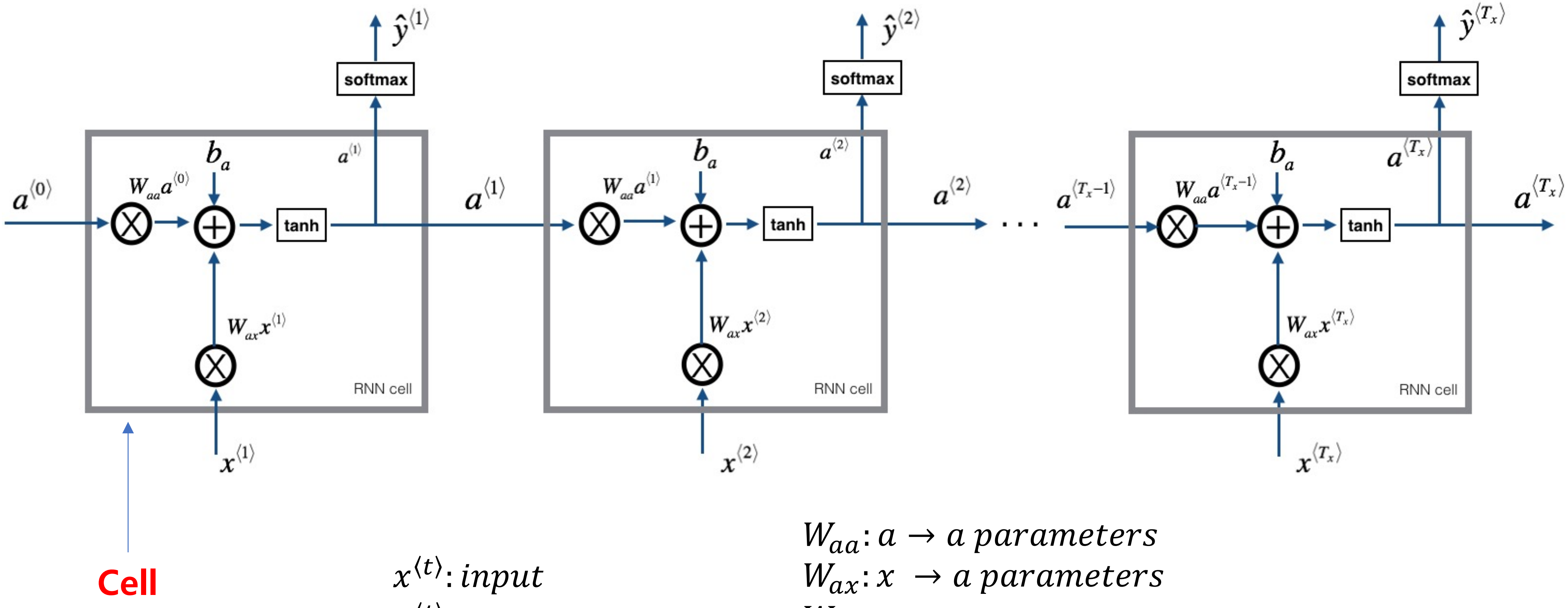
Building a RNN with numpy

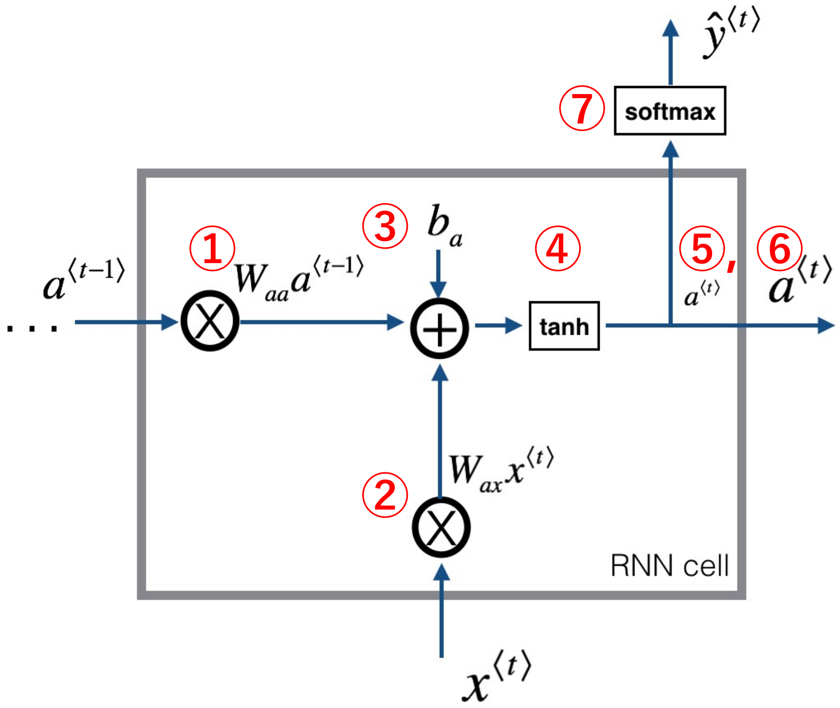
Structure of RNN



$x^{\langle t \rangle}: input$
$y^{\langle t \rangle}: output$
$a^{\langle t \rangle}: activation$

$W_{aa}: a \rightarrow a\ parameters$
$W_{ax}: x \rightarrow a\ parameters$
$W_{ya}: a \rightarrow y\ parameters$
$b_a: bias\ for\ tanh\ activation$
$b_y: bais\ for\ softmax\ activation$

RNN: Forward process (cell)



- **Given**
  $x^{\langle t \rangle}: input$
  $a^{\langle t-1 \rangle}: activation$

- **Parameters**
  $W_{ax}: x \rightarrow a\ parameters$
  $W_{aa}: a \rightarrow a\ parameters$
  $W_{ya}: a \rightarrow y\ parameters$
  $b_a: bias\ for\ tanh\ activation$
  $b_y: bais\ for\ softmax\ activation$

- **Computations**

④ ② ① ③

$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y)$$

⑦ ⑤ ⑥

- **Python code**

parameters

```
# Retrieve parameters from "parameters"
Wax = parameters["Wax"]
Waa = parameters["Waa"]
Wya = parameters["Wya"]
ba = parameters["ba"]
by = parameters["by"]
```

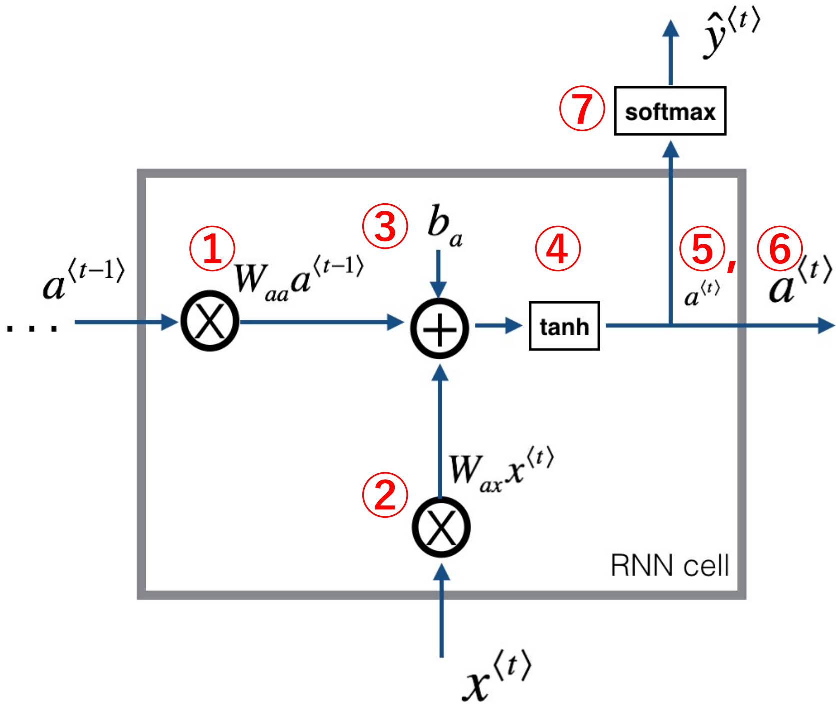
Computations

```
a_next = np.tanh(np.dot(Waa, a_prev) + np.dot(Wax, xt) + ba)
```

```
yt_pred = softmax(np.dot(Wya, a_next) + by)
```

→
```
def rnn_cell_forward(xt, a_prev, parameters):
```

RNN: Forward process (cell)



- **Given**
  $x^{\langle t \rangle}: input$
  $a^{\langle t-1 \rangle}: activation$

- **Parameters**
  $W_{ax}: x \rightarrow a\ parameters$
  $W_{aa}: a \rightarrow a\ parameters$
  $W_{ya}: a \rightarrow y\ parameters$
  $b_a: bias\ for\ tanh\ activation$
  $b_y: bais\ for\ softmax\ activation$

- **Computations**

$$a^{\langle t \rangle} = \overset{④}{\tanh}(\overset{②}{W_{ax}x^{\langle t \rangle}} + \overset{①}{W_{aa}a^{\langle t-1 \rangle}} + \overset{③}{b_a})$$

$$\hat{y}^{\langle t \rangle} = \underset{⑦}{soft\max}(\underset{⑤}{W_{ya}a^{\langle t \rangle}} + \underset{⑥}{b_y})$$

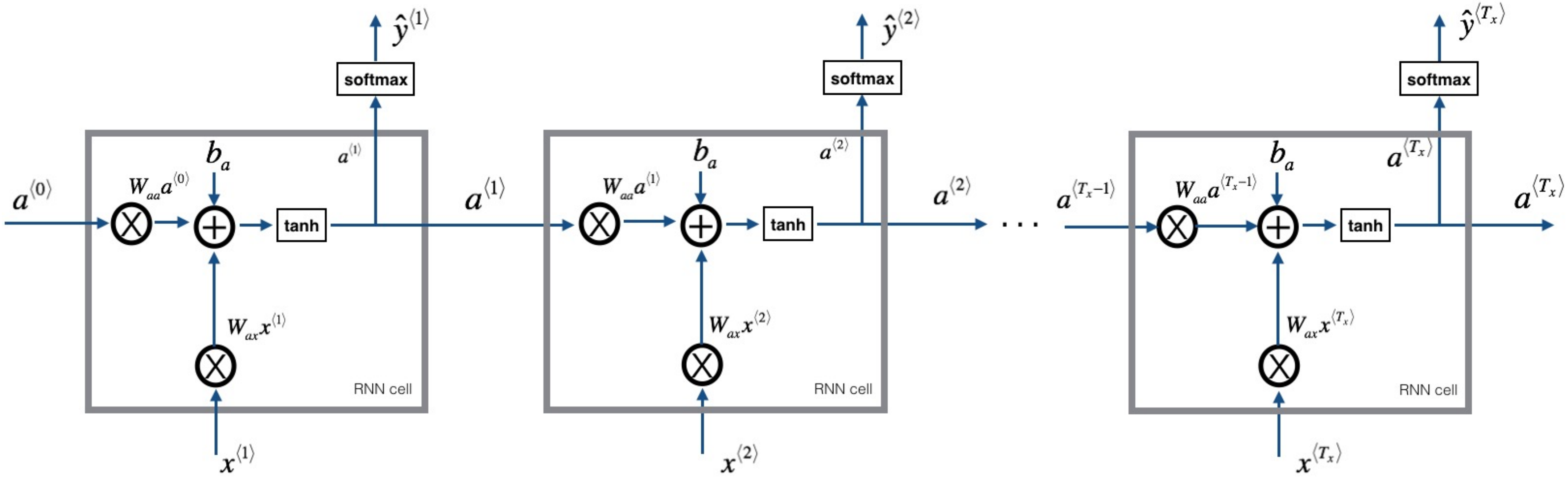
- **Python code**

```
np.random.seed(1)
xt = np.random.randn(3,10)
a_prev = np.random.randn(5,10)
Waa = np.random.randn(5,5)
Wax = np.random.randn(5,3)
Wya = np.random.randn(2,5)
ba = np.random.randn(5,1)
by = np.random.randn(2,1)
parameters = {"Waa": Waa, "Wax": Wax, "Wya": Wya, "ba": ba, "by": by}
```

```
a_next, yt_pred, cache = rnn_cell_forward(xt, a_prev, parameters)
```

a_prev = $a^{\langle t-1 \rangle}$
a_next = $a^{\langle t \rangle}$

RNN: Forward process (network)



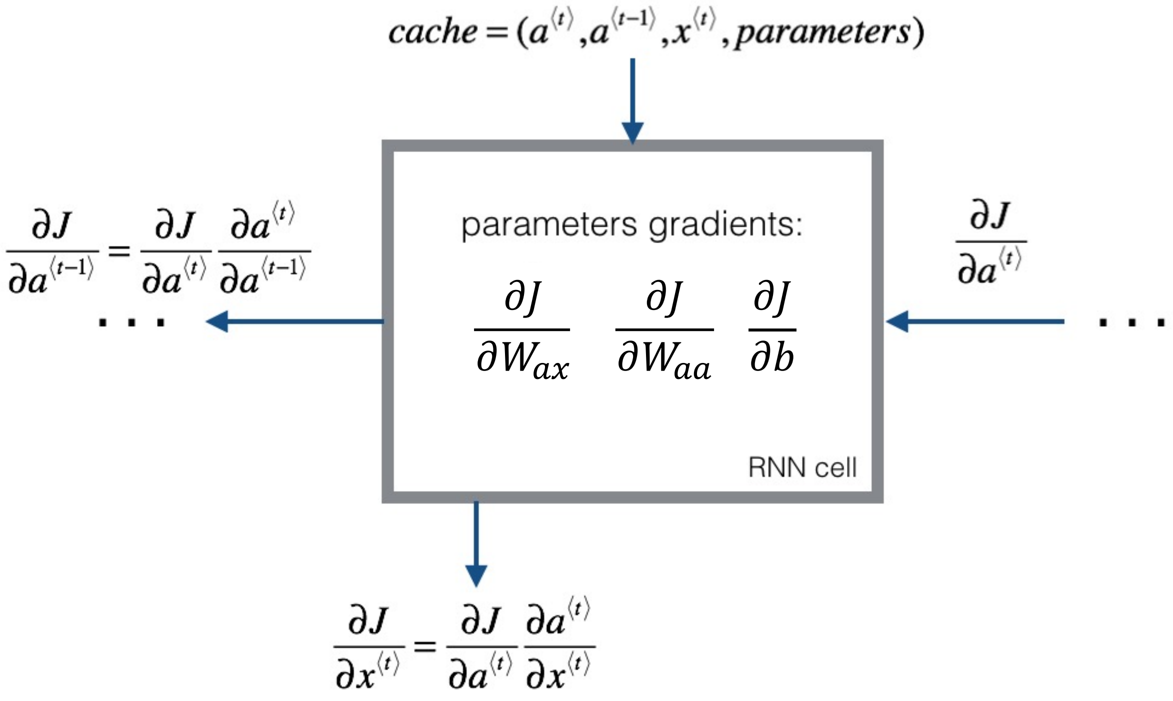

```
rnn_cell_forward(xt, a_prev, parameters)
```

```
rnn_cell_forward(xt, a_prev, parameters)
```

```
rnn_cell_forward(xt, a_prev, parameters)
```

Cellでのforwardを繰り返すだけ

RNN: Backward process (cell)



$$a^{\langle t \rangle} = \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{ax}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) x^{\langle t \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{aa}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) a^{\langle t-1 \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial b} = \sum_{batch} (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial x^{\langle t \rangle}} = W_{ax}{}^T . (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

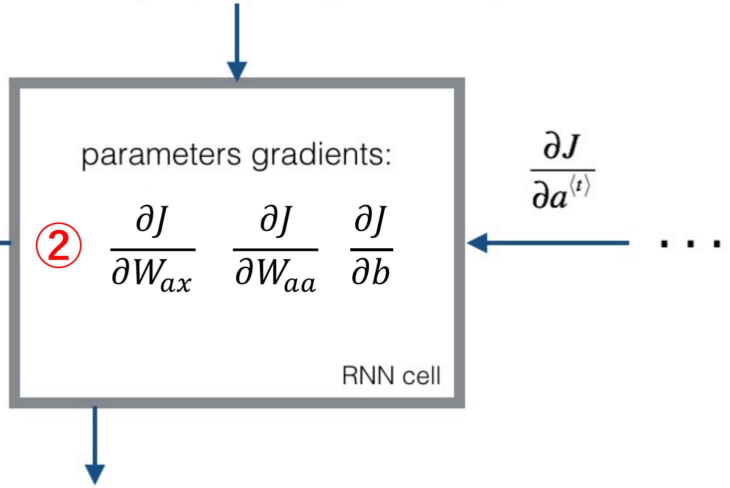
$$\frac{\partial a^{\langle t \rangle}}{\partial a^{\langle t-1 \rangle}} = W_{aa}{}^T . (1 - \tanh(W_{ax} x^{\langle t-1 \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$x^{\langle t \rangle}: input$
$y^{\langle t \rangle}: output$
$a^{\langle t \rangle}: activation$

$W_{aa}: a \rightarrow a\ parameters$
$W_{ax}: x \rightarrow a\ parameters$
$b_a: bias\ for\ tanh\ activation$
$b_y: bais\ for\ softmax\ activation$
$J: loss\ function$

RNN: Backward process (cell)

$$cache = (a^{\langle t\rangle}, a^{\langle t-1\rangle}, x^{\langle t\rangle}, parameters)$$



③ $\frac{\partial J}{\partial a^{\langle t-1\rangle}} = \frac{\partial J}{\partial a^{\langle t\rangle}} \frac{\partial a^{\langle t\rangle}}{\partial a^{\langle t-1\rangle}}$ ...

② $\frac{\partial J}{\partial W_{ax}}$ $\frac{\partial J}{\partial W_{aa}}$ $\frac{\partial J}{\partial b}$

$\frac{\partial J}{\partial a^{\langle t\rangle}}$ ...

① $\frac{\partial J}{\partial x^{\langle t\rangle}} = \frac{\partial J}{\partial a^{\langle t\rangle}} \frac{\partial a^{\langle t\rangle}}{\partial x^{\langle t\rangle}}$

$$a^{\langle t\rangle} = \tanh(W_{ax}x^{\langle t\rangle} + W_{aa}a^{\langle t-1\rangle} + b)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$$

$$\left.\begin{aligned}
\frac{\partial a^{\langle t\rangle}}{\partial W_{ax}} &= (1 - \tanh(W_{ax}x^{\langle t\rangle} + W_{aa}a^{\langle t-1\rangle} + b)^2)x^{\langle t\rangle T} \\
\frac{\partial a^{\langle t\rangle}}{\partial W_{aa}} &= (1 - \tanh(W_{ax}x^{\langle t\rangle} + W_{aa}a^{\langle t-1\rangle} + b)^2)a^{\langle t-1\rangle T} \\
\frac{\partial a^{\langle t\rangle}}{\partial b} &= \sum_{batch}(1 - \tanh(W_{ax}x^{\langle t\rangle} + W_{aa}a^{\langle t-1\rangle} + b)^2)
\end{aligned}\right\} ②$$

$$\frac{\partial a^{\langle t\rangle}}{\partial x^{\langle t\rangle}} = W_{ax}{}^T.(1 - \tanh(W_{ax}x^{\langle t\rangle} + W_{aa}a^{\langle t-1\rangle} + b)^2) \rightarrow ①$$

$$\frac{\partial a^{\langle t\rangle}}{\partial a^{\langle t-1\rangle}} = W_{aa}{}^T.(1 - \tanh(W_{ax}x^{\langle t-1\rangle} + W_{aa}a^{\langle t-1\rangle} + b)^2) \rightarrow ③$$

①

$$\frac{\partial J}{\partial x^{\langle t\rangle}} = \frac{\partial J}{\partial a^{\langle t\rangle}} * \frac{\partial a^{\langle t\rangle}}{\partial x^{\langle t\rangle}}$$

$$\frac{\partial J}{\partial a^{\langle t\rangle}} = da_{next}$$

$$\frac{\partial a^{\langle t\rangle}}{\partial x^{\langle t\rangle}} = W_{ax}^T(1 - a^{\langle t\rangle 2})$$

$$\frac{\partial J}{\partial x^{\langle t\rangle}} = W_{ax}^T\underline{(1 - a^{\langle t\rangle 2})da_{next}}$$

```
dtanh = (1- a_next**2) * da_next
```

```
dxt = np.dot(Wax.T, dtanh)
```

②

$$\frac{\partial J}{\partial W_{ax}} = \frac{\partial J}{\partial a^{\langle t\rangle}} * \frac{\partial a^{\langle t\rangle}}{\partial W_{ax}}$$

$$\frac{\partial J}{\partial a^{\langle t\rangle}} = da_{next}$$

$$\frac{\partial a^{\langle t\rangle}}{\partial W_{ax}} = (1 - a^{\langle t\rangle 2})x^{\langle t\rangle T}$$

$$\frac{\partial J}{\partial x^{\langle t\rangle}} = (1 - a^{\langle t\rangle 2})da_{next}x^{\langle t\rangle T}$$

```
dWax = np.dot(dtanh, xt.T)
```

③

$$\frac{\partial J}{\partial a^{\langle t-1\rangle}} = \frac{\partial J}{\partial a^{\langle t\rangle}} * \frac{\partial a^{\langle t\rangle}}{\partial a^{\langle t-1\rangle}}$$

$$\frac{\partial J}{\partial a^{\langle t\rangle}} = da_{next}$$

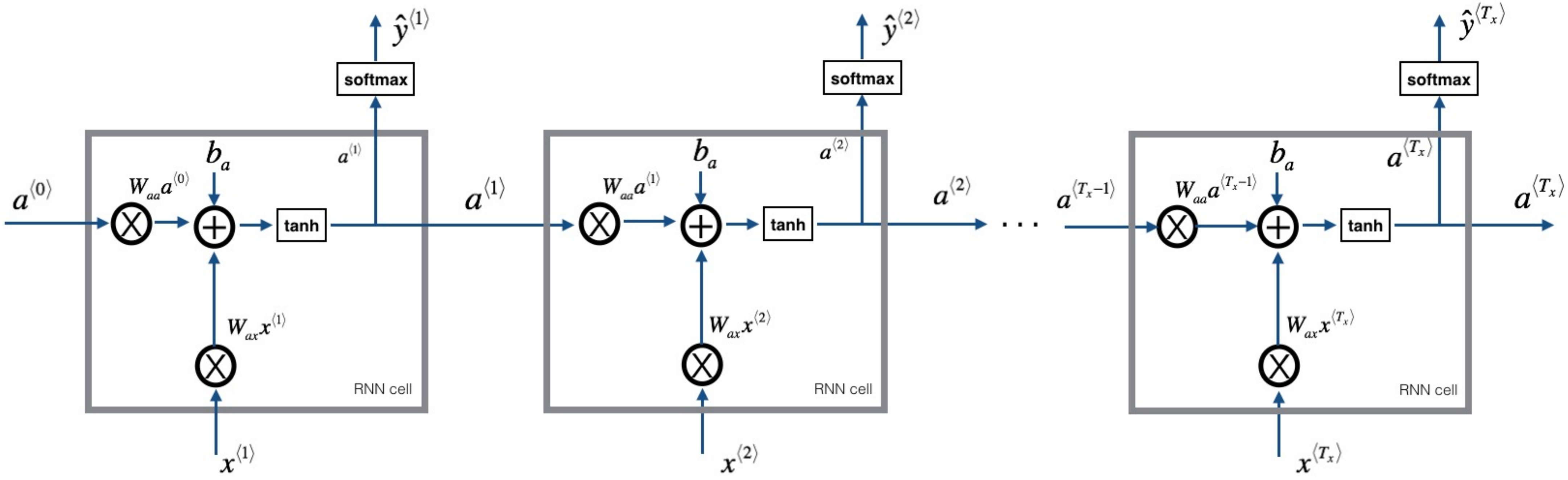
$$\frac{\partial a^{\langle t\rangle}}{\partial a^{\langle t-1\rangle}} = W_{aa}(1 - a^{\langle t\rangle 2})$$

$$\frac{\partial J}{\partial a^{\langle t-1\rangle}} = W_{aa}(1 - a^{\langle t\rangle 2})da_{next}$$

```
def rnn_cell_backward(da_next, cache):
```

```
da_prev = np.dot(Waa.T, dtanh)
```

RNN: Backward process (network)




```python
def rnn_cell_backward(da_next, cache):
```

```python
def rnn_cell_backward(da_next, cache):
```

```python
def rnn_cell_backward(da_next, cache):
```

Cellでのbackwardを繰り返すだけ