_{i1}^1}) & 1-tanh^2(\boldsymbol{s_{i2}^1}) & \dots & 1-tanh^2(\boldsymbol{s_{ij}^1}) \end{bmatrix}$$

$$\frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{s_2}} = \begin{bmatrix} \sigma(\boldsymbol{s_{11}^2})\cdot(1-\sigma(\boldsymbol{s_{11}^2})) & \sigma(\boldsymbol{s_{12}^2})\cdot(1-\sigma(\boldsymbol{s_{12}^2})) & \dots & \sigma(\boldsymbol{s_{1j}^2})\cdot(1-\sigma(\boldsymbol{s_{1j}^2})) \\ \sigma(\boldsymbol{s_{21}^2})\cdot(1-\sigma(\boldsymbol{s_{21}^2})) & \sigma(\boldsymbol{s_{22}^2})\cdot(1-\sigma(\boldsymbol{s_{22}^2})) & \dots & \sigma(\boldsymbol{s_{2j}^2})\cdot(1-\sigma(\boldsymbol{s_{2j}^2})) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(\boldsymbol{s_{i1}^2})\cdot(1-\sigma(\boldsymbol{s_{i1}^2})) & \sigma(\boldsymbol{s_{i2}^2})\cdot(1-\sigma(\boldsymbol{s_{i2}^2})) & \dots & \sigma(\boldsymbol{s_{ij}^2})\cdot(1-\sigma(\boldsymbol{s_{ij}^2})) \end{bmatrix}$$

$\frac{\partial C}{\partial \tilde{\boldsymbol{y}}}$ will not change.

(b) (a) (b) and (c) will be the same as 1.2(a).
(b) For (d), $\frac{\partial \boldsymbol{a_1}}{\boldsymbol{s_1}}$, $\frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{s_2}}$ will not change.
$\frac{\partial C}{\partial \tilde{\boldsymbol{y}}} = -\frac{1}{K}(\frac{\boldsymbol{y}}{\tilde{\boldsymbol{y}}} - \frac{1-\boldsymbol{y}}{1-\tilde{\boldsymbol{y}}})$

(c) Compared with `tanh`, `ReLU` function is easier to derive while maintaining non-linearity. When training a deep network, doing back propagation will be faster. Meanwhile, since `ReLU` won't saturate correct predictions, vanishing gradient is less likely to happen in a deep network.

1.2 Conceptual Questions

(a) Softmax function outputs a probability distribution controlled by variable β aka "coldness". When $\beta \rightarrow \infty$, the output will become an argmax function. Therefore softmax is actually softargmax.

(b) See fig. 1

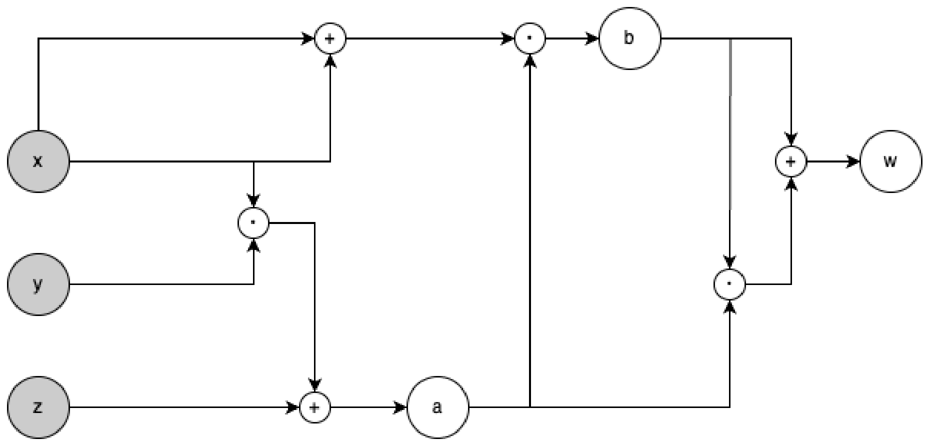

Figure 1: Computational Graph

(c) See fig. 2

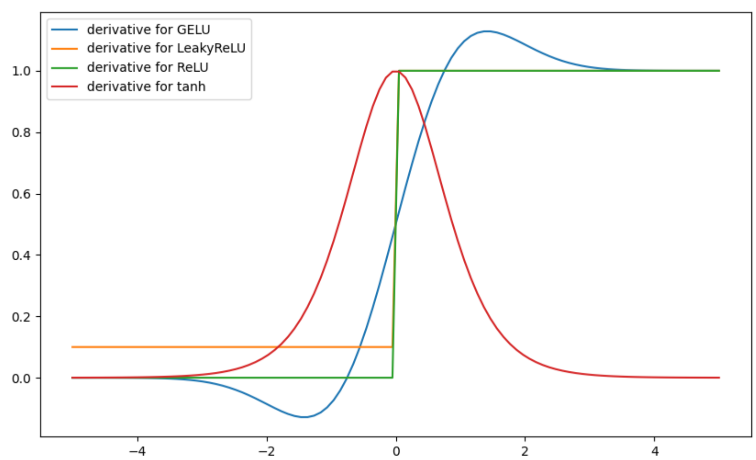

Figure 2: Derivatives for GELU, LeakyReLU, ReLU, tanh

(d) (a) $J_f = \frac{\partial \boldsymbol{W_1 x}}{\partial \boldsymbol{x}} = \boldsymbol{W_1}$
$J_g = \frac{\partial \boldsymbol{W_2 x}}{\partial \boldsymbol{x}} = \boldsymbol{W_2}$

(b) $J_h = \frac{\partial f(\boldsymbol{x}) + g(\boldsymbol{x})}{\boldsymbol{x}} = J_f + J_g = \boldsymbol{W_1} + \boldsymbol{W_2}$

(c) $J_h = \boldsymbol{W_1} + \boldsymbol{W_2} = \boldsymbol{2W_1} = \boldsymbol{2W_2}$

(e) (a) $J_f = \frac{\partial \boldsymbol{W_1 x}}{\partial \boldsymbol{x}} = \boldsymbol{W_1}$
$J_g = \frac{\partial \boldsymbol{W_2 x}}{\partial \boldsymbol{x}} = \boldsymbol{W_2}$

(b) $J_h = \frac{\partial g(f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})} \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{W_2 W_1}$

(c) $J_h = \boldsymbol{W_2 W_1} = \boldsymbol{W_1^2} = \boldsymbol{W_2^2}$

1.3 Deriving Loss Functions

1. For Perceptron, $\tilde{y}$ is either -1 or 1. Notice that $(y - \tilde{y}x_i)$ is always a non-positive number. By add a negative sign and summing up all cases, we get the loss function $-(y - \tilde{y})\Sigma_{i=1}^d w_i x_i$

2. For Adaline / Least Mean Squares, we want to minimize the distance between $\boldsymbol{y}$ and $\boldsymbol{\tilde{y}}$, which is $||y - \tilde{y}||_2$. For sake of mathematical convenience, we set the loss function to be $\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\tilde{y}})^2$.

3. For Logistic Regression, we want to maximize the case where $\texttt{tanh}(b + w_i x_i) = y_i$. Since y_i is either 1 or -1, and $\texttt{tanh}$ function is odd, we know the previous function is the same as

$$\begin{aligned} \texttt{tanh}(y_i(b + w_i x_i) &= 1 \\ \frac{2}{1 + \texttt{exp}(-y_i(b + w_i x_i)} &= 1 \\ 1 + \texttt{exp}(-y_i(b + w_i x_i) &= 1 \\ \texttt{log}(1 + \texttt{exp}(-y_i(b + w_i x_i)) &= 0 \end{aligned}$$

Here we can also ignore b since it won't affect the gradient. We sum up all cases, and multiply it by -2 for mathematical convience, we get the loss function $-2\texttt{log}(1 + \texttt{exp}(-y\Sigma_{i=1}^d w_i x_i)$.