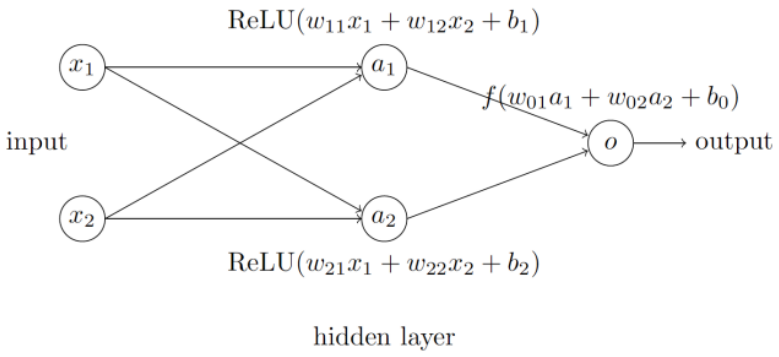
1. [25pts] Multi-layer Perception

Suppose that we apply neural networks on a problem which has boolean inputs $x \in \{0,1\}^p$ and boolean output $y \in \{0,1\}$. The network structure example is showed as below. In this example we set $p = 2$, single hidden layer with 2 neurons, activation function $ReLU(u) = u$ if $u > 0$ otherwise 0, and an additional threshold function (e.g., $f(v) = 1$ if $v > 0$, otherwise $f(v) = 0$) for output layer.



$\mathrm{ReLU}(w_{11}x_1 + w_{12}x_2 + b_1)$
x_1
a_1
$f(w_{01}a_1 + w_{02}a_2 + b_0)$
input
o
output
x_2
a_2
$\mathrm{ReLU}(w_{21}x_1 + w_{22}x_2 + b_2)$
hidden layer

(1) [5pts] Using the structure and settings of neural network above, show that such a simple neural network could output the function $x_1 XOR\ x_2$ (equals to 0 if $x_1 = x_2$ and otherwise 1), which is impossible for linear models. State the values of parameters (i.e., w_{ij} and b_i) you found.

(2) [20pts] Now we allow the number of neurons in the hidden layer to be more than 2 but finite. Retain the structure and other settings. Show that such a neural network with single hidden layer could output an arbitrary binary function $h: \{0,1\}^p \mapsto \{0,1\}$. You can apply threshold function after each neuron in the hidden layer.

(1) Using the structure and settings of NN above, we can obtain the weight matrix and bias matrix of hidden layer

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix}, \boldsymbol{b}^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

The weight matrix and bias matrix of output layer are

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}, \boldsymbol{b}^{(2)} = [b_0]$$

Then we use

$$\boldsymbol{X} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

denotes the four possible inputs. Substitute into NN, the output of the first layer is

$$\begin{aligned}\boldsymbol{H}^{(1)} &= \mathrm{ReLU}\left(\boldsymbol{W}^{(1)^T}\boldsymbol{X} + \boldsymbol{B}^{(1)}\right) \\ &= \mathrm{ReLU}\left(\begin{bmatrix} 0 & w_{12} & w_{11} & w_{11}+w_{12} \\ 0 & w_{22} & w_{21} & w_{21}+w_{22} \end{bmatrix} + \begin{bmatrix} b_1 & b_1 & b_1 & b_1 \\ b_2 & b_2 & b_2 & b_2 \end{bmatrix}\right) \\ &= \mathrm{ReLU}\left(\begin{bmatrix} b_1 & w_{12}+b_1 & w_{11}+b_1 & w_{11}+w_{12}+b_1 \\ b_2 & w_{22}+b_2 & w_{21}+b_2 & w_{21}+w_{22}+b_2 \end{bmatrix}\right)\end{aligned}$$

At the second layer, let its activation function be the identity function, then solve the equation

$$\boldsymbol{H}^{(2)} = {\boldsymbol{W}^{(2)}}^T \boldsymbol{H}^{(1)} + \boldsymbol{B}^{(2)} = [0 \quad 1 \quad 1 \quad 0]$$

Namely

$$\begin{cases} w_{01}\mathrm{ReLU}(b_1) + w_{02}\mathrm{ReLU}(b_2) = 0 \\ w_{01}\mathrm{ReLU}(w_{12} + b_1) + w_{02}\mathrm{ReLU}(w_{22} + b_2) = 1 \\ w_{01}\mathrm{ReLU}(w_{11} + b_1) + w_{02}\mathrm{ReLU}(w_{21} + b_2) = 1 \\ w_{01}\mathrm{ReLU}(w_{11} + w_{12} + b_1) + w_{02}\mathrm{ReLU}(w_{21} + w_{22} + b_2) = 0 \end{cases}$$

This system of equations has multiple solutions, such as

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \boldsymbol{b}^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \boldsymbol{b}^{(2)} = [0]$$

However, xor is not impossible for linear model, because, for the xor problem, the convex sets formed by the positive instance point set and the negative instance point set may intersect, which will lead to linear inseparability.

(2) The original question is equivalent to proving that any Boolean function of p-element input can be approximated by a neural network with ReLU as the activation function.

Lemma 1. *Any Boolean function containing p variables can be uniquely expressed as a Disjunctive Normal Form(DNF)*

Lemma 2. *A single hidden neuron can represent any conjunctive normal form.*

Proof. We can represent a neuron of the hidden layer as $a_k = ReLU\left(\sum_{i=1}^{p} w_{ki}X_i + b_k\right)$.

Consider an arbitrary Boolean variable X_i, if it appears in the form of positive (X_i) in the conjunction normal form (CNF), then set $w_i = 1$; if it appears in the form of negative ($\overline{X_i}$), then set $w_i = -1$; if not in the CNF, set $w_i = 0$; and set$b = 1 - p$. It can be seen that when the ReLU activation function is used, the hidden unit will be activated (output 1) if and only if all the Boolean variables that appear meet the conditions, otherwise it will output 0, which is consistent with the definition of the CNF.

Lemma 1 and *Lemma 2* show that a Boolean function containing p variables needs to be represented by at most 2^{p-1} neurons.

Then, if $o = f\left(\sum_{i=1}^{n} w_{0i}a_i + b_0\right)$ where n is the number of hidden neurons represents the output layer, then let $w_{0i} = 1, b_0 = 0$. In this way, if and only when all hidden neurons are not activated, 0 will be output, otherwise a positive number will be output, which plays the role of disjunction.

But in fact, the above conclusion can be generalized. We can prove that a neuron network with a sigmoidal function or ReLU can approximate an arbitrary function. It's easy to prove that sigmoidal function can be used as an activation function. How about ReLU? Does it also have the same ability to approximate the fitting function as sigmoid?

Lemma 3. *The ReLU function is discriminatory*[1].

$Proof.$ Let μ be a signed Borel measure, and assume the following holds for any $y \in R$ and $\theta \in R$:

$$\int ReLU(yx+\theta)d\mu(x)=0$$

We want to show that $\mu = 0$. For that, we will construct a sigmoid bounded, continuous (and therefore Borel measurable) function from subtracting two ReLU functions with different parameters. In particular, consider the function

$$f(x)=\begin{cases}0 & \text{if } x<0\\ x & \text{if } x\in[0,1]\\ 1 & \text{if } x>1\end{cases}$$

Then any function of the form $g(x)=f(yx+\theta)$ with $y\neq 0$ can be described as

$$g(x)=ReLU(yx+\theta_1)-ReLU(yx+\theta_2)$$

By setting $\theta_1=-\theta/y$ and $\theta_2=(1-\theta)/y$. If $y=0$, then instead set

$$g(x)=f(\theta)=\begin{cases}ReLU\big(f(\theta)\big) & \text{if } f(\theta)\geq 0\\ -ReLU\big(-f(\theta)\big) & \text{if } f(\theta)\leq 0\end{cases}$$

Which means that for any $y\in R,\theta\in R$

$$\begin{aligned}\int f(yx+\theta)d\mu(x)&=\int ReLU(yx+\theta_1)-ReLU(yx+\theta_2)d\mu(x)\\&=\int ReLU(yx+\theta_1)d\mu(x)-\int ReLU(yx+\theta_2)d\mu(x)\\&=0-0=0\end{aligned}$$

Theorem 1 *Let f be a continuous discriminatory function. Then a neural network with f as the activation function is a universal approximator.*

$Proof.$ For the sake of contradiction, assume $\sum_n f$ is not dense in $C(I_n)$. It follows that$\overline{\sum_n f}\neq C(I_n)$. We then apply the corollary of the Hahn-Banach theorem[2] to conclude that there exists some continuous linear functional $F: C(I_n)\to R$ such that $F\neq 0$ but $F(g)=0$ for any $g\in\overline{\sum_n f}$. By the Riesz Representation Theorem[3], there exists some Borel measure μ such that

$$F(g)=\int_{I_n} g(x)d\mu(x) \qquad \text{for all } g\in C(I_n).$$

[1] Cybenkot, G. "Approximation by Superpositions of a Sigmoidal Function ∗." (2006)

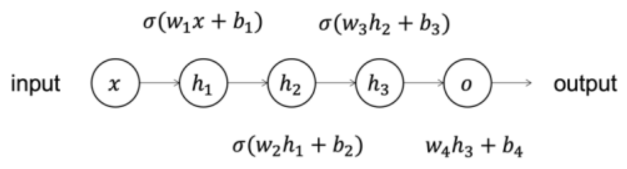
[2] https://en.wikipedia.org/wiki/Hahn%E2%80%93Banach_theorem

[3] https://en.wikipedia.org/wiki/Riesz_representation_theorem

However, since for any y and θ the function $f(y \cdot x + \theta)$ is an element of $\overline{\sum_n f}$., this means that for all $y \in R^n$, and $\theta \in R$ we have $\int f(y.x + 0)d\mu(x) = 0$, which means that $\mu = 0$ (since f is discriminatory) and therefore $F(g) = 0$ for any $g \in C(I_n)$. This contradicts the corollary of the Hahn-Banach theorem, and thus finishes the proof.

2. [10pts] Gradient explosion and gradient vanishing

As shown in the figure below, the neural network has three hidden layers, each with only one neuron, and the activation function is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Let the input be $x = 3$. Use backpropagation to calculate the gradient, and experience the gradient explosion and gradient vanishing issues [1].



(1) [5pts] If $w_1 = 100, w_2 = 150, w_3 = 200, w_4 = 200, b_1 = -300, b_2 = -75, b_3 = -100, b_4 = 10$, calculate the gradient $\frac{\partial o}{\partial b_1}$.

(2) [5pts] If $w_1 = 0.2, w_2 = 0.5, w_3 = 0.3, w_4 = 0.6, b_1 = 1, b_2 = 2, b_3 = 2, b_4 = 1$, calculate the gradient $\frac{\partial o}{\partial b_1}$.

(1)

$$h_1 = \sigma(w_1x + b_1) = \sigma(0) = \frac{1}{2}$$

$$h_2 = \sigma(w_2h_1 + b_2) = \sigma(0) = \frac{1}{2}$$

In the same way, $h_3 = 1/2, o = 110$.

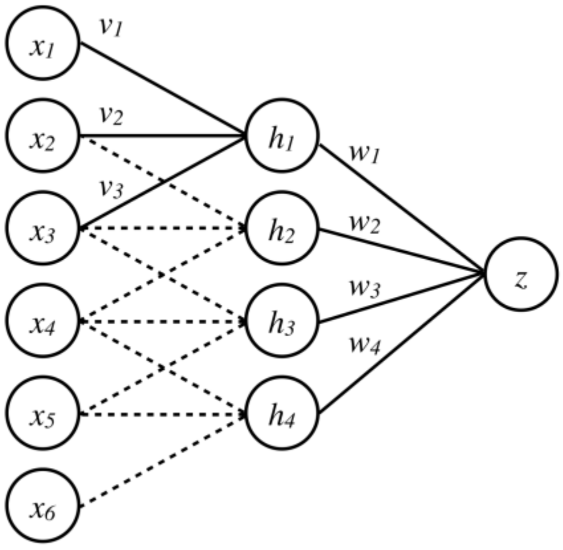
$$\begin{aligned}\frac{\partial o}{\partial b_1} &= \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1} \\ &= w_4 \cdot w_3\sigma'(w_3h_2 + b_3) \cdot w_2\sigma'(w_2h_1 + b_2) \cdot \sigma'(w_1x + b_1) \\ &= w_4 \cdot \prod_{i=2}^{3} w_i\sigma(w_ih_{i-1} + b_i)\big(1 - \sigma(w_ih_{i-1} + b_i)\big) \cdot \sigma(w_1x + b_1)\big(1 - \sigma(w_1x + b_1)\big) \\ &= w_4 \cdot \prod_{i=2}^{3} w_ih_i(1 - h_i) \cdot h_1(1 - h_1) = 93750\end{aligned}$$

(2) By applying the same method in (1), we have $h_1 \approx 0.832, h_2 \approx 0.918, h_3 \approx 0.88, o = h_4 \approx 1.828$.

$$\frac{\partial o}{\partial b_1} = w_4 \cdot \prod_{i=2}^{3} w_ih_i(1 - h_i) \cdot h_1(1 - h_1) \approx 0.000100$$

3. [25pts] CNN

Consider this **one-dimensional** convolutional neural network architecture.



In the first layer, we have a one-dimensional convolution with a single filter of size 3 such that $h_i = \sigma\left(\sum_{j=1}^{3} v_j\, x_{i+j-1}\right)$. The second layer is fully connected, such that $z = \sigma\left(\sum_{i=1}^{4} w_i\, h_i\right)$. The hidden units and output unit's activation function $\sigma(x)$ is the logistic (sigmoid) function with derivative $\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right)$. We perform gradient descent on the loss function $L = (y - z)^2$, where y is the training label for x.

a) [5pts] What is the total number of parameters in this neural network? Recall that convolutional layers share weights. There are no bias terms.

b) [10pts] Compute $\frac{\partial R}{\partial w}$

c) [10pts] Compute $\frac{\partial R}{\partial v_i}$

a) There are $3 + 4 = 7$ parameters, namely $v_1, v_2, v_3, w_1, w_2, w_3$ and w_4.

b)

$$\frac{\partial R}{\partial w} = \frac{\partial R}{\partial z} \cdot \frac{\partial z}{\partial h^T w} \cdot \frac{\partial h^T w}{\partial w} = -2(y - z) \cdot z(1 - z) \cdot h = 2(z - y)z(1 - z)h$$

c)

$$\frac{\partial R}{\partial v_i} = \frac{\partial R}{\partial z} \cdot \frac{\partial z}{\partial h^T w} \cdot \sum_{j=1}^{4} \frac{\partial h^T w}{\partial h_j} \cdot \frac{\partial h_j}{\partial v^T x} \cdot \frac{\partial v^T x}{\partial v_j}$$

$$= -2(y - z) \cdot z(1 - z) \cdot \sum_{j=1}^{4} w_j h_j \left(1 - h_j\right) x_{i+j-1}$$

4. [30pts] CNN

As shown in the figure below, the neural network consists of a convolutional layer and a fully connected layer, with input as $\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}$, convolutional kernel (filter) as $\boldsymbol{K} = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$, convolution result as $\boldsymbol{Y}=\boldsymbol{K} * \boldsymbol{X} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$, and output as $o = \sigma(w_{11}y_{11} + w_{12}y_{12} + w_{21}y_{21} + w_{22}y_{22}), \sigma$ is sigmoid function. The sample label is z, and E is the mean squared error ($E = \frac{1}{2}\|z - o\|^2$). Find the derivative of the convolutional kernel [2].

$$\begin{tikzcd}
\text{input} & \boxed{X} \arrow[r, "\boldsymbol{K} * \boldsymbol{X}"] & \boxed{Y} \arrow[r, "\boldsymbol{W} \odot \mathbf{Y}"] & \boxed{o} \arrow[r] & \text{output}
\end{tikzcd}$$

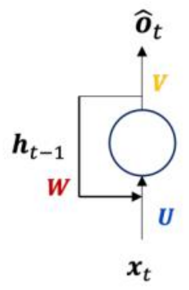
Where, "*" is convolution, "$\odot$" is Hadamard product which is element-wise product.

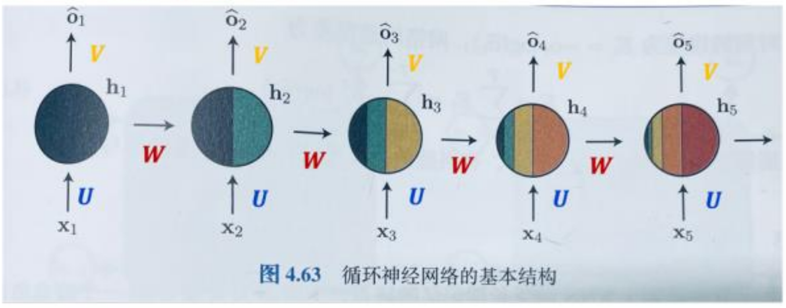
Solution

$$\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{K}} &= \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial \boldsymbol{Y}} \cdot \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{K}} \\
&= (z - o) \cdot \sigma'(\boldsymbol{W} \odot \boldsymbol{Y})\boldsymbol{W} \cdot \mathbb{I}_{2\times 2} * \boldsymbol{X} \\
&= (z - o) \cdot o(1 - o) \cdot \boldsymbol{W} \cdot \begin{bmatrix} x_{11} + x_{22} & x_{12} + x_{23} \\ x_{21} + x_{32} & x_{22} + x_{33} \end{bmatrix}
\end{aligned}$$

5. [30pts] RNN: BPTT

Provide the detailed derivation process of the BPTT (Backpropagation Through Time) algorithm, ensuring that the symbols are consistent with those in the Lecture PPT.





$\hat{o}_1$ $\hat{o}_2$ $\hat{o}_3$ $\hat{o}_4$ $\hat{o}_5$
V
h_1 h_2 h_3 h_4 h_5
W
U
x_1 x_2 x_3 x_4 x_5

图 4.63 循环神经网络的基本结构

Solution

Let's explain the notations we are going to use later.

- $\mathcal{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t+1}, \dots, \boldsymbol{x}_T)$: Input sequence with length of T
- $\boldsymbol{x}_t \in R^{n\times 1}$: Input at time t.
- $\boldsymbol{h}_t \in R^{m\times 1}$: The hidden state at time t.
- $\hat{\boldsymbol{o}}_t \in R^{k\times 1}$: The output at time t.

- $\boldsymbol{o}_t \in R^{k\times 1}$: The real label of each output at time t.
- $\boldsymbol{U} \in R^{m\times n}$: The weights from input layer to hidden layer.
- $\boldsymbol{V} \in R^{k\times m}$: The weights from hidden layer to output layer.
- $\boldsymbol{W} \in R^{m\times m}$: The weights from t_{i-1} to t_i.
- $\boldsymbol{z}_t \in R^{k\times 1}$: All of the input to output layer.

Relations among them:

$$\begin{aligned}
&\boldsymbol{s}_t = \boldsymbol{U}\cdot\boldsymbol{x}_t + \boldsymbol{W}\cdot\boldsymbol{h}_{t-1}\\
&\boldsymbol{h}_t = \tanh(\boldsymbol{s}_t) = \frac{e^{\boldsymbol{s}_t}-e^{-\boldsymbol{s}_t}}{e^{\boldsymbol{s}_t}+e^{-\boldsymbol{s}_t}}\\
&\boldsymbol{z}_t = \boldsymbol{V}\cdot\boldsymbol{h}_t\\
&\widehat{\boldsymbol{o}}_t = g(\boldsymbol{V}\cdot\boldsymbol{h}_t) = \operatorname{softmax}(\boldsymbol{z}_t)\\
&E_t = -\boldsymbol{o}_t\log\widehat{\boldsymbol{o}}_t\\
&E = \sum_{t=1}^{T}E_t = \sum_{t=1}^{T}-\boldsymbol{o}_t\log\widehat{\boldsymbol{o}}_t
\end{aligned}$$

BPTT is similar to BP and is a gradient descent algorithm that backpropagates in time. In RNN, our purpose is to obtain $\frac{\partial E}{\partial U},\frac{\partial E}{\partial W},\frac{\partial E}{\partial V}$, and optimize the three parameters U, V, and W according to these three rates of change.

1. **Calculate $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{V}}$**

$$\frac{\partial E}{\partial \boldsymbol{V}} = \sum_{t=1}^{T}\frac{\partial E}{\partial \boldsymbol{z}_t}\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{V}} = \sum_{t=1}^{T}\sum_{k=1}^{T}\frac{\partial E}{\partial E_k}\frac{\partial E_k}{\partial \widehat{\boldsymbol{o}}_k}\frac{\partial \widehat{\boldsymbol{o}}_k}{\partial \boldsymbol{z}_t}\cdot\boldsymbol{h}_t$$

Here

$$\frac{\partial E}{\partial E_k} = 1, \qquad \frac{\partial E_k}{\partial \widehat{\boldsymbol{o}}_k} = -\frac{\boldsymbol{o}_k}{\widehat{\boldsymbol{o}}_k}$$

For $\frac{\partial \widehat{\boldsymbol{o}}_k}{\partial \boldsymbol{z}_t}$:

$$\widehat{\boldsymbol{o}}_k = \operatorname{softmax}(\boldsymbol{z}_k) = \frac{\mathrm{e}^{\boldsymbol{z}_k}}{\sum_{i=1}^{T}e^{\boldsymbol{z}_i}}$$

When $k = t$,

$$\begin{aligned}
\frac{\partial \widehat{\boldsymbol{o}}_t}{\partial \boldsymbol{z}_t} &= \frac{\partial}{\boldsymbol{z}_t}\left(\frac{\mathrm{e}^{\boldsymbol{z}_t}}{\sum_{i=1}^{T}e^{\boldsymbol{z}_i}}\right) = \frac{e^{\boldsymbol{z}_t}\left(\sum_{i=1}^{T}e^{\boldsymbol{z}_i}\right)-e^{\boldsymbol{z}_t}e^{\boldsymbol{z}_t}}{\left(\sum_{i=1}^{T}e^{\boldsymbol{z}_i}\right)^2}\\
&= \frac{e^{\boldsymbol{z}_t}}{\sum_{i=1}^{T}e^{\boldsymbol{z}_i}}\left(1-\frac{e^{\boldsymbol{z}_t}}{\sum_{i=1}^{T}e^{\boldsymbol{z}_i}}\right) = \widehat{\boldsymbol{o}}_t(1-\widehat{\boldsymbol{o}}_t)
\end{aligned}$$

When $k \neq t$,

$$\frac{\partial \widehat{\boldsymbol{o}}_k}{\partial \boldsymbol{z}_t} = \frac{\partial}{\boldsymbol{z}_t}\left(\frac{\mathrm{e}^{\boldsymbol{z}_k}}{\sum_{i=1}^{T} e^{\boldsymbol{z}_i}}\right) = \frac{0 - e^{\boldsymbol{z}_k} e^{\boldsymbol{z}_t}}{(\sum_{i=1}^{T} e^{\boldsymbol{z}_i})^2}$$
$$= -\frac{e^{\boldsymbol{z}_k}}{\sum_{i=1}^{T} e^{\boldsymbol{z}_i}} \frac{e^{\boldsymbol{z}_t}}{\sum_{i=1}^{T} e^{\boldsymbol{z}_i}} = -\widehat{\boldsymbol{o}}_t \widehat{\boldsymbol{o}}_k$$

Eventually

$$\frac{\partial E}{\partial \boldsymbol{V}} = \sum_{t=1}^{T}\left(-\boldsymbol{o}_t(1-\widehat{\boldsymbol{o}}_t) + \sum_{k=1,k\neq t}^{T} \boldsymbol{o}_k \widehat{\boldsymbol{o}}_t\right) \cdot \boldsymbol{h}_t$$
$$= \sum_{t=1}^{T}\left(\widehat{\boldsymbol{o}}_t \sum_{k=1}^{T} \boldsymbol{o}_k - \boldsymbol{o}_t\right) \cdot \boldsymbol{h}_t$$
$$= \sum_{t=1}^{T} (\widehat{\boldsymbol{o}}_t - \boldsymbol{o}_t) \cdot \boldsymbol{h}_t$$

2. **Calculate $\frac{\partial E}{\partial \boldsymbol{U}}$**

$$\frac{\partial E}{\partial \boldsymbol{U}} = \sum_{t=1}^{T} \frac{\partial E}{\partial \boldsymbol{s}_t} \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{U}}$$

Here

$$\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{U}} = \boldsymbol{x}_t$$

For $\frac{\partial E}{\partial \boldsymbol{s}_t}$, note that $\boldsymbol{h}_t$ is depend on $\boldsymbol{h}_{t-1}$, so we should calculate $\frac{\partial E}{\partial \boldsymbol{s}_T}$ first.

$$\frac{\partial E}{\partial \boldsymbol{s}_T} = \frac{\partial E}{\partial \boldsymbol{z}_T} \frac{\partial \boldsymbol{z}_T}{\partial \boldsymbol{s}_T} = \frac{\partial E}{\partial \boldsymbol{z}_T} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{s}_t} = \frac{\partial E}{\partial \boldsymbol{z}_T} \cdot \boldsymbol{V} \cdot \operatorname{diag}(\boldsymbol{1} - \tanh^2 \boldsymbol{s}_T)$$

Then

$$\frac{\partial E}{\partial \boldsymbol{s}_t} = \frac{\partial E}{\partial \boldsymbol{s}_{t+1}} \frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{s}_t} + \frac{\partial E}{\partial \boldsymbol{z}_t} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{s}_t} = \frac{\partial E}{\partial \boldsymbol{s}_{t+1}} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{s}_t} \frac{\partial \boldsymbol{s}_{t+1}}{\partial \boldsymbol{h}_t} + \frac{\partial E}{\partial \boldsymbol{z}_t} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{s}_t} \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{h}_t}$$
$$= \operatorname{diag}(\boldsymbol{1} - \tanh^2 \boldsymbol{s}_t)\left(\frac{\partial E}{\partial \boldsymbol{s}_{t+1}} \cdot \boldsymbol{W} + \frac{\partial E}{\partial \boldsymbol{z}_t} \cdot \boldsymbol{V}\right), \quad t = T-1, \cdots, 2, 1$$

3. **Calculate $\frac{\partial E}{\partial \boldsymbol{W}}$**

$$\frac{\partial E}{\partial \boldsymbol{W}} = \sum_{t=1}^{T} \frac{\partial E}{\partial \boldsymbol{s}_t} \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{W}} = \sum_{t=1}^{T} \frac{\partial E}{\partial \boldsymbol{s}_t} \boldsymbol{h}_{t-1}$$