Chapter 11

RECURRENT NEURAL NETWORK

CONTENTS

Recurrent neural networks (RNNs) is particularly of use in temporally sequential data alternative to the usual time series data analysis so that RNNs can either classify the entire sequence or generate the subsequent data. The input features are an ordered list of vectors $\boldsymbol{x}_t$, for $t = 0, 1, \cdots$, and the main difference, in terms of the network structure, from the usual MLP is that, some outputs of the hidden layers will be fed backwardly as the inputs for itself or for previous layers. RNNs are often used in text mining and speech recognition because sentences of words, punctuation marks, and texts are naturally appearing in a sequential order.

Let us introduce RNN in a more formal manner. As aforementioned, it does not just feed-forward, and it even contains network loops other than proceeding from the input layer to the output layer, the network ascends along the layer hierachy, it also processes data with the time passing measured in $t = 0, 1, \cdots$. Each neuron j of a recurrent (hidden) layer d_ℓ has a real-valued vector state $h_{j,t}^{(\ell)}$, which can be regarded as the memory of this neuron unit j. By then, each unit j in each layer ℓ receives two sources of inputs:

1. $\boldsymbol{h}_t^{(\ell-1)}$: A vector of states from the previous layer $\ell - 1$ at the time step t;
2. $\boldsymbol{h}_{t-1}^{(\ell)}$: A vector of states of output from this present layer ℓ but at the previous time step $t - 1$.

11.1 More on Matrix Calculus

In Section 3.1, we have defined a set of notations for the gradient of a function f with respect to a vector or a matrix. However, since the backpropagation through time in RNNs requires a heavy use of chain rules for functions taking values as vector or matrix, for the sake of convenience, we here adopt another set of notations:

1. If $\boldsymbol{x} \in \mathbb{R}^N$ is a vector variable and $f(\boldsymbol{x})$ is a real-valued function differentiable in $\boldsymbol{x}$, then
$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) := \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{pmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_N} \end{pmatrix} \in \mathbb{R}^N .$$
2. If $\boldsymbol{x} \in \mathbb{R}^N$ is a vector variable and $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M$ is a vector-valued function differentiable in $\boldsymbol{x}$, then
$$\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) := \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{pmatrix} \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \dfrac{\partial f_M(\boldsymbol{x})}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_N} & \cdots & \dfrac{\partial f_M(\boldsymbol{x})}{\partial x_N} \end{pmatrix} = \left(\begin{array}{c|c|c|c} \dfrac{\partial f_1(\boldsymbol{x})}{\partial \boldsymbol{x}} & \dfrac{\partial f_2(\boldsymbol{x})}{\partial \boldsymbol{x}} & \cdots & \dfrac{\partial f_M(\boldsymbol{x})}{\partial \boldsymbol{x}} \end{array} \right) \in \mathbb{R}^{N \times M} .$$
3. If $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ is a matrix variable and $\boldsymbol{f}(\boldsymbol{X}) \in \mathbb{R}^K$ is a vector-valued function differentiable in $\boldsymbol{X}$, then
$$\nabla_{\boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X}) := \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}} = \begin{pmatrix} \dfrac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial x_{1,1}} & \cdots & \dfrac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial x_{1,M}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial x_{N,1}} & \cdots & \dfrac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial x_{N,M}} \end{pmatrix} \in \mathbb{R}^{N \times M \times K} ,$$
which is a 3^{rd} order tensor.

In backpropagation of MLP, we only consider the chain rule in scalar form as discussed in Subsubsection 9.1.1.2. In general, the chain rule against the independent variable $x \in \mathbb{R}$ on another scalar function $\mathcal{E}$ that depends x through another function h is
$$\frac{\partial \mathcal{E}}{\partial x} = \frac{\partial \mathcal{E}}{\partial h} \frac{\partial h}{\partial x} = \frac{\partial h}{\partial x} \frac{\partial \mathcal{E}}{\partial h}$$
However, when dealing with a column vector $\boldsymbol{x} \in \mathbb{R}^N$, the chain rule builds "from the right to the left", *i.e.*
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \frac{\partial \mathcal{E}}{\partial \boldsymbol{h}} , \tag{11.1.1}$$
the product here is a matrix multiplication. Before we proceed forward, we first introduce three more mathematical operations:

Definition 11.1.1 (*Kronecker product* $\otimes$)**.** Given two real matrices $\boldsymbol{A} = (a_{n,m})_{n=1,m=1}^{N,M}$ of dimensions $N \times M$ and $\boldsymbol{B} = (b_{k\ell})_{k=1,\ell=1}^{K,L}$ of dimensions $K \times L$, their Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is defined as another matrix of dimensions $(N \cdot K) \times (M \cdot L)$ that
$$\boldsymbol{A} \otimes \boldsymbol{B} = \left(\begin{array}{c|c|c} a_{11}\boldsymbol{B} & \cdots & a_{1M}\boldsymbol{B} \\ \hline \vdots & \ddots & \vdots \\ \hline a_{N1}\boldsymbol{B} & \cdots & a_{NM}\boldsymbol{B} \end{array} \right) \in \mathbb{R}^{(N \cdot K) \times (M \cdot L)} ,$$

with entries, for $n = 1, \cdots, N$, $m = 1, \cdots, M$, $k = 1, \cdots, K$, and $\ell = 1, \cdots, L$,

$$(\boldsymbol{A} \otimes \boldsymbol{B})_{K\cdot(n-1)+k, L\cdot(m-1)+\ell} := a_{nm} \cdot b_{k\ell} \, .$$

Definition 11.1.2 (*vectorization*)**.** Let $\boldsymbol{X} = (\boldsymbol{x}_{nm})_{n=1,m=1}^{N,M}$ be a third order tensor in $\mathbb{V}^{N\times M}$, where $\boldsymbol{x}_{nm}$ takes vector-value in a vector space $\mathbb{V}$, then the vectorized form $\text{vec}(\boldsymbol{X})$ of $\boldsymbol{X}$ is a second order tensor

$$\text{vec}(\boldsymbol{X}) := (\boldsymbol{x}_{11}, \cdots, \boldsymbol{x}_{N1}, \boldsymbol{x}_{12}, \cdots, \boldsymbol{x}_{N2}, \cdots, \boldsymbol{x}_{1M}, \cdots, \boldsymbol{x}_{NM})^\top \in \mathbb{V}^{N\cdot M} \, .$$

That is to say we append consecutive columns together to form a long column vector. Besides, its inverse operation $\text{vec}^{-1} : \mathbb{V}^{N\cdot M} \to \mathbb{V}^{N\times M}$ can be defined by splitting the long column of entries into equal length of N of sub-vectors, each of which are then arranged as consecutive columns in order to form a third order tensor, and so

$$\text{vec}^{-1}[\text{vec}(\boldsymbol{X})] = \boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_{11} & \cdots & \boldsymbol{x}_{1M} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_{N1} & \cdots & \boldsymbol{x}_{NM} \end{pmatrix} \in \mathbb{V}^{N\times M} \, .$$

Lemma 11.1.1. Let $\boldsymbol{A} \in \mathbb{R}^{K\times N}$, $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M) \in \mathbb{R}^{N\times M}$, where each $\boldsymbol{x}_m \in \mathbb{R}^N$ for $m = 1, \cdots, M$, and $\boldsymbol{B} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_L) \in \mathbb{R}^{M\times L}$ so that $b_\ell \in \mathbb{R}^M$ for $\ell = 1, \cdots, L$, then we have

$$\text{vec}(\boldsymbol{AXB}) = (\boldsymbol{B}^\top \otimes \boldsymbol{A})\text{vec}(\boldsymbol{X}) \, .$$

Proof. From Definition 11.1.2, we have $\mathbb{V} = \mathbb{R}$, we then consider the ℓ^{th} column, for $\ell = 1, \cdots, L$, of $\boldsymbol{AXB}$:

$$(\boldsymbol{AXB})_{\cdot,\ell} = \boldsymbol{AXb}_\ell = \boldsymbol{A}\sum_{m=1}^{M} \boldsymbol{x}_m b_{m,\ell} = \sum_{m=1}^{M} (b_{m,\ell}\boldsymbol{A})\boldsymbol{x}_m = \begin{pmatrix} b_{1,\ell}\boldsymbol{A} & b_{2,\ell}\boldsymbol{A} & \cdots & b_{M,\ell}\boldsymbol{A} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_M \end{pmatrix}$$

$$= \left(\begin{pmatrix} b_{1,\ell} & b_{2,\ell} & \cdots & b_{M,\ell} \end{pmatrix} \otimes \boldsymbol{A} \right) \text{vec}(\boldsymbol{X}) = (\boldsymbol{b}_\ell^\top \otimes \boldsymbol{A})\text{vec}(\boldsymbol{X}) \, .$$

Finally, stacking the columns together vertically, we see that

$$\boldsymbol{AXB} = \begin{pmatrix} \boldsymbol{b}_1^\top \otimes \boldsymbol{A} \\ \boldsymbol{b}_2^\top \otimes \boldsymbol{A} \\ \vdots \\ \boldsymbol{b}_L^\top \otimes \boldsymbol{A} \end{pmatrix} \text{vec}(\boldsymbol{X}) = \left[\begin{pmatrix} \boldsymbol{b}_1^\top \\ \boldsymbol{b}_2^\top \\ \vdots \\ \boldsymbol{b}_L^\top \end{pmatrix} \otimes \boldsymbol{A} \right] \text{vec}(\boldsymbol{X}) = (\boldsymbol{B}^\top \otimes \boldsymbol{A})\text{vec}(\boldsymbol{X}) \, ,$$

therefore the claim is concluded. □

Lemma 11.1.2. Let $\boldsymbol{f}(\boldsymbol{X}) : \mathbb{R}^{N\times M} \to \mathbb{R}^K$ be a vector-valued function differentiable in a matrix variable $\boldsymbol{X}$, then $(f \circ \text{vec}^{-1})$ is differentiable in $\text{vec}(\boldsymbol{X})$, and as a long vector in $(\mathbb{R}^K)^{N\times M}$ with an entries in $\mathbb{V} = (\mathbb{R}^K)^\top$,

$$\text{vec}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) = \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \text{vec}(\boldsymbol{X})} \in \left[(\mathbb{R}^K)^\top\right]^{N\times M} = \mathbb{R}^{N\times M\times K} \, .$$

Proof. Using Definition 11.1.2, since

$$\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}} = \left(\frac{\partial f_k(\boldsymbol{X})}{\partial x_{nm}}\right)_{n=1,m=1,k=1}^{N,M,K} = \left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{nm}}\right)_{n=1,m=1}^{N,M} , \qquad \text{where} \quad \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial x_{nm}} \in \mathbb{V} = \mathbb{R}^K \, ,$$

we have

$$\text{vec}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) = \left(\begin{array}{ccc|ccc|c|ccc} \frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{11}} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{N1}} & \frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{12}} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{N2}} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{1M}} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{X})^\top}{\partial x_{NM}} \end{array}\right) = \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \text{vec}(\boldsymbol{X})}\,.$$

□

Remark 11.1.1. To ease the notational complexity, let $\boldsymbol{A} = (a_{nm})_{n=1,m=1}^{N,M} \in \mathbb{R}^{N\times M}$ and $\boldsymbol{v} \in \mathbb{R}^K$, in the following we abuse the notation that the third order tensor can be rewritten:

$$\begin{pmatrix} a_{11}\boldsymbol{v}^\top & \cdots & a_{1M}\boldsymbol{v}^\top \\ \vdots & \ddots & \vdots \\ a_{N1}\boldsymbol{v}^\top & \cdots & a_{NM}\boldsymbol{v}^\top \end{pmatrix} := \begin{pmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} \end{pmatrix} \boldsymbol{v}^\top \in \left[(\mathbb{R}^K)^\top\right]^{N\times M} = \mathbb{R}^{N\times M\times K}\,.$$

Using the afore-defined notations, we here provide some examples that are commonly evaluated in DNN:

1. Note that the derivative of a vector $\boldsymbol{x} \in \mathbb{R}^N$ with respect to its own is the identity matrix $\boldsymbol{I}_N$. Therefore, if we differentiate $\boldsymbol{x} \odot \boldsymbol{b}$ with respect to $\boldsymbol{x}$, then
$$\frac{\partial \boldsymbol{x}\odot\boldsymbol{b}}{\partial \boldsymbol{x}} = \text{diag}\{\boldsymbol{b}\} \in \mathbb{R}^{N\times N}\,, \tag{11.1.2}$$
where $\text{diag}\{\boldsymbol{b}\}$ is the diagonal matrix of the vector $\boldsymbol{b} = (b_1, \cdots, b_N)^\top \in \mathbb{R}^N$, *i.e.*
$$\text{diag}\left\{\boldsymbol{b}\right\} = \begin{pmatrix} b_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & b_N \end{pmatrix} \in \mathbb{R}^{N\times N}\,.$$
Moreover, if we multiply it with a vector $\boldsymbol{c} \in \mathbb{R}^N$, then
$$\text{diag}\left\{\boldsymbol{b}\right\}\boldsymbol{c} = \boldsymbol{b}\odot\boldsymbol{c}\,. \tag{11.1.3}$$

2. In the training of MLP, we often evaluate $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{A}\boldsymbol{x})$, where $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ does not involve $\boldsymbol{x}$, using (11.1.1), then
$$\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{A}\boldsymbol{x}} = \boldsymbol{A}^\top \frac{\partial \boldsymbol{\sigma}(\boldsymbol{A}\boldsymbol{x})}{\partial(\boldsymbol{A}\boldsymbol{x})}\,,$$
where we recall Property 9 in Section 3.4. Moreover, in MLP, $\sigma_i(\boldsymbol{A}\boldsymbol{x})$ and $(\boldsymbol{A}\boldsymbol{x})_j$ are often assumed to be independent for $i \neq j$, while $\sigma_i(\boldsymbol{A}\boldsymbol{x})$ solely depends on $(\boldsymbol{A}\boldsymbol{x})_i$, then we can further reduce the derivative to
$$\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{A}^\top \text{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{A}\boldsymbol{x})\right\}\,.$$

3. To be seen in most RNN problems, consider $\boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{a} \in \mathbb{R}^N$, where $\boldsymbol{a} \in \mathbb{R}^M$ is a fixed vector and $\boldsymbol{X} \in \mathbb{R}^{N\times M}$ is a matrix variable. We then consider the gradient of $\boldsymbol{f}(\boldsymbol{X})$ with respect to a vectorized $\boldsymbol{X}$, $\text{vec}(\boldsymbol{X}) = (x_{11}, \cdots, x_{N1}, x_{12}, \cdots, x_{N2}, \cdots, x_{1M}, \cdots, x_{NM})^\top \in \mathbb{R}^{N\cdot M}$. By Lemma 11.1.1, we can rewrite, being a $\mathbb{R}^N$-vector, $\boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{I}_N\boldsymbol{X}\boldsymbol{a} = \text{vec}(\boldsymbol{I}_N\boldsymbol{X}\boldsymbol{a}) = (\boldsymbol{a}^\top \otimes \boldsymbol{I}_N)\text{vec}(\boldsymbol{X})$, then one can use, again, Property 9 in Section 3.4,
$$\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \text{vec}(\boldsymbol{X})} = \frac{\partial(\boldsymbol{a}^\top\otimes\boldsymbol{I}_N)\text{vec}(\boldsymbol{X})}{\partial \text{vec}(\boldsymbol{X})} = (\boldsymbol{a}^\top\otimes\boldsymbol{I}_N)^\top = \left(\begin{array}{c|c|c} a_1\boldsymbol{I}_N & \cdots & a_M\boldsymbol{I}_N \end{array}\right)^\top \in \left[(\mathbb{R}^N)^\top\right]^{(N\cdot M)} = \mathbb{R}^{(N\cdot M)\times N}\,.$$
By Lemma 11.1.2, we have
$$\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}} = \text{vec}^{-1}\left[\text{vec}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)\right] = \text{vec}^{-1}\left[\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \text{vec}(\boldsymbol{X})}\right] = \text{vec}^{-1}(\boldsymbol{a}\otimes\boldsymbol{I}_N) \in \left[(\mathbb{R}^N)^\top\right]^{N\times M} = \mathbb{R}^{N\times M\times N}\,,$$

which is actually a third order tensor; to further see its effect, let $\boldsymbol{e}_k = (0, \cdots, 0, 1, 0, \cdots 0)^\top \in \mathbb{R}^N$ taking a value of 1 only at the k^{th} position or otherwise zero, then

$$\left(\begin{array}{c} a_1\boldsymbol{I}_N \\ \hline a_2\boldsymbol{I}_N \\ \hline \vdots \\ \hline a_M\boldsymbol{I}_N \end{array}\right) \overset{\text{vec}^{-1}}{\mapsto} \left(\begin{array}{cccc} a_1\boldsymbol{I}_N & a_2\boldsymbol{I}_N & \cdots & a_M\boldsymbol{I}_N \end{array}\right) = \begin{pmatrix} a_1\boldsymbol{e}_1^\top & a_2\boldsymbol{e}_1^\top & \cdots & a_M\boldsymbol{e}_1^\top \\ a_1\boldsymbol{e}_2^\top & a_2\boldsymbol{e}_2^\top & \cdots & a_M\boldsymbol{e}_2^\top \\ \vdots & \vdots & \ddots & \vdots \\ a_1\boldsymbol{e}_N^\top & a_2\boldsymbol{e}_N^\top & \cdots & a_M\boldsymbol{e}_N^\top \end{pmatrix}$$

$$= \sum_{k=1}^{N} \left(\begin{array}{c|c|c|c} a_1\boldsymbol{e}_k & a_2\boldsymbol{e}_k & \cdots & a_M\boldsymbol{e}_k \end{array}\right) \boldsymbol{e}_k^\top \in \left[(\mathbb{R}^N)^\top\right]^{N\times M},$$

defined in Remark 11.1.1, where, for $k = 1, \cdots, N$,

$$\left(\begin{array}{c|c|c|c} a_1\boldsymbol{e}_k & a_2\boldsymbol{e}_k & \cdots & a_M\boldsymbol{e}_k \end{array}\right) = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_M \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{N\times M},$$

which only have the k^{th} row with possibly non-zero entries; also see Figure 11.1.1. Moreover, if we further multiply $\nabla_{\boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X})$ by a vector $\boldsymbol{b} \in \mathbb{R}^N$ on right, then

$$\begin{aligned} \frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\boldsymbol{b} = \frac{\partial \boldsymbol{X}\boldsymbol{a}}{\partial \boldsymbol{X}}\boldsymbol{b} &= \text{vec}^{-1}(\boldsymbol{a}\otimes\boldsymbol{I}_N)\boldsymbol{b} = \left(\begin{array}{cccc} a_1\boldsymbol{I}_N & a_2\boldsymbol{I}_N & \cdots & a_M\boldsymbol{I}_N \end{array}\right)\boldsymbol{b} \\ &= \left(\begin{array}{c|c|c|c} a_1\boldsymbol{b} & a_2\boldsymbol{b} & \cdots & a_M\boldsymbol{b} \end{array}\right) = \boldsymbol{b}\boldsymbol{a}^\top \in \mathbb{R}^{N\times M}. \end{aligned} \tag{11.1.4}$$

First slice coordinates for $\boldsymbol{e}_1^\top$

a_1	a_2	$\cdots$	a_M
0	0	$\cdots$	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$
0	0	$\cdots$	0

Second slice coordinates for $\boldsymbol{e}_2^\top$

0	0	$\cdots$	0
a_1	a_2	$\cdots$	a_M
$\vdots$	$\vdots$	$\ddots$	$\vdots$
0	0	$\cdots$	0

$\cdots$

M^{th} slice coordinates for $\boldsymbol{e}_N^\top$

0	0	$\cdots$	0
0	0	$\cdots$	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_1	a_2	$\cdots$	a_M

Figure 11.1.1: N slices of the first two dimensions in $\mathbb{R}^{N\times M}$ of $\nabla_{\boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X}) = \nabla_{\boldsymbol{X}}\boldsymbol{X}\boldsymbol{a}$.

Example 11.1.1. Using the set of notations above and (11.1.1), the backpropagation of a MLP discussed in Subsubsection 9.1.1.2 can be rewritten as:

$$\begin{aligned} \boldsymbol{\delta}^{(L,t)} &:= \frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{z}^{(L,t)}} = \frac{\partial \boldsymbol{a}^{(L,t)}}{\partial \boldsymbol{z}^{(L,t)}}\frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{a}^{(L,t)}} = \frac{\partial \boldsymbol{f}_L(\boldsymbol{z}^{(L,t)})}{\partial \boldsymbol{z}^{(L,t)}}\frac{\partial}{\partial \boldsymbol{a}^{(L,t)}}\left(\frac{1}{2}(\boldsymbol{y}^{(t)} - \boldsymbol{a}^{(L,t)})^\top(\boldsymbol{y}^{(t)} - \boldsymbol{a}^{(L,t)})\right) \\ &= \text{diag}\left\{\boldsymbol{f}_L'(\boldsymbol{z}^{(L,t)})\right\}(\boldsymbol{a}^{(L,t)} - \boldsymbol{y}^{(t)}) = \boldsymbol{f}_L'(\boldsymbol{z}^{(L,t)}) \odot (\boldsymbol{a}^{(L,t)} - \boldsymbol{y}^{(t)}), \end{aligned}$$

the second last equality is directly followed from (11.1.2). Moreover,

$$\boldsymbol{\delta}^{(\ell,t)} := \frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{z}^{(\ell,t)}} = \frac{\partial \boldsymbol{z}^{(\ell+1,t)}}{\partial \boldsymbol{z}^{(\ell,t)}} \frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{z}^{(\ell+1,t)}} = \frac{\partial \boldsymbol{a}^{(\ell,t)}}{\partial \boldsymbol{z}^{(\ell,t)}} \frac{\partial \boldsymbol{z}^{(\ell+1,t)}}{\partial \boldsymbol{a}^{(\ell,t)}} \frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{z}^{(\ell+1,t)}}$$
$$= \operatorname{diag}\{\boldsymbol{f}'_\ell(\boldsymbol{z}^{(\ell,t)})\} \left(\boldsymbol{W}^{(\ell+1,t-1)}\right)^\top \boldsymbol{\delta}^{(\ell+1,t)} = \left\{ \left(\boldsymbol{W}^{(\ell+1,t-1)}\right)^\top \boldsymbol{\delta}^{(\ell+1,t)} \right\} \odot \boldsymbol{f}'_\ell(\boldsymbol{z}^{(\ell,t)}) \,,$$

where the second last equality follows by (11.1.2) for the first term and for the second term by Property 9 in Section 3.4. Finally, for the weights, by using (11.1.4), we have

$$\frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{W}^{(\ell,t-1)}} = \frac{\partial \boldsymbol{z}^{(\ell,t)}}{\partial \boldsymbol{W}^{(\ell,t-1)}} \frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{z}^{(\ell,t)}} = \frac{\partial (\boldsymbol{W}^{(\ell,t-1)} \boldsymbol{a}^{(\ell-1,t)})}{\partial \boldsymbol{W}^{(\ell,t-1)}} \boldsymbol{\delta}^{(\ell,t)} = \boldsymbol{\delta}^{(\ell,t)} \left(\boldsymbol{a}^{(\ell-1,t)}\right)^\top = \boldsymbol{\delta}^{(\ell,t)} \left(\boldsymbol{f}_{\ell-1}(\boldsymbol{z}^{(\ell-1,t)})\right)^\top ;$$

and then that for biases

$$\frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{b}^{(\ell,t-1)}} = \frac{\partial \boldsymbol{z}^{(\ell,t)}}{\partial \boldsymbol{b}^{(\ell,t-1)}} \frac{\partial \mathcal{E}^{(t)}}{\partial \boldsymbol{z}^{(\ell,t)}} = \frac{\partial \boldsymbol{b}^{(\ell,t-1)}}{\partial \boldsymbol{b}^{(\ell,t-1)}} \boldsymbol{\delta}^{(\ell,t)} = \boldsymbol{I}_{d_\ell} \boldsymbol{\delta}^{(\ell,t)} = \boldsymbol{\delta}^{(\ell,t)} \,.$$

11.2 Process in RNN

As aforementioned, RNN takes a sequential data matrix $\boldsymbol{X}$ as input, in which each datum is essentially a sequence of vectors $\boldsymbol{x}_t \in \mathbb{R}^d$, for $t = 1, \cdots, D$ and some $d \in \mathbb{Z}^+$, and we say that the data input is of a length D:

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_D) \,.$$

For instance, if the input sample $\boldsymbol{X}$ is a text sentence, then the feature vector $\boldsymbol{x}_t$, for each $t = 1, \cdots, D$, represents a word in a sentence at time t. For instance, Table 11.2.1 shows a sequence of input feature vectors of length 8 in a movie review for sentiment analysis, in which a positive review indicating a favourable consideration has a label of $y = 1$; otherwise, $y = 0$, implies an inferior performance.

Data	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	$\boldsymbol{x}_6$	$\boldsymbol{x}_7$	$\boldsymbol{x}_8$	$\boldsymbol{y}$
1	The	first	half	was	very	boring	.		0
2	Great	performance	by	all	the	leading	actors	.	1
3	The	visual	effects	were	stunning	.			1
4	The	movie	was	a	waste	of	time	.	0

Table 11.2.1: The inputs and labels of an RNN in a movie review for sentiment analysis.

11.2.1 Forward Propagation of a Temporal Dataset

The forward propagation of a L-layer RNN takes the form, for $t = 1, 2, \cdots, D$,

(*Cell output*) $$\boldsymbol{h}_t^{(\ell)} = \boldsymbol{f}_\ell(\boldsymbol{W}_h^{(\ell)} \boldsymbol{h}_t^{(\ell-1)} + \boldsymbol{V}_h^{(\ell)} \boldsymbol{h}_{t-1}^{(\ell)} + \boldsymbol{b}_h^{(\ell)}) \,, \qquad \text{for} \quad \ell = 1, \cdots, L-1 \,; \tag{11.2.1a}$$

(*Final output*) $$\boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{W}_o \boldsymbol{h}_t^{(L-1)} + \boldsymbol{b}_o) \,. \tag{11.2.1b}$$

For any single layer ℓ, every neuron or now called *memory cell* uses the same set of coefficients $\boldsymbol{W}_h^{(\ell)}, \boldsymbol{V}_h^{(\ell)}$, and $\boldsymbol{b}_h^{(\ell)}$ being independent of time t, it means that the model is temporal homogeneous. For example, Figure 11.2.1 illustrates the first three timestep of a 2-layer RNN with the activation function $\boldsymbol{f}_\ell$, for $\ell = 1, 2$, being the hyperbolic tangent function:

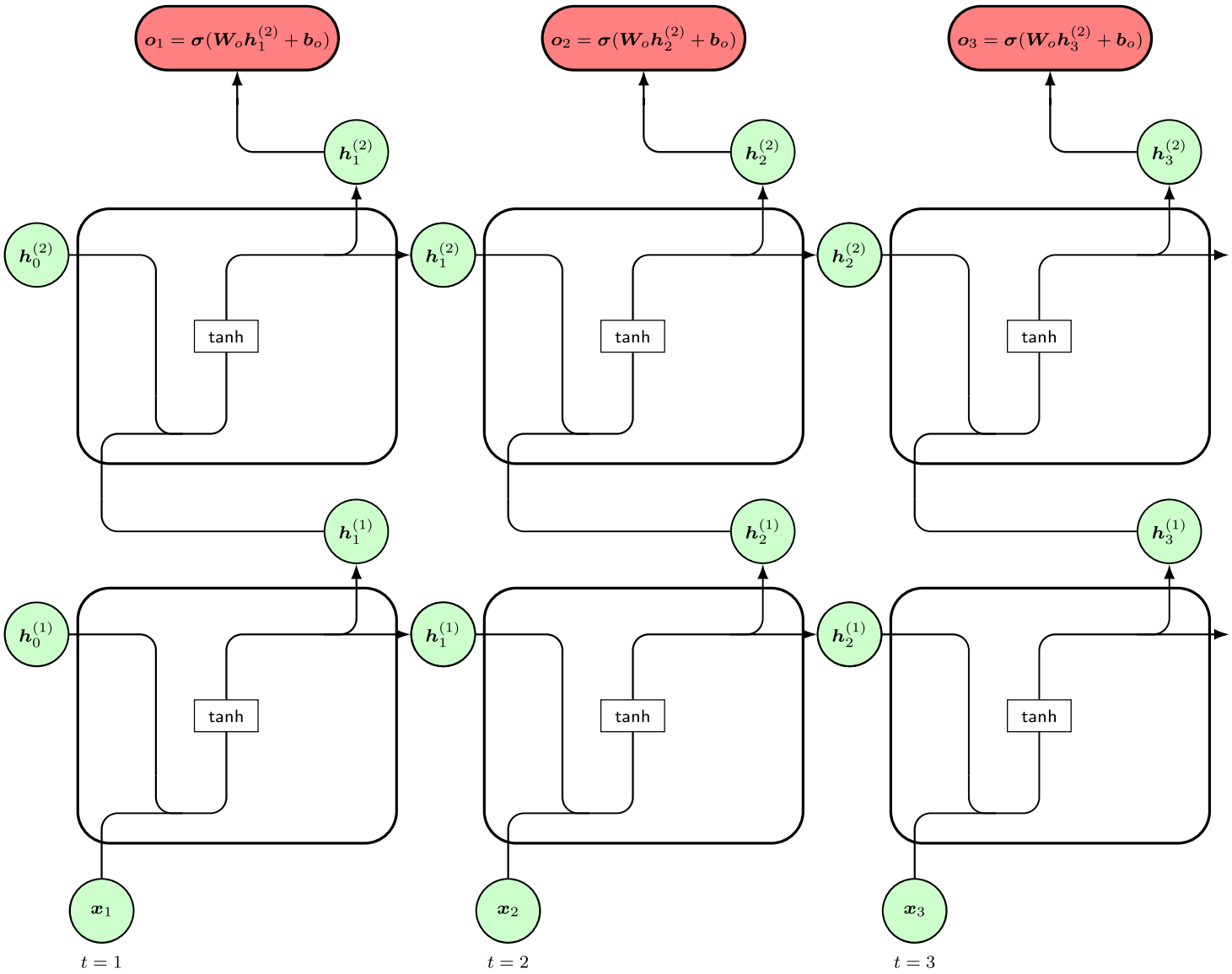

Figure 11.2.1: The first three timestep of a two-layer RNN; also see [4].

The dimension of $\boldsymbol{W}_o^{(\ell)}$ is chosen such that the multiplicative product of the matrix $\boldsymbol{W}_o^{(\ell)}$ and the vector $\boldsymbol{h}_t^{(\ell)}$ has the same dimension as that of the bias vector $\boldsymbol{b}_o^{(\ell)}$; in return, their choices also depend on the very dimension of the output label $\boldsymbol{y}$ in the training dataset. In a typical MLP, the weights and biases are updated through the backpropagation of the training dataset, as introduced in Subsubsection 9.1.1.2; but for this temporal dependent RNN, a modified version of the usual backpropagation is now considered to update $\boldsymbol{W}_h^{(\ell)}$, $\boldsymbol{V}_h^{(\ell)}$, $\boldsymbol{b}_h^{(\ell)}$, $\boldsymbol{W}_o^{(\ell)}$, and $\boldsymbol{b}_o^{(\ell)}$, which is called *backpropagation through time*; see Subsection 11.2.3 for a detailed discussion. Lastly, the initial states $\boldsymbol{h}_0^{(\ell)}$ can be fixed parameters, *i.e.* initializing $\boldsymbol{h}_0^{(\ell)}$ with random numbers, or learnable parameter, *i.e.* initializing $\boldsymbol{h}_0^{(\ell,0)}$ with random numbers and then updating $\boldsymbol{h}_0^{(\ell,k)}$ for each iteration k with backpropagation through time. For simplicity, to avoid unnecessary technicalities, we shall only consider fixed initial states $\boldsymbol{h}_0^{(\ell)}$ in the rest of our discussions.

11.2.2 An Illustrative Example of RNN

Figure 11.2.2 considers a particular timestep t of the 2-layer RNN shown in Figure 11.2.1:

1. The first (leftmost) layer receives as input a feature vector $\boldsymbol{x}_t = (x_{1,t}, x_{2,t})^\top \in \mathbb{R}^2$ and the output of the previous timestep from the same layer $\boldsymbol{h}_{t-1}^{(\ell)} = (h_{1,t-1}^{(\ell)}, h_{2,t-1}^{(\ell)}) \in \mathbb{R}^2$; and
2. The second layer receives the output of the first layer as input $\boldsymbol{h}_t^{(\ell-1)} = (h_{1,t}^{(\ell-1)}, h_{2,t}^{(\ell-1)})^\top \in \mathbb{R}^2$.

Therefore, for $d = 2$, we have, for $\ell = 1, 2$,

$$\boldsymbol{W}_h^{(\ell)} = \begin{pmatrix} \boldsymbol{\omega}_{h,1}^{(\ell)} & \boldsymbol{\omega}_{h,2}^{(\ell)} \end{pmatrix}^\top \qquad \text{and} \qquad \boldsymbol{b}_h^{(\ell)} = (b_{h,1}^{(\ell)}, b_{h,2}^{(\ell)})^\top .$$

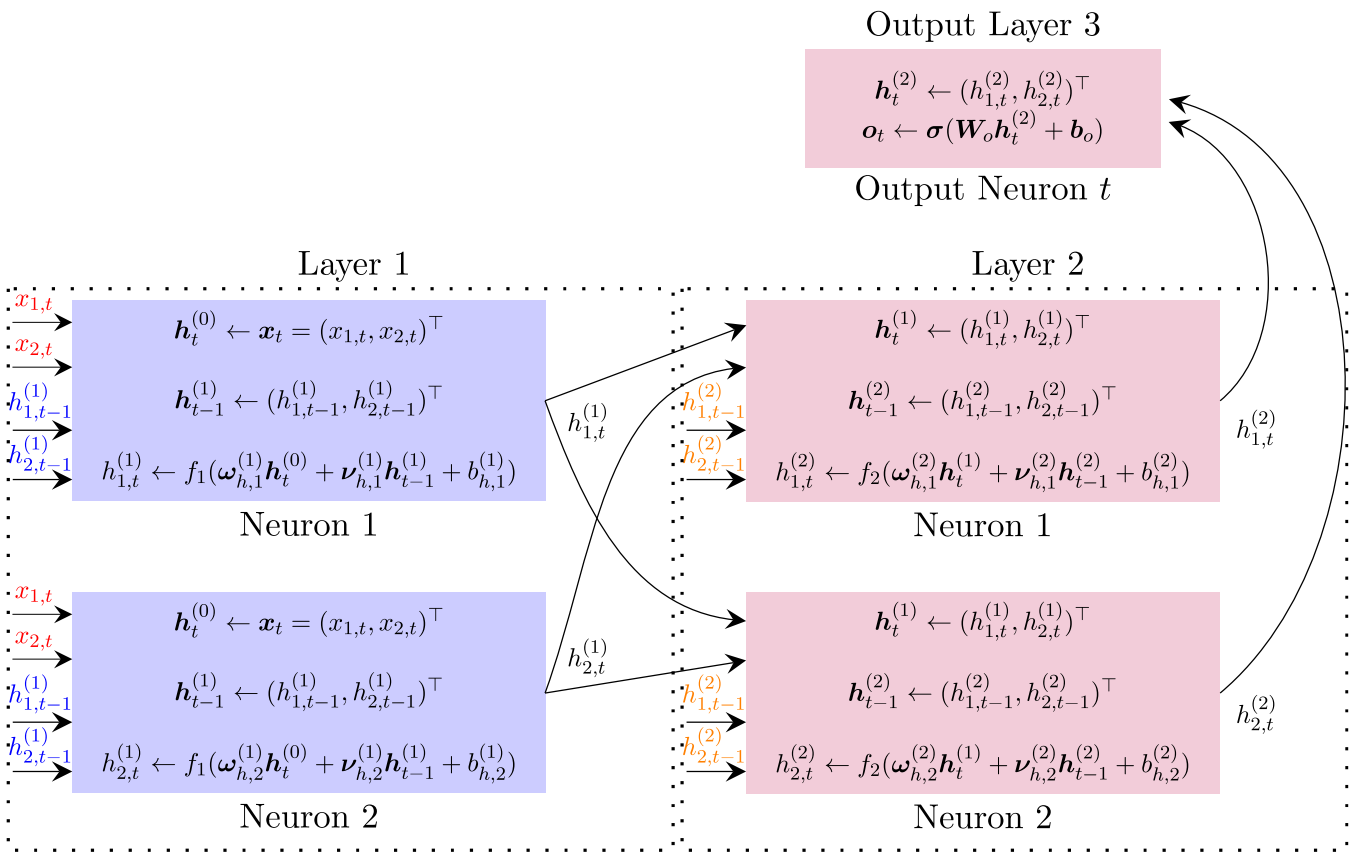

Figure 11.2.2: A folded RNN: The input feature vector is two-dimensional, each hidden layer has two neuron units, inspired by [1].

For RNN, to update the state $h_{j,t}^{(\ell)}$ at each timestep $t = 1, \cdots, D$ in each neuron unit j of the layer ℓ:

1. calculate a linear combination of the inputs $\boldsymbol{h}_t^{(\ell-1)}$ from the previous layer and $\boldsymbol{h}_{t-1}^{(\ell)}$ from the same layer but the last timestep with the respective parameter vectors $\boldsymbol{\omega}_{h,j}^{(\ell)}$, and $\boldsymbol{\nu}_{h,j}^{(\ell)}$, and the bias $b_{h,j}^{(\ell)}$, in other words, $(\boldsymbol{\omega}_{h,j}^{(\ell)})^\top \boldsymbol{h}_t^{(\ell-1)} + (\boldsymbol{\nu}_{h,j}^{(\ell)})^\top \boldsymbol{h}_{t-1}^{(\ell)} + b_{h,j}^{(\ell)}$; then
2. apply an activation function f_1 to obtain the current output $h_{j,t}^{(\ell)}$ of this neuron unit j, and so $h_{j,t}^{(\ell)} = f_\ell[(\boldsymbol{\omega}_{h,j}^{(\ell)})^\top \boldsymbol{h}_t^{(\ell-1)} + (\boldsymbol{\nu}_{h,j}^{(\ell)})^\top \boldsymbol{h}_{t-1}^{(\ell)} + b_{h,j}^{(\ell)}]$ with $\boldsymbol{h}_t^{(0)} = \boldsymbol{x}_t$, for $t = 1, \cdots, D$. Some common choices for f_ℓ are sigmoid type activation functions, such as the commonly used hyperbolic tangent *tanh*; while ReLU is not so prevalent in RNNs.

Finally, in the output layer $L = 3$, we shall apply another activation function $\boldsymbol{\sigma}$ to the output of all neurons that returns a different vector $\boldsymbol{o}_t$ of the same input dimension of the arguments, *i.e.* $\boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{W}_o \boldsymbol{h}_t^{(2)} + \boldsymbol{b}_o)$.

For instance, a typical choice for $\boldsymbol{\sigma}$ in a classification problem (with d_L classes) is the softmax function:

$$\boldsymbol{\sigma}(\boldsymbol{z}^{(L)}) = (\sigma_1(\boldsymbol{z}^{(L)}), \cdots, \sigma_{d_L}(\boldsymbol{z}^{(L)}))^\top, \qquad \text{where} \quad \sigma_i(\boldsymbol{z}^{(L)}) = \frac{\exp\left\{z_i^{(L)}\right\}}{\sum_{j=1}^{d_L} \exp\left\{z_j^{(L)}\right\}},$$

where d_L is the total number of neuron units in this layer L.

Figure 11.2.3 illustrates an unfolded 2-layer RNN model shown in Figure 11.2.1. We shall use this model to illustrate more about the feed forward procedure.

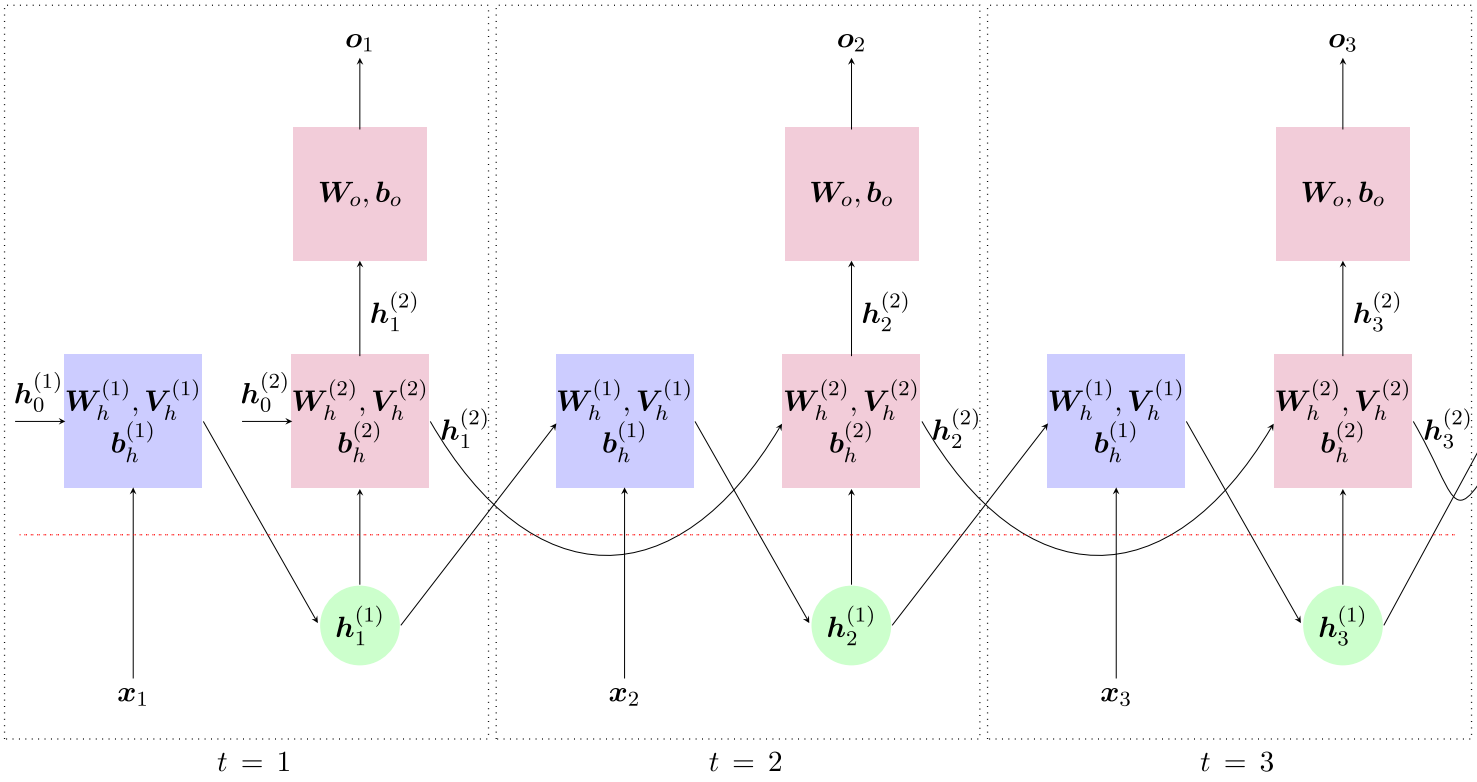

Figure 11.2.3: Unfolding the 2-layer RNN shown in Figure 11.2.1 with $t = 1, 2, 3$, inspired by [1].

11.2.3 Backpropagation through Time

For the sake of convenience, we only illustrate the main idea through one example with MSE as the loss function, and we only consider the calibration for a 1-layer RNN, see Figure 11.2.4. Therefore, the individual temporal error function $\mathcal{E}_t$ at timestep t is simply,

$$\mathcal{E}_t = \frac{1}{2}(\boldsymbol{y}_t - \boldsymbol{o}_t)^\top (\boldsymbol{y}_t - \boldsymbol{o}_t)\,,$$

where $\boldsymbol{y}_t$ is the label at time t, $\boldsymbol{o}_t$ is the predicted output. The total loss function is then given by:

$$\mathcal{E} = \sum_{t=1}^{D} \mathcal{E}_t\,.$$

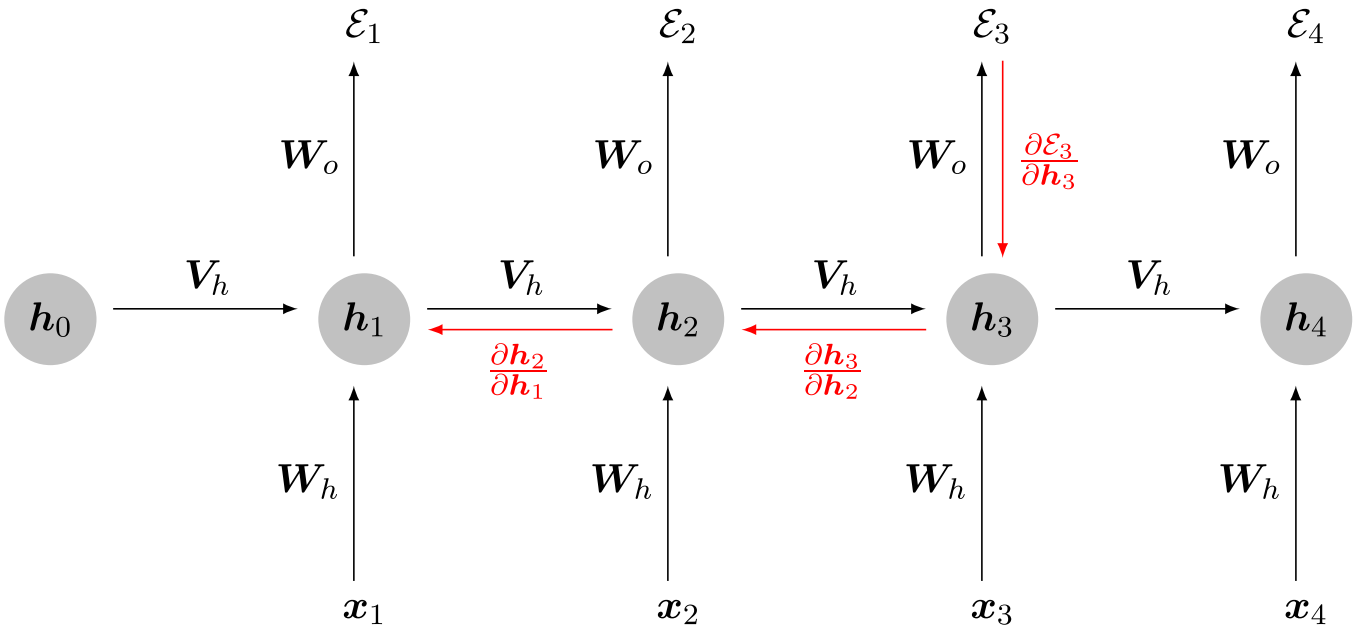

Figure 11.2.4: 1-layer RNN with length $D = 4$: Backpropagation path for $\nabla_{\boldsymbol{W}_h}\mathcal{E}_3$, $\nabla_{\boldsymbol{V}_h}\mathcal{E}_3$, and $\nabla_{\boldsymbol{b}_h}\mathcal{E}_3$.

Recall the notations adopted in (11.2.1a) and (11.2.1b):

(*Cell output*) $$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{W}_h\boldsymbol{x}_t + \boldsymbol{V}_h\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)\,; \tag{11.2.2a}$$

(*Final output*) $$\boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{W}_o\boldsymbol{h}_t + \boldsymbol{b}_o)\,. \tag{11.2.2b}$$

By the law of total derivative, the gradient for parameters $\boldsymbol{W}_o$ is:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_o} &= \sum_{t=1}^{D} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_o} = \sum_{t=1}^{D} \frac{\partial (\boldsymbol{W}_o\boldsymbol{h}_t)}{\partial \boldsymbol{W}_o}\frac{\partial \boldsymbol{o}_t}{\partial (\boldsymbol{W}_o\boldsymbol{h}_t)}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{o}_t} = \sum_{t=1}^{D} \frac{\partial (\boldsymbol{W}_o\boldsymbol{h}_t)}{\partial \boldsymbol{W}_o}\operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{W}_o\boldsymbol{h}_t + \boldsymbol{b}_o)\right\}(\boldsymbol{o}_t - \boldsymbol{y}_t) \\ &= \sum_{t=1}^{D} \left\{(\boldsymbol{o}_t - \boldsymbol{y}_t) \odot \boldsymbol{\sigma}'(\boldsymbol{W}_o\boldsymbol{h}_t + \boldsymbol{b}_o)\right\}\boldsymbol{h}_t^\top\,, \end{aligned} \tag{11.2.3}$$

by using (11.1.4); and that of $\boldsymbol{b}_o$ is

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_o} &= \sum_{t=1}^{D} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_o} = \sum_{t=1}^{D} \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{b}_o}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{o}_t} = \sum_{t=1}^{D} \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{W}_o\boldsymbol{h}_t + \boldsymbol{b}_o)\right\}(\boldsymbol{o}_t - \boldsymbol{y}_t) \\ &= \sum_{t=1}^{D} (\boldsymbol{o}_t - \boldsymbol{y}_t) \odot \boldsymbol{\sigma}'(\boldsymbol{W}_o\boldsymbol{h}_t + \boldsymbol{b}_o)\,. \end{aligned} \tag{11.2.4}$$

On the other hand, the gradients of $\mathcal{E}$ with respect to the parameters $\boldsymbol{W}_h$, $\boldsymbol{V}_h$, and $\boldsymbol{b}_h$ are more complicated

to derive, because $\boldsymbol{h}_t$ also depends on the output of the same hidden layer, but at the previous timestep at $t-1$, *i.e.* $\boldsymbol{h}_{t-1}$, while $\boldsymbol{h}_{t-1}$ itself is also a function of the layer's output, but again even at an earlier timestep $t-2$, *i.e.* $\boldsymbol{h}_{t-2}$, same arguments goes until it reaches the first hidden timestep input $\boldsymbol{h}_0$ and the original feature inputs $\boldsymbol{x}$'s. And this is precisely what we meant by propagating the error rate back through time.

For the sake of computational convenience, denote the error rate and the weighted sum by, respectively:

$$\boldsymbol{\delta}_{t,\tau} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} \qquad \text{and} \qquad \boldsymbol{z}_\tau^h = \boldsymbol{W}_h \boldsymbol{x}_\tau + \boldsymbol{V}_h \boldsymbol{h}_{\tau-1} + \boldsymbol{b}_h\,, \quad \text{for} \quad \tau = 1, \cdots, t\,.$$

Note that directly from (11.2.1b) and Property 9 in Section 3.4,

$$\begin{aligned}\boldsymbol{\delta}_{t,t} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} &= \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{h}_t}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{o}_t} = \frac{\partial (\boldsymbol{W}_o \boldsymbol{h}_t)}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{o}_t}{\partial (\boldsymbol{W}_o \boldsymbol{h}_t)}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{o}_t} = \boldsymbol{W}_o^\top \text{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{W}_o \boldsymbol{h}_t + \boldsymbol{b}_o)\right\}(\boldsymbol{o}_t - \boldsymbol{y}_t) \\ &= \boldsymbol{W}_o^\top \left\{(\boldsymbol{o}_t - \boldsymbol{y}_t) \odot \boldsymbol{\sigma}'(\boldsymbol{W}_o \boldsymbol{h}_t + \boldsymbol{b}_o)\right\}\,. \end{aligned} \tag{11.2.5}$$

The next step is to backpropagate the error rate to the previous timestep $t-1$:

$$\boldsymbol{\delta}_{t,t-1} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_{t-1}} = \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} = \frac{\partial \boldsymbol{z}_t^h}{\partial \boldsymbol{h}_{t-1}}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{z}_t^h}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} = \boldsymbol{V}_h^\top \text{diag}\left\{\boldsymbol{f}'(\boldsymbol{z}_t^h)\right\}\boldsymbol{\delta}_{t,t} = \boldsymbol{V}_h^\top \left\{\boldsymbol{\delta}_{t,t} \odot \boldsymbol{f}'(\boldsymbol{z}_t^h)\right\}. \tag{11.2.6}$$

Similarly, inductively, for $\tau = 1, \cdots, t-1$, we obtain

$$\boldsymbol{\delta}_{t,\tau} = \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \frac{\partial \boldsymbol{z}_{\tau+1}^h}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{z}_{\tau+1}^h}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_{\tau+1}} = \boldsymbol{V}_h^\top \left\{\boldsymbol{\delta}_{t,\tau+1} \odot \boldsymbol{f}'(\boldsymbol{z}_{\tau+1}^h)\right\}. \tag{11.2.7}$$

Figure 11.2.4 illustrates the path of the backpropagation through time of $\nabla_{\boldsymbol{W}_h}\mathcal{E}_3$, $\nabla_{\boldsymbol{V}_h}\mathcal{E}_3$, and $\nabla_{\boldsymbol{b}_h}\mathcal{E}_3$, indicated in red, of a 1-layer RNN; in particular, $\nabla_{\boldsymbol{W}_h}\mathcal{E}_3$ is computed as:

$$\begin{aligned}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{W}_h} &= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_h}\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_3} + \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{W}_h}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_3} + \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{W}_h}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_3} = \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_h}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_1} + \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{W}_h}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_2} + \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{W}_h}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_3} \\ &= \sum_{\tau=1}^{3}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{W}_h}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_\tau} = \sum_{\tau=1}^{3}\frac{\partial \boldsymbol{z}_\tau^h}{\partial \boldsymbol{W}_h}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{z}_\tau^h}\frac{\partial \mathcal{E}_3}{\partial \boldsymbol{h}_\tau}\,. \end{aligned} \tag{11.2.8}$$

Therefore, using similar calculations of (11.2.3), the gradient of $\mathcal{E}$ with respect to the parameters $\boldsymbol{W}_h$, $\boldsymbol{V}_h$, and $\boldsymbol{b}_h$ are, respectively:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_h} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_h} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{z}_\tau^h}{\partial \boldsymbol{W}_h}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{z}_\tau^h}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial (\boldsymbol{W}_h \boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_h}\text{diag}\{\boldsymbol{f}'(\boldsymbol{z}_\tau^h)\}\boldsymbol{\delta}_{t,\tau} \\ &= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\{\boldsymbol{f}'(\boldsymbol{z}_\tau^h) \odot \boldsymbol{\delta}_{t,\tau}\}\boldsymbol{x}_\tau^\top\,, \end{aligned} \tag{11.2.9}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_h} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_h} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{z}_\tau^h}{\partial \boldsymbol{V}_h}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{z}_\tau^h}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial (\boldsymbol{V}_h \boldsymbol{h}_{\tau-1})}{\partial \boldsymbol{V}_h}\text{diag}\{\boldsymbol{f}'(\boldsymbol{z}_\tau^h)\}\boldsymbol{\delta}_{t,\tau} \\ &= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\{\boldsymbol{f}'(\boldsymbol{z}_\tau^h) \odot \boldsymbol{\delta}_{t,\tau}\}\boldsymbol{h}_{\tau-1}^\top\,, \end{aligned} \tag{11.2.10}$$

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_h} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_h} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{z}_\tau^h}{\partial \boldsymbol{b}_h}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{z}_\tau^h}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\text{diag}\{\boldsymbol{f}'(\boldsymbol{z}_\tau^h)\}\boldsymbol{\delta}_{t,\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\boldsymbol{f}'(\boldsymbol{z}_\tau^h) \odot \boldsymbol{\delta}_{t,\tau}\,. \tag{11.2.11}$$

In multilayer RNN, the backpropagations happen through both layers and time. One should pay attention that each $\boldsymbol{h}_t^{(\ell)}$ is a function of both $\boldsymbol{h}_{t-1}^{(\ell)}$ and $\boldsymbol{h}_t^{(\ell-1)}$, while $\boldsymbol{h}_{t-1}^{(\ell)}$ and $\boldsymbol{h}_t^{(\ell-1)}$ depend on the pairs $(\boldsymbol{h}_{t-2}^{(\ell)}, h_{t-1}^{(\ell-1)})$ and $(\boldsymbol{h}_{t-1}^{(\ell-1)}, \boldsymbol{h}_t^{(\ell-2)})$, respectively, the same dependence structure will be carried on until it reaches to the utmost left and floor boundaries; for instance, as depicted in Figure 11.2.5, the interdependence as a function of $\mathcal{E}$ of a 2-layer RNN with $D = 4$, the backprobagation mechanism on $\nabla_{\boldsymbol{W}_h^{(1)}}\mathcal{E}_3$, $\nabla_{\boldsymbol{V}_h^{(1)}}\mathcal{E}_3$, and $\nabla_{\boldsymbol{b}_h^{(1)}}\mathcal{E}_3$ are

illustrated.

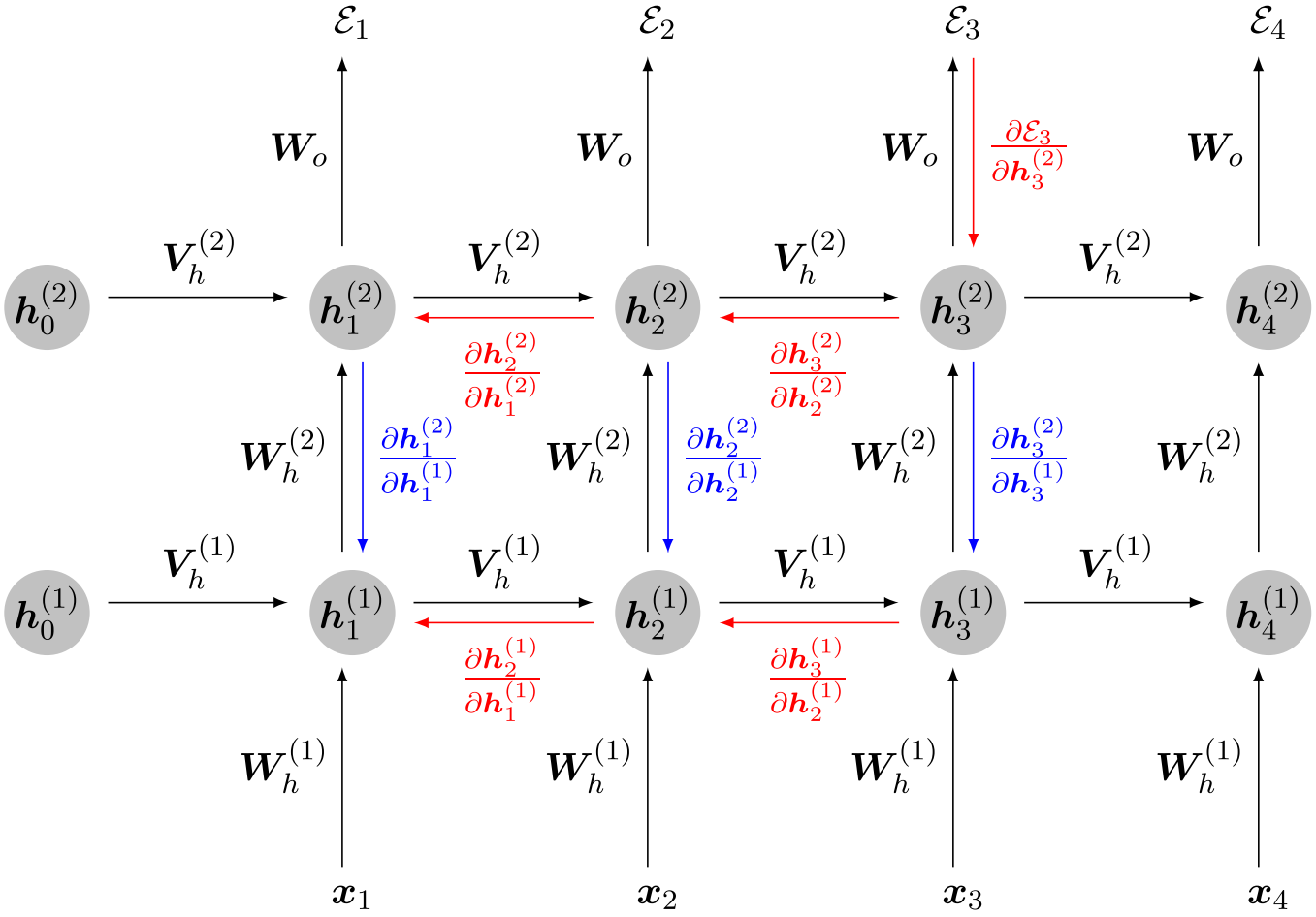

Figure 11.2.5: A 2-layer RNN with length $D = 4$: Backpropagation path of $\nabla_{\boldsymbol{W}_h^{(1)}} \mathcal{E}_3$, $\nabla_{\boldsymbol{V}_h^{(1)}} \mathcal{E}_3$, and $\nabla_{\boldsymbol{b}_h^{(1)}} \mathcal{E}_3$.

From (11.2.5), we know that the gradient of the temporal error $\mathcal{E}_t$ with respect to $\boldsymbol{h}_t^{(2)}$ and $\boldsymbol{h}_\tau^{(2)}$, for $\tau = 1, \cdots, t-1$, are, respectively:

$$\boldsymbol{\delta}_{t,t}^{(2)} = \boldsymbol{W}_o^\top \left\{ (\boldsymbol{o}_t - \boldsymbol{y}_t) \odot \boldsymbol{\sigma}'(\boldsymbol{W}_o \boldsymbol{h}_t^{(2)} + \boldsymbol{b}_o) \right\} \qquad \text{and} \qquad \boldsymbol{\delta}_{t,\tau}^{(2)} = (\boldsymbol{V}_h^{(2)})^\top \left\{ \boldsymbol{\delta}_{t,\tau+1}^{(2)} \odot \boldsymbol{f}'(\boldsymbol{z}_{\tau+1}^{h,(2)}) \right\}.$$

We now consider the temporal error $\mathcal{E}_t$ with respect to $\boldsymbol{h}_t^{(1)}$:

$$\begin{aligned} \boldsymbol{\delta}_{t,t}^{(1)} &= \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t^{(1)}} = \frac{\partial \boldsymbol{h}_t^{(2)}}{\partial \boldsymbol{h}_t^{(1)}} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t^{(2)}} = \frac{\partial \boldsymbol{z}_t^{h,(2)}}{\partial \boldsymbol{h}_t^{(1)}} \frac{\partial \boldsymbol{h}_t^{(2)}}{\partial \boldsymbol{z}_t^{h,(2)}} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t^{(2)}} \\ &= \frac{\partial (\boldsymbol{W}_h^{(2)} \boldsymbol{h}_t^{(1)})}{\partial \boldsymbol{h}_t^{(1)}} \operatorname{diag}\{\boldsymbol{f}_2'(\boldsymbol{z}_t^{h,(2)})\} \boldsymbol{\delta}_{t,t}^{(2)} = (\boldsymbol{W}_h^{(2)})^\top \left\{ \boldsymbol{f}_2'(\boldsymbol{z}_t^{h,(2)}) \odot \boldsymbol{\delta}_{t,t}^{(2)} \right\}. \end{aligned}$$

However, for $\boldsymbol{\delta}_{t,t-1}^{(1)}$, we first note in Figure 11.2.5 that there are two paths, one red path and one blue path, leading to the same neuron on the previous timestep $t-1$; to this end, we shall first consider, for $t = 1, \cdots, D$,

$$\frac{\partial \boldsymbol{h}_t^{(2)}}{\partial \boldsymbol{h}_t^{(1)}} = \frac{\partial \boldsymbol{z}_t^{h,(2)}}{\partial \boldsymbol{h}_t^{(1)}} \frac{\partial \boldsymbol{h}_t^{(2)}}{\partial \boldsymbol{z}_t^{h,(2)}} = \frac{\partial (\boldsymbol{W}_h^{(2)} \boldsymbol{h}_{t-1}^{(1)})}{\partial \boldsymbol{h}_{t-1}^{(1)}} \operatorname{diag}\left\{ \boldsymbol{f}_2'(\boldsymbol{z}_t^{h,(2)}) \right\} = (\boldsymbol{W}_h^{(2)})^\top \operatorname{diag}\left\{ \boldsymbol{f}_2'(\boldsymbol{z}_t^{h,(2)}) \right\},$$

using Property 9 in Section 3.4. Therefore, using which and the intermediate step in (11.2.6), we have

$$\begin{aligned}\boldsymbol{\delta}_{t,t-1}^{(1)} &= \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_{t-1}^{(1)}} = \frac{\partial \boldsymbol{h}_{t-1}^{(2)}}{\partial \boldsymbol{h}_{t-1}^{(1)}}\frac{\partial \boldsymbol{h}_t^{(2)}}{\partial \boldsymbol{h}_{t-1}^{(2)}}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t^{(2)}} + \frac{\partial \boldsymbol{h}_t^{(1)}}{\partial \boldsymbol{h}_{t-1}^{(1)}}\frac{\partial \boldsymbol{h}_t^{(2)}}{\partial \boldsymbol{h}_t^{(1)}}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t^{(2)}} \\ &= (\boldsymbol{W}_h^{(2)})^\top \text{diag}\left\{\boldsymbol{f}_2'(\boldsymbol{z}_{t-1}^{h,(2)})\right\}(\boldsymbol{V}_h^{(2)})^\top \text{diag}\left\{\boldsymbol{f}_2'(\boldsymbol{z}_t^{h,(2)})\right\}\boldsymbol{\delta}_{t,t}^{(2)} \\ &\quad + (\boldsymbol{V}_h^{(1)})^\top \text{diag}\left\{\boldsymbol{f}_1'(\boldsymbol{z}_t^{h,(1)})\right\}(\boldsymbol{W}_h^{(2)})^\top \text{diag}\left\{\boldsymbol{f}_2'(\boldsymbol{z}_t^{h,(2)})\right\}\boldsymbol{\delta}_{t,t}^{(2)}\,.\end{aligned}$$

For that of the timestep $t-2$, the computation becomes tedious, we observe in Figure 11.2.5 that there are three paths. In general, for computing the gradient of the temporal error $\mathcal{E}_t$ with respect to $\boldsymbol{h}_\tau^{(1)}$, for $\tau = 1, \cdots, t$, in a 2-layer RNN has $t-\tau+1$ number of paths, and we shall end the computation here.

11.3 Problems with RNN

1. **Vanishing /Exploding Gradient**: From Figure 11.2.3, the same set of parameters will be used for calibration for each t of length D time series data, if this length of the input sequence D increases, as explained in last Section 11.2, backpropagation has to "unfold" the network through time. For instance, from (11.2.7), the error rate has to backpropagate through time from timestep D to the timestep 1, namely:

$$\boldsymbol{\delta}_{D,1} = \boldsymbol{V}_h^\top\left\{\boldsymbol{\delta}_{D,1+1}\odot \boldsymbol{f}'(\boldsymbol{z}_{1+1}^h)\right\} = \underbrace{\boldsymbol{V}_h^\top\left\{\cdots\left\{\boldsymbol{V}_h^\top\right.\right.}_{D-1 \text{ entries}}\left\{\boldsymbol{\delta}_{D,D}\odot \boldsymbol{f}'(\boldsymbol{z}_D^h)\right\}\odot\cdots\Big\}\odot \boldsymbol{f}'(\boldsymbol{z}_2^h)\Big\}, \qquad (11.3.1)$$

   where $\boldsymbol{\delta}_{D,D}$ is given in (11.2.5) with $t=D$; that means the error rate at the timestep 1 is the product of $D-1$ matrices $\boldsymbol{V}_h^\top$. For if $\boldsymbol{V}_h^\top$ is small and if D is also very large, then this induces a vanishing gradient problem that much affects the parameter update. On the other hand, if $\boldsymbol{V}_h^\top$ is large, then this will lead to an exploding gradient problem. Note that the problem still exist no matter what the activation function is chosen. On the other hand, as mentioned before in Section 9.2, *tanh* and *softmax* (*sigmoid*) suffer from the vanishing gradient problem in MLP, and the same problem occurs in RNN that the error rate in (11.3.1) is also a product of $\boldsymbol{f}'(\boldsymbol{z}_D^h)\odot\cdots\odot\boldsymbol{f}'(\boldsymbol{z}_2^h)$. Note that, with f being the sigmoid (resp. tanh) activation function, we have $f'(\boldsymbol{z}_t^h)\le 1/4$ (resp. $f'(\boldsymbol{z}_t^h)\le 1$), for $t=1,\cdots,D$. Therefore, the product of $\boldsymbol{f}'(\boldsymbol{z}_D^h)\odot\cdots\odot\boldsymbol{f}'(\boldsymbol{z}_2^h)$ is capped at $(1/4)^{D-1}$ (resp. 1) and this may trigger the vanishing gradient problem.

2. **Long-term Dependence**: As the length of the time series data grows, the feature vectors from the beginning of the sequence tend to be "forgotten" in the sense that the state of each unit, which serves as network's memory, is more notably affected by the feature vectors that read recently. Consider the two individual temporal error, $\mathcal{E}_1$ and $\mathcal{E}_D$, for the hidden state of $\boldsymbol{h}$ from timestep 1 and timestep D, respectively, and they are

$$\boldsymbol{h}_1 = \boldsymbol{f}(\boldsymbol{W}_h\boldsymbol{x}_0 + \boldsymbol{V}_h\boldsymbol{h}_0 + \boldsymbol{b}_h) \qquad \text{and} \qquad \boldsymbol{h}_D = \boldsymbol{f}(\boldsymbol{W}_h\boldsymbol{x}_D + \boldsymbol{V}_h\boldsymbol{h}_{D-1} + \boldsymbol{b}_h)\,,$$

   while $\boldsymbol{h}_{D-1}$ is a function of both previous timestep output $\boldsymbol{h}_{D-2}$ and $\boldsymbol{x}_{D-1}$, which in turn is a function of another previous timestep objects and so forth until $\boldsymbol{h}_0$ and $\boldsymbol{x}_1$. Hence, as D is large, the impact of $\boldsymbol{x}_1$ on $\boldsymbol{h}_D$ is much smaller than its impact on $\boldsymbol{h}_1$, that means being forgotten; indeed, for 1-layer

RNN, mathematically,

$$\frac{\partial \boldsymbol{h}_D}{\partial \boldsymbol{h}_1} = \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\cdots\frac{\partial \boldsymbol{h}_{D-1}}{\partial \boldsymbol{h}_{D-2}}\frac{\partial \boldsymbol{h}_D}{\partial \boldsymbol{h}_{D-1}}\,.$$

Since each

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} = \frac{\partial \boldsymbol{z}_t^h}{\partial \boldsymbol{h}_{t-1}}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{z}_t^h} = \boldsymbol{V}_h^\top \text{diag}\left\{\boldsymbol{f}'(\boldsymbol{z}_t^h)\right\}\,,$$

then

$$\frac{\partial \boldsymbol{h}_D}{\partial \boldsymbol{h}_1} = \boldsymbol{V}_h^\top \text{diag}\left\{\boldsymbol{f}'(\boldsymbol{z}_2^h)\right\}\frac{\partial \boldsymbol{h}_D}{\partial \boldsymbol{h}_2} = \boldsymbol{V}_h^\top \text{diag}\left\{\boldsymbol{f}'(\boldsymbol{z}_2^h)\right\}\boldsymbol{V}_h^\top \text{diag}\left\{\boldsymbol{f}'(\boldsymbol{z}_3^h)\right\}\cdots\boldsymbol{V}_h^\top \text{diag}\left\{\boldsymbol{f}'(\boldsymbol{z}_D^h)\right\}\,.$$

when one applies RNNs to text or speech recognition, the cause-effect link between distant words in a long sentence can be lost. One possible solution to this short memory problem is to use **gated RNNs** or **LSTM** to be discussed in the next two sections.

11.4 Gated RNNs

To remedy the shortcoming of short memory nature of the general RNNs introduced in the last section, *Gated RNNs* was invented in Cho et al. (2014), which included the recently popular long short-term memory (LSTM) networks and those based on the gated recurrent unit (GRU). Gated RNNs use activation function to determine which information is stored in their units for future use; in particular, the trained model can "read" the input temporal sequence of feature vectors, based on which the model decides at some earlier time step t to keep this specific information obtained from the feature vectors for the use at a later time. This stored information can be used later to analyze with the feature vectors recently acquired from the near end of the time series data. For example, if the input text starts with the word "she", a language processing RNN model could decide to store the information about the gender of the subject so as to interpret properly the possible meaning of "their" to be seen observed in the whole sentence segment.

In a gated RNN, there are different units that make decisions about what informations to store, or when to allow to read, write or erase them. These decisions are learnt from the dataset and are implemented through the concept of "logic" gates. A *gated recurrent unit* (GRU) composes of a reset gate, update gate, cell input and cell output: for instance, in the first layer of a gated recurrent unit (GRU), it takes in two inputs:

1. a feature vector $\boldsymbol{x}_t =: \boldsymbol{h}_t^{(0)}$; and
2. a vector from the same or other memory cells in the same (first) layer but from the previous time step, $\boldsymbol{h}_{t-1}^{(1)}$.

The following Figure 11.4.1 shows a memory cell of a GRU:

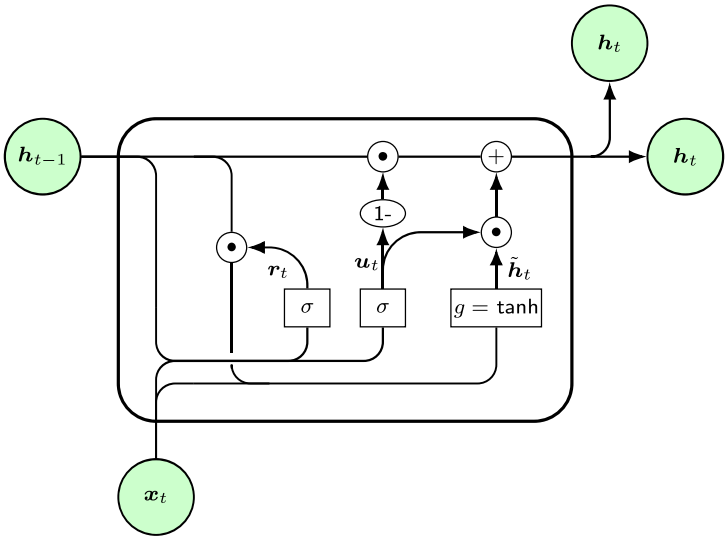

Figure 11.4.1: A GRU memory cell; also see [4].

(*Reset gate*) $$\boldsymbol{r}_t = \boldsymbol{\sigma}(\boldsymbol{W}_r\boldsymbol{x}_t + \boldsymbol{V}_r\boldsymbol{h}_{t-1} + \boldsymbol{b}_r)\,; \tag{11.4.1a}$$

(*Update gate*) $$\boldsymbol{u}_t = \boldsymbol{\sigma}(\boldsymbol{W}_u\boldsymbol{x}_t + \boldsymbol{V}_u\boldsymbol{h}_{t-1} + \boldsymbol{b}_u)\,; \tag{11.4.1b}$$

(*Cell input*) $$\tilde{\boldsymbol{h}}_t = \boldsymbol{g}(\boldsymbol{W}_{\tilde{h}}\boldsymbol{x}_t + \boldsymbol{V}_{\tilde{h}}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_{\tilde{h}})\,; \tag{11.4.1c}$$

(*Cell output*) $$\boldsymbol{h}_t = \boldsymbol{u}_t \odot \tilde{\boldsymbol{h}}_t + (\boldsymbol{1} - \boldsymbol{u}_t) \odot \boldsymbol{h}_{t-1}\,. \tag{11.4.1d}$$

where in the following, we describe more in detail about the very nature:

1. $\boldsymbol{x}_t$ stands for the input feature vector at the timestep t;
2. Matrices $\boldsymbol{W}_r, \boldsymbol{W}_u, \boldsymbol{W}_{\tilde{h}}$ correspond to the weights of inputs $\boldsymbol{x}_t$ used in the reset gate, update gate, and feedback cell input, respectively;
3. Matrices $\boldsymbol{V}_r, \boldsymbol{V}_u, \boldsymbol{V}_{\tilde{h}}$ correspond to the weights of the connections between the cell output activated value $\boldsymbol{h}_{t-1}$ being fed back, but one-step delay with the reset gate, update gate, and cell input, respectively;
4. $\boldsymbol{b}_r, \boldsymbol{b}_u, \boldsymbol{b}_{\tilde{h}}$ stands for bias vectors for the reset gate, update gate, and cell input, respectively;
5. Activation functions $\boldsymbol{\sigma}$ and $\boldsymbol{g}$ are used for the gates and cell input, respectively. Typically, the logistic function is adopted for $\boldsymbol{\sigma}$ and the hyperbolic tangent *tanh* is used for $\boldsymbol{g}$;
6. **Update Gate**: if $\boldsymbol{u}_t \approx \boldsymbol{0}$, then this memory cell, *Memory Gated Unit* (MGU), keeps the value of the cell output $\boldsymbol{h}_{t-1}$ from the previous time step $t-1$; indeed $\boldsymbol{h}_t \approx \boldsymbol{h}_{t-1}$ from (11.4.1d), *i.e.* $\boldsymbol{h}_{t-1}$. It helps to capture the long-term dependence in the time series data;
7. **Reset Gate**: if $\boldsymbol{u}_t \approx \boldsymbol{1}$, the value of the MGU is overwritten by a new value $\tilde{h}_t$, and this is regarded as capturing short-term dependence in time series.

Note that the two gates are mimicking the effect given by a logic gate in electric circuit, namely whenever a threshold is surpassed, an operational signal is fired; otherwise, the state is off. This is also the name of this gated recurrent unit coming from. In particular, in the cell input, we have:

1. if $\boldsymbol{r}_t = \boldsymbol{0}$, essentially only information of $\boldsymbol{x}_t$ is used to calculate $\tilde{\boldsymbol{h}}_t$;
2. if $\boldsymbol{r}_t = \boldsymbol{1}$, $\boldsymbol{h}_{t-1}$ would interfee, and even override the effects of $\boldsymbol{x}_t$ especially when $\|\boldsymbol{V}_{\tilde{h}}\|_2$ is large, that means the previously older memory will reset the state all again and then cover x_t.

A GRU takes an input $\boldsymbol{x}_t$ and stores it for some time, by then all $\boldsymbol{u}_t = \boldsymbol{0}$, for $\tau = 1, \cdots, t$, which is somehow equivalent to applying the identity function ($f(x) = x$) to the input. On the other hand, as the gradient of the identity function is a constant vector of "1", and so it is uniformly bounded away from zero, therefore, when a network with GRUs is trained with backpropagation through time, the gradient does not vanish. To see this effect, let us introduce the backpropagation for this GRU.

11.4.1 Backpropagation through Time

For simplicity, we only consider a 1-layer GRU as before, where we adopt the RNN final output in (11.2.1b) as the output layer, denote the error rates through time be

$$\boldsymbol{\delta}_{t,\tau} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau}\,, \qquad \tau = 1, \cdots, t\,,$$

where $\boldsymbol{\delta}_{t,t}$ is given in (11.2.5) as a RNN final output is set. Denote the respective weighted sums $\boldsymbol{z}_t$ for different gates and inputs be:

$$\begin{aligned}
\boldsymbol{z}_t^r &= \boldsymbol{W}_r\boldsymbol{x}_t + \boldsymbol{V}_r\boldsymbol{h}_{t-1} + \boldsymbol{b}_r\,, && \text{where} \quad \boldsymbol{r}_t = \boldsymbol{\sigma}(\boldsymbol{z}_t^r)\,;\\
\boldsymbol{z}_t^u &= \boldsymbol{W}_u\boldsymbol{x}_t + \boldsymbol{V}_u\boldsymbol{h}_{t-1} + \boldsymbol{b}_u\,, && \text{where} \quad \boldsymbol{u}_t = \boldsymbol{\sigma}(\boldsymbol{z}_t^u)\,;\\
\boldsymbol{z}_t^{\tilde{h}} &= \boldsymbol{W}_{\tilde{h}}\boldsymbol{x}_t + \boldsymbol{V}_{\tilde{h}}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_{\tilde{h}}\,, && \text{where} \quad \tilde{\boldsymbol{h}}_t = \boldsymbol{g}(\boldsymbol{z}_t^{\tilde{h}})\,.
\end{aligned}$$

In general, to backpropagate the error rate from $\tau+1$ to τ, we have by simple chain rule:

$$\boldsymbol{\delta}_{t,\tau} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_{\tau+1}} = \frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\boldsymbol{\delta}_{t,\tau+1}\,.$$

Recall that the cell output equation in (11.4.1d), we further define a differentiable function $\boldsymbol{f}:(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\mapsto \boldsymbol{x}\odot\boldsymbol{y}+(1-\boldsymbol{x})\odot\boldsymbol{z}$, and also note that both $\boldsymbol{u}_{\tau+1}$, and $\tilde{\boldsymbol{h}}_{\tau+1}$ are a function of $\boldsymbol{h}_\tau$, a simple application of the law of total derivative gives:

$$\begin{aligned}
\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau} &= \frac{\partial \boldsymbol{f}(\boldsymbol{u}_{\tau+1},\tilde{\boldsymbol{h}}_{\tau+1},\boldsymbol{h}_{\tau+1})}{\partial \boldsymbol{h}_\tau} + \frac{\partial \boldsymbol{u}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{f}(\boldsymbol{u}_{\tau+1},\tilde{\boldsymbol{h}}_{\tau+1},\boldsymbol{h}_{\tau+1})}{\partial \boldsymbol{u}_{\tau+1}} + \frac{\partial \tilde{\boldsymbol{h}}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{f}(\boldsymbol{u}_{\tau+1},\tilde{\boldsymbol{h}}_{\tau+1},\boldsymbol{h}_{\tau+1})}{\partial \tilde{\boldsymbol{h}}_{\tau+1}}\\
&= \mathrm{diag}\{\boldsymbol{1}-\boldsymbol{u}_{\tau+1}\} + \frac{\partial \boldsymbol{u}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{u}_{\tau+1}} + \frac{\partial \boldsymbol{z}_{\tau+1}^{\tilde{h}}}{\partial \boldsymbol{h}_\tau}\frac{\partial \tilde{\boldsymbol{h}}_{\tau+1}}{\partial \boldsymbol{z}_{\tau+1}^{\tilde{h}}}\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \tilde{\boldsymbol{h}}_{\tau+1}}\,, \qquad (11.4.2)
\end{aligned}$$

where we use (11.1.3) that $(\boldsymbol{1}-\boldsymbol{u}_{\tau+1})\odot\boldsymbol{h}_\tau = \mathrm{diag}\{\boldsymbol{1}-\boldsymbol{u}_{\tau+1}\}\boldsymbol{h}_\tau$, and then Property 9 in Section 3.4. We further notice that $\boldsymbol{r}_{\tau+1}$ is also a function of $\boldsymbol{h}_\tau$ in $\boldsymbol{z}_{\tau+1}^{\tilde{h}}$, therefore in particular, the law of total derivative give:

$$\frac{\partial \boldsymbol{z}_{\tau+1}^{\tilde{h}}}{\partial \boldsymbol{h}_\tau} = \frac{\partial(\boldsymbol{V}_{\tilde{h}}\mathrm{diag}\{\boldsymbol{h}_\tau\}\boldsymbol{r}_{\tau+1})}{\partial \boldsymbol{h}_\tau} = \frac{\partial(\boldsymbol{V}_{\tilde{h}}\mathrm{diag}\{\boldsymbol{r}_{\tau+1}\}\boldsymbol{h}_\tau)}{\partial \boldsymbol{h}_\tau} = \frac{\partial \boldsymbol{r}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\left(\boldsymbol{V}_{\tilde{h}}\mathrm{diag}\{\boldsymbol{h}_\tau\}\right)^\top + \left(\boldsymbol{V}_{\tilde{h}}\mathrm{diag}\{\boldsymbol{r}_{\tau+1}\}\right)^\top\,.$$

Therefore, (11.4.2) can be further computed as:

$$\begin{aligned}
\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau} &= \mathrm{diag}\{\boldsymbol{1}-\boldsymbol{u}_{\tau+1}\} + \boldsymbol{V}_u^\top\mathrm{diag}\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^u)\}\mathrm{diag}\left\{\tilde{\boldsymbol{h}}_{\tau+1}-\boldsymbol{h}_{\tau+1}\right\}\\
&\quad + \left[\frac{\partial \boldsymbol{r}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\left(\boldsymbol{V}_{\tilde{h}}\mathrm{diag}\{\boldsymbol{h}_\tau\}\right)^\top + \left(\boldsymbol{V}_{\tilde{h}}\mathrm{diag}\{\boldsymbol{r}_{\tau+1}\}\right)^\top\right]\mathrm{diag}\{\boldsymbol{g}'(\boldsymbol{z}_{\tau+1}^{\tilde{h}})\}\mathrm{diag}\{\boldsymbol{u}_{\tau+1}\}\\
&= \mathrm{diag}\{\boldsymbol{1}-\boldsymbol{u}_{\tau+1}\} + \boldsymbol{V}_u^\top\mathrm{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^u)\odot\left(\tilde{\boldsymbol{h}}_{\tau+1}-\boldsymbol{h}_{\tau+1}\right)\right\}\\
&\quad + \left[\boldsymbol{V}_r^\top\mathrm{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^r)\odot\boldsymbol{h}_\tau\right\} + \mathrm{diag}\{\boldsymbol{r}_{\tau+1}\}\right]\boldsymbol{V}_{\tilde{h}}^\top\mathrm{diag}\left\{\boldsymbol{g}'(\boldsymbol{z}_{\tau+1}^{\tilde{h}})\odot\boldsymbol{u}_{\tau+1}\right\}.
\end{aligned}$$

Therefore, for $\tau=1,\cdots,t-1$, we have

$$\begin{aligned}
\boldsymbol{\delta}_{t,\tau} &= \boldsymbol{\delta}_{t,\tau+1}\odot(\boldsymbol{1}-\boldsymbol{u}_{\tau+1}) + \boldsymbol{V}_u^\top\left(\boldsymbol{\delta}_{t,\tau+1}\odot\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^u)\odot\left(\tilde{\boldsymbol{h}}_{\tau+1}-\boldsymbol{h}_{\tau+1}\right)\right)\\
&\quad + \left[\boldsymbol{V}_r^\top\mathrm{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^r)\odot\boldsymbol{h}_\tau\right\} + \mathrm{diag}\{\boldsymbol{r}_{\tau+1}\}\right]\boldsymbol{V}_{\tilde{h}}^\top\left(\boldsymbol{\delta}_{t,\tau+1}\odot\boldsymbol{g}'(\boldsymbol{z}_{\tau+1}^{\tilde{h}})\odot\boldsymbol{u}_{\tau+1}\right). \qquad (11.4.3)
\end{aligned}$$

Consider the parameter updates in the cell input,

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_{\tilde{h}}} &= \sum_{t=1}^D\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_{\tilde{h}}} = \sum_{t=1}^D\sum_{\tau=1}^t\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{W}_{\tilde{h}}}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^D\sum_{\tau=1}^t\frac{\partial \tilde{\boldsymbol{h}}_\tau}{\partial \boldsymbol{W}_{\tilde{h}}}\frac{\partial \boldsymbol{h}_\tau}{\partial \tilde{\boldsymbol{h}}_\tau}\boldsymbol{\delta}_{t,\tau}\\
&= \sum_{t=1}^D\sum_{\tau=1}^t\frac{\partial(\boldsymbol{W}_{\tilde{h}}\boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_{\tilde{h}}}\mathrm{diag}\{\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\}\mathrm{diag}(\boldsymbol{u}_\tau)\boldsymbol{\delta}_{t,\tau} = \sum_{t=1}^D\sum_{\tau=1}^t\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\odot\boldsymbol{u}_\tau\}\boldsymbol{x}_\tau^\top\,,\\
\frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_{\tilde{h}}} &= \sum_{t=1}^D\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_{\tilde{h}}} = \sum_{t=1}^D\sum_{\tau=1}^t\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\odot\boldsymbol{u}_\tau\}\boldsymbol{h}_{\tau-1}^\top\,,\\
\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_{\tilde{h}}} &= \sum_{t=1}^D\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_{\tilde{h}}} = \sum_{t=1}^D\sum_{\tau=1}^t\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\odot\boldsymbol{u}_\tau\}\,.
\end{aligned}$$

While for the parameter updates in the update gate, likewise we have:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_u} &= \sum_{t=1}^{D} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_u} = \sum_{t=1}^{D}\sum_{\tau=1}^{t} \frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{W}_u}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t} \frac{\partial \boldsymbol{u}_\tau}{\partial \boldsymbol{W}_u}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{u}_\tau}\boldsymbol{\delta}_{t,\tau} \\ &= \sum_{t=1}^{D}\sum_{\tau=1}^{t} \frac{\partial (\boldsymbol{W}_u \boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_u}\text{diag}\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^u)\}\text{diag}\{\tilde{\boldsymbol{h}}_\tau - \boldsymbol{h}_{\tau-1}\}\boldsymbol{\delta}_{t,\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^u)\odot\left(\tilde{\boldsymbol{h}}_\tau - \boldsymbol{h}_{\tau-1}\right)\right\}\boldsymbol{x}_\tau^\top, \\ \frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_u} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_u} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^u)\odot\left(\tilde{\boldsymbol{h}}_\tau - \boldsymbol{h}_{\tau-1}\right)\right\}\boldsymbol{h}_{\tau-1}^\top, \\ \frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_u} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_u} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^u)\odot\left(\tilde{\boldsymbol{h}}_\tau - \boldsymbol{h}_{\tau-1}\right)\right\}.\end{aligned}$$

Finally, for the parameter updates in the reset gate,

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_r} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_r} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{W}_r}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{r}_\tau}{\partial \boldsymbol{W}_r}\frac{\partial \tilde{\boldsymbol{h}}_\tau}{\partial \boldsymbol{r}_\tau}\frac{\partial \boldsymbol{h}_\tau}{\partial \tilde{\boldsymbol{h}}_\tau}\boldsymbol{\delta}_{t,\tau} \\ &= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial (\boldsymbol{W}_r \boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_r}\text{diag}\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^r)\}\boldsymbol{V}_{\tilde{h}}^\top\text{diag}\{\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\}\text{diag}\{\boldsymbol{u}_\tau\}\boldsymbol{\delta}_{t,\tau} \\ &= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^r)\odot\left(\boldsymbol{V}_{\tilde{h}}^\top\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\odot\boldsymbol{u}_\tau\}\right)\right\}\boldsymbol{x}_\tau^\top, \\ \frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_r} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_r} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^r)\odot\left(\boldsymbol{V}_{\tilde{h}}^\top\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\odot\boldsymbol{u}_\tau\}\right)\right\}\boldsymbol{h}_{\tau-1}^\top, \\ \frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_r} &= \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_r} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^r)\odot\left(\boldsymbol{V}_{\tilde{h}}^\top\{\boldsymbol{\delta}_{t,\tau}\odot\boldsymbol{g}'(\boldsymbol{z}_\tau^{\tilde{h}})\odot\boldsymbol{u}_\tau\}\right)\right\}.\end{aligned}$$

From (11.4.3), we observe that the cell error rate can be expressed in the form:

$$\boldsymbol{\delta}_{t,\tau} = (\mathbf{1} - \boldsymbol{u}_{\tau+1})\odot\boldsymbol{\delta}_{t,\tau+1} + \boldsymbol{A}_{\tau+1}(\boldsymbol{V}_u) + \boldsymbol{B}_{\tau+1}(\boldsymbol{V}_{\tilde{h}}, \boldsymbol{V}_r),$$

where the first term does not depend on the weights $\boldsymbol{V}$'s, and

$$\begin{aligned}\boldsymbol{A}_{\tau+1}(\boldsymbol{V}_u) &= \boldsymbol{V}_u^\top\text{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^u)\odot\left(\tilde{\boldsymbol{h}}_{\tau+1} - \boldsymbol{h}_{\tau+1}\right)\right\}; \\ \boldsymbol{B}_{\tau+1}(\boldsymbol{V}_{\tilde{h}}, \boldsymbol{V}_r) &= \left[\boldsymbol{V}_r^\top\text{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^r)\odot\boldsymbol{h}_\tau\right\} + \text{diag}\{\boldsymbol{r}_{\tau+1}\}\right]\boldsymbol{V}_{\tilde{h}}^\top\left(\boldsymbol{\delta}_{t,\tau+1}\odot\boldsymbol{g}'(\boldsymbol{z}_{\tau+1}^{\tilde{h}})\odot\boldsymbol{u}_{\tau+1}\right).\end{aligned}$$

Therefore, if the value of the update gate $\boldsymbol{u}_\tau$ is close to $\mathbf{0}$ for all $\tau = t-1, \cdots, 1$, then with $\boldsymbol{\sigma}$ being the logistic activation function, we have $\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^u) \approx \mathbf{0}$, and hence the cell error rate in (11.4.3) is simply $\boldsymbol{\delta}_{t,\tau} \approx \boldsymbol{\delta}_{t,\tau+1}$. In general, the error rates of a fully-trained network are close to zero such that the updates of the parameters in the network are small. However, when suffering from the vanishing gradient problem, the error rates diminish to zero in the process of backpropagating through time even when the model is not fully-trained, *i.e.* the optimal parameters are not yet attained. With $\boldsymbol{u}_\tau \approx \mathbf{0}$ in the gated RNN, we have a stable error rates at all timestep τ as $\boldsymbol{\delta}_{t,\tau} \approx \boldsymbol{\delta}_{t,\tau+1}$ across $\tau = 1, \cdots, t-1$, and hence, as long as $\boldsymbol{\delta}_{t,t}$ is significantly different from zero, the gated RNN avoids the vanishing gradient problem.

11.4.2 Example of Gated RNN: LSTM

Long Short-Term Memory (LSTM), first proposed in Hochreiter and Schmidhuber (1997), makes use of memory cells (central processing and storage units) that allow a constant propagation flow of error rates

through layers and time during the training of the network; in this way, as GRU, LSTM can avoid the vanishing gradient problem, and hence it allows for a *uniform credit assignment*, which means the backpropagation of error rates to inputs without scaling to be described further in Subsection 11.4.3. The following Figure 11.4.2 shows a memory cell of LSTM.

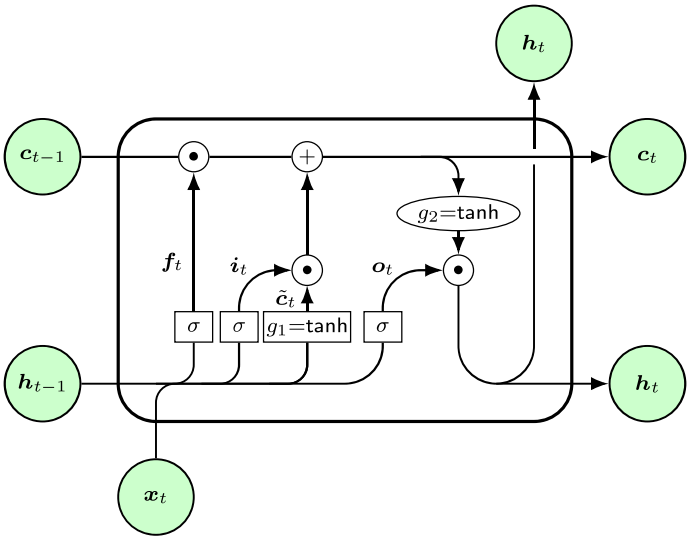

Figure 11.4.2: LSTM memory cell; also see [4].

(*Forget gate*) $$\boldsymbol{f}_t = \boldsymbol{\sigma}(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{V}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f)\,; \tag{11.4.4a}$$

(*Input gate*) $$\boldsymbol{i}_t = \boldsymbol{\sigma}(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{V}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i)\,; \tag{11.4.4b}$$

(*Cell input*) $$\tilde{\boldsymbol{c}}_t = \boldsymbol{g}_1(\boldsymbol{W}_{\tilde{c}} \boldsymbol{x}_t + \boldsymbol{V}_{\tilde{c}} \boldsymbol{h}_{t-1} + \boldsymbol{b}_{\tilde{c}})\,; \tag{11.4.4c}$$

(*Output gate*) $$\boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{V}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o)\,; \tag{11.4.4d}$$

(*Cell state*) $$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}\,; \tag{11.4.4e}$$

(*Cell output*) $$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{g}_2(\boldsymbol{c}_t)\,; \tag{11.4.4f}$$

where

1. for this memory cell, $\boldsymbol{x}_t$ stands for the input feature vector at the present timestep t;
2. matrices $\boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_{\tilde{c}}, \boldsymbol{W}_o$ correspond to the weights of inputs $\boldsymbol{x}_t$ used for the forget gate, input gate, cell input and output gate, respectively;
3. matrices $\boldsymbol{V}_f, \boldsymbol{V}_i, \boldsymbol{V}_{\tilde{c}}, \boldsymbol{V}_o$ correspond to the weights of the cell output activated value $\boldsymbol{h}_{t-1}$, fed from the one-step delay of itself for the forget gate, input gate, cell input and output gate, respectively;
4. vectors $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_{\tilde{c}}, \boldsymbol{b}_o$ correspond to the biases of the forget gate, input gate, cell input, and output gate, respectively;
5. activation functions $\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{\sigma}$ are used for the cell input, cell state and all other gates, respectively. Typically, hyperbolic tangent *tanh* is chosen for $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$; while logistic function can be adopted for $\boldsymbol{\sigma}$;
6. **Forget Gate** $\boldsymbol{f}_t$: decides what the last timestep cell input information should be dropped or kept. As an image value under a sigmoid type activation function, ranging in (0, 1), its value closer to 0 means to forget, while if it is closer to 1, it means to keep;

7. **Input Gate $\boldsymbol{i}_t$**: decides whether the input should be updated. As a sigmoid type activation function taking values in (0, 1), if its value is closer to 0 that means no update required but just using the last timestep; otherwise if it is closer to 1, one has to update by using $\tilde{\boldsymbol{c}}_t$; and
8. **Output Gate $\boldsymbol{o}_t$**: decides whether the current cell state will previously stored input value provide a non-zero output of the cell, which serves as an input for the next hidden layer.
9. with the input gate $\boldsymbol{i}$ and forget gate $\boldsymbol{f}$ taking values in $(0, 1)$, the cell state $\boldsymbol{c}_t$ represents the long term memory. If the forget gate is close to 1, then we keep all the cell state "memories" and further accumulate it with the (short-term information) cell input $\tilde{\boldsymbol{c}}$ through the input gate when forward propagating through time.
10. with $\boldsymbol{g}_2$ taking values in $(-1, 1)$, the cell output $\boldsymbol{h}_t$ represents the short-term memory.

11.4.2.1 Backpropagation through Time in LSTM

For illustration of the main idea, we only consider a 1-layer LSTM, where we adopt the RNN final output in (11.2.1b) as the output layer. As there are two products generated by the memory cell, namely the cell output and the cell state, we denote the two corresponding error rates be

$$\boldsymbol{\delta}_{t,\tau} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} \quad \text{and} \quad \boldsymbol{\lambda}_{t,\tau} := \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_\tau}\,, \qquad \tau = 1, \cdots, t\,,$$

where $\boldsymbol{\delta}_{t,t}$ is given in (11.2.5) as a RNN final output is set. Consider the cell output equation:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{g}_2(\boldsymbol{c}_t)\,.$$

the respective partial derivatives of the error with respect to $\boldsymbol{o}_t$ and $\boldsymbol{c}_t$ are:

$$\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{o}_t} = \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{o}_t}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} = \operatorname{diag}\left\{\boldsymbol{g}_2(\boldsymbol{c}_t)\right\}\boldsymbol{\delta}_{t,t} = \boldsymbol{\delta}_{t,t} \odot \boldsymbol{g}_2(\boldsymbol{c}_t)\,; \tag{11.4.5}$$

$$\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_t} = \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{c}_t}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} = \operatorname{diag}\left\{\boldsymbol{o}_t \odot \boldsymbol{g}_2'(\boldsymbol{c}_t)\right\}\boldsymbol{\delta}_{t,t} = \boldsymbol{\delta}_{t,t} \odot \boldsymbol{o}_t \odot \boldsymbol{g}_2'(\boldsymbol{c}_t) =: \boldsymbol{\lambda}_{t,t}\,. \tag{11.4.6}$$

Denote the respective weighted sums $\boldsymbol{z}_t$ for different gates and inputs by:

$$\begin{aligned}
\boldsymbol{z}_t^f &= \boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{V}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\,, && \text{where} \quad \boldsymbol{f}_t = \boldsymbol{\sigma}(\boldsymbol{z}_t^f)\,;\\
\boldsymbol{z}_t^i &= \boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{V}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\,, && \text{where} \quad \boldsymbol{i}_t = \boldsymbol{\sigma}(\boldsymbol{z}_t^i)\,;\\
\boldsymbol{z}_t^{\tilde{c}} &= \boldsymbol{W}_{\tilde{c}}\boldsymbol{x}_t + \boldsymbol{V}_{\tilde{c}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\tilde{c}}\,, && \text{where} \quad \tilde{\boldsymbol{c}}_t = \boldsymbol{g}_1(\boldsymbol{z}_t^{\tilde{c}})\,;\\
\boldsymbol{z}_t^o &= \boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{V}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\,, && \text{where} \quad \boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{z}_t^o)\,.
\end{aligned}$$

To backpropagate the cell error rate $\boldsymbol{\lambda}_{t,\tau+1}$ from $\tau+1$ to τ, for $\tau = t-1, \cdots, 1$,

$$\boldsymbol{\lambda}_{t,\tau} = \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_\tau} = \frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{c}_\tau}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_{\tau+1}} = \frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{c}_\tau}\boldsymbol{\lambda}_{t,\tau+1}\,. \tag{11.4.7}$$

Recall (11.4.4e) at $\tau+1$, we have $\boldsymbol{c}_{\tau+1} = \boldsymbol{i}_{\tau+1} \odot \tilde{\boldsymbol{c}}_{\tau+1} + \boldsymbol{f}_{\tau+1} \odot \boldsymbol{c}_\tau$. On the other hand, at the timestep τ in (11.4.4f), we have $\boldsymbol{h}_\tau = \boldsymbol{o}_\tau \odot \boldsymbol{g}_2(\boldsymbol{c}_\tau)$ with which all $\boldsymbol{i}_{\tau+1}, \tilde{\boldsymbol{c}}_{\tau+1}$, and $\boldsymbol{f}_{\tau+1}$ are functions of $\boldsymbol{c}_\tau$. Therefore, by the law of total derivative, similar to deriving (11.4.2),

$$\begin{aligned}
&\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{c}_\tau} = \frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{c}_\tau}\frac{\partial \boldsymbol{i}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{i}_{\tau+1}} + \frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{c}_\tau}\frac{\partial \tilde{\boldsymbol{c}}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \tilde{\boldsymbol{c}}_{\tau+1}} + \frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{c}_\tau}\frac{\partial \boldsymbol{f}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{f}_{\tau+1}} + \operatorname{diag}\left\{\boldsymbol{f}_{\tau+1}\right\}\\
&= \operatorname{diag}\left\{\boldsymbol{o}_\tau \odot \boldsymbol{g}_2'(\boldsymbol{c}_\tau)\right\}\left\{\frac{\partial \boldsymbol{i}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\operatorname{diag}\left\{\tilde{\boldsymbol{c}}_{\tau+1}\right\} + \frac{\partial \tilde{\boldsymbol{c}}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\operatorname{diag}\left\{\boldsymbol{i}_{\tau+1}\right\} + \frac{\partial \boldsymbol{f}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\operatorname{diag}\left\{\boldsymbol{c}_\tau\right\}\right\} + \operatorname{diag}\left\{\boldsymbol{f}_{\tau+1}\right\}\\
&= \operatorname{diag}\left\{\boldsymbol{o}_\tau \odot \boldsymbol{g}_2'(\boldsymbol{c}_\tau)\right\}\left\{\boldsymbol{V}_i^\top \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^i) \odot \tilde{\boldsymbol{c}}_{\tau+1}\right\} + \boldsymbol{V}_{\tilde{c}}^\top \operatorname{diag}\left\{\boldsymbol{g}_1'(\boldsymbol{z}_{\tau+1}^{\tilde{c}}) \odot \boldsymbol{i}_{\tau+1}\right\} + \boldsymbol{V}_f^\top \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^f) \odot \boldsymbol{c}_\tau\right\}\right\}\\
&\quad + \operatorname{diag}\left\{\boldsymbol{f}_{\tau+1}\right\}\,.
\end{aligned}$$

Therefore, for $\tau = 1, \cdots, t-1$, the cell error rate in (11.4.7) can be further computed as:

$$\boldsymbol{\lambda}_{t,\tau} = \text{diag}\left\{\boldsymbol{o}_\tau \odot \boldsymbol{g}_2'(\boldsymbol{c}_\tau)\right\}\left[\boldsymbol{V}_i^\top\left\{\boldsymbol{\lambda}_{t,\tau+1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^i) \odot \tilde{\boldsymbol{c}}_{\tau+1}\right\} + \boldsymbol{V}_{\tilde{c}}^\top\left\{\boldsymbol{\lambda}_{t,\tau+1} \odot \boldsymbol{g}_1'(\boldsymbol{z}_{\tau+1}^{\tilde{c}}) \odot \boldsymbol{i}_{\tau+1}\right\}\right.$$
$$\left. + \boldsymbol{V}_f^\top\left\{\boldsymbol{\lambda}_{t,\tau+1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\tau+1}^f) \odot \boldsymbol{c}_\tau\right\}\right] + \boldsymbol{f}_{\tau+1} \odot \boldsymbol{\lambda}_{t,\tau+1}\,. \tag{11.4.8}$$

Consider the parameter updates in the forget gate,

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_f} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_f} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{c}_\tau}{\partial \boldsymbol{W}_f}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{f}_\tau}{\partial \boldsymbol{W}_f}\frac{\partial \boldsymbol{c}_\tau}{\partial \boldsymbol{f}_\tau}\boldsymbol{\lambda}_{t,\tau}$$
$$= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial(\boldsymbol{W}_f\boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_f}\text{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^f)\right\}\left(\text{diag}\left\{\boldsymbol{c}_{\tau-1}\right\}\boldsymbol{\lambda}_{t,\tau}\right) = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \boldsymbol{c}_{\tau-1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_\tau^f)\right\}\boldsymbol{x}_\tau^\top\,,$$
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_f} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_f} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \boldsymbol{c}_{\tau-1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_\tau^f)\right\}\boldsymbol{h}_{\tau-1}^\top\,,$$
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_f} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_f} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \boldsymbol{c}_{\tau-1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_\tau^f)\right\}\,.$$

Likewise, for the parameter updates in the input gate,

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_i} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_i} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{c}_\tau}{\partial \boldsymbol{W}_i}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{i}_\tau}{\partial \boldsymbol{W}_i}\frac{\partial \boldsymbol{c}_\tau}{\partial \boldsymbol{i}_\tau}\boldsymbol{\lambda}_{t,\tau}$$
$$= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial(\boldsymbol{W}_i\boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_i}\text{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}_\tau^i)\right\}\text{diag}\left\{\tilde{\boldsymbol{c}}_\tau\right\}\boldsymbol{\lambda}_{t,\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \tilde{\boldsymbol{c}}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}_\tau^i)\right\}\boldsymbol{x}_\tau^\top\,,$$
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_i} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_i} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \tilde{\boldsymbol{c}}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}_\tau^i)\right\}\boldsymbol{h}_{\tau-1}^\top\,,$$
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_i} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_i} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \tilde{\boldsymbol{c}}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}_\tau^i)\right\}\,.$$

Similarly, for the parameter updates in the cell input,

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_{\tilde{c}}} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_{\tilde{c}}} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{c}_\tau}{\partial \boldsymbol{W}_{\tilde{c}}}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{c}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \tilde{\boldsymbol{c}}_\tau}{\partial \boldsymbol{W}_{\tilde{c}}}\frac{\partial \boldsymbol{c}_\tau}{\partial \tilde{\boldsymbol{c}}_\tau}\boldsymbol{\lambda}_{t,\tau}$$
$$= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial(\boldsymbol{W}_{\tilde{c}}\boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_{\tilde{c}}}\text{diag}\left\{\boldsymbol{g}_1'(\boldsymbol{z}_\tau^{\tilde{c}})\right\}\text{diag}\left\{\boldsymbol{i}_\tau\right\}\boldsymbol{\lambda}_{t,\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \boldsymbol{i}_\tau \odot \boldsymbol{g}_1'(\boldsymbol{z}_\tau^{\tilde{c}})\right\}\boldsymbol{x}_\tau^\top\,,$$
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_{\tilde{c}}} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_{\tilde{c}}} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \boldsymbol{i}_\tau \odot \boldsymbol{g}_1'(\boldsymbol{z}_\tau^{\tilde{c}})\right\}\boldsymbol{h}_{\tau-1}^\top\,,$$
$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_{\tilde{c}}} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_{\tilde{c}}} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\lambda}_{t,\tau} \odot \boldsymbol{i}_\tau \odot \boldsymbol{g}_1'(\boldsymbol{z}_\tau^{\tilde{c}})\right\}\,.$$

Finally, there is still an output gate to be dealt, which involves more argument as follows. The next step is to backpropagate the state error rate $\boldsymbol{\delta}_{t,\tau+1}$ from $\tau+1$ to τ, *i.e.*

$$\boldsymbol{\delta}_{t,\tau} = \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_{\tau+1}} = \frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\boldsymbol{\delta}_{t,\tau+1}\,.$$

Note that $\boldsymbol{o}_{\tau+1}$ and $\boldsymbol{c}_{\tau+1}$ are functions of $\boldsymbol{h}_\tau$ from (11.4.4d) and (11.4.4e), which is further dealt to (11.4.4a), (11.4.4b) and (11.4.4c). From (11.4.4f) at $\tau+1$ that $\boldsymbol{h}_{\tau+1} = \boldsymbol{o}_{\tau+1} \odot \boldsymbol{g}_2(\boldsymbol{c}_{\tau+1})$, by law of total derivative, we

have

$$\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{h}_\tau} = \frac{\partial \boldsymbol{o}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{o}_{\tau+1}} + \frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{h}_{\tau+1}}{\partial \boldsymbol{c}_{\tau+1}} = \boldsymbol{V}_o^\top \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}^o_{\tau+1})\right\}\operatorname{diag}\left\{\mathbf{1}\odot \boldsymbol{g}_2(\boldsymbol{c}_{\tau+1})\right\} + \frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\operatorname{diag}\left\{\boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1})\right\}$$

$$= \boldsymbol{V}_o^\top \operatorname{diag}\left\{\boldsymbol{g}_2(\boldsymbol{c}_{\tau+1})\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_{\tau+1})\right\} + \frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\operatorname{diag}\left\{\boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1})\right\}\,.$$

To compute $\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{h}_\tau}$, we first recall that $\boldsymbol{f}_{\tau+1}$, $\boldsymbol{i}_{\tau+1}$, and $\tilde{\boldsymbol{c}}_{\tau+1}$ are functions of $\boldsymbol{h}_\tau$ from (11.4.4a), (11.4.4b), and (11.4.4c) respectively. Then, applying the law of total derivative to (11.4.4e), we obtain

$$\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{h}_\tau} = \frac{\partial \boldsymbol{i}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{i}_{\tau+1}} + \frac{\partial \boldsymbol{f}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \boldsymbol{f}_{\tau+1}} + \frac{\partial \tilde{\boldsymbol{c}}_{\tau+1}}{\partial \boldsymbol{h}_\tau}\frac{\partial \boldsymbol{c}_{\tau+1}}{\partial \tilde{\boldsymbol{c}}_{\tau+1}}$$

$$= \boldsymbol{V}_i^\top \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}^i_{\tau+1})\right\}\operatorname{diag}\left\{\tilde{\boldsymbol{c}}_{\tau+1}\right\} + \boldsymbol{V}_f^\top \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}^f_{\tau+1})\right\}\operatorname{diag}\left\{\boldsymbol{c}_\tau\right\} + \boldsymbol{V}_{\tilde{c}}^\top \operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}^{\tilde{c}}_{\tau+1})\right\}\operatorname{diag}\left\{\boldsymbol{i}_{\tau+1}\right\}$$

$$= \boldsymbol{V}_i^\top \operatorname{diag}\left\{\tilde{\boldsymbol{c}}_{\tau+1}\odot \boldsymbol{\sigma}'(\boldsymbol{z}^i_{\tau+1})\right\} + \boldsymbol{V}_f^\top \operatorname{diag}\left\{\boldsymbol{c}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}^f_{\tau+1})\right\} + \boldsymbol{V}_{\tilde{c}}^\top \operatorname{diag}\left\{\boldsymbol{i}_{\tau+1}\odot \boldsymbol{\sigma}'(\boldsymbol{z}^{\tilde{c}}_{\tau+1})\right\}\,.$$

Therefore, for $\tau = 1, \cdots, t-1$, we have

$$\boldsymbol{\delta}_{t,\tau} = \boldsymbol{V}_o^\top\left\{\boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{g}_2(\boldsymbol{c}_{\tau+1})\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_{\tau+1})\right\} + \left[\boldsymbol{V}_i^\top \operatorname{diag}\left\{\tilde{\boldsymbol{c}}_{\tau+1}\odot \boldsymbol{\sigma}'(\boldsymbol{z}^i_{\tau+1})\right\}\right.$$

$$\left. + \boldsymbol{V}_f^\top \operatorname{diag}\left\{\boldsymbol{c}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}^f_{\tau+1})\right\} + \boldsymbol{V}_{\tilde{c}}^\top \operatorname{diag}\left\{\boldsymbol{i}_{\tau+1}\odot \boldsymbol{g}_1'(\boldsymbol{z}^{\tilde{c}}_{\tau+1})\right\}\right]\left\{\boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1})\right\}$$

$$= \boldsymbol{V}_o^\top\left\{\boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{g}_2(\boldsymbol{c}_{\tau+1})\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_{\tau+1})\right\} + \boldsymbol{V}_i^\top\left\{\boldsymbol{\delta}_{t,\tau+1}\odot \tilde{\boldsymbol{c}}_{\tau+1}\odot \boldsymbol{\sigma}'(\boldsymbol{z}^i_{\tau+1})\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1})\right\}$$

$$+ \boldsymbol{V}_f^\top\left\{\boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{c}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}^f_{\tau+1})\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1})\right\} + \boldsymbol{V}_{\tilde{c}}^\top\left\{\boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{i}_{\tau+1}\odot \boldsymbol{g}_1'(\boldsymbol{z}^{\tilde{c}}_{\tau+1})\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1})\right\}$$

$$= \begin{pmatrix} \boldsymbol{V}_o^\top & \boldsymbol{V}_i^\top & \boldsymbol{V}_f^\top & \boldsymbol{V}_{\tilde{c}}^\top \end{pmatrix}\begin{pmatrix} \boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{g}_2(\boldsymbol{c}_{\tau+1})\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_{\tau+1}) \\ \boldsymbol{\delta}_{t,\tau+1}\odot \tilde{\boldsymbol{c}}_{\tau+1}\odot \boldsymbol{\sigma}'(\boldsymbol{z}^i_{\tau+1})\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1}) \\ \boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{c}_\tau \odot \boldsymbol{\sigma}'(\boldsymbol{z}^f_{\tau+1})\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1}) \\ \boldsymbol{\delta}_{t,\tau+1}\odot \boldsymbol{i}_{\tau+1}\odot \boldsymbol{g}_1'(\boldsymbol{z}^{\tilde{c}}_{\tau+1})\odot \boldsymbol{o}_{\tau+1}\odot \boldsymbol{g}_2'(\boldsymbol{c}_{\tau+1}) \end{pmatrix}\,.$$

Eventually, for the parameter updates in the output gate, we see:

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_o} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{W}_o} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial \boldsymbol{h}_\tau}{\partial \boldsymbol{W}_o}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_\tau} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\frac{\partial (\boldsymbol{W}_o\boldsymbol{x}_\tau)}{\partial \boldsymbol{W}_o}\operatorname{diag}\left\{\boldsymbol{\sigma}'(\boldsymbol{z}^o_\tau)\right\}\operatorname{diag}\left\{\boldsymbol{g}_2(\boldsymbol{c}_\tau)\right\}\boldsymbol{\delta}_{t,\tau}$$

$$= \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\delta}_{t,\tau}\odot \boldsymbol{g}_2(\boldsymbol{c}_\tau)\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_\tau)\right\}\boldsymbol{x}_\tau^\top\,,$$

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{V}_o} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{V}_o} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\delta}_{t,\tau}\odot \boldsymbol{g}_2(\boldsymbol{c}_\tau)\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_\tau)\right\}\boldsymbol{h}_{\tau-1}^\top\,,$$

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{b}_o} = \sum_{t=1}^{D}\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{b}_o} = \sum_{t=1}^{D}\sum_{\tau=1}^{t}\left\{\boldsymbol{\delta}_{t,\tau}\odot \boldsymbol{g}_2(\boldsymbol{c}_\tau)\odot \boldsymbol{\sigma}'(\boldsymbol{z}^o_\tau)\right\}\,.$$

11.4.3 Uniform Credit Assignment

Uniform credit assignment stands for the property that LSTM can consider all input information at each phase of learning, no matter where they are located in the input temporal sequence. From (11.4.8), we observe that the cell error rate can be written in the form:

$$\boldsymbol{\lambda}_{t,\tau} = \operatorname{diag}\left\{\boldsymbol{o}_\tau \odot \boldsymbol{g}_2'(\boldsymbol{c}_\tau)\right\}\left(\boldsymbol{A}_{\tau+1}(\boldsymbol{V}_i) + \boldsymbol{B}_{\tau+1}(\boldsymbol{V}_f) + \boldsymbol{C}_{\tau+1}(\boldsymbol{V}_{\tilde{c}})\right) + \boldsymbol{f}_{\tau+1}\boldsymbol{\lambda}_{t,\tau+1}\,,$$

where the last term does not depend on the weights $\boldsymbol{V}$, and

$$\boldsymbol{A}_{\tau+1}(\boldsymbol{V}_i) = \boldsymbol{V}_i^\top \Big\{ \boldsymbol{\lambda}_{t,\tau+1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^i_{\tau+1}) \odot \tilde{\boldsymbol{c}}_{\tau+1} \Big\};$$

$$\boldsymbol{B}_{\tau+1}(\boldsymbol{V}_f) = \boldsymbol{V}_f^\top \Big\{ \boldsymbol{\lambda}_{t,\tau+1} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^f_{\tau+1}) \odot \boldsymbol{c}_{\tau} \Big\};$$

$$\boldsymbol{C}_{\tau+1}(\boldsymbol{V}_{\tilde{c}}) = \boldsymbol{V}_{\tilde{c}}^\top \Big\{ \boldsymbol{\lambda}_{t,\tau+1} \odot \boldsymbol{g}_1'(\boldsymbol{z}^{\tilde{c}}_{\tau+1}) \odot \boldsymbol{i}_{\tau+1} \Big\}.$$

Therefore, if the value of the forget gate $\boldsymbol{f}$ is close to $\mathbf{1}$ and that of the input gate $\boldsymbol{i}$ is close to $\mathbf{0}$, then both $\boldsymbol{z}^f$ and $\boldsymbol{z}^i$ take extreme values, very large in positive and in negative, respectively. Therefore, $\boldsymbol{\sigma}'(\boldsymbol{z}^f) \approx \mathbf{0}$ and $\boldsymbol{\sigma}'(\boldsymbol{z}^i) \approx \mathbf{0}$, as a result $\boldsymbol{A}_{\tau+1}(\boldsymbol{V}_i)$, $\boldsymbol{B}_{\tau+1}(\boldsymbol{V}_f)$ and $\boldsymbol{C}_{\tau+1}(\boldsymbol{V}_{\tilde{c}})$ are all very close to $\mathbf{0}$. Hence, if an output cell error occurs only at the end of the sequence, *i.e.* $\boldsymbol{\lambda}_{t,t}$ is significantly different from zero, such a memory cell, via backpropagation through time, supplies the same cell error rate $\boldsymbol{\lambda}_{t,\tau}$ in every timestep $\tau = 1, \cdots, t$ that would not vanish and stabilize. Thus, all weights $\boldsymbol{W}$'s, $\boldsymbol{V}$'s, and biases $\boldsymbol{b}$'s of a memory cell are updated by the same cell error rate $\boldsymbol{\lambda}_{t,t}$, and thus, LSTM avoids the vanishing gradient problem through time.

Moreover, with the input gate $\boldsymbol{i}$ being close to zero, the memory cell just sums up all the fed-in inputs it receives during scanning the input temporal sequence, *i.e.* $\boldsymbol{i}_\tau = \mathbf{0}$, for all $\tau = 1, \cdots, t$,

$$\boldsymbol{c}_\tau = \boldsymbol{i}_\tau \odot \tilde{\boldsymbol{c}}_\tau + \boldsymbol{f}_\tau \odot \boldsymbol{c}_{\tau-1} \qquad \mapsto \qquad \boldsymbol{c}_\tau = \tilde{\boldsymbol{c}}_\tau + \boldsymbol{f}_\tau \odot \boldsymbol{c}_{\tau-1}.$$

Thus, such a memory cell is the same as a unit that reads in all sequence elements at one time.

Bibliography

[1] Burkov, A. (2019). *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada.

[2] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[3] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[4] Olah, C. (2015). Understanding lstm networks.