Deep Learning: Homework #1

Due on March 7, 2021

Professor Zhen Li

Peng Deng

Problem 1

Cross entropy is often used as the objective function when training neural networks in classification problems. Suppose the training set includes N training pairs $D = \left\{\left(\boldsymbol{x}_i^{(\text{train})}, y_i^{(\text{train})}\right)\right\}_{i=1}^{N}$, where $\boldsymbol{x}_i^{(\text{train})}$ is a training sample and $y_i^{(\text{train})} \in \{1, \ldots, c\}$ is its corresponding class label. $\boldsymbol{z}_i$ is the output of the network given input $\boldsymbol{x}_i^{(\text{train})}$ and the nonlinearity of the output layer is softmax. $\boldsymbol{z}_i$ is a c dimensional vector, $z_{i,k} \in [0, 1]$ and $\sum_{k=1}^{c} z_{i,k} = 1$. The questions are as follows.

(1) Write the objective function of cross-entropy with softmax activation function, and calculate the gradient of hidden-to-output and input-to-hidden weights as we introduced in class.

(2) Verify it is equivalent to the negative log-likelihood on the training set, assuming the training samples are independent.

Solution

Subproblem (1)

We use one-hot encoding to encode the target value for the training sample $\boldsymbol{x}_i^{(\text{train})}$ as $\boldsymbol{t}_i (i = 1, 2, \ldots N)$, which is a c dimensional vector. Besides, it is defined as follows

$$t_{i,k} = \begin{cases} 1 & (k = y_i^{(\text{train})}) \\ 0 & (k \neq y_i^{(\text{train})}) \end{cases} \tag{1}$$

- The objective function can be written as follow. We use $\boldsymbol{W}$ to denote all the weights in the network.

$$J(\boldsymbol{W}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{c} t_{i,k} \log(z_{i,k}) \tag{2}$$

- The gradient of hidden-to-output weights is calculated as follow. We use net to denote the net value in the output layer. We set δ to denote the sensitivity in the output layer. We use w to denote the weights of hidden-to-output.

$$\begin{aligned} \frac{\partial J}{\partial w_{kj}} &= \sum_{i=1}^{N} \frac{\partial J}{\partial net_{i,k}} \frac{\partial net_{i,k}}{\partial w_{kj}} \\ &= \sum_{i=1}^{N} -\delta_{i,k} \frac{\partial net_{i,k}}{\partial w_{kj}} \end{aligned} \tag{3}$$

Then, we can calculate $\delta_{i,k}$ ans $\boldsymbol{\delta}_i$ as follow

$$\begin{aligned} \delta_{i,k} &= -\frac{\partial J}{\partial net_{i,k}} = -\frac{\partial J}{\partial z_{i,k}} \frac{\partial z_{i,k}}{\partial net_{i,k}} \\ \boldsymbol{\delta}_i &= -\frac{\partial J}{\partial \boldsymbol{net}_i} = -\frac{\partial J}{\partial \boldsymbol{z}_i} \frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{net}_i} \end{aligned} \tag{4}$$

Set $m = y_i^{(\text{train})}$, then we have $t_{i,m} = 1$, and $\boldsymbol{t}_i = (0, \cdots, 1, \cdots, 0)$. Then we can get $\frac{\partial J}{\partial \boldsymbol{z}_i}$ as follow

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{z}_i} &= \begin{pmatrix} \frac{\partial J}{\partial z_{i,1}} & \frac{\partial J}{\partial z_{i,2}} & \cdots & \frac{\partial J}{\partial z_{i,c}} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & \cdots & -\frac{1}{N}\frac{1}{z_{i,m}} & \cdots & 0 \end{pmatrix} \end{aligned} \tag{5}$$

Then we can calculate $\frac{\partial \boldsymbol{z}_i}{\boldsymbol{net}_i}$ as follow

$$\begin{aligned}\frac{\partial \boldsymbol{z}_i}{\boldsymbol{net}_i} &= \begin{pmatrix} \frac{\partial z_{i,1}}{\partial net_{i,1}} & \frac{\partial z_{i,1}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,1}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,1}}{\partial net_{i,c}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial z_{i,m}}{\partial net_{i,1}} & \frac{\partial z_{i,m}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,m}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,m}}{\partial net_{i,c}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial z_{i,c}}{\partial net_{i,1}} & \frac{\partial z_{i,c}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,c}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,c}}{\partial net_{i,c}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial z_{i,1}}{\partial net_{i,1}} & \frac{\partial z_{i,1}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,1}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,1}}{\partial net_{i,c}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ -z_{i,m} \cdot z_{i,1} & -z_{i,m} \cdot z_{i,2} & \cdots & z_{i,m} - z_{i,m}^2 & \cdots & -z_{i,m} \cdot z_{i,c} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial z_{i,c}}{\partial net_{i,1}} & \frac{\partial z_{i,c}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,c}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,c}}{\partial net_{i,c}} \end{pmatrix} \end{aligned} \tag{6}$$

By combination of equation 5 and equation 6, we can recalculate $\delta_{i,k}$ ans $\boldsymbol{\delta}_i$ as follow

$$\begin{aligned}\boldsymbol{\delta}_i &= -\frac{\partial J}{\partial \boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\boldsymbol{net}_i} \\ &= -\begin{pmatrix} 0 & 0 & \cdots & -\frac{1}{N}\frac{1}{z_{i,m}} & \cdots & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial z_{i,1}}{\partial net_{i,1}} & \frac{\partial z_{i,1}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,1}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,1}}{\partial net_{i,c}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ -z_{i,m} \cdot z_{i,1} & -z_{i,m} \cdot z_{i,2} & \cdots & z_{i,m} - z_{i,m}^2 & \cdots & -z_{i,m} \cdot z_{i,c} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial z_{i,c}}{\partial net_{i,1}} & \frac{\partial z_{i,c}}{\partial net_{i,2}} & \cdots & \frac{\partial z_{i,c}}{\partial net_{i,m}} & \cdots & \frac{\partial z_{i,c}}{\partial net_{i,c}} \end{pmatrix} \\ &= \begin{pmatrix} -\frac{1}{N}z_{i,1} & -\frac{1}{N}z_{i,2} & \cdots & \frac{1}{N}(1 - z_{i,m}) & \cdots & -\frac{1}{N}z_{i,c} \end{pmatrix} \\ &= \frac{1}{N}\begin{pmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 \end{pmatrix} - \begin{pmatrix} z_{i,1} & z_{i,2} & \cdots & z_{i,m} & \cdots & z_{i,c} \end{pmatrix} \\ &= \frac{1}{N}(\boldsymbol{t}_i - \boldsymbol{z}_i) \end{aligned}$$

$$\delta_{i,k} = \frac{1}{N}(t_{i,k} - z_{i,k}) \tag{7}$$

Back to equation 3, then we can recalculate the gradient of hidden-to-output weights as follow

$$\begin{aligned}\frac{\partial J}{\partial w_{kj}} &= \sum_{i=1}^{N} -\delta_{i,k}\frac{\partial net_{i,k}}{\partial w_{kj}} \\ &= \sum_{i=1}^{N} -\frac{1}{N}(t_{i,k} - z_{i,k})\, y_{i,j} \\ &= -\frac{1}{N}\sum_{i=1}^{N}(t_{i,k} - z_{i,k})\, y_{i,j} \end{aligned} \tag{8}$$

Thus, we can write down the matrix format of the gradient of hidden-to-output weights as follow. We suppose there are n_H unit in the hidden layer.

$$\frac{\partial J}{\partial \boldsymbol{w}} = -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{t}_i - \boldsymbol{z}_i\right)^{\top}\boldsymbol{y}_i \tag{9}$$

Where $\left(\boldsymbol{t}_i - \boldsymbol{z}_i\right)^{\top}$ is a $c \times 1$ column vector and $\boldsymbol{y}_i$ is a $1 \times n_H$ row vector.

- We suppose $\boldsymbol{x}_i^{(\text{train})}$ is a d dimensional vector, and we use q to denote the q^{th} unit. The gradient of input-to-hidden weights is calculated as follow. We use $net^{'}$ to denote the net value in the hidden layer. We set $\delta^{'}$ to denote the sensitivity in the output layer. We use $w^{'}$ to denote the weights of hidden-to-output.

$$\begin{aligned}\frac{\partial J}{\partial w_{jq}^{'}} &= \sum_{i=1}^{N}\frac{\partial J}{\partial net_{i,j}^{'}}\frac{\partial net_{i,j}^{'}}{\partial w_{jq}^{'}} \\ &= \sum_{i=1}^{N} -\delta_{i,j}^{'}\frac{\partial net_{i,j}^{'}}{\partial w_{jq}^{'}}\end{aligned} \tag{10}$$

Then, we can calculate $\delta_{i,j}^{'}$ ans $\boldsymbol{\delta}_i^{'}$ as follow

$$\begin{aligned}\delta_{i,j}^{'} &= -\frac{\partial J}{\partial net_{i,j}^{'}} = -\frac{\partial J}{\partial y_{i,j}}\frac{\partial y_{i,j}}{\partial net_{i,j}^{'}} \\ \boldsymbol{\delta}_i^{'} &= -\frac{\partial J}{\partial \boldsymbol{net}_i^{'}} = -\frac{\partial J}{\partial \boldsymbol{y}_i}\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{net}_i^{'}}\end{aligned} \tag{11}$$

Then we can get $\frac{\partial J}{\partial \boldsymbol{y}_i}$ as follows

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{y}_i} &= \frac{\partial J}{\partial \boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{y}_i} = \frac{\partial J}{\partial \boldsymbol{z}_i}\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{net}_i}\frac{\partial \boldsymbol{net}_i}{\partial \boldsymbol{y}_i} \\ &= -\boldsymbol{\delta}_i\frac{\partial \boldsymbol{net}_i}{\partial \boldsymbol{y}_i} = -\boldsymbol{\delta}_i\boldsymbol{w}\end{aligned} \tag{12}$$

Where $\boldsymbol{\delta}_i$ is a $1 \times c$ row vector and $\boldsymbol{w}$ is a $c \times n_H$ matrix. We can get $\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{net}_i^{'}}$ as follow

$$\begin{aligned}\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{net}_i^{'}} &= \begin{pmatrix} \frac{\partial y_{i,1}}{\partial net_{i,1}^{'}} & \frac{\partial y_{i,1}}{\partial net_{i,2}^{'}} & \cdots & \frac{\partial y_{i,1}}{\partial net_{i,n_H}^{'}} \\ \frac{\partial y_{i,2}}{\partial net_{i,1}^{'}} & \frac{\partial y_{i,2}}{\partial net_{i,2}^{'}} & \cdots & \frac{\partial y_{i,2}}{\partial net_{i,n_H}^{'}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{i,n_H}}{\partial net_{i,1}^{'}} & \frac{\partial y_{i,n_H}}{\partial net_{i,2}^{'}} & \cdots & \frac{\partial y_{i,n_H}}{\partial net_{i,n_H}^{'}} \end{pmatrix} \\ &= \begin{pmatrix} y_{i,1} - y_{i,1}^2 & -y_{i,1}\cdot y_{i,2} & \cdots & -y_{i,1}\cdot y_{i,n_H} \\ -y_{i,2}\cdot y_{i,1} & y_{i,2} - y_{i,2}^2 & \cdots & -y_{i,2}\cdot y_{i,n_H} \\ \vdots & \vdots & \ddots & \vdots \\ -y_{i,n_H}\cdot y_{i,1} & -y_{i,n_H}\cdot y_{i,2} & \cdots & y_{i,n_H} - y_{i,n_H}^2 \end{pmatrix} \\ &= \operatorname{diag}\left(\boldsymbol{y}_i\right) - \boldsymbol{y}_i^{\top}\boldsymbol{y}_i\end{aligned} \tag{13}$$

By combination of equation 12 and equation 13, we can recalculate $\delta_{i,k}^{'}$ ans $\boldsymbol{\delta}_i^{'}$ as follow

$$\begin{aligned}\boldsymbol{\delta}_i^{'} &= -\frac{\partial J}{\partial \boldsymbol{y}_i}\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{net}_i^{'}} \\ &= \boldsymbol{\delta}_i \boldsymbol{w}\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{net}_i^{'}} \\ &= \frac{1}{N}\left(\boldsymbol{t}_i - \boldsymbol{z}_i\right)\boldsymbol{w}\left(\operatorname{diag}\left(\boldsymbol{y}_i\right) - \boldsymbol{y}_i^{\top}\boldsymbol{y}_i\right)\end{aligned} \tag{14}$$

Back to equation 10, then we can recalculate the gradient of input-to-hidden weights as follow

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{w}^{'}} &= -\sum_{i=1}^{N}\boldsymbol{\delta}_i^{'\top}\boldsymbol{x}_i = -\sum_{i=1}^{N}\left(\boldsymbol{\delta}_i\boldsymbol{w}\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{net}_i^{'}}\right)^{\top}\boldsymbol{x}_i \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left(\operatorname{diag}\left(\boldsymbol{y}_i\right) - \boldsymbol{y}_i^{\top}\boldsymbol{y}_i\right)^{\top}\boldsymbol{w}^{\top}\left(\boldsymbol{t}_i - \boldsymbol{z}_i\right)^{\top}\boldsymbol{x}_i \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left(\operatorname{diag}\left(\boldsymbol{y}_i\right) - \boldsymbol{y}_i^{\top}\boldsymbol{y}_i\right)\boldsymbol{w}^{\top}\left(\boldsymbol{t}_i - \boldsymbol{z}_i\right)^{\top}\boldsymbol{x}_i\end{aligned} \tag{15}$$

Subproblem (2)

We suppose $z_{i,k} \in [0,1]$ and $\sum_{k=1}^{c} z_{i,k} = 1$, which is the probability of $\boldsymbol{x}_i^{(\text{train})}$ belongs to class k. Thus, we can get the likelihood of $\boldsymbol{x}_i^{(\text{train})}$ as follow

$$p_i = \prod_{k=1}^{c}\left(z_{i,k}\right)^{t_{i,k}} \tag{16}$$

Then we can get the likelihood function as follow

$$L = \prod_{i=1}^{N} p_i = \prod_{i=1}^{N}\prod_{k=1}^{c}\left(z_{i,k}\right)^{t_{i,k}} \tag{17}$$

Then we can get the log-likelihood function as follow

$$\begin{aligned}\log L &= \sum_{i-1}^{N}\log\left(\prod_{k=1}^{c}\left(z_{i,k}\right)^{t_{i,k}}\right) \\ &= \sum_{i-1}^{N}\sum_{k=1}^{c} t_{i,k}\log z_{i,k}\end{aligned} \tag{18}$$

According to equation 2 and equation 18, we can find that

$$\begin{aligned}J\left(\boldsymbol{W}\right) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{c} t_{i,k}\log\left(z_{i,k}\right) \\ \log L &= \sum_{i-1}^{N}\sum_{k=1}^{c} t_{i,k}\log z_{i,k}\end{aligned} \tag{19}$$

Because we know that $J\left(\boldsymbol{W}\right)$ can be expressed as follow as well

$$J\left(\boldsymbol{W}\right) = -\sum_{i=1}^{N}\sum_{k=1}^{c} t_{i,k}\log\left(z_{i,k}\right) \tag{20}$$

Thus, we can see that $J\left(\boldsymbol{W}\right)$ is equivalent to the negative log-likelihood on the training set.

Problem 2

x_1 and x_2 are two input variables, and y is the target variable to be predicted. The network structure is shown in Figure 1(a). $h_{11} = f_{11}(x_1), h_{12} = f_{12}(x_2)$, and $y = g(h_{11}, h_{12})$. The questions are as follows.

(1) Assuming $x_1 \in [0,1]$ and $x_2 \in [0,1]$, in order to obtain the decision regions in Figure 1(b), decide functions f_{11}, f_{12}, and g.

(2) Now we extend the range of x_1 and x_2 to [0,2] . Please add one more layer to Figure 1(a) in order to obtain the decision regions in Figure 1(c) .

(3) Although the decision boundaries in Figure 1(c) look complicated, there exist regularity and global structure. Please explain such regularity and global structure. Based on your observation, draw the decision boundaries when the range of x_1 and x_2 to $[0,4]$.

(4) Following the question above and assuming the range of x_1 and x_2 is extended to $[0, 2^n]$, draw the network structure and the transform function in each layer, in order to obtain the decision regions with the same regularity and global structure in Figure 1(b) and (c). The complexity of computation units should be $O(n)$.

(5) Assuming the range of x_1 and x_2 is $[0, 2^n]$ and only one hidden layer is allowed, specify the network structure and transform functions.

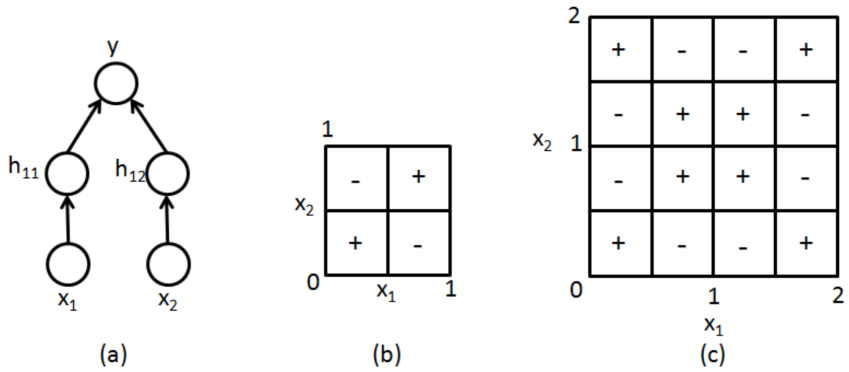


Figure 1 Figure for problem 2

Solution

Subproblem (1)

We can decide the functions as follows

$$\begin{aligned} f_{11}(x) = x - \frac{1}{2} \quad &\Longrightarrow h_{11} = f_{11}(x_1) = x_1 - \frac{1}{2} \quad \Longrightarrow h_{11} \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\ f_{12}(x) = x - \frac{1}{2} \quad &\Longrightarrow h_{12} = f_{12}(x_2) = x_2 - \frac{1}{2} \quad \Longrightarrow h_{12} \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\ g(u,v) = u \cdot v \quad &\Longrightarrow y = g(h_{11}, h_{12}) = h_{11} \cdot h_{12} \end{aligned} \tag{21}$$

Thus, when $y > 0$, we classify the point to positive class, and when $y < 0$, we classify the point to negative class.

Subproblem (2)

When we extend the range of x_1 and x_2 to [0,2], we can add one more layer to Figure 1(a) and then we can obtain the decision regions in Figure 1(c) . The new network structure is showed as Figure 2.

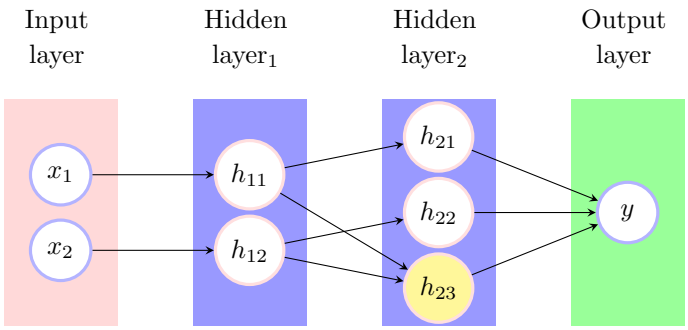


Figure 2 The new network structure

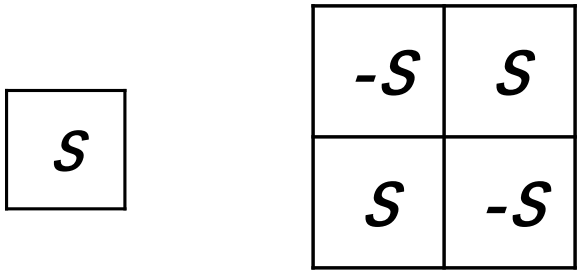
Besides, the relationships between units in different layers are decided as follows. The yellow unit is used to transmit symbols $(+,-)$.

$$\begin{aligned}
&h_{11} = x_1 - 1 \quad \Longrightarrow h_{11} \in [-1,1] \\
&h_{12} = x_2 - 1 \quad \Longrightarrow h_{12} \in [-1,1] \\
&h_{21} = h_{11} - \operatorname{sgn}(h_{11}) \cdot \frac{1}{2} \quad \Longrightarrow h_{21} \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\
&h_{22} = h_{12} - \operatorname{sgn}(h_{12}) \cdot \frac{1}{2} \quad \Longrightarrow h_{22} \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\
&h_{23} = h_{11} \cdot h_{12} \\
&y = h_{21} \cdot h_{22} \cdot h_{23}
\end{aligned} \tag{22}$$

Thus, when $y > 0$, we classify the point to positive class, and when $y < 0$, we classify the point to negative class.

Subproblem (3)

As we can see, there is a recurrence process in this problem and we can get the recurrence formula bwtween two states as follow. We define the decision boundaries when $n = i$ as Figure 3(a), and then we can get decision boundaries when $n = i + 1$ as Figure 3(b). Thus, we get the recurrence formula of the decision boundaries between state $n = i$ and state $n = i + 1$ as Figure 3, which is the regularity and global structure. Then we can draw the decision boundaries when the range of x_1 and x_2 to $[0, 4]$ as Figure 4, and we can see in this situation, $n = 2$.



(a) The state $n = i$ (b) The state $n = i + 1$

Figure 3 The recurrence formula of the decision boundaries between state $n = i$ and state $n = i + 1$

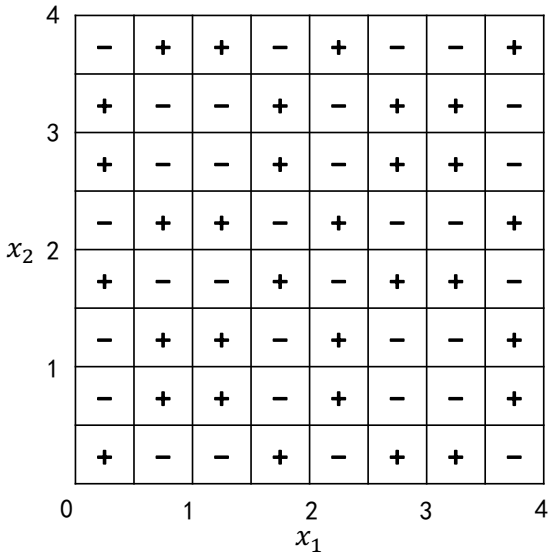


Figure 4 The decision boundaries when the range of x_1 and x_2 to $[0, 4]$

Subproblem (4)

When the range of x_1 and x_2 is extended to $[0, 2^n]$, the network structure is showed as Figure 5.

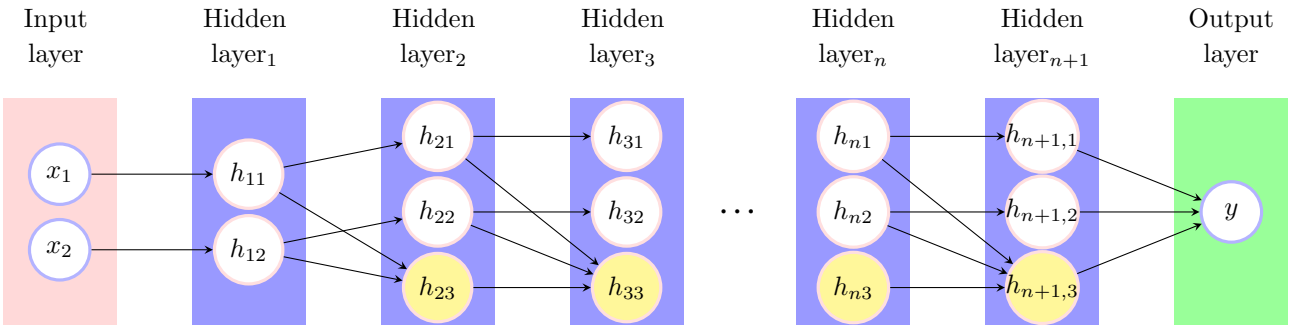


Figure 5 The network structure

The relationships between units in different layers are decided as follows. The yellow unit is used to transmit symbols $(+, -)$.

◦ For input to hidden layer

$$\begin{aligned} h_{11} &= x_1 - 2^{n-1} \quad \Longrightarrow h_{11} \in \left[-2^{n-1}, 2^{n-1}\right] \\ h_{12} &= x_2 - 2^{n-1} \quad \Longrightarrow h_{12} \in \left[-2^{n-1}, 2^{n-1}\right] \end{aligned} \tag{23}$$

◦ For hidden layer 1 to hidden layer 2

$$\begin{aligned} h_{21} &= h_{11} - \operatorname{sng}(h_{11}) \cdot 2^{n-2} \quad \Longrightarrow h_{21} \in \left[-2^{n-2}, 2^{n-2}\right] \\ h_{22} &= h_{12} - \operatorname{sng}(h_{12}) \cdot 2^{n-2} \quad \Longrightarrow h_{22} \in \left[-2^{n-2}, 2^{n-2}\right] \\ h_{23} &= h_{11} \cdot h_{12} \end{aligned} \tag{24}$$

◦ For hidden layer i to hidden layer $i+1$, $i = 2, 3, \cdots, n$

$$\begin{aligned} h_{i+1,1} &= h_{i1} - \operatorname{sng}(h_{i1}) \cdot 2^{n-i-1} \quad \Longrightarrow h_{i+1,1} \in \left[-2^{n-i-1}, 2^{n-i-1}\right] \\ h_{i+1,2} &= h_{i2} - \operatorname{sng}(h_{i2}) \cdot 2^{n-i-1} \quad \Longrightarrow h_{i+1,2} \in \left[-2^{n-i-1}, 2^{n-i-1}\right] \\ h_{i+1,3} &= h_{i1} \cdot h_{i2} \cdot h_{i3} \end{aligned} \tag{25}$$

◦ For the last hidden layer (hidden layer n+1) to the output layer

$$y = h_{n+1,1} \cdot h_{n+1,2} \cdot h_{n+1,3} \tag{26}$$

Thus, when $y > 0$, we classify the point to positive class, and when $y < 0$, we classify the point to negative class.
Then, we can get the complexity of computation units as follow. We define the number of units as Q.

$$\begin{aligned} Q &= Q_{\text{input}} + Q_{\text{hidden}} + Q_{\text{output}} \\ &= 2 + (n+1) \cdot 3 - 1 + 1 \\ &= 3n + 5 \end{aligned} \tag{27}$$

Thus, we can see the the complexity of computation units is $O(n)$. With only one hidden layer, the network structure is specified as Figure 6.

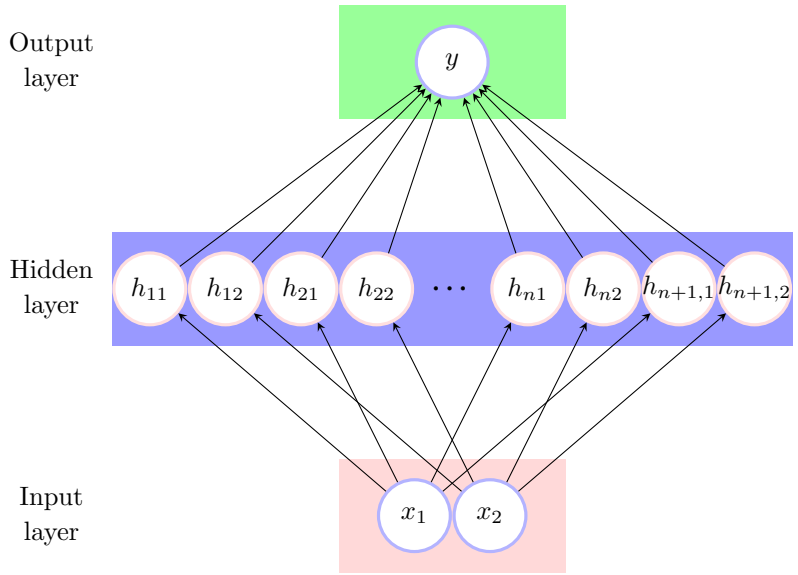


Figure 6 The network structure with only one hidden layer

The relationships between units in different layers are decided as follows.
◦ For input layer to h_{11}, h_{12}

$$\begin{aligned} h_{11} &= x_1 - 2^{n-1} \quad \Longrightarrow h_{11} \in \left[-2^{n-1}, 2^{n-1}\right] \\ h_{12} &= x_2 - 2^{n-1} \quad \Longrightarrow h_{12} \in \left[-2^{n-1}, 2^{n-1}\right] \end{aligned} \tag{28}$$

◦ For input layer to $h_{i+1,1}, h_{i+1,2}$, $i = 1, 3, \cdots, n$.

$$\begin{aligned} h_{i+1,1} &= h_{i1} - \operatorname{sng}(h_{i1}) \cdot 2^{n-i-1} \Longrightarrow \text{we can compute it iteratively and } h_{i+1,1} \in \left[-2^{n-i-1}, 2^{n-i-1}\right] \\ h_{i+1,2} &= h_{i2} - \operatorname{sng}(h_{i2}) \cdot 2^{n-i-1} \Longrightarrow \text{we can compute it iteratively and } h_{i+1,2} \in \left[-2^{n-i-1}, 2^{n-i-1}\right] \end{aligned} \tag{29}$$

◦ For the hidden layer to the output layer

$$y = \prod_{i=1}^{n+1} h_{i1} \cdot h_{i2} \tag{30}$$

Thus, when $y > 0$, we classify the point to positive class, and when $y < 0$, we classify the point to negative class.

Problem 3

Siamese neural nets have an interesting architecture – the same parameters and functions are used to evaluate 2 inputs. As one might expect, Siamese nets are useful to train similarity metrics, evaluations of how "close" inputs are. These nets have been applied to facial recognition tasks with a good deal of success, but in this example, we'll see how to train Siamese nets to learn a distance metric for two inputs, e.g., word vectors. One might imagine training a net to map word vectors across languages, discover synonyms or antonyms, etc.

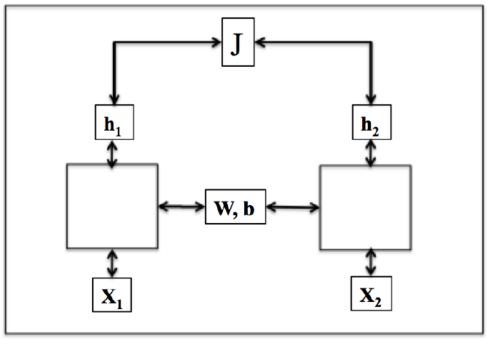


Figure 7 The Siamese neural nets

Here is one such model to evaluate how similar two inputs are using Euclidean distance. There are two inputs $\boldsymbol{x}_1, \boldsymbol{x}_2 \in R^n$, shared parameters $\boldsymbol{W} \in R^{m\times n}$ and $\boldsymbol{b} \in R^m$, and a single hidden layer associated with each input:

$$\boldsymbol{h}_1 = \sigma\left(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b}\right)$$
$$\boldsymbol{h}_2 = \sigma\left(\boldsymbol{W}\boldsymbol{x}_2 + \boldsymbol{b}\right)$$

We evaluate the distance between the two activations h_1, h_2 using Euclidean distance as our similarity metric. The model objective J is

$$J = \frac{1}{2}\left\|h_1 - h_2\right\|_F^2 + \frac{\lambda}{2}\|W\|_F^2$$

where λ is a given regularization parameter. (The Frobenius norm $\|\cdot\|_F$ is a matrix norm defined by $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j}\left|\boldsymbol{A}_{ij}\right|^2}$). The questions are as follows.

(1) Calculate the gradient $\nabla_W J$ and $\nabla_b J$.

(2) Write out the (vanilla) gradient descent update rules for the model parameters for a single training example (with arbitrary step size α).

(3) If $\boldsymbol{W} \in R^{10\times 5}$ and $\boldsymbol{b} \in R^{10\times 1}$, how many parameters does the model have?

Now imagine you wanted to see how ReLU/sigmoid nonlinearities might affect training on single input. But instead of training two separate nets, you want to train a psuedo-Siamese net like Figure 8.

Problem 3 continued on next page...

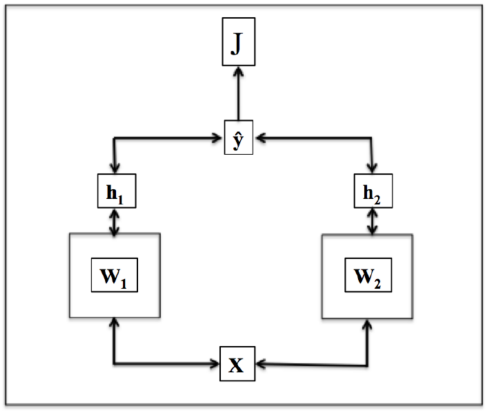


Figure 8 The pseudo-Siamese neural nets

The whole model would be changed to

$$\begin{aligned}\boldsymbol{h}_1 &= \sigma\left(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1\right)\\ \boldsymbol{h}_2 &= \text{Relu}\left(\boldsymbol{W}_2\boldsymbol{x} + \boldsymbol{b}_2\right)\\ \hat{\boldsymbol{y}} &= \text{softmax}\left(\boldsymbol{W}_3\left(\boldsymbol{h}_1 + \boldsymbol{h}_2\right) + \boldsymbol{b}_3\right)\end{aligned}$$

where $\boldsymbol{x} \in R^n, \boldsymbol{W}_1, \boldsymbol{W}_2 \in R^{m\times n}, \boldsymbol{W}_3 \in R^{k\times m}, \boldsymbol{b}_1, \boldsymbol{b}_2 \in R^m$ and $\boldsymbol{b}_3 \in R^k$. We calculate this model for N examples and k classes with cross-entropy loss

$$J = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{k} y_j^i \log\left(\hat{y}_j^i\right)$$

where $\boldsymbol{y}_j$ is the one-hot vector for example j with all probability mass on the correct class and $\hat{y}_j$ are the softmax scores for example j.

(4) Calculate $\nabla_{\boldsymbol{h}_1} J, \nabla_{\boldsymbol{h}_2} J$, and $\nabla_{\boldsymbol{x}} J$.

(5) For $\boldsymbol{W}_1, \boldsymbol{W}_2$, which one is likely to train faster, please explain it.

Solution

Subproblem (1)

∘ For $\nabla_{\boldsymbol{W}} J$

$$\begin{aligned}\frac{\partial J}{\partial w_{ji}} &= \frac{\partial \frac{1}{2}\left\|\boldsymbol{h}_1 - \boldsymbol{h}_2\right\|_F^2}{\boldsymbol{h}_1 - \boldsymbol{h}_2} \cdot \frac{\partial\left(\boldsymbol{h}_1 - \boldsymbol{h}_2\right)}{\partial w_{ji}} + \frac{\partial \frac{\lambda}{2}\|\boldsymbol{W}\|_F^2}{\partial w_{ji}}\\ &= \left(\boldsymbol{h}_1 - \boldsymbol{h}_2\right)^\top\left(\frac{\partial \boldsymbol{h}_1}{\partial w_{ji}} - \frac{\partial \boldsymbol{h}_2}{\partial w_{ji}}\right) + \lambda w_{ji}\\ &= \left(\boldsymbol{h}_1 - \boldsymbol{h}_2\right)^\top\begin{pmatrix}\frac{\partial h_{11}}{\partial w_{ji}} - \frac{\partial h_{21}}{\partial w_{ji}}\\ \frac{\partial h_{12}}{\partial w_{ji}} - \frac{\partial h_{22}}{\partial w_{ji}}\\ \vdots\\ \frac{\partial h_{1m}}{\partial w_{ji}} - \frac{\partial h_{2m}}{\partial w_{ji}}\end{pmatrix} + \lambda w_{ji} = \left(\boldsymbol{h}_1 - \boldsymbol{h}_2\right)^\top\begin{pmatrix}0\\ 0\\ \vdots\\ \sigma'\left(\boldsymbol{w}_j\boldsymbol{x}_1 + b_j\right)\cdot x_{1i} - \sigma'\left(\boldsymbol{w}_j\boldsymbol{x}_2 + b_j\right)\cdot x_{2i}\\ \vdots\\ 0\end{pmatrix} + \lambda w_{ji}\\ &= \left(h_{1j} - h_{2j}\right)\cdot\left(\sigma'\left(\boldsymbol{w}_j\boldsymbol{x}_1 + b_j\right)\cdot x_{1i} - \sigma'\left(\boldsymbol{w}_j\boldsymbol{x}_2 + b_j\right)\cdot x_{2i}\right) + \lambda w_{ji}\end{aligned} \tag{31}$$

Thus, we can get the gradient $\nabla_{\boldsymbol{W}} J$ as follow

$$\nabla_{\boldsymbol{W}} J = \left[(\boldsymbol{h}_1 - \boldsymbol{h}_2) \circ \sigma^{'}(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b})\right] \cdot \boldsymbol{x}_1^{\top} - \left[(\boldsymbol{h}_1 - \boldsymbol{h}_2) \circ \sigma^{'}(\boldsymbol{W}\boldsymbol{x}_2 + \boldsymbol{b})\right] \cdot \boldsymbol{x}_2^{\top} + \lambda \boldsymbol{W} \tag{32}$$

Where $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{b}$ are $m \times 1$ column vectors, and $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ are $n \times 1$ column vectors.
∘ For $\nabla_{\boldsymbol{b}} J$

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{b}} =& \frac{\partial \frac{1}{2}\left\|\boldsymbol{h}_1 - \boldsymbol{h}_2\right\|_F^2}{\boldsymbol{h}_1 - \boldsymbol{h}_2} \cdot \frac{\partial (\boldsymbol{h}_1 - \boldsymbol{h}_2)}{\partial \boldsymbol{b}} \\
=& (\boldsymbol{h}_1 - \boldsymbol{h}_2)^{\top} \cdot \left(\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{b}} - \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{b}}\right) \\
=& (\boldsymbol{h}_1 - \boldsymbol{h}_2)^{\top} \cdot \\
& \begin{pmatrix}
\sigma^{'}(\boldsymbol{w}_1\boldsymbol{x}_1 + b_1) - \sigma^{'}(\boldsymbol{w}_1\boldsymbol{x}_2 + b_1) & 0 & \cdots & 0 \\
0 & \sigma^{'}(\boldsymbol{w}_2\boldsymbol{x}_1 + b_2) - \sigma^{'}(\boldsymbol{w}_1\boldsymbol{x}_2 + b_1) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma^{'}(\boldsymbol{w}_m\boldsymbol{x}_1 + b_m) - \sigma^{'}(\boldsymbol{w}_1\boldsymbol{x}_2 + b_1)
\end{pmatrix} \\
=& (\boldsymbol{h}_1 - \boldsymbol{h}_2)^{\top} \cdot \left[\operatorname{diag}\left(\sigma^{'}(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b})\right) - \operatorname{diag}\left(\sigma^{'}(\boldsymbol{W}\boldsymbol{x}_2 + \boldsymbol{b})\right)\right]
\end{aligned}$$

Thus, we can get $\nabla_{\boldsymbol{b}} J$ as follows

$$\begin{aligned}
\nabla_{\boldsymbol{b}} J &= \left(\frac{\partial J}{\partial \boldsymbol{b}}\right)^{\top} \\
&= \left[\operatorname{diag}\left(\sigma^{'}(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b})\right) - \operatorname{diag}\left(\sigma^{'}(\boldsymbol{W}\boldsymbol{x}_2 + \boldsymbol{b})\right)\right] \cdot (\boldsymbol{h}_1 - \boldsymbol{h}_2)
\end{aligned} \tag{33}$$

Where $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{b}$ are $m \times 1$ column vectors, and $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ are $n \times 1$ column vectors.

Subproblem (2)
We can write out the gradient descent update rules for the model parameters for a single training example ($x_1 \text{and} x_2$) as follow

$$\begin{aligned}
\boldsymbol{W} &= \boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} J \\
&= \boldsymbol{W} - \alpha \left(\left[(\boldsymbol{h}_1 - \boldsymbol{h}_2) \circ \sigma^{'}(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b})\right] \cdot \boldsymbol{x}_1^{\top} - \left[(\boldsymbol{h}_1 - \boldsymbol{h}_2) \circ \sigma^{'}(\boldsymbol{W}\boldsymbol{x}_2 + \boldsymbol{b})\right] \cdot \boldsymbol{x}_2^{\top} + \lambda \boldsymbol{W}\right) \\
\boldsymbol{b} &= \boldsymbol{b} - \alpha \nabla_{\boldsymbol{b}} J \\
&= \boldsymbol{b} - \alpha \left(\left[\operatorname{diag}\left(\sigma^{'}(\boldsymbol{W}\boldsymbol{x}_1 + \boldsymbol{b})\right) - \operatorname{diag}\left(\sigma^{'}(\boldsymbol{W}\boldsymbol{x}_2 + \boldsymbol{b})\right)\right] \cdot (\boldsymbol{h}_1 - \boldsymbol{h}_2)\right)
\end{aligned} \tag{34}$$

Subproblem (3)
Because $\boldsymbol{W} \in R^{10 \times 5}$ and $\boldsymbol{b} \in R^{10 \times 1}$, and there is a hyper-parameter λ, so the number of parameters in the model is

$$10 \times 5 + 10 \times 1 + 1 = 61 \tag{35}$$

Subproblem (4)
We have the loss function as follow

$$\begin{aligned}
J &= -\frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{k} y_j^i \log\left(\hat{y}_j^i\right) \\
&= -\frac{1}{N} \sum_{j=1}^{N} J_j \\
\Longrightarrow J_j &= \sum_{i=1}^{k} y_j^i \log\left(\hat{y}_j^i\right)
\end{aligned} \tag{36}$$

Which means that J_j is the loss for the j^{th} sample. Then we can compute the gradient for sample j. Besides, we define $\hat{\boldsymbol{y}}_j = \text{softmax}(\boldsymbol{net}_j)$. We suppose that the j^{th} sample belongs to the class m, which means $y_j^m = 1$. Besides, we suppose $\boldsymbol{h}_1$, $\boldsymbol{h}_2$ is the vector generated by the j^{th} sample.

∘ For $\nabla_{\boldsymbol{h}_1} J$.

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{h}_1} &= \frac{\partial\left(-\frac{1}{N}\sum_{j=1}^{N} J_j\right)}{\partial \boldsymbol{h}_1} \\ &= -\frac{1}{N}\cdot\frac{\partial J_j}{\partial \boldsymbol{h}_1} \\ &= -\frac{1}{N}\cdot\frac{\partial J_j}{\partial \hat{\boldsymbol{y}}_j}\cdot\frac{\partial \hat{\boldsymbol{y}}_j}{\partial \boldsymbol{net}_j}\cdot\frac{\partial \boldsymbol{net}_j}{\partial \boldsymbol{h}_1}\end{aligned} \tag{37}$$

We can calculate $\frac{\partial J_j}{\partial \hat{\boldsymbol{y}}_j}$ as follow

$$\begin{aligned}\frac{\partial J_j}{\partial \hat{\boldsymbol{y}}_j} &= \frac{\partial \log\left(\hat{y}_j^m\right)}{\partial \hat{\boldsymbol{y}}_j} \\ &= \begin{pmatrix} 0 & 0 & \cdots & \frac{1}{\hat{y}_j^m} & \cdots & 0 \end{pmatrix}\end{aligned} \tag{38}$$

We can calculate $\frac{\partial \hat{\boldsymbol{y}}_j}{\partial \boldsymbol{net}_j}$ as follow

$$\begin{aligned}\frac{\partial \hat{\boldsymbol{y}}_j}{\partial \boldsymbol{net}_j} &= \begin{pmatrix} & & \cdots & \cdots & & \\ \frac{\partial \hat{y}_j^m}{\partial net_j^1} & \frac{\partial \hat{y}_j^m}{\partial net_j^2} & \cdots & \frac{\partial \hat{y}_j^m}{\partial net_j^m} & \cdots & \frac{\partial \hat{y}_j^m}{\partial net_j^k} \\ & & \cdots & \cdots & & \end{pmatrix} \\ &= \begin{pmatrix} & & \cdots & \cdots & & \\ -\hat{y}_j^1\hat{y}_j^m & -\hat{y}_j^2\hat{y}_j^m & \cdots & \hat{y}_j^m - \left(\hat{y}_j^m\right)^2 & \cdots & -\hat{y}_j^k\hat{y}_j^m \\ & & \cdots & \cdots & & \end{pmatrix}\end{aligned} \tag{39}$$

We can calculate $\frac{\partial \boldsymbol{net}_j}{\partial \boldsymbol{h}_1}$ as follow

$$\begin{aligned}\frac{\partial \boldsymbol{net}_j}{\partial \boldsymbol{h}_1} &= \frac{\partial\left(\boldsymbol{W}_3\left(\boldsymbol{h}_1 + \boldsymbol{h}_2\right) + \boldsymbol{b}_3\right)}{\partial \boldsymbol{h}_1} \\ &= \frac{\partial\left(\boldsymbol{W}_3\boldsymbol{h}_1\right)}{\partial \boldsymbol{h}_1} = \boldsymbol{W}_3\end{aligned} \tag{40}$$

Then, by interesting equation 38, equation 39, and equation 40 into equation 37, we can compute $\frac{\partial J_j}{\partial \boldsymbol{h}_1}$ as follow

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{h}_1} &= -\frac{1}{N}\cdot\frac{\partial J_j}{\partial \hat{\boldsymbol{y}}_j}\cdot\frac{\partial \hat{\boldsymbol{y}}_j}{\partial \boldsymbol{net}_j}\cdot\frac{\partial \boldsymbol{net}_j}{\partial \boldsymbol{h}_1} \\ &= -\frac{1}{N}\cdot\begin{pmatrix} 0 & 0 & \cdots & \frac{1}{\hat{y}_j^m} & \cdots & 0 \end{pmatrix}\cdot\begin{pmatrix} & & \cdots & \cdots & & \\ -\hat{y}_j^1\hat{y}_j^m & -\hat{y}_j^2\hat{y}_j^m & \cdots & \hat{y}_j^m - \left(\hat{y}_j^m\right)^2 & \cdots & -\hat{y}_j^k\hat{y}_j^m \\ & & \cdots & \cdots & & \end{pmatrix}\cdot\boldsymbol{W}_3 \\ &= -\frac{1}{N}\cdot\begin{pmatrix} -\hat{y}_j^1 & -\hat{y}_j^2 & \cdots & 1-\hat{y}_j^m & \cdots & -\hat{y}_j^k \end{pmatrix}\cdot\boldsymbol{W}_3 \\ &= -\frac{1}{N}\cdot\left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right)\top\cdot\boldsymbol{W}_3\end{aligned} \tag{41}$$

Problem 3 continued on next page...

Finally, we can get $\nabla_{\boldsymbol{h}_1} J$ as follow

$$\begin{aligned} \nabla_{\boldsymbol{h}_1} J &= \left(\frac{\partial J}{\partial \boldsymbol{h}_1}\right)^\top \\ &= -\frac{1}{N}\boldsymbol{W}_3^\top \left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right) \end{aligned} \tag{42}$$

Where $\boldsymbol{y}_j$ and $\hat{\boldsymbol{y}}_j$ are $k \times 1$ column vector.
∘ For $\nabla_{\boldsymbol{h}_2} J$, it is similar to $\nabla_{\boldsymbol{h}_1} J$. Thus we can get it as follow

$$\begin{aligned} \nabla_{\boldsymbol{h}_2} J &= \left(\frac{\partial J}{\partial \boldsymbol{h}_2}\right)^\top \\ &= -\frac{1}{N}\boldsymbol{W}_3^\top \left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right) \end{aligned} \tag{43}$$

∘ For $\nabla_{\boldsymbol{x}} J$, we consider $\boldsymbol{x}$ is the j^{th} sample, as $\boldsymbol{x}_j$. Besides, in the follow equation, $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are also generated by the j^{th} sample.

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{x}_j} &= \frac{\partial\left(-\frac{1}{N}\sum_{j=1}^{N} J_j\right)}{\partial \boldsymbol{x}_j} \\ &= -\frac{1}{N}\cdot\frac{\partial J_j}{\partial \boldsymbol{x}_j} \\ &= -\frac{1}{N}\cdot\left(\frac{\partial J_j}{\partial \boldsymbol{h}_1}\cdot\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}_j} + \frac{\partial J_j}{\partial \boldsymbol{h}_2}\cdot\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{x}_j}\right) \end{aligned} \tag{44}$$

We can calculate $\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}_j}$ as follows. We define $\hat{\boldsymbol{net}}_{\boldsymbol{j}} = \boldsymbol{W}_1\boldsymbol{x}_j + \boldsymbol{b}_1$.

$$\begin{aligned} \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}_j} &= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{net}_j^{'}}\cdot\frac{\partial \boldsymbol{net}_j^{'}}{\partial \boldsymbol{x}_j} \\ &= \begin{pmatrix} \frac{\partial\sigma\left(\hat{\boldsymbol{net}}_{j,1}\right)}{\partial \hat{\boldsymbol{net}}_{j,1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial\sigma\left(\hat{\boldsymbol{net}}_{j,2}\right)}{\partial \hat{\boldsymbol{net}}_{j,2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial\sigma\left(\hat{\boldsymbol{net}}_{j,m}\right)}{\partial \hat{\boldsymbol{net}}_{j,m}} \end{pmatrix}\cdot \boldsymbol{W}_1 \\ &= \operatorname{diag}\left(\sigma^{'}\left(\hat{\boldsymbol{net}}_j\right)\right)\cdot \boldsymbol{W}_1 \\ &= \operatorname{diag}\left(\sigma^{'}\left(\boldsymbol{W}_1\boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\cdot \boldsymbol{W}_1 \end{aligned} \tag{45}$$

Similarly, we can get $\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{x}_j}$ as follows.

$$\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{x}_j} = \operatorname{diag}\left(\mathrm{ReLU}^{'}\left(\boldsymbol{W}_2\boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\cdot \boldsymbol{W}_2 \tag{46}$$

Back to equation 44, we can calculate $\frac{\partial J}{\partial \boldsymbol{x}_j}$ as follow

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{x}_j} &= -\frac{1}{N}\cdot\left[\left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right)^\top\cdot\boldsymbol{W}_3\cdot\operatorname{diag}\left(\sigma^{'}\left(\boldsymbol{W}_1\boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\cdot\boldsymbol{W}_1 + \left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right)^\top\cdot\boldsymbol{W}_3\cdot\operatorname{diag}\left(\mathrm{ReLU}^{'}\left(\boldsymbol{W}_2\boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\cdot\boldsymbol{W}_2\right] \\ &= -\frac{1}{N}\cdot\left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right)^\top\cdot\boldsymbol{W}_3\cdot\left[\operatorname{diag}\left(\sigma^{'}\left(\boldsymbol{W}_1\boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\cdot\boldsymbol{W}_1 + \operatorname{diag}\left(\mathrm{ReLU}^{'}\left(\boldsymbol{W}_2\boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\cdot\boldsymbol{W}_2\right] \end{aligned} \tag{47}$$

Problem 3 continued on next page...

Thus, we can get $\nabla_{\boldsymbol{x}_j} J$ as follow

$$\begin{aligned}\nabla_{\boldsymbol{x}_j} J &= \left(\frac{\partial J}{\partial \boldsymbol{x}_j}\right)^{\top} \\ &= -\frac{1}{N} \cdot \left[\boldsymbol{W}_1^{\top} \operatorname{diag}\left(\sigma^{'}\left(\boldsymbol{W}_1 \boldsymbol{x}_j + \boldsymbol{b}_1\right)\right) + \boldsymbol{W}_2^{\top} \operatorname{diag}\left(\mathrm{ReLU}^{'}\left(\boldsymbol{W}_2 \boldsymbol{x}_j + \boldsymbol{b}_1\right)\right)\right] \cdot \boldsymbol{W}_3^{\top} \cdot \left(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right)\end{aligned} \tag{48}$$

Subproblem (5)

$\boldsymbol{W}_2$ is likely to train faster. For the sigmod function, the gradient in the positive and negative saturation regions is close to 0, which may cause the vanishing gradient problem and then the training process would be very slow. On the contrary, for the ReLU function, it doesn't have a tendency to saturate, and there will be no particularly small gradients. Thus, the training process would be faster.