THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 8**

Deep Learning 1

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Backward Propagation Algorithm

1. We have a k-class classification problem. The input $\boldsymbol{x}$ is a d-dimensional vector, and the corresponding label $\boldsymbol{y}$ is a one-hot k-dimensional vector with one element being 1 and others being 0. We define the following neural network $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2)$:

$$\boldsymbol{h}_1 = \boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1, \quad \boldsymbol{h}_1 \in \mathbb{R}^s \tag{1}$$

$$\boldsymbol{a}_1 = \mathrm{ReLU}(\boldsymbol{h}_1), \quad \boldsymbol{a}_1 \in \mathbb{R}^s \tag{2}$$

$$\boldsymbol{h}_2 = \mathrm{Concat}(\boldsymbol{a}_1, \boldsymbol{x}), \quad \boldsymbol{h}_2 \in \mathbb{R}^{s+d} \tag{3}$$

$$\boldsymbol{m} \sim \mathrm{Bernoulli}(p), \quad \boldsymbol{m} \in \mathbb{R}^{s+d} \tag{4}$$

$$\boldsymbol{a}_2 = \boldsymbol{h}_2 \odot \boldsymbol{m}, \quad \boldsymbol{a}_2 \in \mathbb{R}^{s+d} \tag{5}$$

$$\boldsymbol{h}_3 = \boldsymbol{W}_2^\top \boldsymbol{a}_2 + \boldsymbol{b}_2, \quad \boldsymbol{h}_3 \in \mathbb{R}^k \tag{6}$$

$$\hat{\boldsymbol{y}} = \mathrm{Softmax}(\boldsymbol{h}_3), \quad \hat{\boldsymbol{y}} \in \mathbb{R}^k \tag{7}$$

where $\mathrm{ReLU}(\cdot)$ is element-wise ReLU activation function, $\mathrm{Concat}(\boldsymbol{a}, \boldsymbol{b})$ means concatenating vector $\boldsymbol{b}$ after vector $\boldsymbol{a}$ to make a vector with larger dimensionality, $\boldsymbol{m}$ is current dropout mask, $\odot$ means element-wise multiplication(Specifically, equation(5) randomly zeroes some of the elements of the input tensor $\boldsymbol{h}_2$ with probability p using $\boldsymbol{m}$ sampled from a Bernoulli distribution. For example, if $\boldsymbol{h}_2 = [a, b, c, d]^\top$, and $\boldsymbol{m} = [e, f, g, h]^\top$, then $\boldsymbol{a}_2 = [ae, bf, cg, dh]^\top$ according to equation(5)), and $\mathrm{Softmax}(\cdot)$ is softmax activation function. The dimensionalities of parameters and internal variables could be inferred from the neural network definition.

We use the following cross-entropy loss:

$$L = -\boldsymbol{y}^\top \log \hat{\boldsymbol{y}}. \tag{8}$$

1.1. Please use backward propagation algorithm to find the following derivatives:

$$\frac{\partial L}{\partial \hat{\boldsymbol{y}}}, \quad \frac{\partial L}{\partial \boldsymbol{h}_3}, \quad \frac{\partial L}{\partial \boldsymbol{W}_2}, \quad \frac{\partial L}{\partial \boldsymbol{b}_2}, \quad \frac{\partial L}{\partial \boldsymbol{a}_2}, \quad \frac{\partial L}{\partial \boldsymbol{h}_2}, \quad \frac{\partial L}{\partial \boldsymbol{a}_1}, \quad \frac{\partial L}{\partial \boldsymbol{h}_1}, \quad \frac{\partial L}{\partial \boldsymbol{W}_1}, \quad \frac{\partial L}{\partial \boldsymbol{b}_1}, \quad \frac{\partial L}{\partial \boldsymbol{x}}. \tag{9}$$

Please express your results in matrices and vectors instead of sum of individual elements.

解: 原题目中 $\boldsymbol{w}_1, \boldsymbol{w}_2$ 实为矩阵, 故改用 $\boldsymbol{W}_1, \boldsymbol{W}_2$ 符号代替之. 为避免符号下标混淆, 本题中一律以右上角小括号内指标表示向量, 矩阵和张量的元素, 如 $\boldsymbol{a}^{(1)}$ 表示向量 $\boldsymbol{a}$ 的第 1 个元素, $\boldsymbol{A}^{(2,3)}$ 表示矩阵 $\boldsymbol{A}$ 的第 2 行第 3 列的元素, $\mathsf{A}^{(1,2,3)}$ 表示张量 A 的第 1 行第 2 列第 3 竖的元素.

交叉熵损失函数 L 对 $\hat{\boldsymbol{y}}$ 的偏导数为

$$\frac{\partial L}{\partial \hat{\boldsymbol{y}}} = \frac{\partial\big(-\boldsymbol{y}^\top \log \hat{\boldsymbol{y}}\big)}{\partial \hat{\boldsymbol{y}}} = -\frac{\partial \log \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{y}}}\boldsymbol{y} = -\operatorname{diag}(\mathbf{1} \oslash \hat{\boldsymbol{y}})\,\boldsymbol{y} = -\boldsymbol{y} \oslash \hat{\boldsymbol{y}} \tag{10}$$

其中, 符号 $\oslash$ 表示逐项除法 (element-wise division).

函数 $\hat{\boldsymbol{y}}$ 对 $\boldsymbol{h}_3$ 的偏导数为

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}_3} = \frac{\partial}{\partial \boldsymbol{h}_3}\left(\frac{\exp(\boldsymbol{h}_3)}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \tag{11}$$

其中, $\forall\, i, j = 1, 2, \ldots, k$, 当 $i \neq j$ 时, 有

$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{y}}^{(i)}}{\partial \boldsymbol{h}_3^{(j)}} &= \frac{\partial}{\partial \boldsymbol{h}_3^{(j)}}\left(\frac{\exp(\boldsymbol{h}_3^{(i)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \\
&= \frac{0 - \exp(\boldsymbol{h}_3^{(i)})\exp(\boldsymbol{h}_3^{(j)})}{[\mathbf{1}^\top \exp(\boldsymbol{h}_3)]^2} \\
&= -\frac{\exp(\boldsymbol{h}_3^{(i)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)} \\
&= -\hat{\boldsymbol{y}}^{(i)}\hat{\boldsymbol{y}}^{(j)}
\end{aligned} \tag{12}$$

当 $i = j$ 时, 有

$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{y}}^{(j)}}{\partial \boldsymbol{h}_3^{(j)}} &= \frac{\partial}{\partial \boldsymbol{h}_3^{(j)}}\left(\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \\
&= \frac{\exp(\boldsymbol{h}_3^{(j)})\mathbf{1}^\top \exp(\boldsymbol{h}_3) - \exp(\boldsymbol{h}_3^{(j)})\exp(\boldsymbol{h}_3^{(j)})}{[\mathbf{1}^\top \exp(\boldsymbol{h}_3)]^2} \\
&= -\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)} + \frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)} \\
&= -\hat{\boldsymbol{y}}^{(j)}\hat{\boldsymbol{y}}^{(j)} + \hat{\boldsymbol{y}}^{(j)}
\end{aligned} \tag{13}$$

综上, $\hat{\boldsymbol{y}}$ 对 $\boldsymbol{h}_3$ 的偏导数为

$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}_3} &= \frac{\partial}{\partial \boldsymbol{h}_3}\left(\frac{\exp(\boldsymbol{h}_3)}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \\
&= -\hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^\top + \operatorname{diag}(\hat{\boldsymbol{y}})
\end{aligned} \tag{14}$$

注意到 $\boldsymbol{y}$ 是 one-hot 向量, 则交叉熵损失函数 L 对 $\boldsymbol{h}_3$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{h}_3} &= \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}_3}\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\ &= [-\hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^\top + \operatorname{diag}(\hat{\boldsymbol{y}})](-\boldsymbol{y} \oslash \hat{\boldsymbol{y}}) \\ &= \hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^\top(\boldsymbol{y} \oslash \hat{\boldsymbol{y}}) - \operatorname{diag}(\hat{\boldsymbol{y}})(\boldsymbol{y} \oslash \hat{\boldsymbol{y}}) \\ &= \hat{\boldsymbol{y}}\sum_{i=1}^{k}\boldsymbol{y}^{(i)} - \boldsymbol{y} \\ &= \hat{\boldsymbol{y}} - \boldsymbol{y}\end{aligned} \tag{15}$$

函数 $\boldsymbol{h}_3$ 对 $\boldsymbol{W}_2$ 的偏导数为 3 阶张量 $\mathsf{W} \in \mathbb{R}^{(s+d)\times k\times k}$, 其元素为

$$\mathsf{W}^{(m,n,i)} = \frac{\partial \boldsymbol{h}_3^{(i)}}{\partial \boldsymbol{W}_2^{(m,n)}} = \frac{\partial\left[(\boldsymbol{W}_2^\top)^{(i,:)}\boldsymbol{a}_2 + \boldsymbol{b}_2^{(i)}\right]}{\partial \boldsymbol{W}_2^{(m,n)}} = \delta_{in}\boldsymbol{a}_2^{(m)}, \quad \forall\ i,n = 1,2,\ldots,k,\ m = 1,2,\ldots,s+d \tag{16}$$

其中, δ_{in} 为 Kronecker delta 符号.

故交叉熵损失函数 L 对 $\boldsymbol{W}_2^{(m,n)}$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}_2^{(m,n)}} &= \sum_{i=1}^{k}\frac{\partial \boldsymbol{h}_3^{(i)}}{\partial \boldsymbol{W}_2^{(m,n)}}\frac{\partial L}{\partial \boldsymbol{h}_3^{(i)}} \\ &= \sum_{i=1}^{k}\delta_{in}\boldsymbol{a}_2^{(m)}\left[\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)}\right] \\ &= \boldsymbol{a}_2^{(m)}\left[\hat{\boldsymbol{y}}^{(n)} - \boldsymbol{y}^{(n)}\right]\end{aligned} \tag{17}$$

所以, 交叉熵损失函数 L 对 $\boldsymbol{W}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_2} = \boldsymbol{a}_2(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top \tag{18}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_2} = \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{b}_2}\frac{\partial L}{\partial \boldsymbol{h}_3} = \frac{\partial(\boldsymbol{W}_2^\top\boldsymbol{a}_2 + \boldsymbol{b}_2)}{\partial \boldsymbol{b}_2}\frac{\partial L}{\partial \boldsymbol{h}_3} = \boldsymbol{I}(\hat{\boldsymbol{y}} - \boldsymbol{y}) = \hat{\boldsymbol{y}} - \boldsymbol{y} \tag{19}$$

交叉熵损失函数 L 对 $\boldsymbol{a}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{a}_2} = \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{a}_2}\frac{\partial L}{\partial \boldsymbol{h}_3} = \boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{20}$$

交叉熵损失函数 L 对 $\boldsymbol{h}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{h}_2} = \frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}_2}\frac{\partial L}{\partial \boldsymbol{a}_2} = \frac{\partial(\boldsymbol{h}_2 \odot \boldsymbol{m})}{\partial \boldsymbol{h}_2}\frac{\partial L}{\partial \boldsymbol{a}_2} = \operatorname{diag}(\boldsymbol{m})\boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) = \boldsymbol{m} \odot [\boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y})] \tag{21}$$

交叉熵损失函数 L 对 $\boldsymbol{a}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{a}_1} = \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{a}_1}\frac{\partial L}{\partial \boldsymbol{h}_2} = \frac{\partial \begin{bmatrix}\boldsymbol{a}_1 \\ \boldsymbol{x}\end{bmatrix}}{\partial \boldsymbol{a}_1}\frac{\partial L}{\partial \boldsymbol{h}_2} = [\boldsymbol{I}_{s\times s} \quad \boldsymbol{0}_{s\times d}]\,\mathrm{diag}(\boldsymbol{m})\boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{22}$$

交叉熵损失函数 L 对 $\boldsymbol{h}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\frac{\partial L}{\partial \boldsymbol{a}_1} = \frac{\partial\ \mathrm{ReLU}(\boldsymbol{h}_1)}{\partial \boldsymbol{h}_1}\frac{\partial L}{\partial \boldsymbol{a}_1} = \mathrm{diag}\left(\frac{\boldsymbol{1} + \mathrm{sgn}(\boldsymbol{h}_1)}{2}\right)\frac{\partial L}{\partial \boldsymbol{a}_1} \tag{23}$$

函数 $\boldsymbol{h}_1$ 对 $\boldsymbol{W}_1$ 的偏导数为

$$\frac{\partial \boldsymbol{h}_1^{(i)}}{\partial \boldsymbol{W}_1^{(m,n)}} = \frac{\partial\left[(\boldsymbol{W}_1^\top)^{(i,:)}\boldsymbol{x} + \boldsymbol{b}_1^{(i)}\right]}{\partial \boldsymbol{W}_1^{(m,n)}} = \delta_{in}\boldsymbol{x}^{(m)}, \quad \forall\ i, n = 1, 2, \ldots, s,\ m = 1, 2, \ldots, d \tag{24}$$

故交叉熵损失函数 L 对 $\boldsymbol{W}_1^{(m,n)}$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}_1^{(m,n)}} &= \sum_{i=1}^{s}\frac{\partial \boldsymbol{h}_1^{(i)}}{\partial \boldsymbol{W}_1^{(m,n)}}\frac{\partial L}{\partial \boldsymbol{h}_1^{(i)}} \\ &= \sum_{i=1}^{s}\frac{\partial L}{\partial \boldsymbol{h}_1^{(i)}}\delta_{in}\boldsymbol{x}^{(m)} \\ &= \boldsymbol{x}^{(m)}\frac{\partial L}{\partial \boldsymbol{h}_1^{(n)}}\end{aligned} \tag{25}$$

所以, 交叉熵损失函数 L 对 $\boldsymbol{W}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_1} = \boldsymbol{x}\left(\frac{\partial L}{\partial \boldsymbol{h}_1}\right)^\top \tag{26}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_1} = \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{b}_1}\frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial(\boldsymbol{W}_1^\top\boldsymbol{x} + \boldsymbol{b}_1)}{\partial \boldsymbol{b}_1}\frac{\partial L}{\partial \boldsymbol{h}_1} = \boldsymbol{I}\frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial L}{\partial \boldsymbol{h}_1} \tag{27}$$

交叉熵损失函数 L 对 $\boldsymbol{x}$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{x}} &= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_1} + \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_2} \\ &= \frac{\partial(\boldsymbol{W}_1^\top\boldsymbol{x} + \boldsymbol{b}_1)}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_1} + \frac{\partial \begin{bmatrix}\boldsymbol{a}_1 \\ \boldsymbol{x}\end{bmatrix}}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_2} \\ &= \boldsymbol{W}_1\frac{\partial L}{\partial \boldsymbol{h}_1} + [\boldsymbol{0}_{d\times s} \quad \boldsymbol{I}_{d\times d}]\frac{\partial L}{\partial \boldsymbol{h}_2}\end{aligned} \tag{28}$$

1.2. Derive the back-propagation updates for convolutional neural networks: In this situation, the input $\boldsymbol{x}$ is an image with size $C_{\text{in}} \times H \times W$, where the three dimensions represent channel, height and width respectively. We define the following simple convolutional neural network:

$$\boldsymbol{u}_1 = \text{Conv2d}(C_{\text{in}}, C_{\text{out}}, k)(\boldsymbol{x}) \tag{29}$$

$$\boldsymbol{h}_1 = \text{MaxPool2d}(N)(\boldsymbol{u}_1) \tag{30}$$

$$\boldsymbol{a}_1 = \text{ReLU}(\boldsymbol{h}_1) \tag{31}$$

$$\boldsymbol{u}_2 = \text{Flatten}(\boldsymbol{a}_1) \tag{32}$$

$$\boldsymbol{h}_2 = \boldsymbol{W}_2^\top \boldsymbol{u}_2 + \boldsymbol{b}_2 \tag{33}$$

$$\hat{\boldsymbol{y}} = \text{Softmax}(\boldsymbol{h}_2) \tag{34}$$

where Conv2d (There are learnable parameters $\boldsymbol{W}_1, \boldsymbol{b}_1$ in this function. See this link for detailed definitions of the function arguments and this link for visualization) and MaxPool2d are convolutional function and pooling function defined in a PyTorch style, Flatten means reshaping the input vector into a one-dimensional vector, and the final loss function is the same with Eq. (8). What is the derivatives for the network parameters (i.e. convolutional and linear weights and biases)?

Hint: Read the following reference material: Notes on Convolutional Neural Networks.

解: 首先确定各个变量的维数, 输入图片 $\boldsymbol{x} \in \mathbb{R}^{C_{\text{in}} \times H \times W}$, Conv2d 函数输出 $\boldsymbol{u}_1 \in \mathbb{R}^{C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}}$, 其中

$$H_{\text{out}} = H - k + 1, \quad W_{\text{out}} = W - k + 1 \tag{35}$$

MaxPool2d 函数输出 $\boldsymbol{h}_1 \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, 其中

$$H_{\text{mp}} = \left\lfloor \frac{H_{\text{out}}}{N} \right\rfloor, \quad W_{\text{mp}} = \left\lfloor \frac{W_{\text{out}}}{N} \right\rfloor \tag{36}$$

ReLU 函数输出 $\boldsymbol{a}_1 \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, Flatten 函数输出 $\boldsymbol{u}_2 \in \mathbb{R}^d$, 其中

$$d = C_{\text{out}} + H_{\text{mp}} + W_{\text{mp}} \tag{37}$$

设 $\boldsymbol{W}_2 \in \mathbb{R}^{d \times s}$, 则 $\boldsymbol{b}_2 \in \mathbb{R}^s$, $\boldsymbol{h}_2 \in \mathbb{R}^s$, $\hat{\boldsymbol{y}} \in \mathbb{R}^s$.

由 1.1 题可知, 交叉熵损失函数 L 对 $\boldsymbol{W}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_2} = \boldsymbol{a}_2(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top \tag{38}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_2} = \hat{\boldsymbol{y}} - \boldsymbol{y} \tag{39}$$

交叉熵损失函数 L 对 $\boldsymbol{u}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{u}_2} = \boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{40}$$

函数 $\boldsymbol{u}_2$ 对 $\boldsymbol{a}_1$ 的偏导数为 4 阶张量 $\mathsf{A} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}} \times d}$, 其元素为

$$\mathsf{A}^{(i,j,k,m)} = \frac{\partial \boldsymbol{u}_2^{(m)}}{\partial \boldsymbol{a}_1^{(i,j,k)}} = \frac{\partial [\text{Flatten}(\boldsymbol{a}_1)]^{(m)}}{\partial \boldsymbol{a}_1^{(i,j,k)}} = \delta_{mn(i,j,k)} \tag{41}$$

其中 $n(\cdot,\cdot,\cdot)$ 是三元函数,

$$n(i,j,k) \triangleq (i-1)H_{\text{mp}}W_{\text{mp}} + (j-1)W_{\text{mp}} + k \tag{42}$$

交叉熵损失函数 L 对 $\boldsymbol{a}_1$ 的偏导数为

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{a}_1^{(i,j,k)}} &= \sum_{m=1}^{d} \frac{\partial \boldsymbol{u}_2^{(m)}}{\partial \boldsymbol{a}_1^{(i,j,k)}} \frac{\partial L}{\partial \boldsymbol{u}_2^{(m)}} \\ &= \sum_{m=1}^{d} \delta_{mn(i,j,k)} \boldsymbol{W}_2^{(m,:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \\ &= \boldsymbol{W}_2^{(n(i,j,k),:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \end{aligned} \tag{43}$$

函数 $\boldsymbol{a}_1$ 对 $\boldsymbol{h}_1$ 的偏导数为 6 阶张量 $\mathsf{H} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}} \times C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, 其元素为

$$\mathsf{H}^{(r,s,t,i,j,k)} = \frac{\partial \boldsymbol{a}_1^{(i,j,k)}}{\partial \boldsymbol{h}_1^{(r,s,t)}} = \frac{\partial \ \text{ReLU}(\boldsymbol{h}_1^{(i,j,k)})}{\partial \boldsymbol{h}_1^{(r,s,t)}} = \delta_{ir}\delta_{js}\delta_{kt} \frac{1 + \text{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \tag{44}$$

交叉熵损失函数 L 对 $\boldsymbol{h}_1$ 的偏导数为

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{h}_1^{(r,s,t)}} &= \sum_{i=1}^{C_{\text{out}}} \sum_{j=1}^{H_{\text{mp}}} \sum_{k=1}^{W_{\text{mp}}} \frac{\partial \boldsymbol{a}_1^{(i,j,k)}}{\partial \boldsymbol{h}_1^{(r,s,t)}} \frac{\partial L}{\partial \boldsymbol{a}_1^{(i,j,k)}} \\ &= \sum_{i=1}^{C_{\text{out}}} \sum_{j=1}^{H_{\text{mp}}} \sum_{k=1}^{W_{\text{mp}}} \delta_{ir}\delta_{js}\delta_{kt} \frac{1 + \text{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \boldsymbol{W}_2^{(n(i,j,k),:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \\ &= \frac{1 + \text{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \boldsymbol{W}_2^{(n(r,s,t),:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \end{aligned} \tag{45}$$

函数 $\boldsymbol{h}_1$ 对 $\boldsymbol{u}_1$ 的偏导数为 6 阶张量 $\mathsf{U} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}} \times C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, 其元素为

$$\begin{aligned} \mathsf{U}^{(p,q,l,r,s,t)} &= \frac{\partial \boldsymbol{h}_1^{(r,s,t)}}{\partial \boldsymbol{u}_1^{(p,q,l)}} \\ &= \frac{\partial \ \text{MaxPool2d}(\boldsymbol{h}_1^{(r,s,t)})}{\partial \boldsymbol{u}_1^{(p,q,l)}} \\ &= \begin{cases} \delta_{\boldsymbol{h}_1^{(r,s,t)}) \boldsymbol{u}_1^{(p,q,l)}}, & \text{if } \ p-1 < \dfrac{r}{N} \leqslant p, \ q-1 < \dfrac{s}{N} \leqslant q, \ l-1 < \dfrac{t}{N} \leqslant l \\ 0, & \text{o.w.} \end{cases} \end{aligned} \tag{46}$$

交叉熵损失函数 L 对 $\boldsymbol{u}_1$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{u}_1^{(p,q,l)}} &= \sum_{r=1}^{C_{\text{out}}}\sum_{s=1}^{H_{\text{mp}}}\sum_{t=1}^{W_{\text{mp}}} \frac{\partial \boldsymbol{h}_1^{(r,s,t)}}{\partial \boldsymbol{u}_1^{(p,q,l)}} \frac{\partial L}{\partial \boldsymbol{h}_1^{(r,s,t)}} \\ &= \sum_{r=1}^{C_{\text{out}}}\sum_{s=1}^{H_{\text{mp}}}\sum_{t=1}^{W_{\text{mp}}} \frac{\partial \boldsymbol{h}_1^{(r,s,t)}}{\partial \boldsymbol{u}_1^{(p,q,l)}} \frac{1+\operatorname{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \boldsymbol{W}_2^{(n(r,s,t),:)}(\hat{\boldsymbol{y}}-\boldsymbol{y})\end{aligned} \tag{47}$$

由函数 Conv2d 定义可知

$$\boldsymbol{u}_1^{(j,:,:)} = \boldsymbol{b}_1^{(j,:,:)} + \sum_{k=1}^{C_{\text{in}}} \boldsymbol{W}_1^{(j,k,:,:)} \star \boldsymbol{x}^{(k,:,:)} \tag{48}$$

其中, $\star$ 为二维互相关 (cross-correlation) 运算符号,

$$\boldsymbol{b}_1 \in \mathbb{R}^{C_{\text{out}}\times H_{\text{out}}\times W_{\text{out}}}, \quad \boldsymbol{W}_1 \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times k\times k} \tag{49}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_1} = \frac{\partial L}{\partial \boldsymbol{u}_1} \tag{50}$$

函数 $\boldsymbol{u}_1$ 对 $\boldsymbol{W}_1$ 的偏导数为 7 阶张量 $\mathsf{W} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times k\times k\times C_{\text{out}}\times H_{\text{out}}\times W_{\text{out}}}$, 难以具体表出, 根据参考文献 [2] 可知可以使用 MATLAB 内置函数直接计算交叉熵损失函数 L 对 $\boldsymbol{W}_1$ 的偏导数.

交叉熵损失函数 L 对 $\boldsymbol{W}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_1^{(m,n,:,:)}} = \text{rot180}\left\{\text{conv2}\left[\boldsymbol{x}^{(n,:,:)},\ \text{rot180}\left(\frac{\partial L}{\partial \boldsymbol{u}_1^{(m,:,:)}}\right),\ \text{'valid'}\right]\right\} \tag{51}$$

其中, rot180 和 conv2 均为 MATLAB 中的函数.

Vanishing and Exploding Gradients

2. In this section we will investigate the vanishing/exploding gradient problem and see why ReLU and ResNet can mitigate this problem. The vanishing/exploding gradient problem often occurs during training very deep neural networks and could make the training process failed.

Suppose the input $\boldsymbol{x}$ and output $\hat{\boldsymbol{y}}$ are both d-dimensional vectors, and the gradient of loss w.r.t. $\hat{\boldsymbol{y}}$: $\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \in \mathbb{R}^d$ is known. Suppose the deep feed-forward neural network has l layers (i.e., $l > 100$), and all internal nodes are also d-dimensional. All weights are initialized with zero-mean Gaussian distribution and all biases are initialized to zero. We now analyze the gradients in the first optimization iteration.

Effect of ReLU. The neural network is defined by:

$$\begin{aligned} \boldsymbol{a}_1 &= \text{Sigmoid}(\boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1) \\ \boldsymbol{a}_2 &= \text{Sigmoid}(\boldsymbol{W}_2^\top \boldsymbol{a}_1 + \boldsymbol{b}_2) \\ &\vdots \\ \hat{\boldsymbol{y}} = \boldsymbol{a}_l &= \text{Sigmoid}(\boldsymbol{W}_l^\top \boldsymbol{a}_{l-1} + \boldsymbol{b}_l) \end{aligned} \tag{52}$$

2.1. Find $\frac{\partial L}{\partial \boldsymbol{W}_1}$.

解: 对 $z \in \mathbb{R}$, Sigmoid 函数定义为

$$\text{Sigmoid}(z) = \frac{1}{1+\exp(-z)} \tag{53}$$

其对 z 的偏导数为

$$\begin{aligned} \frac{\partial\ \text{Sigmoid}(z)}{\partial z} &= \frac{-[-\exp(-z)]}{[1+\exp(z)]^2} \\ &= \frac{1}{1+\exp(-z)} \frac{\exp(-z)}{1+\exp(-z)} \\ &= \text{Sigmoid}(z)[1-\text{Sigmoid}(z)] \end{aligned} \tag{54}$$

对 $\boldsymbol{z} \in \mathbb{R}^d$, Sigmoid 函数可用逐项除法 $\oslash$ 符号表示为

$$\text{Sigmoid}(\boldsymbol{z}) = \boldsymbol{1} \oslash [\boldsymbol{1} + \exp(-\boldsymbol{z})] \tag{55}$$

则由式 (54) 可知 $\text{Sigmoid}(\boldsymbol{z})$ 对 $\boldsymbol{z}$ 的偏导数为

$$\frac{\partial\ \text{Sigmoid}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \text{diag}\left\{\text{Sigmoid}(\boldsymbol{z}) \odot \left[\boldsymbol{1} - \text{Sigmoid}(\boldsymbol{z})\right]\right\} \tag{56}$$

记 $\boldsymbol{a}_0 \triangleq \boldsymbol{x}$, 令

$$\boldsymbol{h}_i \triangleq \boldsymbol{W}_i^\top \boldsymbol{a}_{i-1} + \boldsymbol{b}_i, \quad \forall\ i = 1, 2, \ldots, l \tag{57}$$

则

$$\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}} = \boldsymbol{W}_i, \quad \forall\ i = 1, 2, \ldots, l \tag{58}$$

又由式 (56) 可知

$$\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i} = \frac{\partial\ \text{Sigmoid}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i} = \text{diag}[\boldsymbol{a}_i \odot (\boldsymbol{1} - \boldsymbol{a}_i)], \quad \forall\ i = 1, 2, \ldots, l \tag{59}$$

则由链式法则及式 (18) 可得

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}_1} &= \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{a}_1}\cdots\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_{l-1}}\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \left(\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\right)\left(\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{a}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}_2}\right)\cdots\left(\frac{\partial \boldsymbol{h}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{h}_{l-1}}\right)\left(\frac{\partial \boldsymbol{h}_l}{\partial \boldsymbol{a}_{l-1}}\frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{h}_l}\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\left(\prod_{i=2}^{l}\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}}\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i}\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\operatorname{diag}\left[\boldsymbol{a}_1\odot(\boldsymbol{1}-\boldsymbol{a}_1)\right]\left(\prod_{i=2}^{l}\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\boldsymbol{1}-\boldsymbol{a}_i)\right]\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\left(\prod_{i=2}^{l}\boldsymbol{W}_i\right)\left(\prod_{j=1}^{l}\operatorname{diag}\left[\boldsymbol{a}_j\odot(\boldsymbol{1}-\boldsymbol{a}_j)\right]\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \boldsymbol{x}\left[\left(\prod_{i=2}^{l}\boldsymbol{W}_i\right)\left(\prod_{j=1}^{l}\operatorname{diag}\left[\boldsymbol{a}_j\odot(\boldsymbol{1}-\boldsymbol{a}_j)\right]\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}}\right]^{\top}
\end{aligned} \tag{60}$$

2.2. Suppose $z\in\mathbb{R}$ is a scalar, find the maximum of sigmoid function's derivative: $g(z)=\text{Sigmoid}'(z)$.

解: 由式 (54) 可知

$$g(z)=\text{Sigmoid}'(z)=\text{Sigmoid}(z)[1-\text{Sigmoid}(z)] \tag{61}$$

根据基本不等式可得

$$g(z)=\text{Sigmoid}(z)[1-\text{Sigmoid}(z)]\leqslant\frac{\left[\text{Sigmoid}(z)+1-\text{Sigmoid}(z)\right]^2}{4}=\frac{1}{4} \tag{62}$$

等号成立当且仅当

$$\text{Sigmoid}(z)=1-\text{Sigmoid}(z) \tag{63}$$

即

$$\text{Sigmoid}(z)=\frac{1}{2} \tag{64}$$

即

$$z=0 \tag{65}$$

故当 $z=0$ 时, Sigmoid 函数的梯度 $g(z)$ 取到最大值

$$\max_{z\in\mathbb{R}}g(z)=g(0)=\frac{1}{4} \tag{66}$$

2.3. Since many sigmoid functions' derivatives are multiplied together in $\frac{\partial L}{\partial \boldsymbol{W}_1}$, the values in $\frac{\partial L}{\partial \boldsymbol{W}_1}$ will tend to vanish to zero. Is there any possibility that we can initialize weight matrices carefully to recover the values into a good range for float32 (e.g., -0.01 to 1000)? (*Hint: Since the sigmoid function saturates in both sides,*

in order to keep the gradient scale "reasonable", one needs to make sure the values before sigmoid function lie in non-saturation region.)

解: 由上题可知 Sigmoid(z) 函数的最大梯度位于 $z=0$ 处, 且 Sigmoid(0) = 1/2, 即要求

$$\begin{aligned} \boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1 &= \boldsymbol{0} \\ \boldsymbol{W}_i^\top \frac{\boldsymbol{1}}{2} + \boldsymbol{b}_i &= \boldsymbol{0}, \quad \forall\, i = 2, 3, \ldots, l \end{aligned} \tag{67}$$

初始化时偏置 $\boldsymbol{b}_i = \boldsymbol{0}$, $\forall\, i = 1, 2, \ldots, l$, 则初始化权重矩阵需满足

$$\begin{aligned} \boldsymbol{W}_1^\top \boldsymbol{x} &= \boldsymbol{0} \\ \boldsymbol{W}_i^\top \boldsymbol{1} &= \boldsymbol{0}, \quad \forall\, i = 2, 3, \ldots, l \end{aligned} \tag{68}$$

此时获得的 Sigmoid 函数梯度连乘是最大的, 但由于

$$\max_{z \in \mathbb{R}} g(z) = \frac{1}{4} \tag{69}$$

又 $l > 100$, $\forall\, z_j \in \mathbb{R},\ j = 1, 2, \ldots, l$ 有

$$\prod_{j=1}^{l} g(z_j) < \prod_{j=1}^{l} \max_{z_j \in \mathbb{R}} g(z_j) = 2^{-2l} < 2^{-200} \ll 2^{-126} \tag{70}$$

故当层数很多时, 在梯度都取最大值这种最好的情况下也几乎不可能把梯度值恢复到对 float32 精度友好的数值范围 (e.g., -0.01 to 1000) 内, 甚至会远远超出 float32 精度的表示范围 ($2^{-126} \sim 2^{127}$).

2.4. If our computer supports efficient arbitrary precision floating point calculation, do we need to care about vanishing gradient problem anymore?

解: 需要考虑, 因为即便计算精确, 梯度 $\frac{\partial L}{\partial \boldsymbol{W}_1}$ 过小也会使得权重矩阵的更新过程

$$\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 - \eta \frac{\partial L}{\partial \boldsymbol{W}_1} \tag{71}$$

极为缓慢, 最终导致整个神经网络长时间无法收敛.

2.5. Can we alleviate the vanishing problem by replacing the activation function from sigmoid to ReLU? Why?

解: 可以缓解一部分. 此时

$$\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i} = \frac{\partial\ \mathrm{ReLU}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i} = \mathrm{diag}\left(\frac{\boldsymbol{1} + \mathrm{sgn}(\boldsymbol{a}_j)}{2}\right), \quad \forall\, i = 1, 2, \ldots, l \tag{72}$$

则

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{W}_1} &= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1} \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1} \left(\prod_{i=2}^{l} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}} \frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i}\right) \frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\ &= \boldsymbol{x} \left[\left(\prod_{i=2}^{l} \boldsymbol{W}_i\right) \left(\prod_{j=1}^{l} \mathrm{diag}\left(\frac{\boldsymbol{1} + \mathrm{sgn}(\boldsymbol{a}_j)}{2}\right)\right) \frac{\partial L}{\partial \hat{\boldsymbol{y}}}\right]^\top \end{aligned} \tag{73}$$

所以, ReLU 函数梯度连乘的部分

$$\prod_{j=1}^{l} \operatorname{diag}\left(\frac{\mathbf{1}+\operatorname{sgn}(\boldsymbol{a}_j)}{2}\right)$$

不会消失, 但若 $\det(\boldsymbol{W}_i)$ 过小, 则权重矩阵连乘的部分

$$\prod_{i=2}^{l} \boldsymbol{W}_i$$

会消失, 这也会导致梯度的消失. 因此, 把 Sigmoid 函数换成 ReLU 函数可以缓解一部分梯度消失的问题, 但不能完全避免这个问题.

Effect of ResNet. The neural network now has skip connections:

$$\begin{aligned}
\boldsymbol{a}_1 &= \operatorname{Sigmoid}(\boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{x} \\
\boldsymbol{a}_2 &= \operatorname{Sigmoid}(\boldsymbol{W}_2^\top \boldsymbol{a}_1 + \boldsymbol{b}_2) + \boldsymbol{a}_1 \\
&\vdots \\
\hat{\boldsymbol{y}} = \boldsymbol{a}_l &= \operatorname{Sigmoid}(\boldsymbol{W}_l^\top \boldsymbol{a}_{l-1} + \boldsymbol{b}_l) + \boldsymbol{a}_{l-1}
\end{aligned} \tag{74}$$

2.7. Find $\frac{\partial L}{\partial \boldsymbol{W}_1}$ and explain why the gradients can not easily vanish.

解: 与 2.1 题类似, 由链式法则及式 (18) 可得

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}_1} &= \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{a}_1}\cdots\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_{l-1}}\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \left(\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\right)\left(\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{a}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}_2}+\boldsymbol{I}\right)\cdots\left(\frac{\partial \boldsymbol{h}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{h}_{l-1}}+\boldsymbol{I}\right)\left(\frac{\partial \boldsymbol{h}_l}{\partial \boldsymbol{a}_{l-1}}\frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{h}_l}+\boldsymbol{I}\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\left[\prod_{i=2}^{l}\left(\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}}\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i}+\boldsymbol{I}\right)\right]\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\operatorname{diag}\left[\boldsymbol{a}_1\odot(\mathbf{1}-\boldsymbol{a}_1)\right]\left[\prod_{i=2}^{l}\left(\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\mathbf{1}-\boldsymbol{a}_i)\right]+\boldsymbol{I}\right)\right]\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \boldsymbol{x}\left\{\operatorname{diag}\left[\boldsymbol{a}_1\odot(\mathbf{1}-\boldsymbol{a}_1)\right]\left[\prod_{i=2}^{l}\left(\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\mathbf{1}-\boldsymbol{a}_i)\right]+\boldsymbol{I}\right)\right]\frac{\partial L}{\partial \hat{\boldsymbol{y}}}\right\}^\top
\end{aligned} \tag{75}$$

所以, 连乘部分

$$\prod_{i=2}^{l}\left(\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\mathbf{1}-\boldsymbol{a}_i)\right]+\boldsymbol{I}\right)$$

中的每一项都加上了一个单位矩阵, 则此部分轻易不会消失, 因此 ResNet 可以有效防止梯度消失的问题.

Programming: Image Classification with CIFAR-10

Please install PyTorch and run the CIFAR-10 tutorial. If you want to use TensorFlow or other deep learning frameworks, please find corresponding CIFAR-10 tutorial for that framework and run it.

If you successfully run the tutorial, write "Success" in the report and otherwise please write "Failed". For this assignment report, only this single word is required, no details/accuracies/codes are needed. The assignment intends to get you familiar with the deep learning library. If you have already trained some deep classifiers using any deep learning library before, you can safely ignore the tutorial and directly write "Success" in the report.

解: Success.

参考文献

[1] Dimitri P. Bertsekas. Nonlinear Programming: 3rd Edition[M]. Athena Scientific, 2016. ISBN: 978-1-886529-05-2, http://www.athenasc.com/nonlinbook.html.

[2] Bouvrie, Jake. Notes on Convolutional Neural Networks. CogPrints.org, 2006. http://cogprints.org/5869/1/cnn_tutorial.pdf.