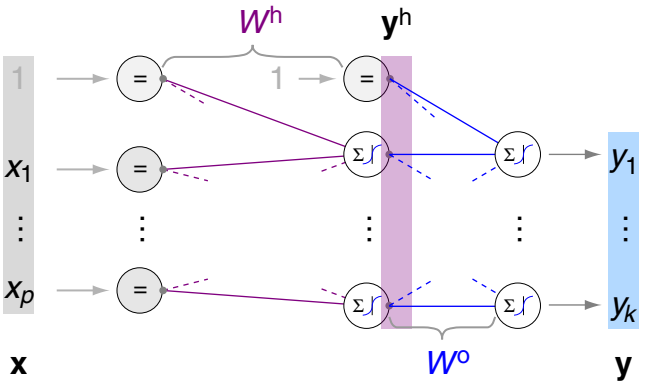
Chapter ML:IX

IX. Deep Learning for Sequence Processing

- Introduction to Deep Learning
- RNN Sequence Encoding
- RNN Sequence Decoding
- RNNs for Machine Translation
- Long-Term Dependencies
- Attention Mechanism
- Transformer Architecture
- Transformer Advanced
- Language Models
- Language Model Training

RNN Sequence Decoding

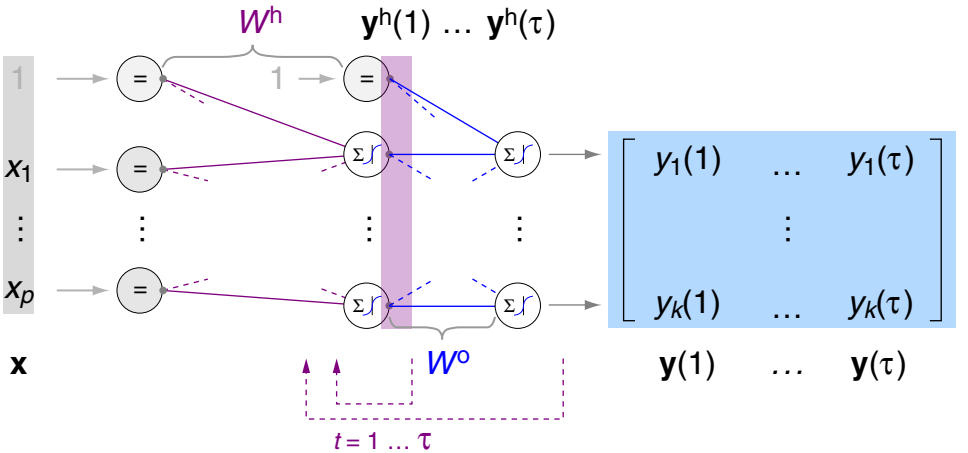
From Vector Output to Sequence Output



- One p-dimensional input vector $\mathbf{x}$.
- One hidden layer (general: $d-1$ hidden layers, i.e., d active layers).
- One k-dimensional output vector $\mathbf{y}(\mathbf{x})$.

RNN Sequence Decoding

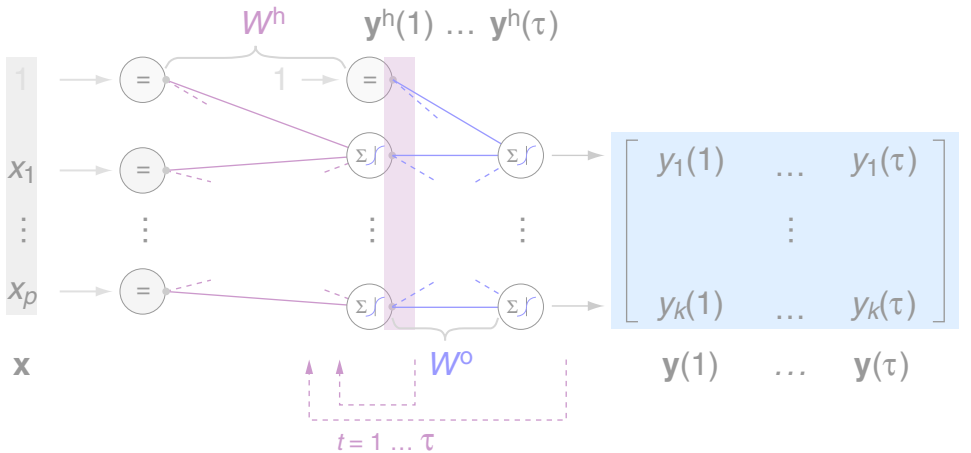
From Vector Output to Sequence Output



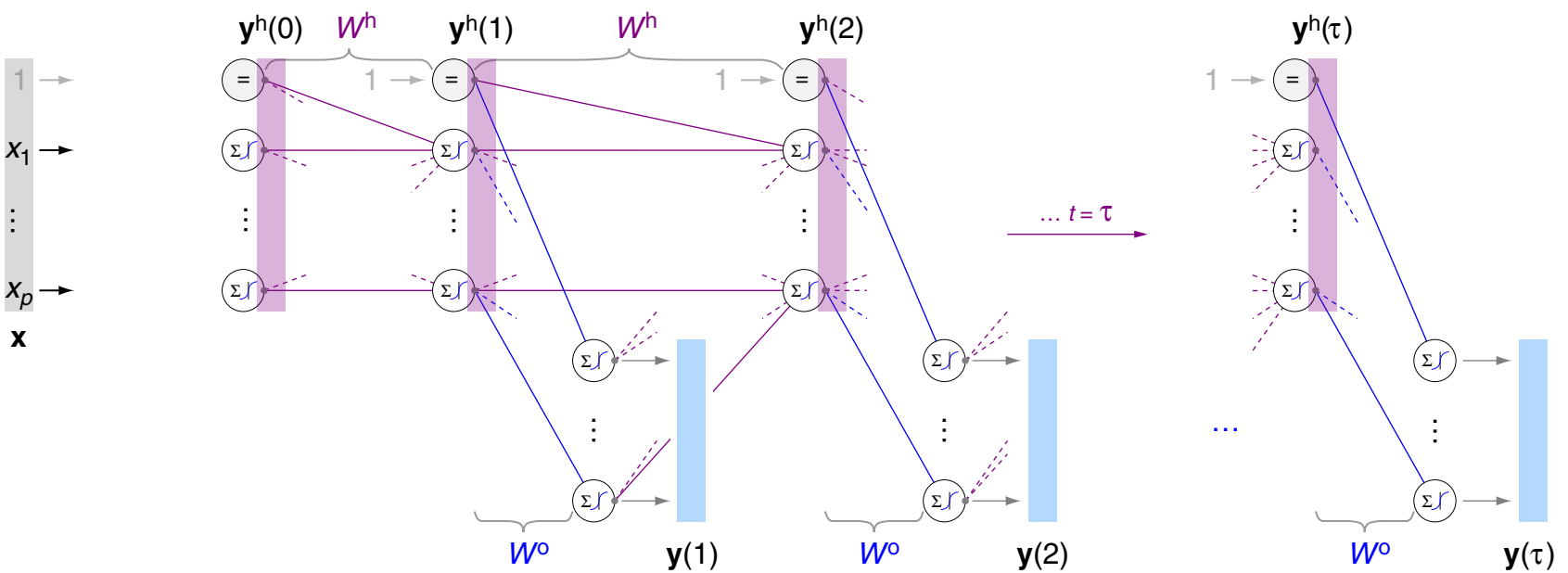
- One p-dimensional input vector $\mathbf{x}$.
- One hidden and one output layer, which are recurrently updated.
- Sequence of k-dimen. output vectors $[\mathbf{y}(\mathbf{x}, 1), \ldots, \mathbf{y}(\mathbf{x}, \tau)]$ or $[\mathbf{y}(1), \ldots, \mathbf{y}(\tau)]$.

RNN Sequence Decoding

From Vector Output to Sequence Output

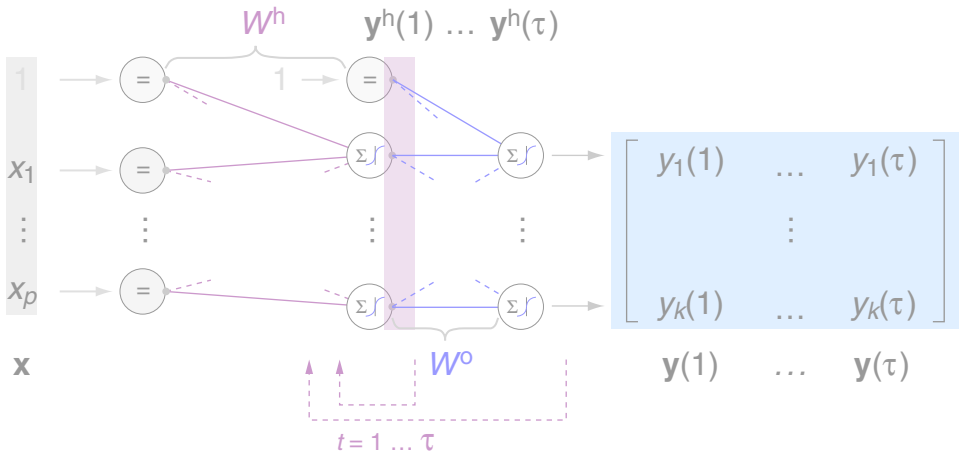


Unfold hidden and output layer

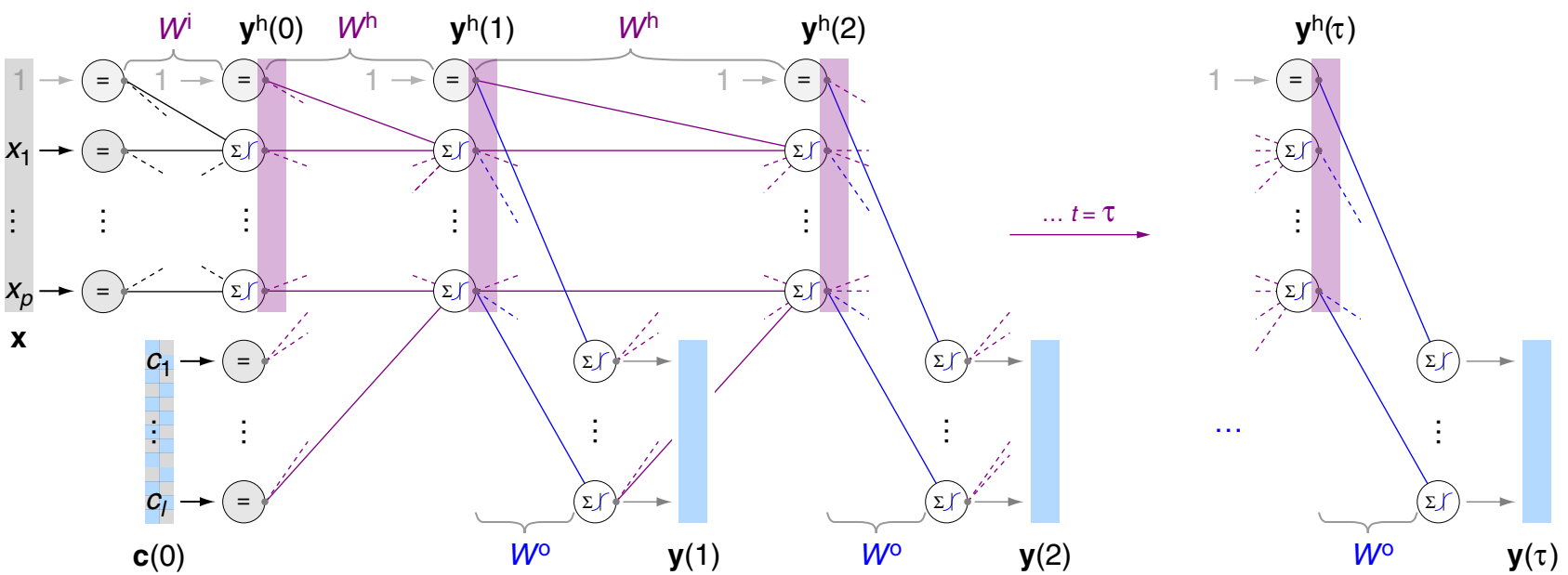


RNN Sequence Decoding

From Vector Output to Sequence Output



Unfold hidden and output layer



Remarks:

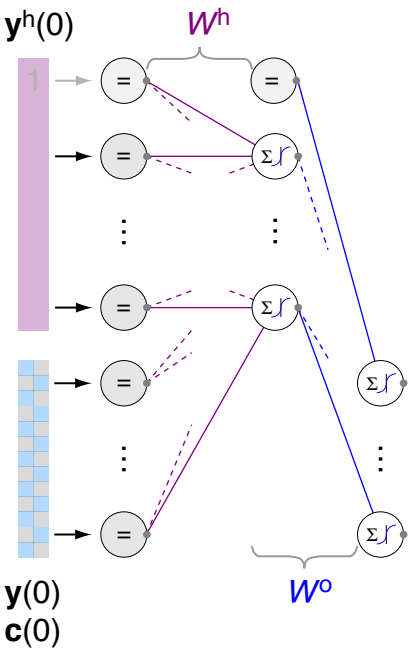
- An output sequence is written in brackets, $[\mathbf{y}(1), \ldots, \mathbf{y}(\tau)]$, where $\mathbf{y}(t), t = 1, \ldots, \tau$, denotes the output vector at time step t.
- The words in the output sequence are usually one-hot-encoded, i.e., by a k-dimensional output vector with a "1" whose position indicates the word, and zeros elsewhere.
- If the input, $\mathbf{x}$, is clear from the context, we usually note $\mathbf{y}(\mathbf{x}, t)$ as $\mathbf{y}(t)$.
- The matrix W^{i} is necessary to embed the typically low-dimensional input vector $\mathbf{x}$ regarding the high-dimensional hidden vectors $\mathbf{y}^h$: $\mathbf{y}^{\mathsf{h}}(0) = \boldsymbol{\sigma}\left(W^{\mathsf{i}}\mathbf{x}\right)$.
- The parameter τ in $\mathbf{y}(\tau)$ is unknown. More specifically, the generation process terminates at that time step τ for which $\mathbf{y}(\tau) = (0, 0, \ldots, 0, 1)^T$ ($\widehat{=}$ <end>).

  τ (the length of the computed output) does not have to be equal to T (the length of the target or ground truth).

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 0 → 1

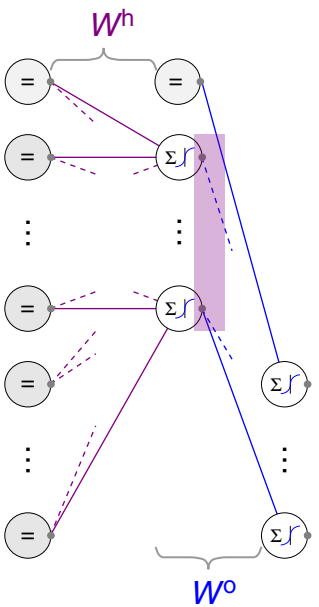


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 0 → 1

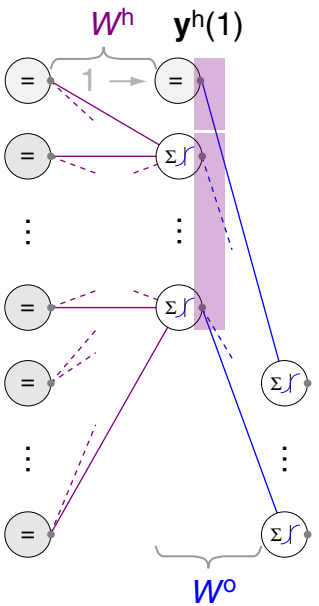


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 0 → 1

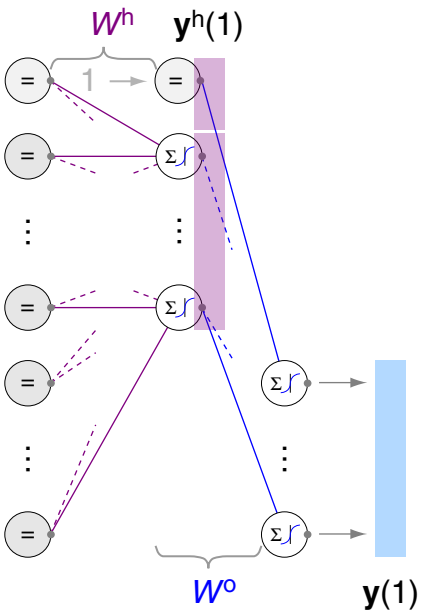


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 0 → 1

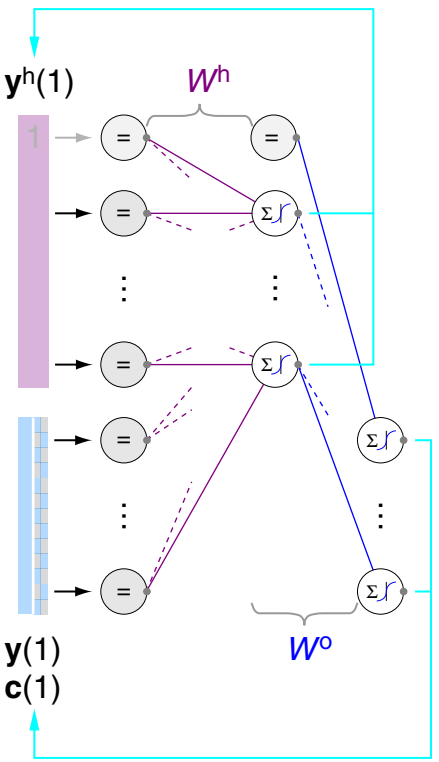


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 0 → 1

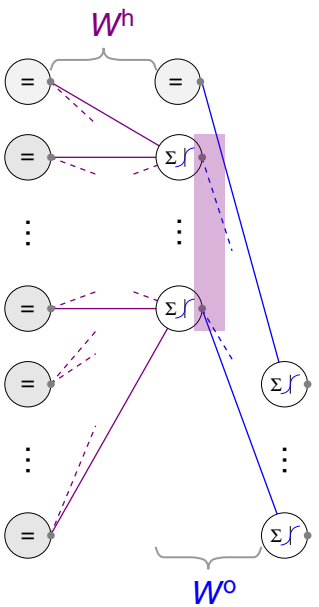


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 1 → 2

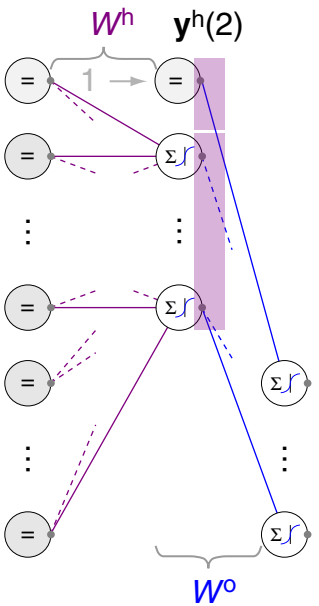


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 1 → 2

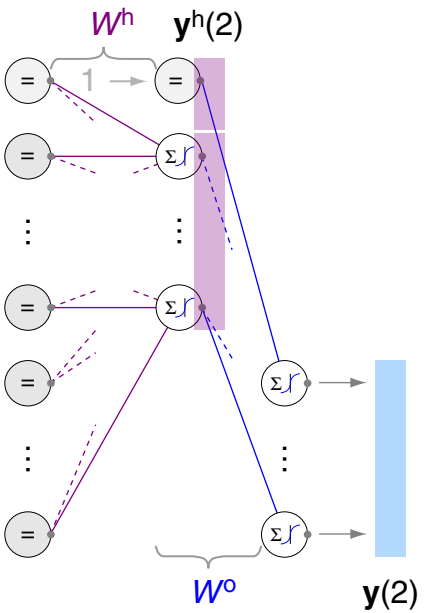


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 1 → 2

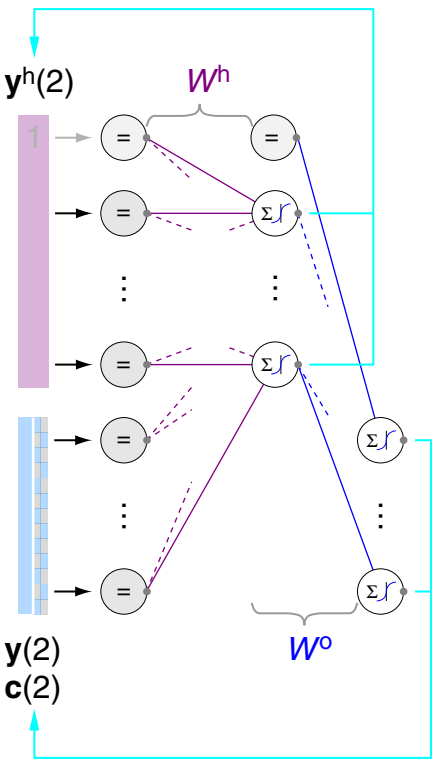


Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation [decoding overview]

t: 1 → 2

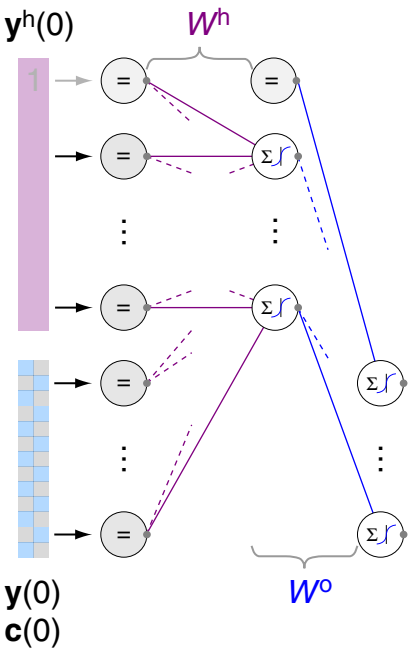


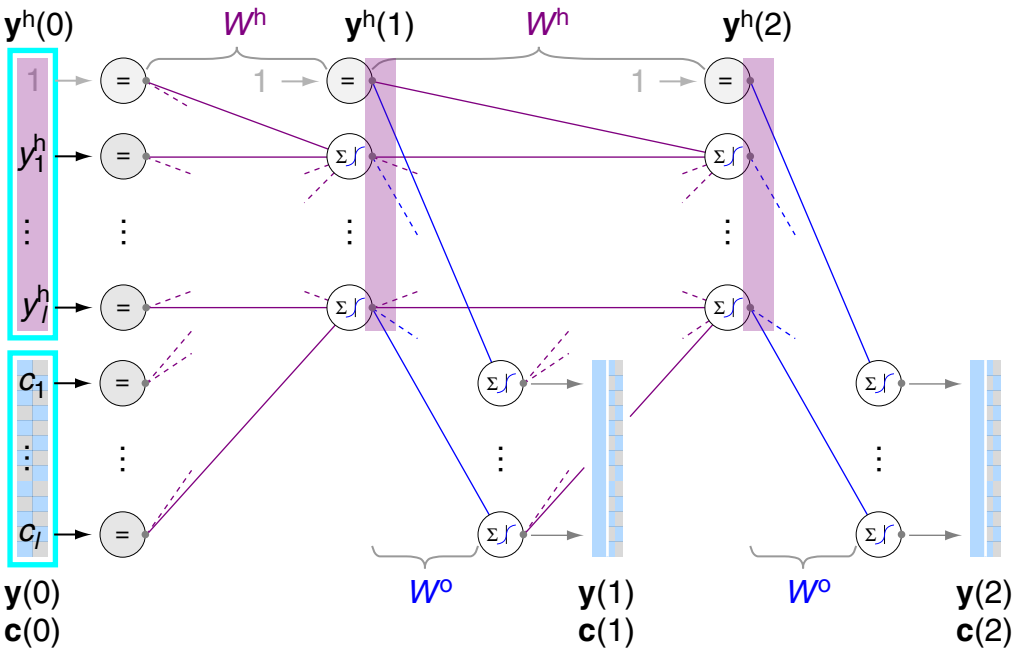
Output generation over t.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 0 → 1





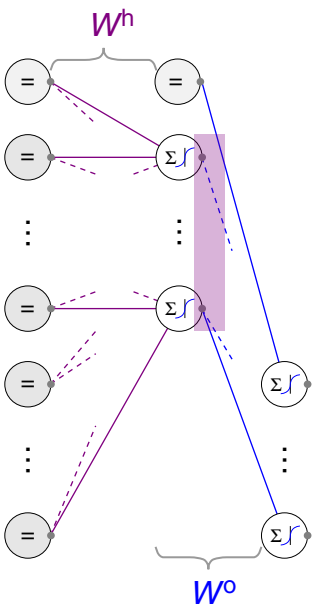
Output generation over t.

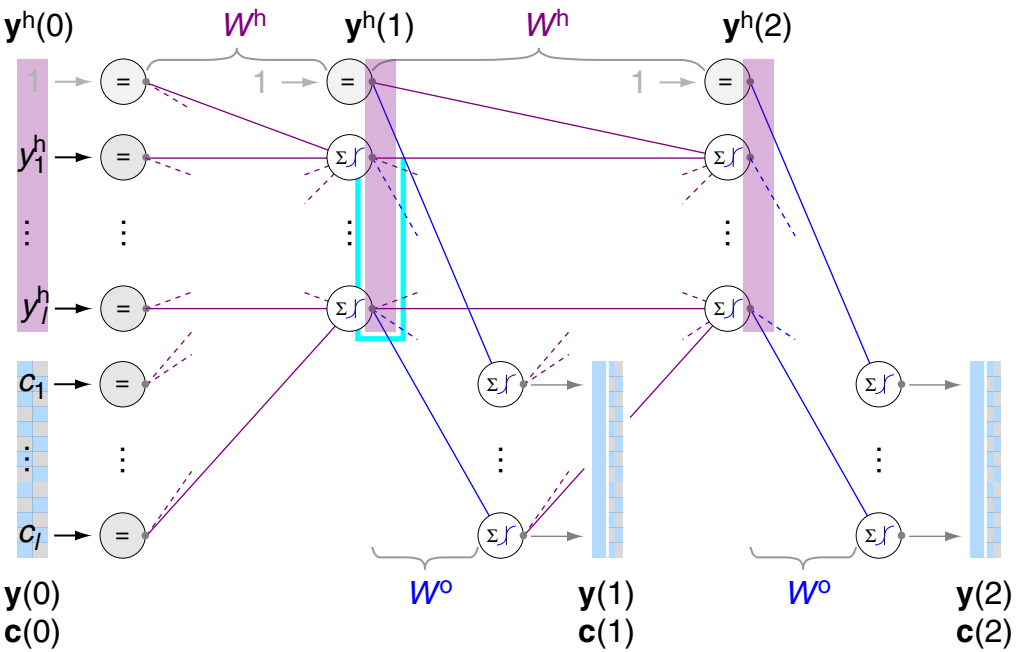
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 0 → 1





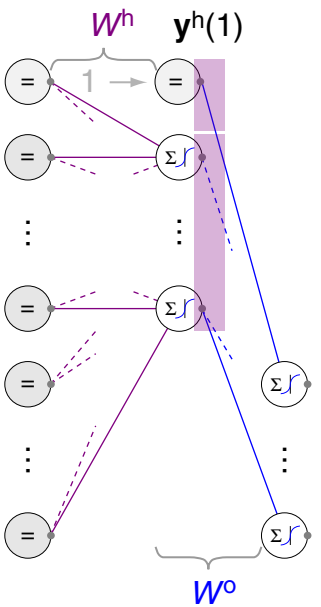
Output generation over t.

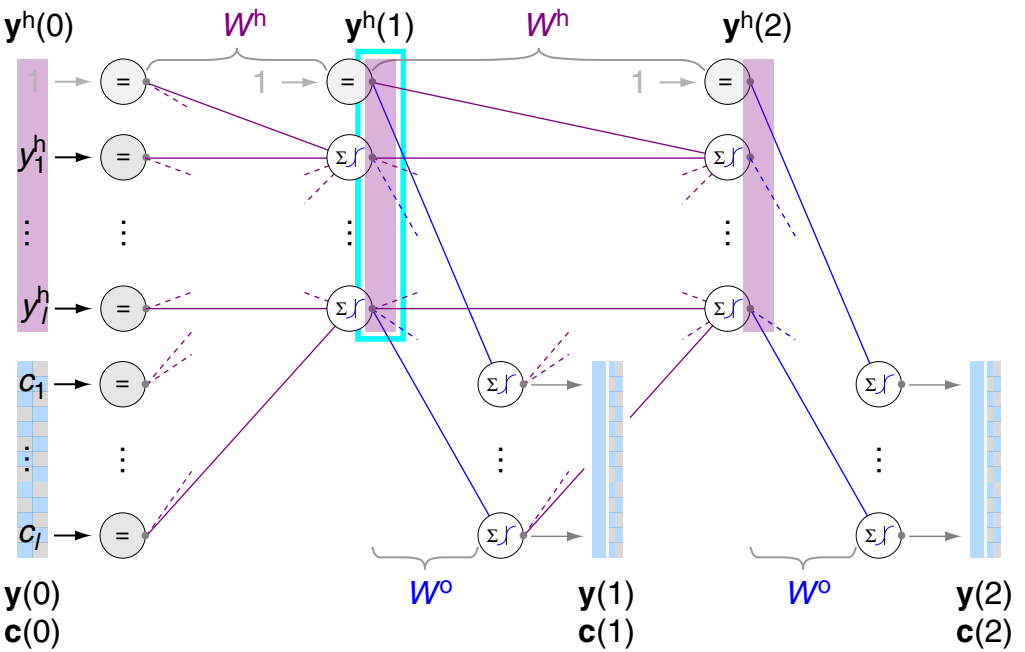
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 0 → 1





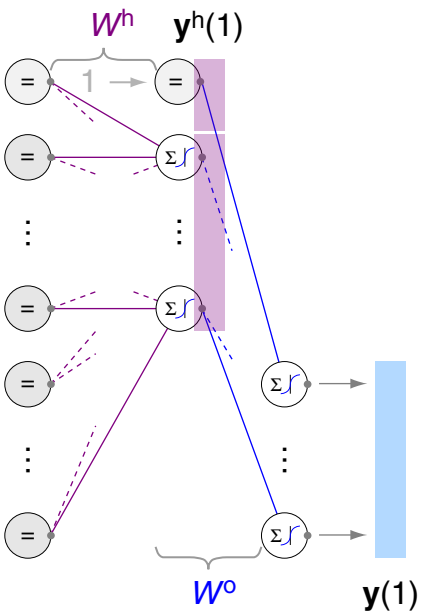
Output generation over t.

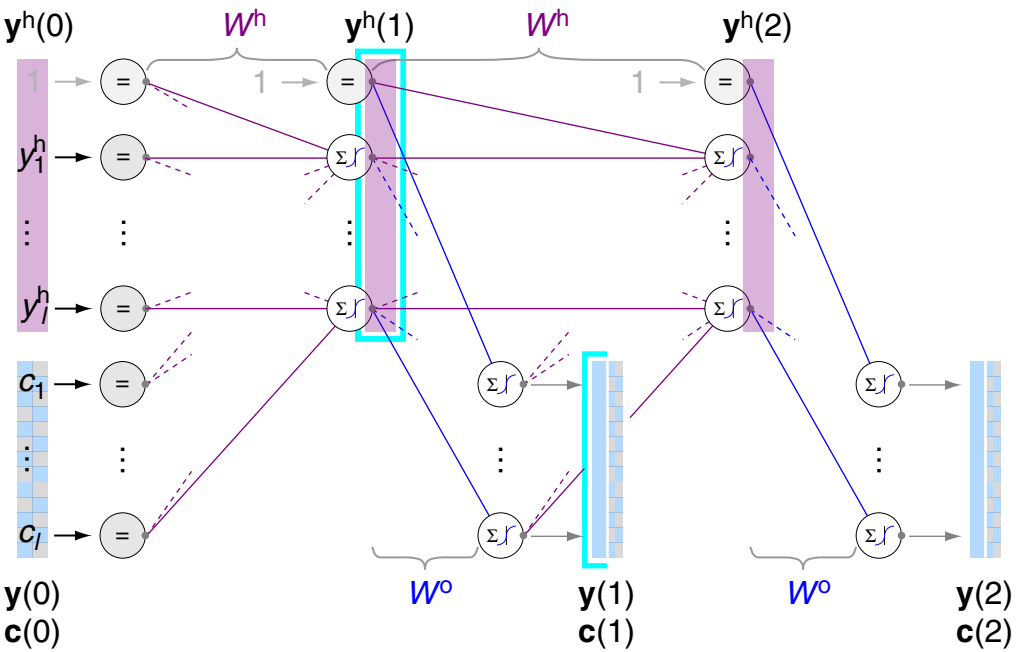
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 0 → 1



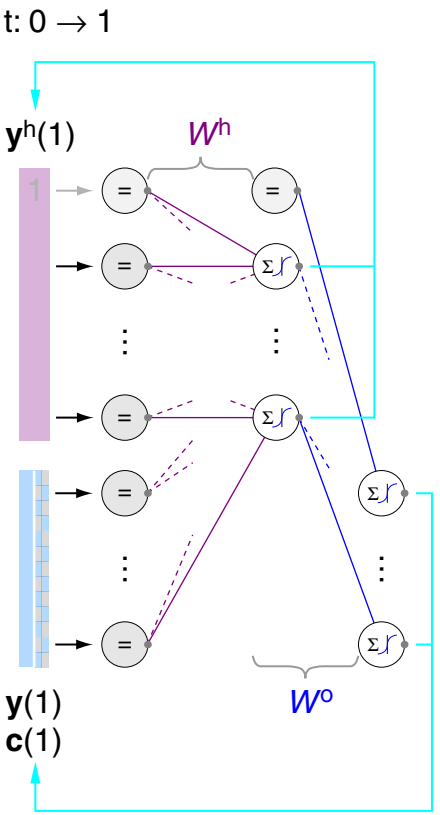


Output generation over t.

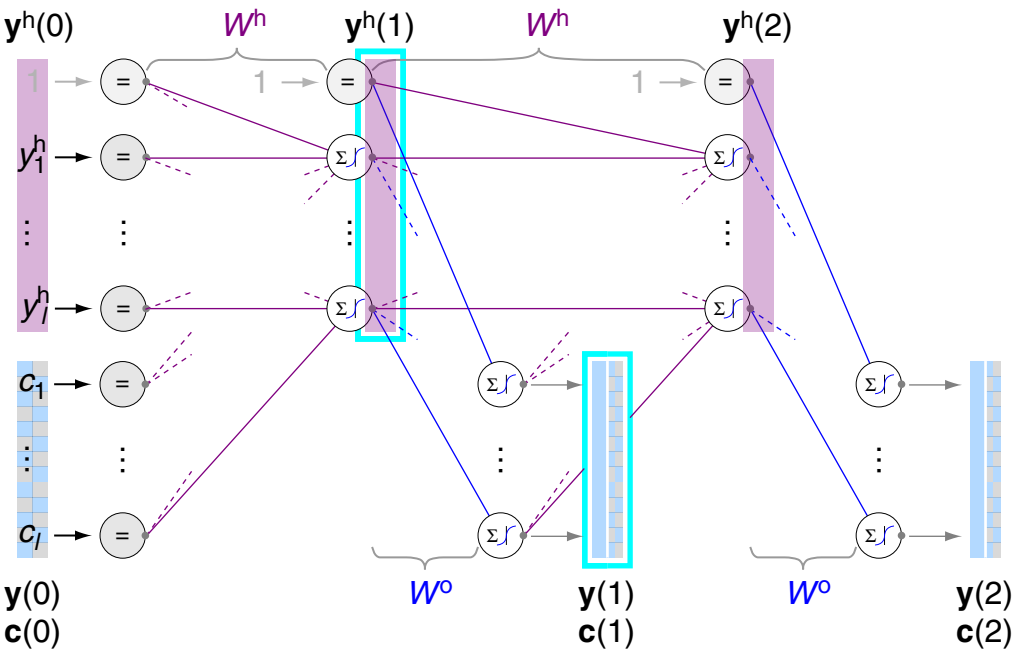
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]



Output generation over t.

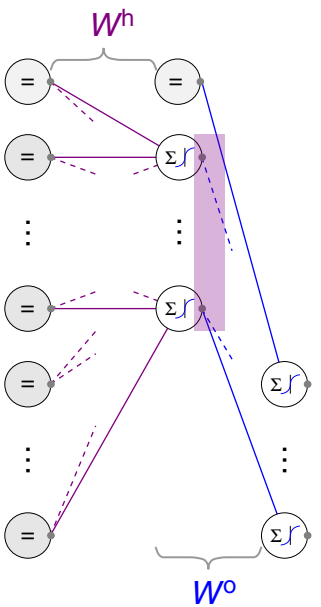


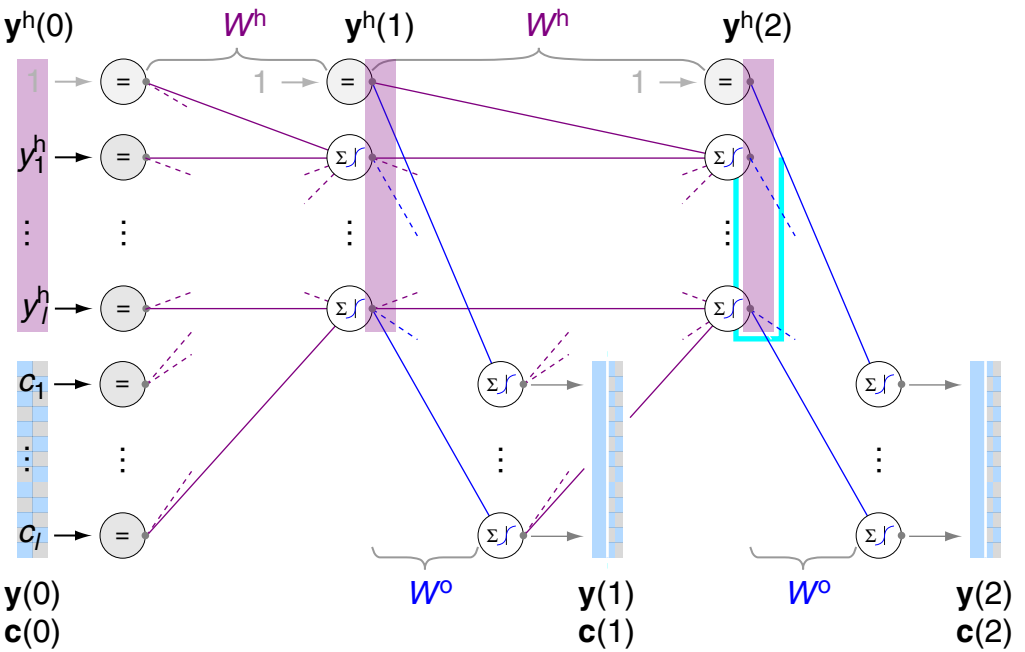
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 1 → 2





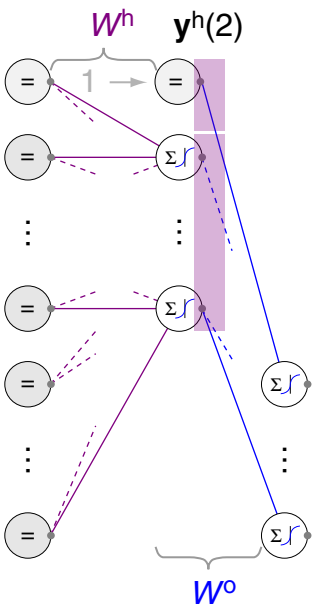
Output generation over t.

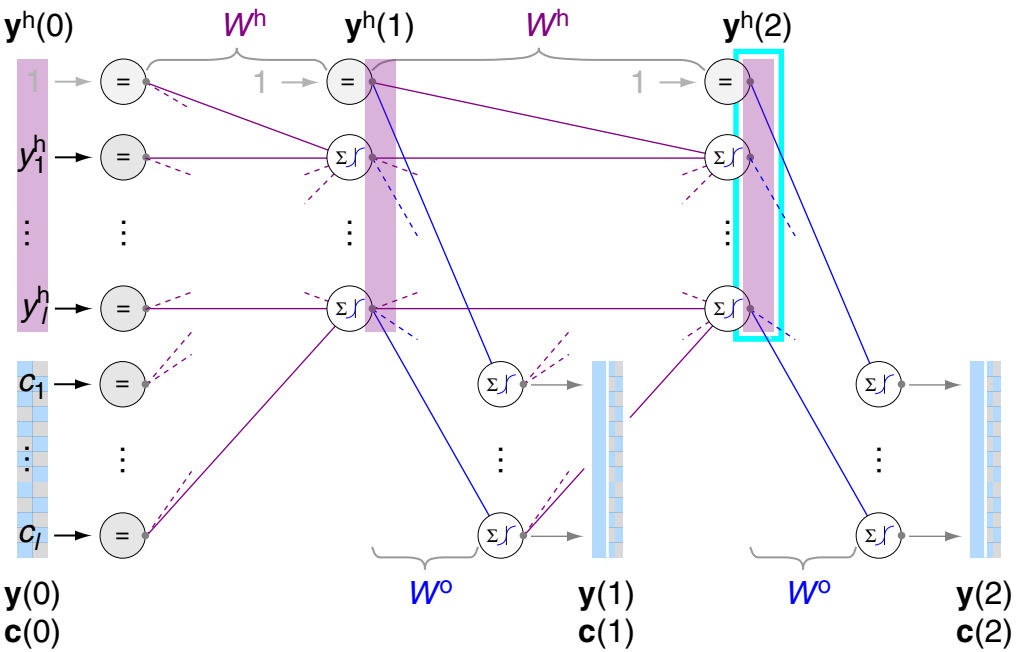
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 1 → 2





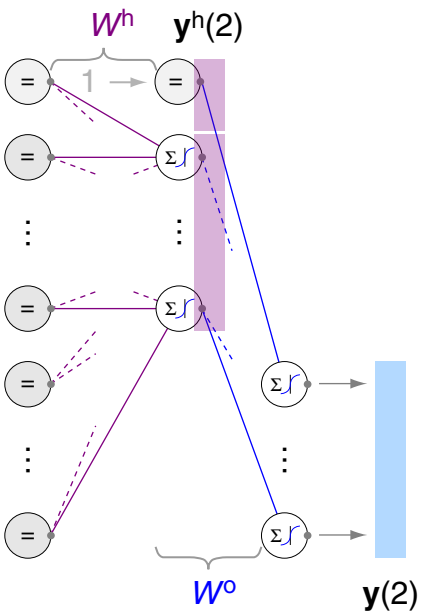
Output generation over t.

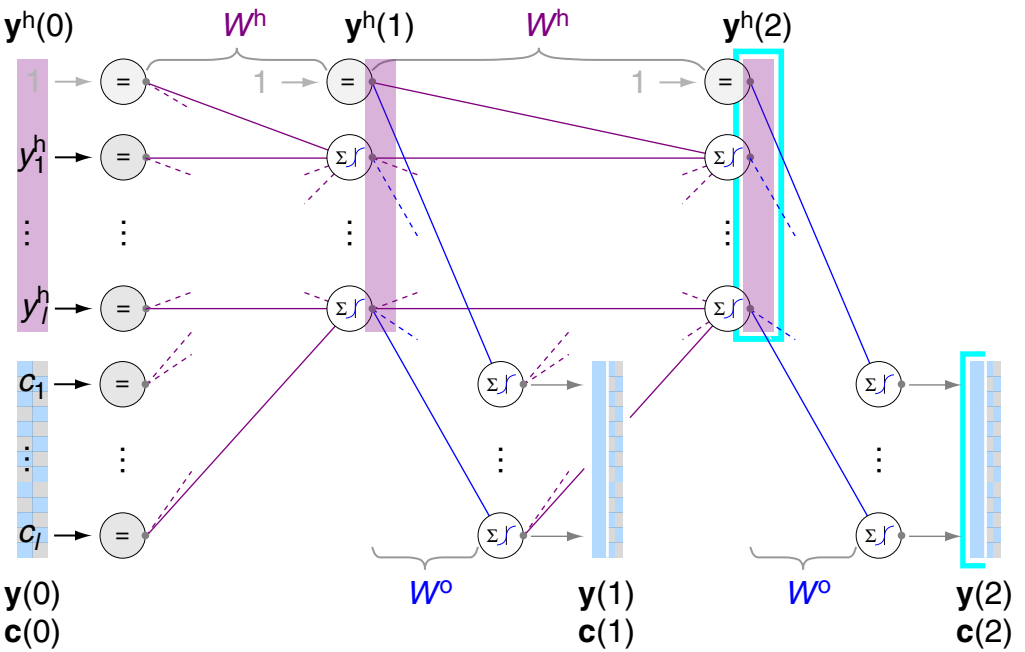
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]

t: 1 → 2



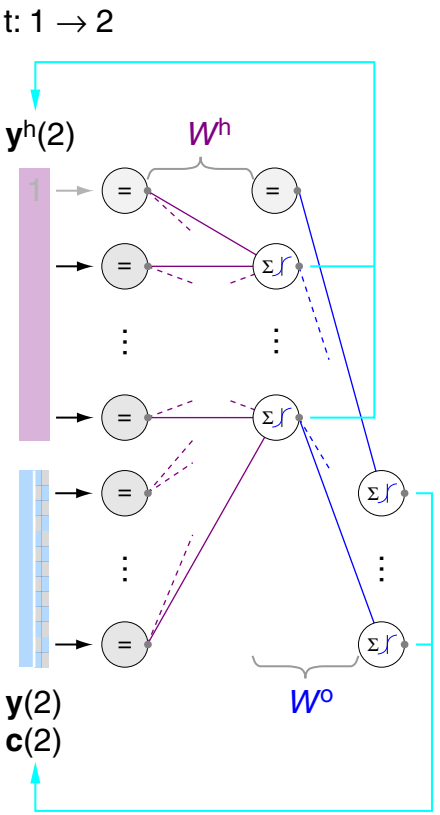


Output generation over t.

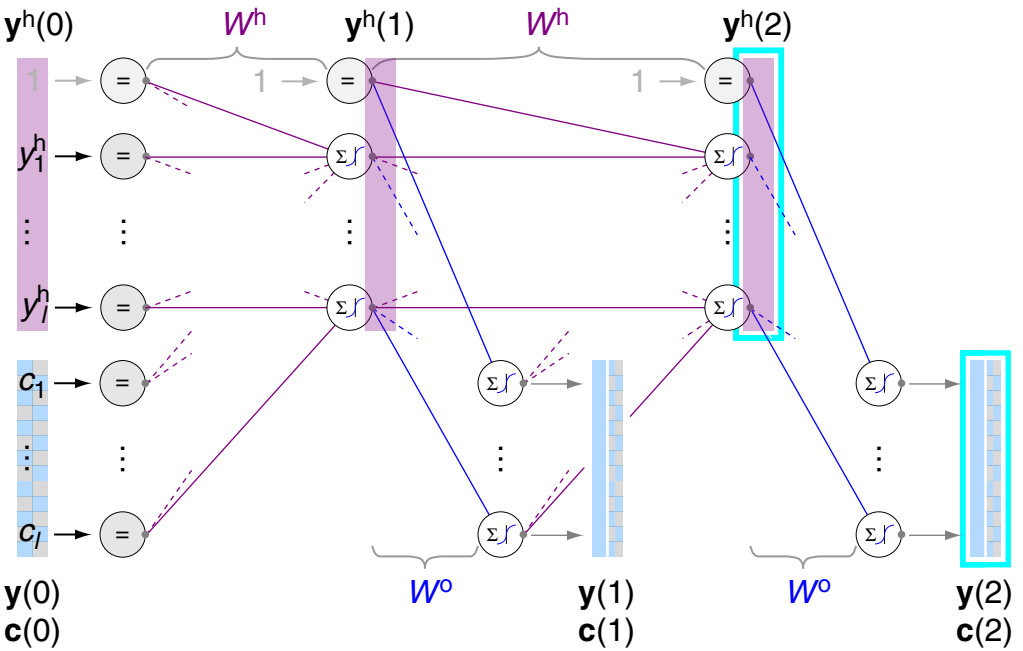
Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]



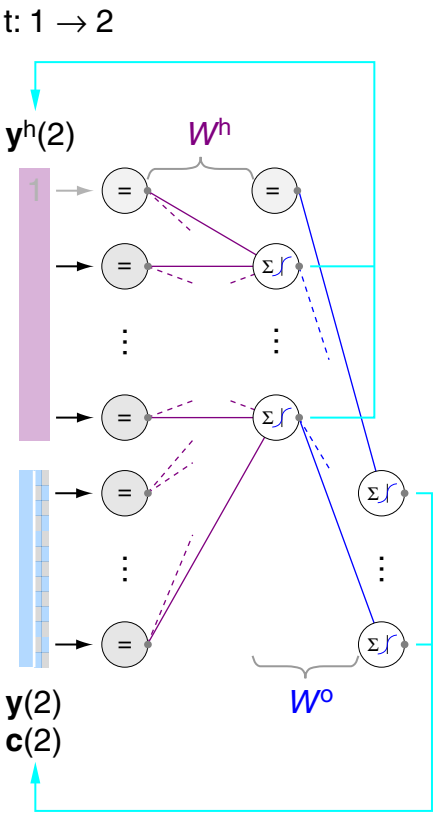
Output generation over t.

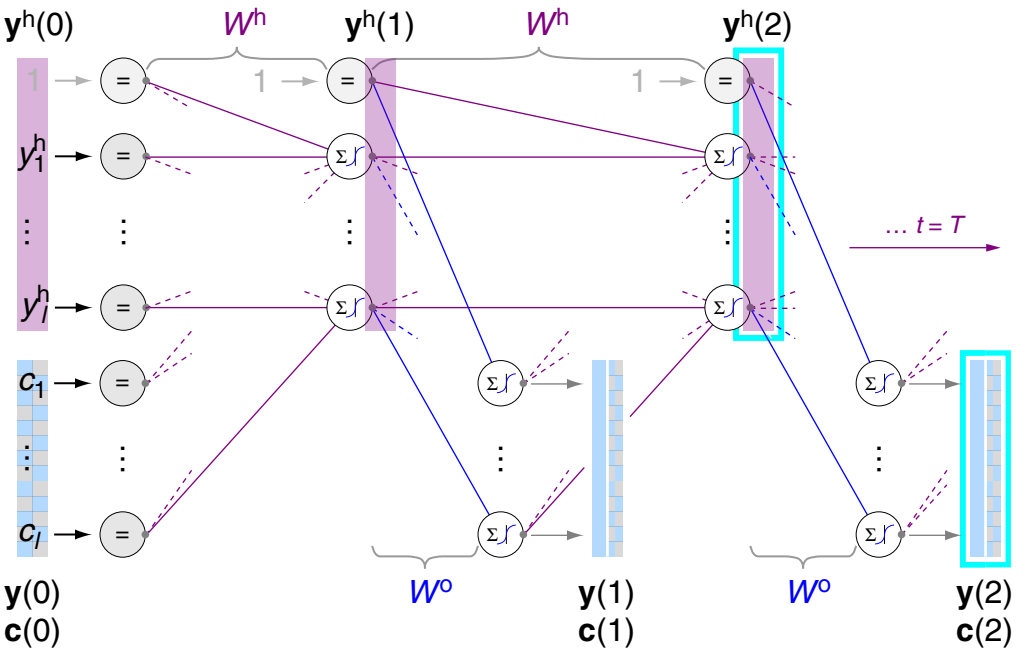


Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Recurrent Output Generation Unfolded [decoding overview]





Output generation over t.

Hidden and output layer at subsequent time steps.

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)\,],\,\mathbf{y}(1)],\,\mathbf{y}(2)],\ldots],\,\mathbf{y}(\tau{-}1)],\quad \mathbf{x} = \begin{pmatrix}0\\ \vdots\\ 1\\ 0\\ \vdots\end{pmatrix} \mathrel{\widehat{=}}$ `word_11`

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}}$ `<start>`, $\quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(4) \mathrel{\widehat{=}}$ `<end>`

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)\,],\ \mathbf{y}(1)\,],\ \mathbf{y}(2)\,],\ldots\,],\ \mathbf{y}(\tau-1)\,],\quad \mathbf{x} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix} \widehat{=}$ `word_11`

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \widehat{=}$ `<start>`, $\mathbf{y}(\tau) \widehat{=} \mathbf{c}(4) \widehat{=}$ `<end>`

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)\,],\ \mathbf{y}(1)\,],\ \mathbf{y}(2)\,],\ldots\,],\ \mathbf{y}(\tau-1)\,],\quad \mathbf{x} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix} \widehat{=}$ `word_11`

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \widehat{=}$ `<start>`, $\mathbf{y}(\tau) \widehat{=} \mathbf{c}(4) \widehat{=}$ `<end>`

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)\,],\ \mathbf{y}(1)\,],\ \mathbf{y}(2)\,],\ldots\,],\ \mathbf{y}(\tau-1)\,],\quad \mathbf{x} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix} \widehat{=}$ `word_11`

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \widehat{=}$ `<start>`, $\mathbf{y}(\tau) \widehat{=} \mathbf{c}(4) \widehat{=}$ `<end>`

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)\,],\ \mathbf{y}(1)\,],\ \mathbf{y}(2)\,],\ldots\,],\ \mathbf{y}(\tau-1)\,],\quad \mathbf{x} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix} \widehat{=}$ `word_11`

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \widehat{=}$ `<start>`, $\mathbf{y}(\tau) \widehat{=} \mathbf{c}(4) \widehat{=}$ `<end>`

RNN Sequence Decoding

Class-to-Sequence: Text Generation

I $\rightarrow$ love my cat.

Cats $\rightarrow$ and dogs lap water.

It $\rightarrow$ is raining cats and dogs.

Vocabulary: (`allowed always and are been cat cats dogs enemies have i is it lap love my natural not raining water <start> <end>`)

Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)\,],\ \mathbf{y}(1)\,],\ \mathbf{y}(2)\,],\ldots\,],\ \mathbf{y}(\tau-1)\,],\quad \mathbf{x} = \begin{pmatrix}0\\ \vdots\\ 1\\ 0\\ \vdots\end{pmatrix} \mathrel{\widehat{=}}$ `word_11`

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}}$ `<start>`, $\quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(4) \mathrel{\widehat{=}}$ `<end>`

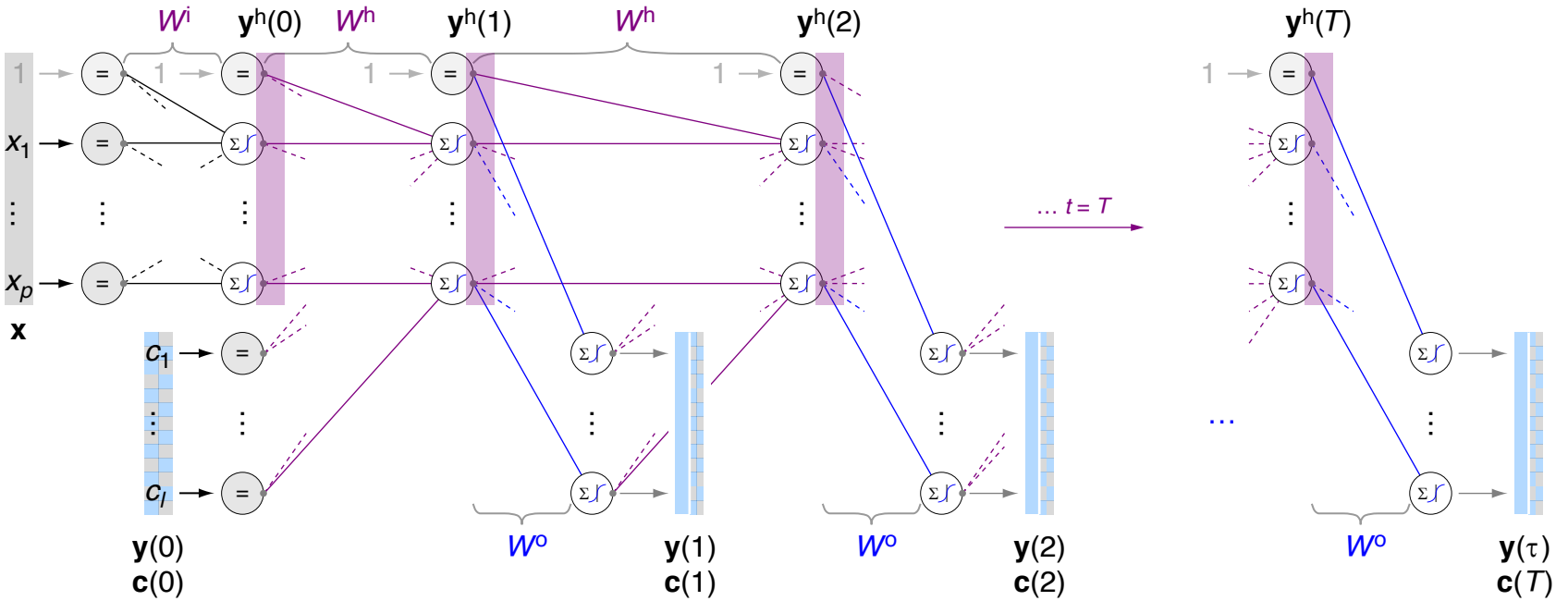
Target: $[\mathbf{c}(1), \ldots, \mathbf{c}(4)] = \left[\begin{pmatrix}0\\ \vdots\\ 1\\ 0\\ \vdots\end{pmatrix}, \begin{pmatrix}0\\ \vdots\\ 1\\ 0\end{pmatrix}, \begin{pmatrix}0\\ 1\\ 0\\ \vdots\end{pmatrix}, \begin{pmatrix}0\\ \vdots\\ 1\end{pmatrix}\right]$

$\mathrel{\widehat{=}}$ [`word_15, word_16, word_6, word_22`]

$\mathrel{\widehat{=}}$ love my cat

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(\tau-1)]$

Hidden:

$\mathbf{y}^{h}(0) = \boldsymbol{\sigma}\left(W^{i}\mathbf{x}\right)$

$\mathbf{y}^{h}(t) = \boldsymbol{\sigma}\left(W^{h}\begin{pmatrix}\mathbf{y}^{h}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, \tau$

Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

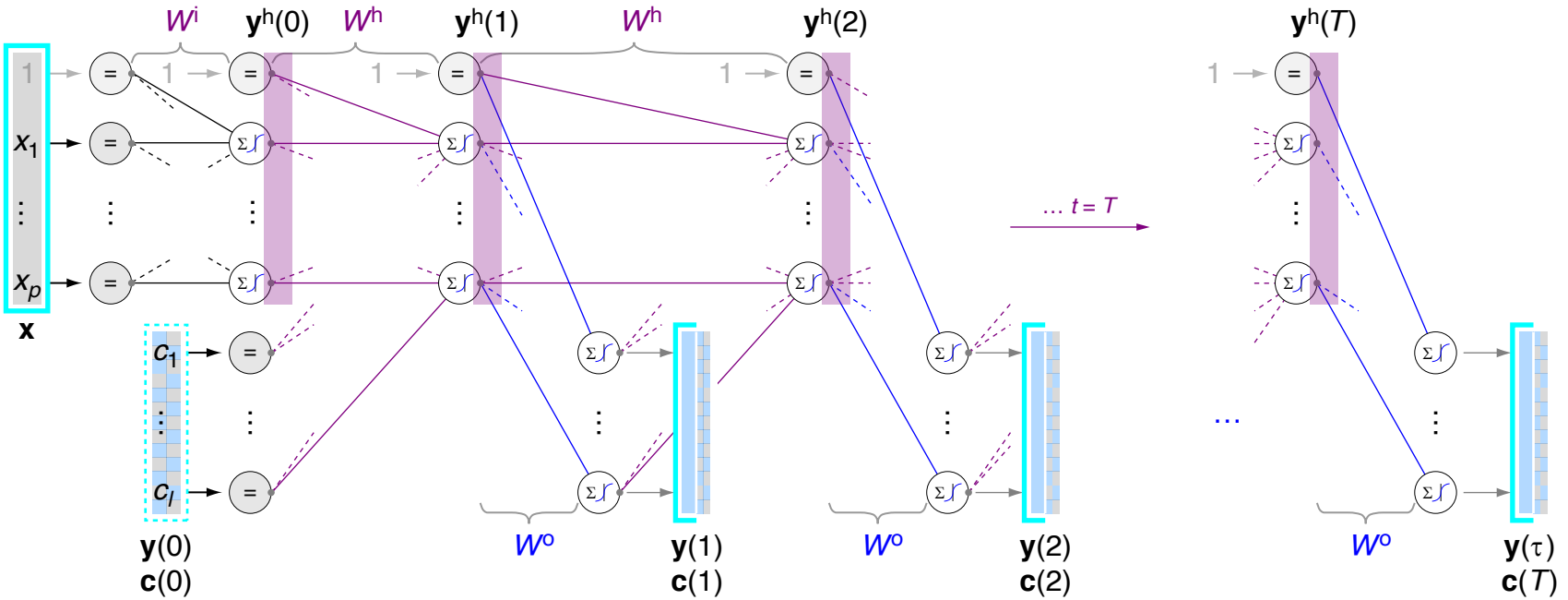
$\mathbf{c}(T) \mathrel{\widehat{=}}$ <end>

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\Delta}\left(W^{o}\mathbf{y}^{h}(t)\right), t = 1, \ldots, \tau$

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(\tau-1)]$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\triangle}\left(W^{\mathrm{o}}\mathbf{y}^{\mathrm{h}}(t)\right), t = 1, \ldots, \tau$

Hidden:

$\mathbf{y}^{\mathrm{h}}(0) = \boldsymbol{\sigma}\left(W^{\mathrm{i}}\mathbf{x}\right)$

$\mathbf{y}^{\mathrm{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathrm{h}}\begin{pmatrix}\mathbf{y}^{\mathrm{h}}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, \tau$

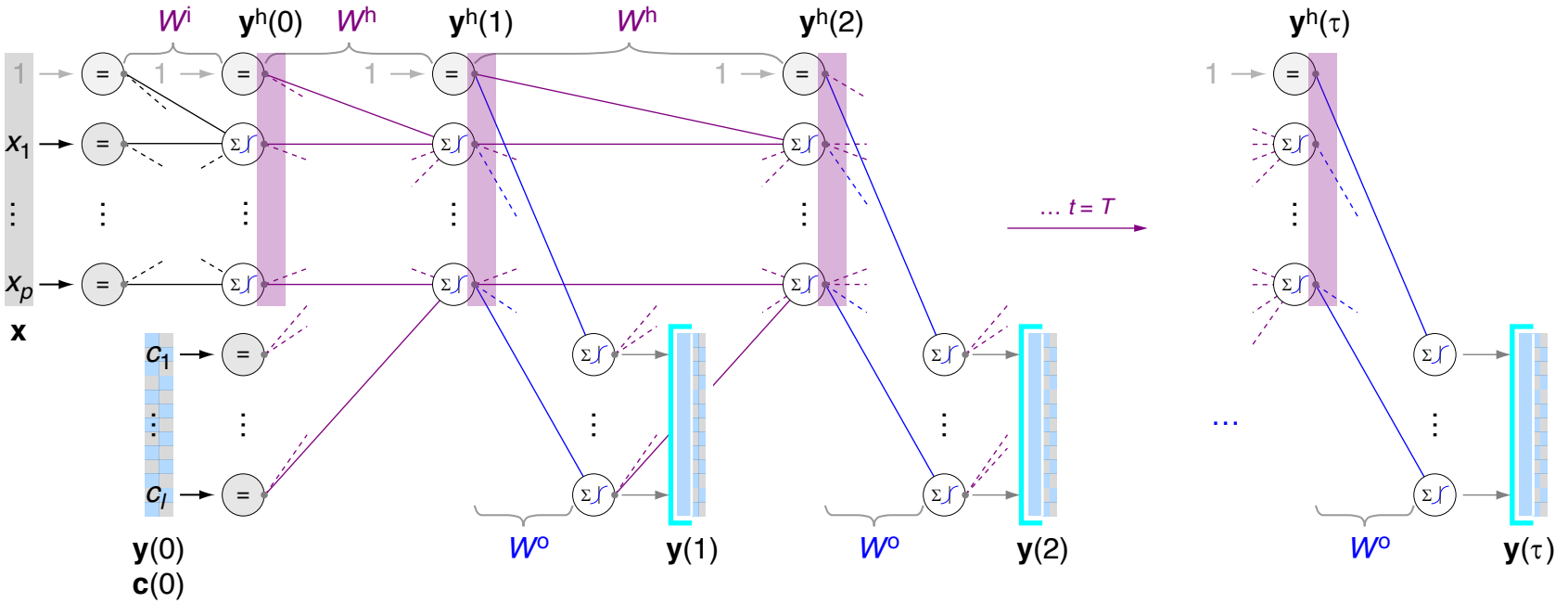
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

$\mathbf{c}(T) \mathrel{\widehat{=}} \texttt{<end>}$

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(\tau-1)]$

Hidden:

$\mathbf{y}^{h}(0) = \boldsymbol{\sigma}\left(W^{i}\,\mathbf{x}\right)$

$\mathbf{y}^{h}(t) = \boldsymbol{\sigma}\left(W^{h}\begin{pmatrix}\mathbf{y}^{h}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, \tau$

Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

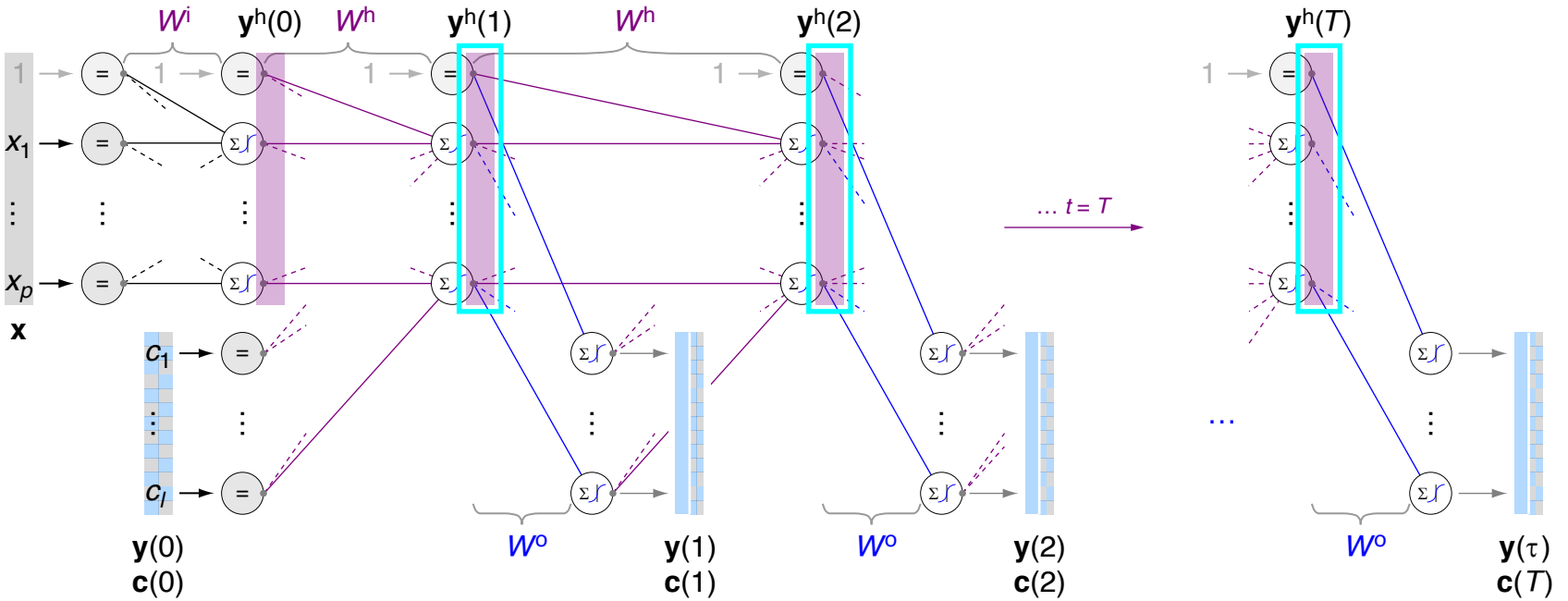
$\mathbf{c}(T) \mathrel{\widehat{=}}$ <end>

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\Delta}\left(W^{o}\,\mathbf{y}^{h}(t)\right), t = 1, \ldots, \tau$

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(\tau-1)]$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\triangle}\left(W^{o}\,\mathbf{y}^{h}(t)\right), t = 1, \ldots, \tau$

Hidden:

$\mathbf{y}^{h}(0) = \boldsymbol{\sigma}\left(W^{i}\,\mathbf{x}\right)$

$\mathbf{y}^{h}(t) = \boldsymbol{\sigma}\left(W^{h}\begin{pmatrix}\mathbf{y}^{h}(t-1)\\ \mathbf{c}(t-1)\end{pmatrix}\right), t = 1, \ldots, T$

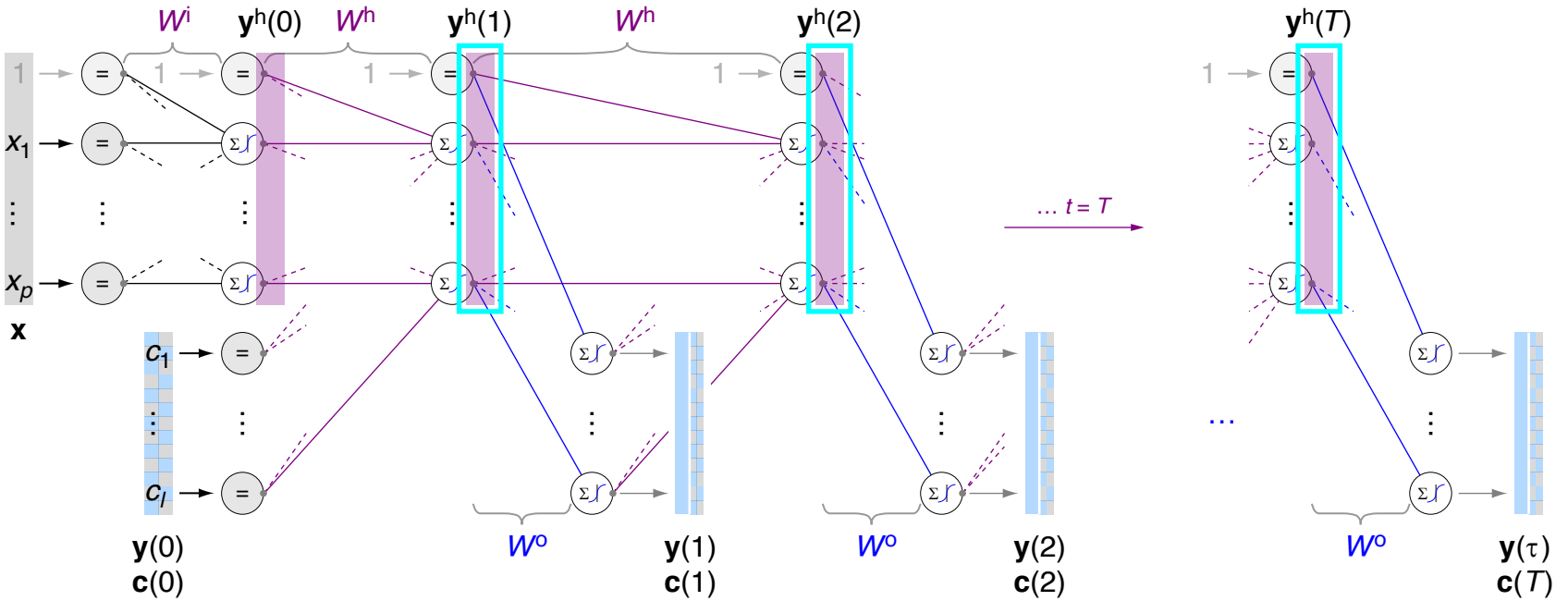
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

$\mathbf{c}(T) \mathrel{\widehat{=}}$ <end>

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(\tau-1)]$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\triangle}\left(W^{\mathsf{o}}\,\mathbf{y}^{\mathsf{h}}(t)\right), t = 1, \ldots, \tau$

Hidden:

$\mathbf{y}^{\mathsf{h}}(0) = \boldsymbol{\sigma}\left(W^{\mathsf{i}}\,\mathbf{x}\right)$

$\mathbf{y}^{\mathsf{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{h}}\begin{pmatrix}\mathbf{y}^{\mathsf{h}}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, \tau$

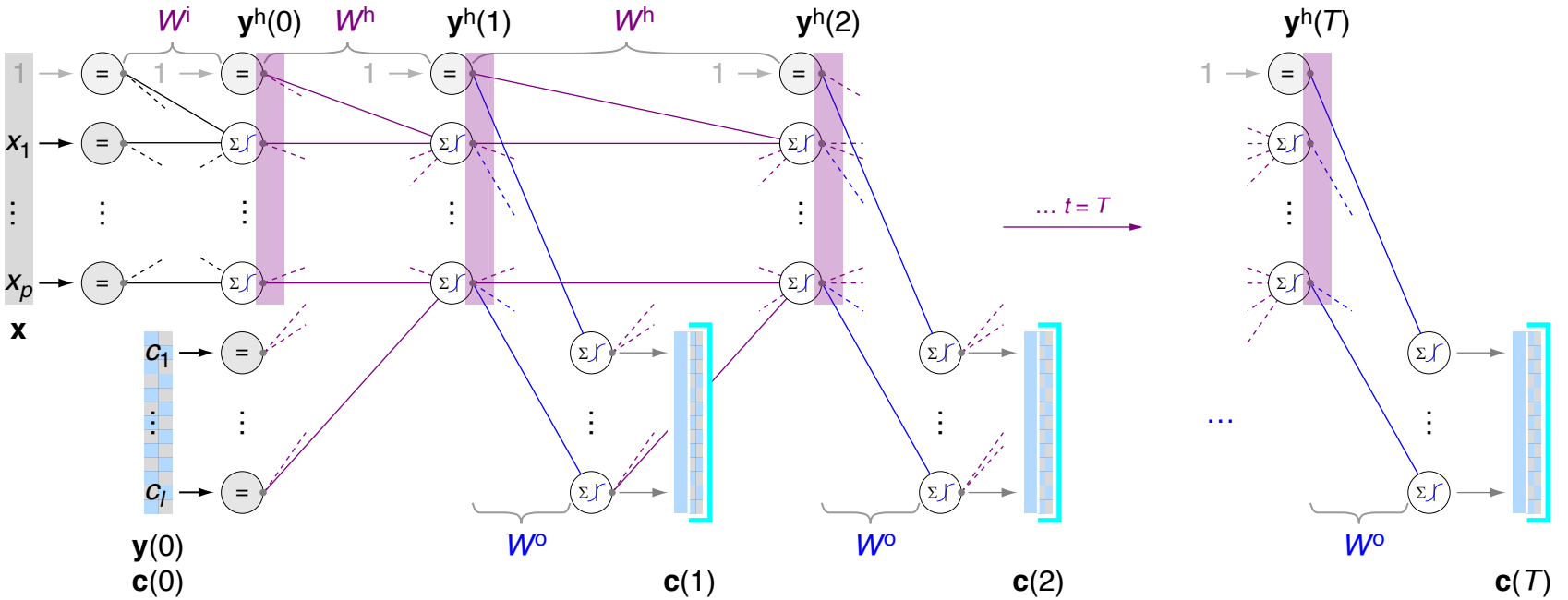
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

$\mathbf{c}(T) \mathrel{\widehat{=}} \texttt{<end>}$

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(\tau-1)]$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\triangle}\left(W^{o}\,\mathbf{y}^{h}(t)\right), t = 1, \ldots, \tau$

Hidden:

$\mathbf{y}^{h}(0) = \boldsymbol{\sigma}\left(W^{i}\,\mathbf{x}\right)$

$\mathbf{y}^{h}(t) = \boldsymbol{\sigma}\left(W^{h}\begin{pmatrix}\mathbf{y}^{h}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, \tau$

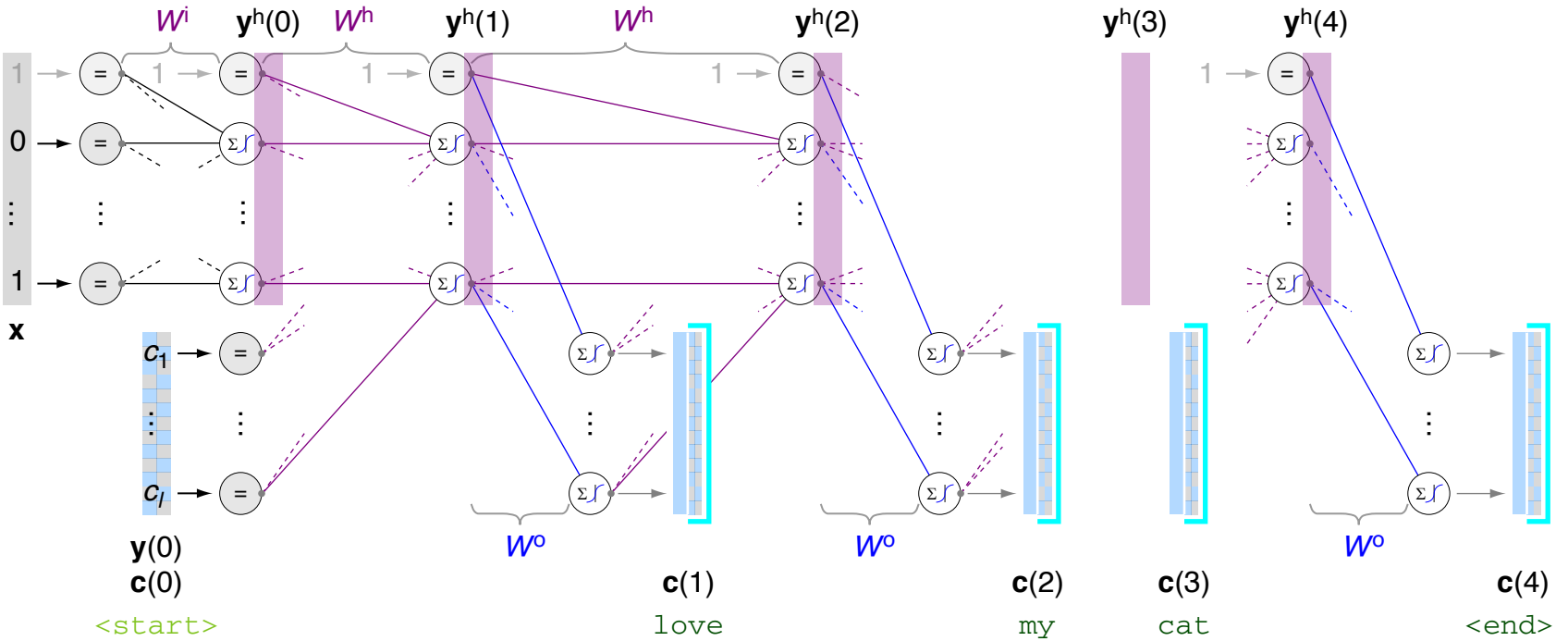
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

$\mathbf{c}(T) \mathrel{\widehat{=}}$ <end>

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(3)]$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\Delta}\left(W^{\mathsf{o}}\,\mathbf{y}^{\mathsf{h}}(t)\right), t = 1, \ldots, 4$

Hidden:

$\mathbf{y}^{\mathsf{h}}(0) = \boldsymbol{\sigma}\left(W^{\mathsf{i}}\,\mathbf{x}\right)$

$\mathbf{y}^{\mathsf{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{h}}\begin{pmatrix}\mathbf{y}^{\mathsf{h}}(t-1)\\ \mathbf{c}(t-1)\end{pmatrix}\right), t = 1, \ldots, 4$

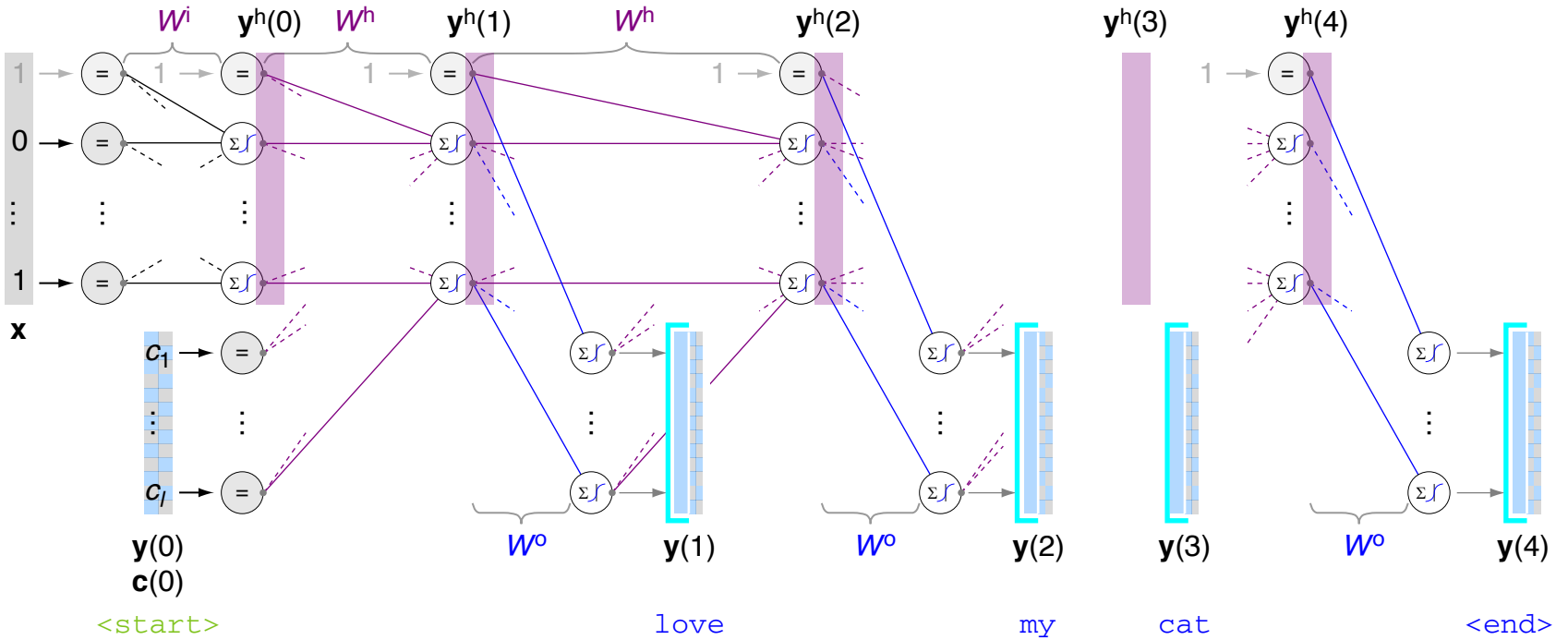
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(4)]$

$\mathbf{c}(4) \mathrel{\widehat{=}}$ <end>

RNN Sequence Decoding

Class-to-Sequence Mapping with RNNs



Input:

$\mathbf{x}, [\mathbf{y}(1), \ldots, \mathbf{y}(3)]$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_{\Delta}\left(W^{\mathsf{o}}\,\mathbf{y}^{\mathsf{h}}(t)\right), t = 1, \ldots, 4$

Hidden:

$\mathbf{y}^{\mathsf{h}}(0) = \boldsymbol{\sigma}\left(W^{\mathsf{i}}\,\mathbf{x}\right)$

$\mathbf{y}^{\mathsf{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{h}}\begin{pmatrix}\mathbf{y}^{\mathsf{h}}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, 4$

Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(4)]$

$\mathbf{c}(4) \mathrel{\widehat{=}}$ <end>

Remarks:

- We denote $\mathbf{y}(0)$ not as input since it is predefined and does not contain any "actual knowledge". In particular, $\mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}}$ `<start>`.

- At training time the calculation of $\mathbf{y}^h(t)$ usually considers the ground truth $\mathbf{c}(t-1)$:

$$\mathbf{y}^{\mathsf{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{h}}\begin{pmatrix}\mathbf{y}^{\mathsf{h}}(t-1)\\ \mathbf{c}(t-1)\end{pmatrix}\right)$$

- At test time ("production mode") the calculation of $\mathbf{y}^h(t)$ has to consider the output $\mathbf{y}(t-1)$:

$$\mathbf{y}^{\mathsf{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{h}}\begin{pmatrix}\mathbf{y}^{\mathsf{h}}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right)$$

RNN Sequence Decoding

The IGD Algorithm for Class-to-Sequence Tasks [IGD$_{seq2c}$]

Algorithm:	IGD$_{c2seq}$	Incremental Gradient Descent for RNNs at class2seq tasks.
Input:	D	Multiset of examples $(\mathbf{x}, [\mathbf{c}(1), \ldots, \mathbf{c}(T)])$ with $\mathbf{x} \in \{0,1\}^p$, $\mathbf{c}(t) \in \mathbf{R}^k$.
	η	Learning rate, a small positive constant.
Output:	$W^{\mathsf{i}}, W^{\mathsf{h}}, W^{\mathsf{o}}$	Weights matrices. (= hypothesis)

1. *initialize_random_weights*$(W^{\mathsf{i}}, W^{\mathsf{h}}, W^{\mathsf{o}})$, $t_{\text{epoch}} = 0$
2. **REPEAT**
3. $t_{\text{epoch}} = t_{\text{epoch}} + 1$
4. FOREACH $(\mathbf{x}, [\mathbf{c}(1), \ldots, \mathbf{c}(T)]) \in D$ DO
5. Model function evaluation.
6. Calculation of residual vectors.
7a. Calculation of derivative of the loss.
7b.
8. Parameter update $\widehat{=}$ one gradient step down.
9. ENDDO
10. **UNTIL**(*convergence*$(D, \mathbf{y}(\cdot), t_{\text{epoch}})$)
11. *return*$(W^{\mathsf{i}}, W^{\mathsf{h}}, W^{\mathsf{o}})$

RNN Sequence Decoding

The IGD Algorithm for Class-to-Sequence Tasks [IGD$_{seq2c}$]

Algorithm:	IGD$_{c2seq}$	Incremental Gradient Descent for RNNs at class2seq tasks.
Input:	D	Multiset of examples $(\mathbf{x}, [\mathbf{c}(1), \ldots, \mathbf{c}(T)])$ with $\mathbf{x} \in \{0,1\}^p$, $\mathbf{c}(t) \in \mathbf{R}^k$.
	η	Learning rate, a small positive constant.
Output:	$W^{\mathsf{i}}, W^{\mathsf{h}}, W^{\mathsf{o}}$	Weights matrices. (= hypothesis)

1. *initialize_random_weights*$(W^{\mathsf{i}}, W^{\mathsf{h}}, W^{\mathsf{o}})$, $t_{\text{epoch}} = 0$
2. **REPEAT**
3. $\quad t_{\text{epoch}} = t_{\text{epoch}} + 1$
4. $\quad$ FOREACH $(\mathbf{x}, [\mathbf{c}(1), \ldots, \mathbf{c}(T)]) \in D$ DO
5. $\quad\quad \mathbf{y}^{\mathsf{h}}(0) = \boldsymbol{\sigma}\left(W^{\mathsf{i}}\,\mathbf{x}\right)$
$\quad\quad$ FOR $t = 1$ TO T DO // forward propagation
$$\quad\quad\quad \mathbf{y}^{\mathsf{h}}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{h}}\begin{pmatrix}\mathbf{y}^{\mathsf{h}}(t-1)\\ \mathbf{c}(t-1)\end{pmatrix}\right),\quad \mathbf{y}(t) = \boldsymbol{\sigma}_{\vartriangle}\left(W^{\mathsf{o}}\,\mathbf{y}^{\mathsf{h}}(t)\right)$$
$\quad\quad$ ENDDO
6. $\quad\quad [\boldsymbol{\delta}(1), \ldots, \boldsymbol{\delta}(T)] = [\mathbf{c}(1), \ldots, \mathbf{c}(T)] \ominus [\mathbf{y}(1), \ldots, \mathbf{y}(\tau)]$ // consider that T may $\neq \tau$
7a. $\quad\quad \ell(\mathbf{w}) = \sum_t l(\boldsymbol{\delta}(t)) + \frac{\lambda}{n} R(\mathbf{w}),\quad \nabla\ell(\mathbf{w}) = \text{autodiff}(\ell(), \mathbf{w})$ // backprop. (7a+7b)
7b. $\quad\quad {}_{\vartriangle}W^{\mathsf{i}} = \eta \cdot \nabla^{\mathsf{i}}\ell(\mathbf{w}),\quad {}_{\vartriangle}W^{\mathsf{h}} = \eta \cdot \nabla^{\mathsf{h}}\ell(\mathbf{w}),\quad {}_{\vartriangle}W^{\mathsf{o}} = \eta \cdot \nabla^{\mathsf{o}}\ell(\mathbf{w})$
8. $\quad\quad W^{\mathsf{i}} = W^{\mathsf{i}} + {}_{\vartriangle}W^{\mathsf{i}},\quad W^{\mathsf{h}} = W^{\mathsf{h}} + {}_{\vartriangle}W^{\mathsf{h}},\quad W^{\mathsf{o}} = W^{\mathsf{o}} + {}_{\vartriangle}W^{\mathsf{o}}$
9. $\quad$ ENDDO
10. **UNTIL**(*convergence*$(D, \mathbf{y}(\,\cdot\,), t_{\text{epoch}})$)
11. *return*$(W^{\mathsf{i}}, W^{\mathsf{h}}, W^{\mathsf{o}})$

Remarks:

- We use the operator »$\ominus$« to compare the two sequences $[\mathbf{c}(t)]$ and $[\mathbf{y}(t)]$ of possibly different length. Note that the concrete semantics of »$\ominus$« is left open here.