Chapter ML:IX (continued)

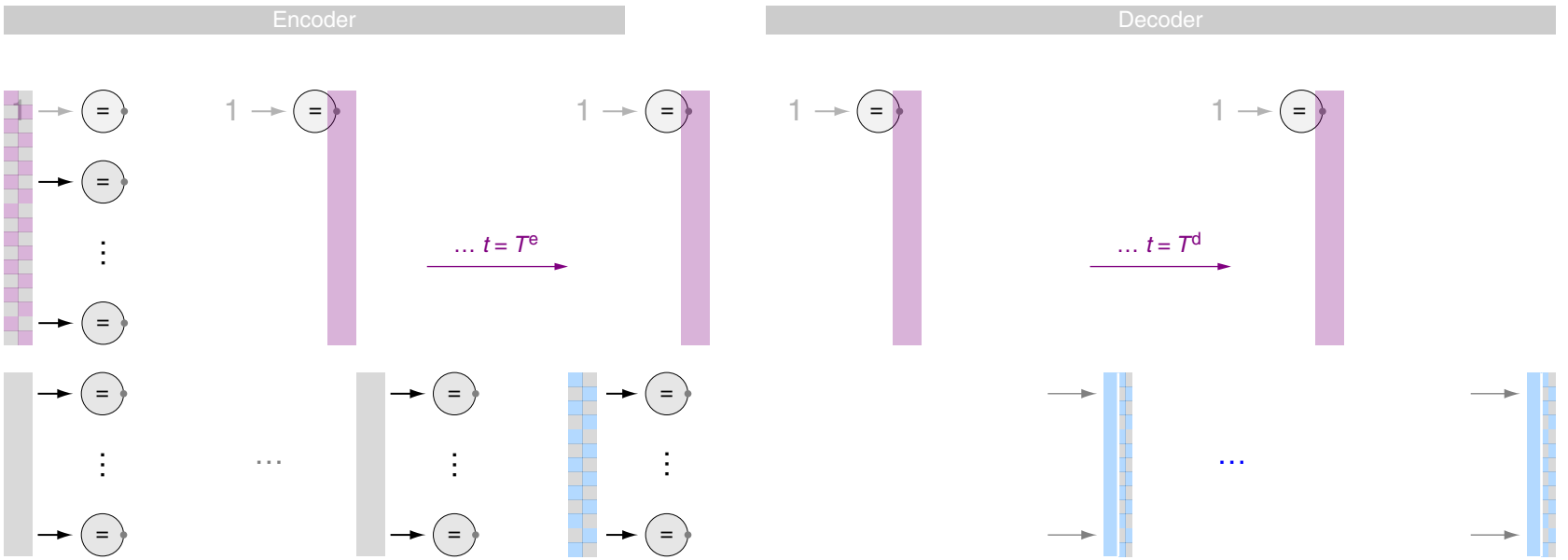
IX. Deep Learning

- Elements of Deep Learning
- Convolutional Neural Networks
- Autoencoder Networks
- Recurrent Neural Networks
- Long-Term Dependencies
- RNNs for Machine Translation
- Attention Mechanism
- Self Attention and Transformers
- Transformer Language Models

Recurrent Neural Networks

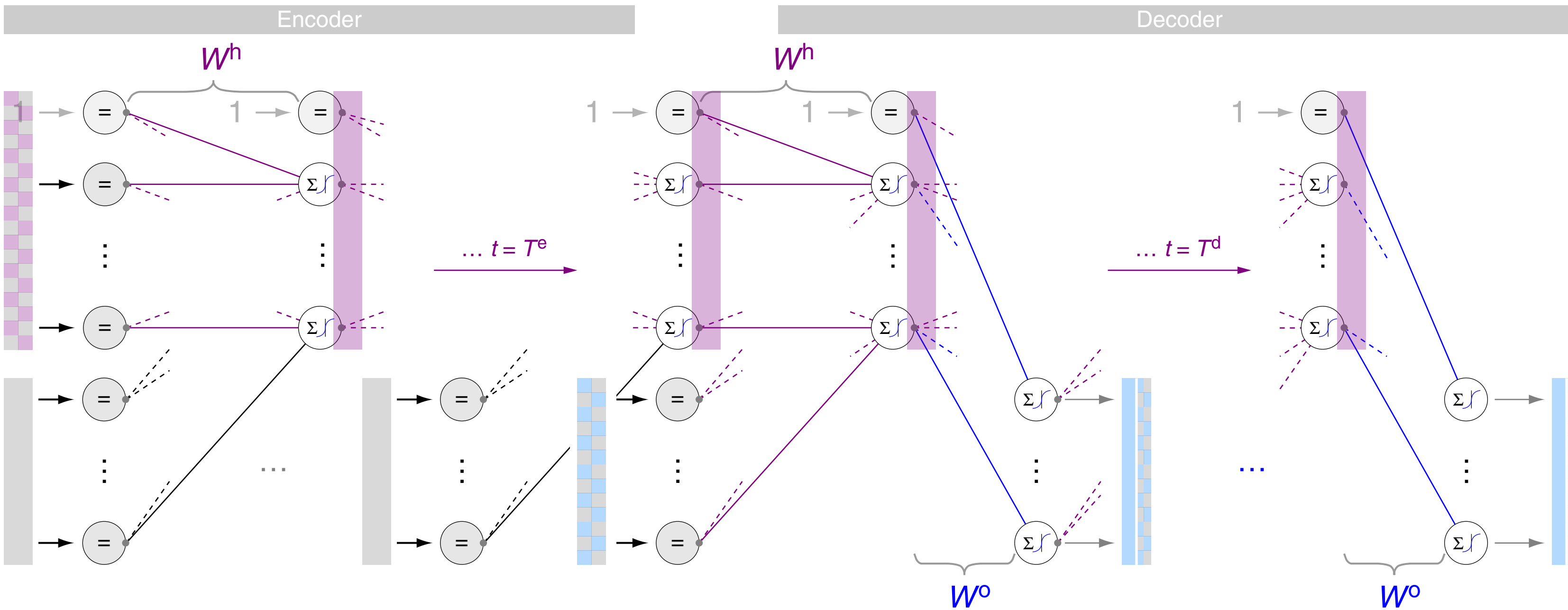
Notation I (color scheme)

[notation: color, graph, language]



Recurrent Neural Networks

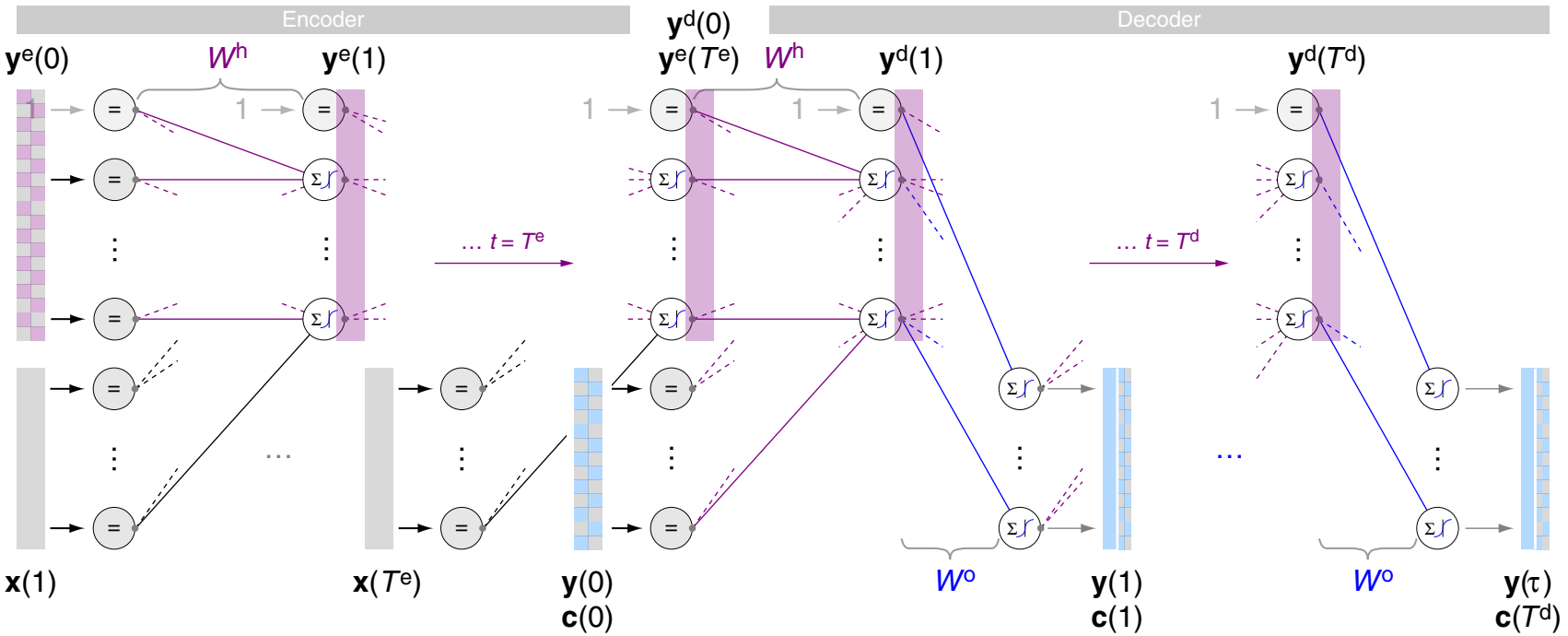
Notation I (color scheme)   [notation: color, graph, language]



Recurrent Neural Networks

Notation I (color scheme)

[notation: color, graph, language]



$\mathbf{x}(t)$ input
`in_word`

$\mathbf{y}^{h}(t)$ hidden
$\mathbf{y}^{e}(t)$ hidden encoder
$\mathbf{y}^{d}(t)$ hidden decoder

predefined hidden

$\mathbf{y}^{a}(t)$ attention

$\mathbf{y}(t)$ output
`out_word` (test)

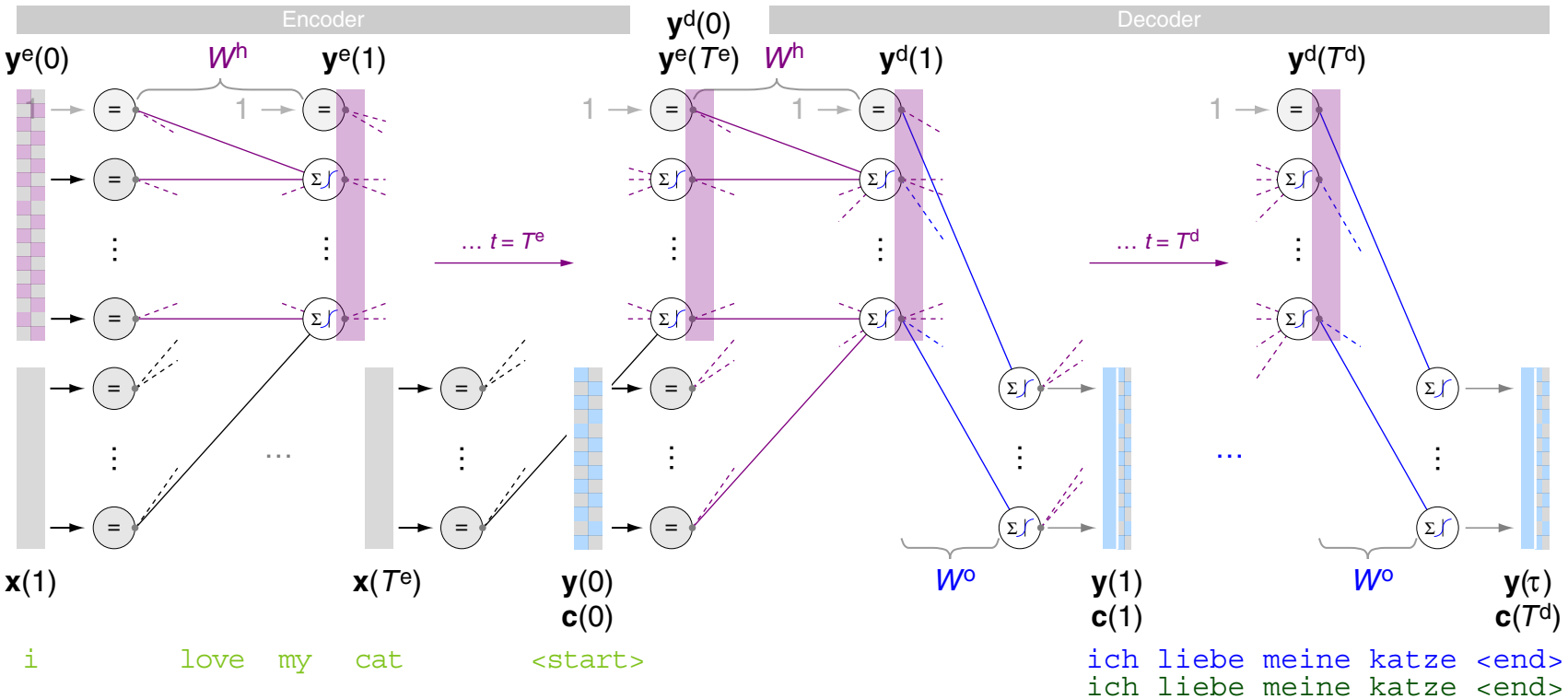
$\mathbf{c}(t)$ target
`out_word` (training)

$\mathbf{y}(t)$ output or $\mathbf{c}(t)$ target
(depends on training or test phase)

Recurrent Neural Networks

Notation I (color scheme)

[notation: color, graph, language]



$\mathbf{x}(t)$ input
in_word

$\mathbf{y}^h(t)$ hidden
$\mathbf{y}^e(t)$ hidden encoder
$\mathbf{y}^d(t)$ hidden decoder

predefined hidden

$\mathbf{y}^a(t)$ attention

$\mathbf{y}(t)$ output
out_word (test)

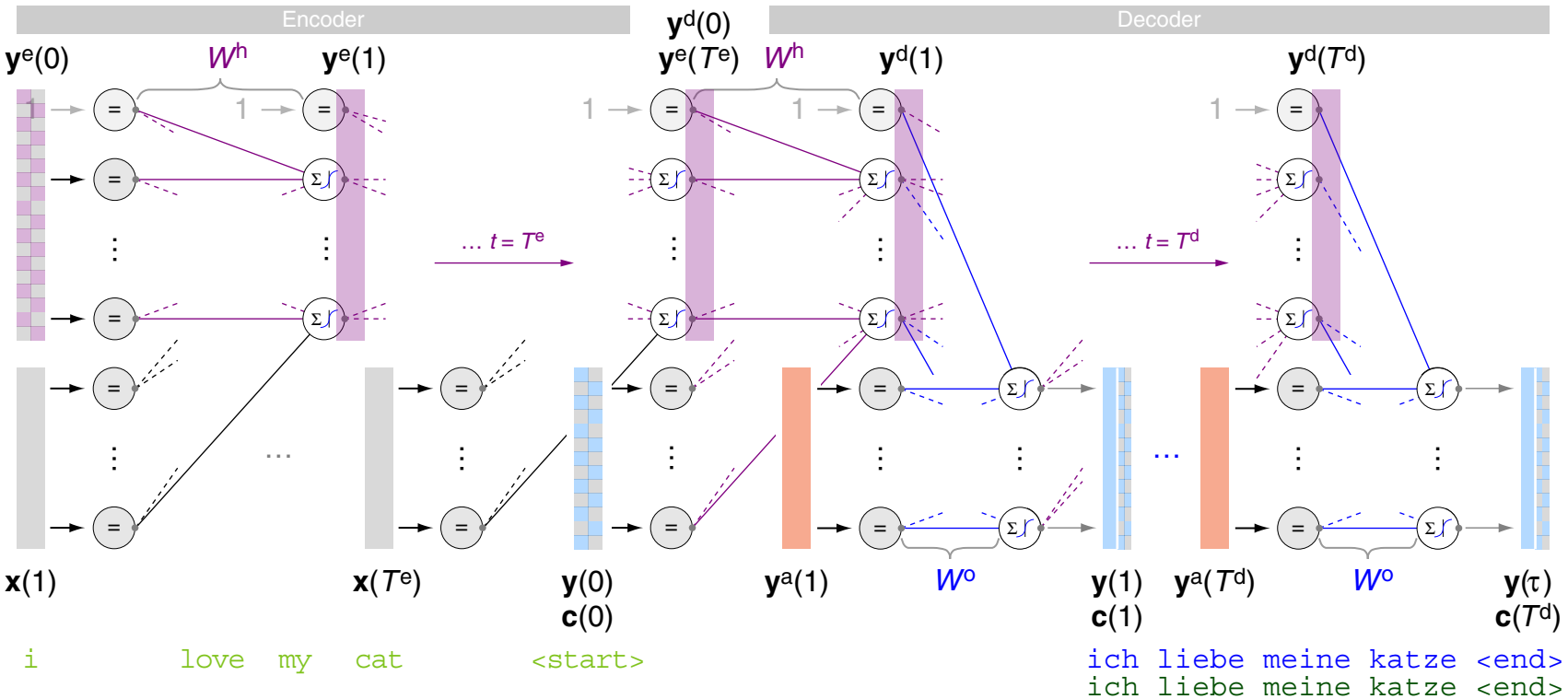
$\mathbf{c}(t)$ target
out_word (training)

$\mathbf{y}(t)$ output or $\mathbf{c}(t)$ target
(depends on training or test phase)

Recurrent Neural Networks

Notation I (color scheme)

[notation: color, graph, language]



$\mathbf{x}(t)$ input
in_word

$\mathbf{y}^h(t)$ hidden
$\mathbf{y}^e(t)$ hidden encoder
$\mathbf{y}^d(t)$ hidden decoder

predefined hidden

$\mathbf{y}^a(t)$ attention

$\mathbf{y}(t)$ output
out_word (test)

$\mathbf{c}(t)$ target
out_word (training)

$\mathbf{y}(t)$ output or $\mathbf{c}(t)$ target
(depends on training or test phase)

Remarks:

- A hidden vector is the result of an intermediate computation in a multilayer network. If sequences are processed, hidden vectors may be distinguished as hidden *encoder* vectors (which consider the input at a certain time step) and hidden *decoder* vectors (which generate the output at a certain time step).
- A predefined hidden vector is used to for initialization purposes for the first hidden layer in a sequence-processing multilayer network. Typically, it is a constant vector of zeroes.
- Attention vectors combine the information in hidden vectors in a way that is specific to a certain time step. Keyword: vanishing gradient problem
- A target vector (or target vector sequence) encodes the desired output.
  Keywords: supervised learning, ground truth

Recurrent Neural Networks

Types of Learning Tasks [Recap]

(S1) sequence $\rightarrow$ class

sentence $\rightarrow \{\oplus, \ominus\}$

`i love my cat` $\rightarrow \oplus$

(S2) class $\rightarrow$ sequence

$\{\oplus, \ominus\} \rightarrow$ sentence

$\oplus \rightarrow$ `i love my cat`

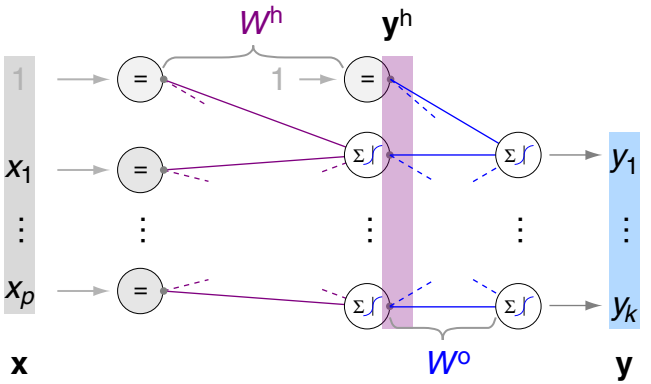
(S3) sequence $\rightarrow$ sequence

English sentence $\rightarrow$ German sentence

`i love my cat` $\rightarrow$ `ich liebe meine katze`

Recurrent Neural Networks

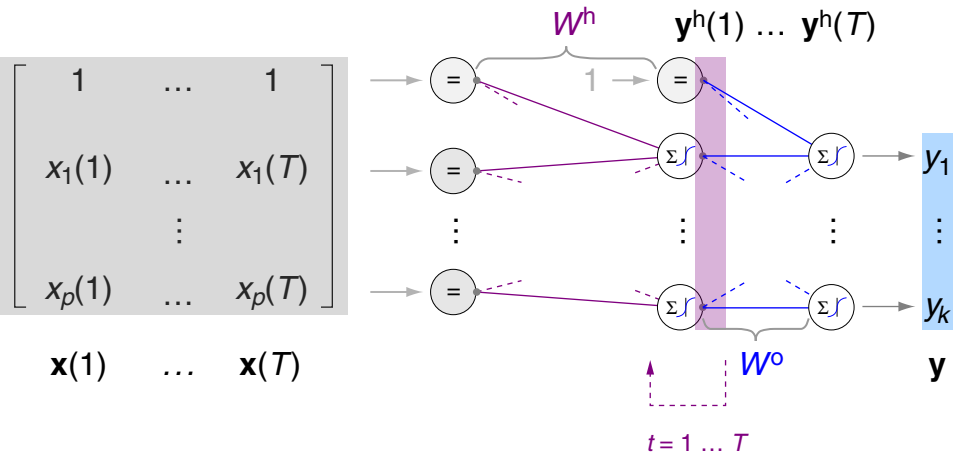
RNN Sequence Encoding



- 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 $\mathbf{y}(\mathbf{x})$.

Recurrent Neural Networks

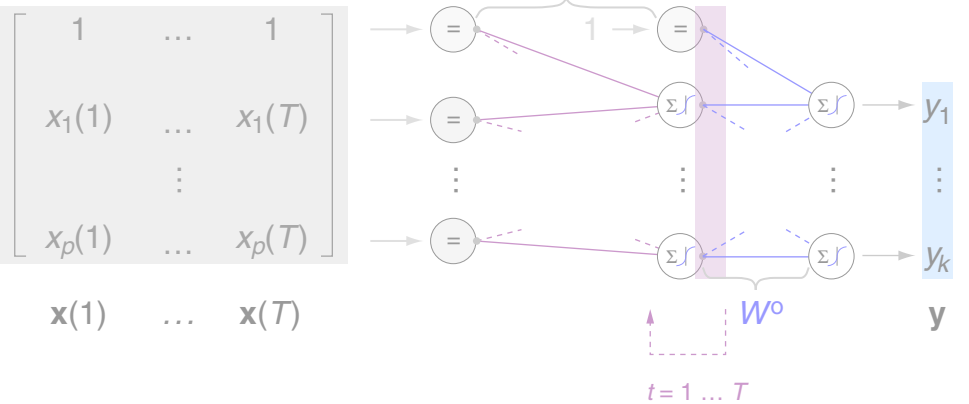
RNN Sequence Encoding (continued)



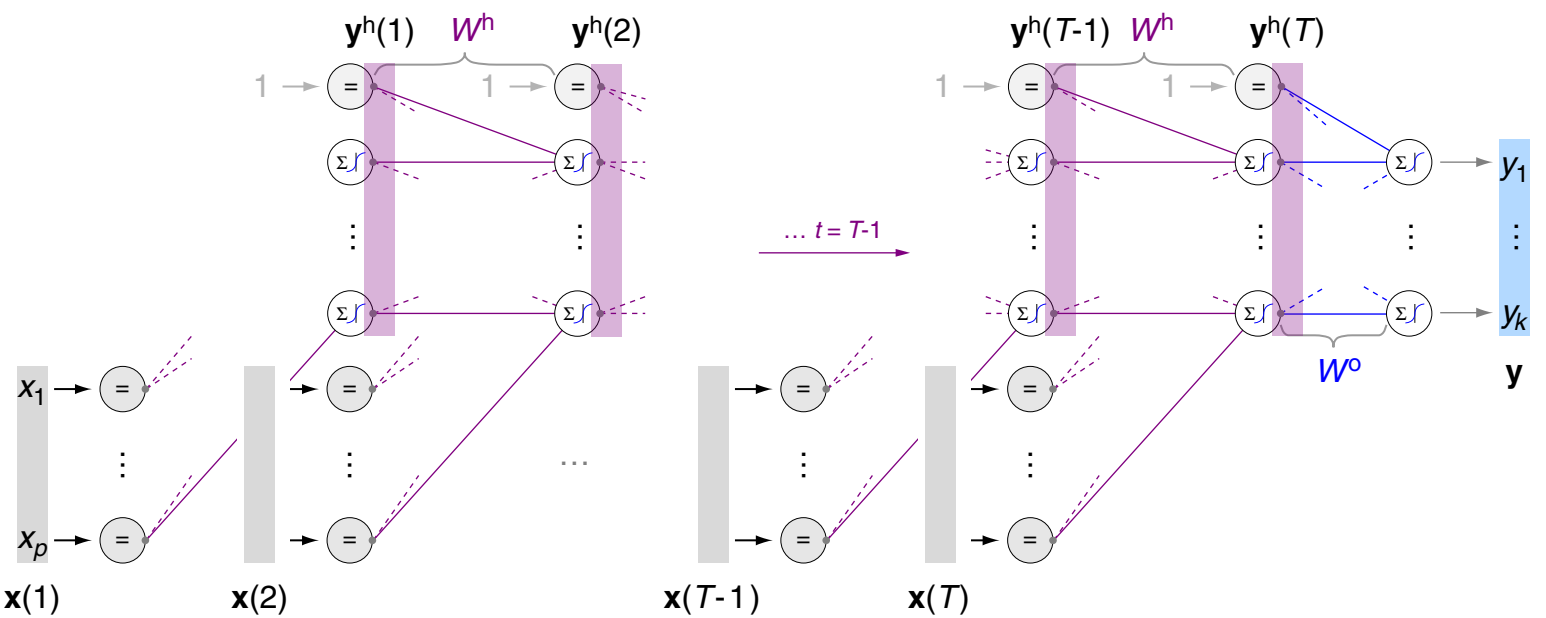
- Sequence of p-dimensional input vectors $[\mathbf{x}(1), \ldots, \mathbf{x}(T)]$.
- One hidden layer that is recurrently updated.
- One k-dimensional output vector $\mathbf{y}([\mathbf{x}(1), \ldots, \mathbf{x}(T)])$ or $\mathbf{y}$.

Recurrent Neural Networks

RNN Sequence Encoding (continued)

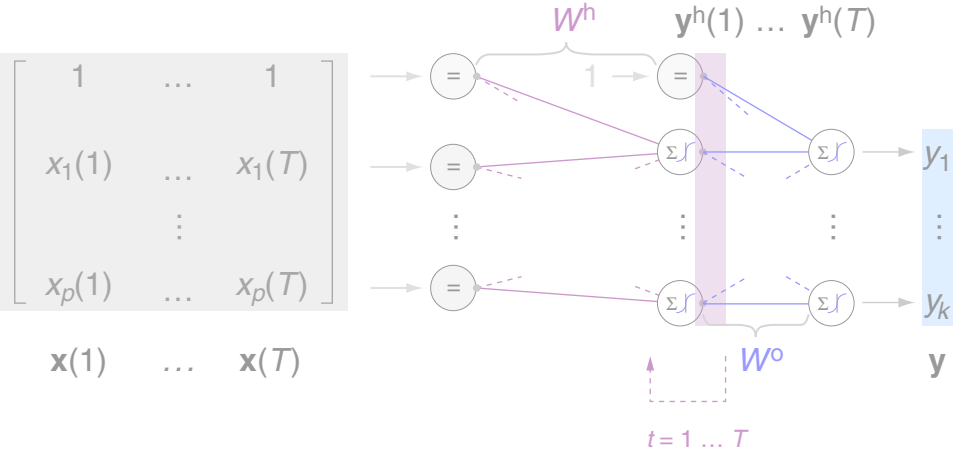


Unfold hidden layer

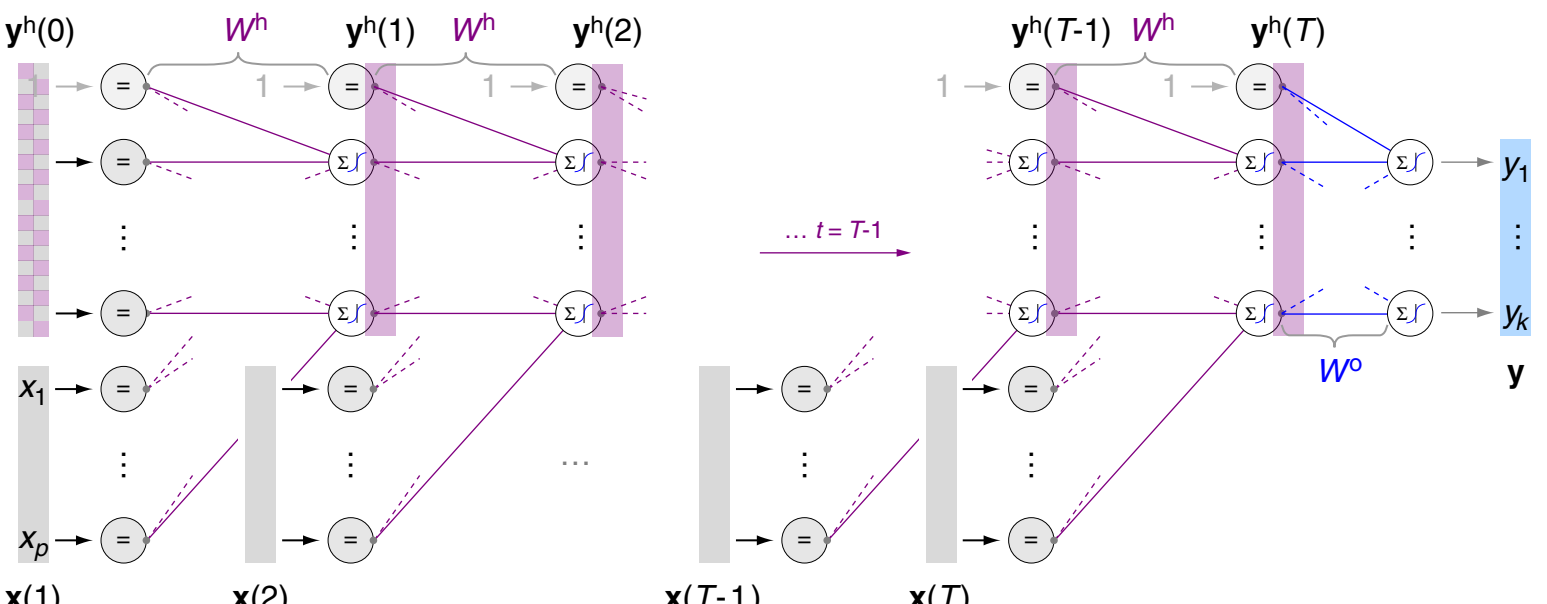


Recurrent Neural Networks

RNN Sequence Encoding (continued)



Unfold hidden layer



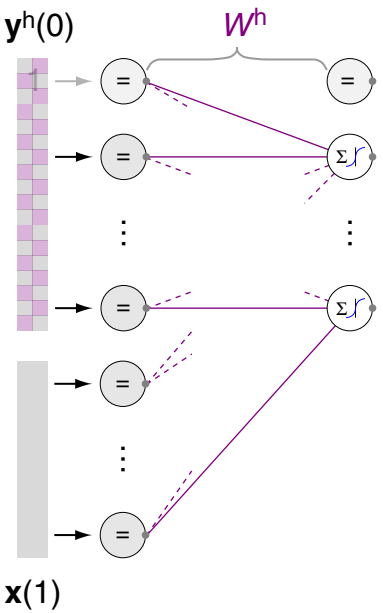
Remarks:

- An input sequence is written in brackets, $[\mathbf{x}(1), \ldots, \mathbf{x}(T)]$, where $\mathbf{x}(t), t = 1, \ldots, T$, denotes the input vector at time step t.
- The words in the input sequence are usually one-hot-encoded, i.e., by a p-dimensional input vector with a "1" whose position indicates the word, and zeros elsewhere.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1

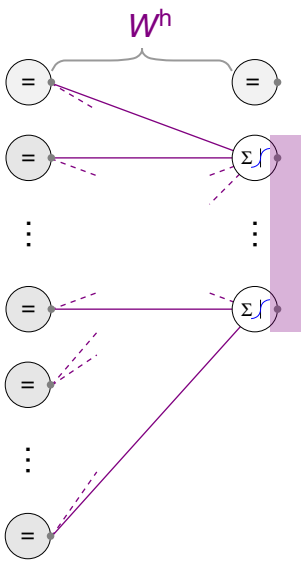


Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1

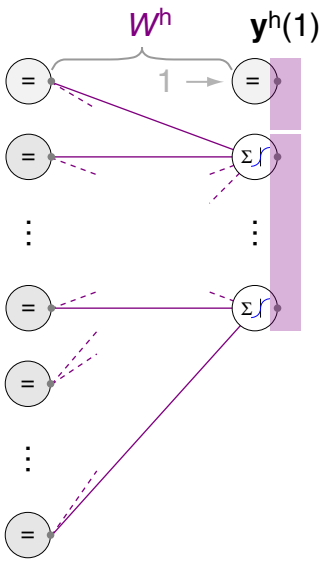


Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1

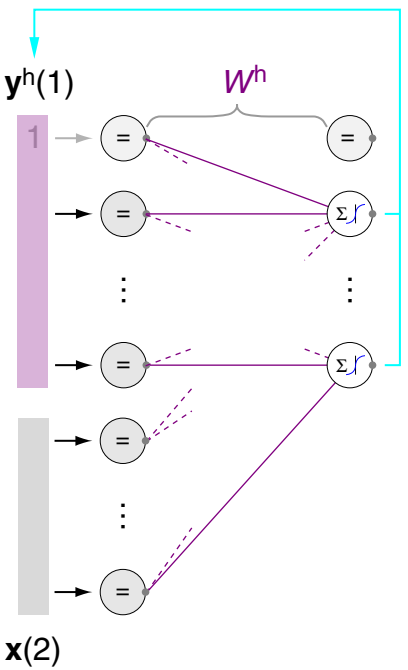


Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1

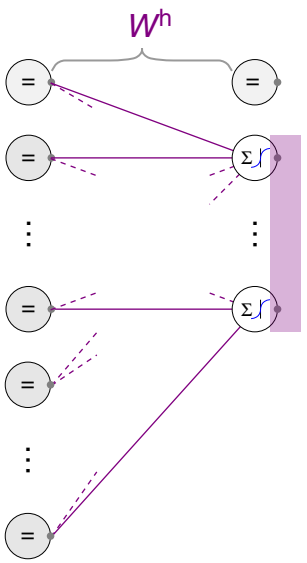


Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 1 → 2

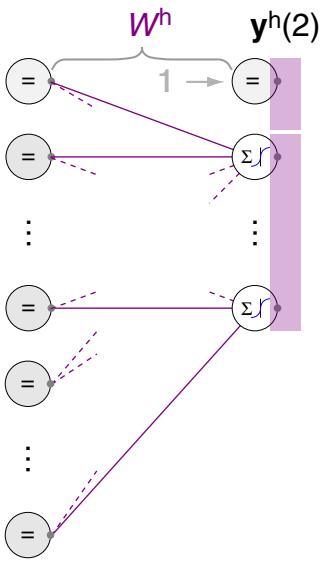


Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 1 → 2

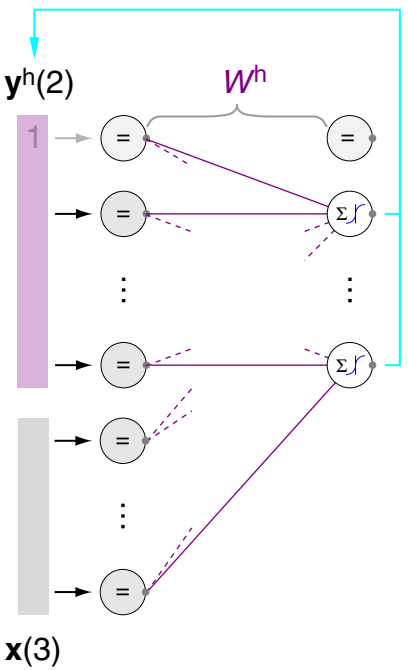


Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 1 → 2

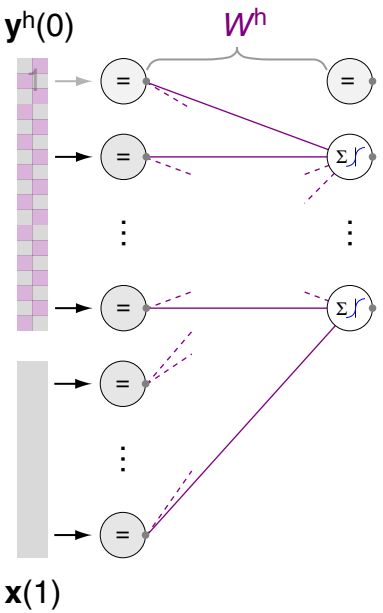


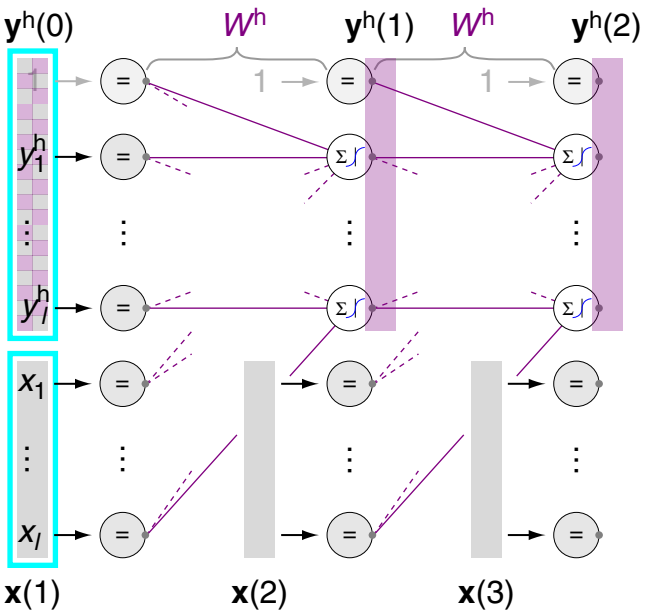
Input encoding over t.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1





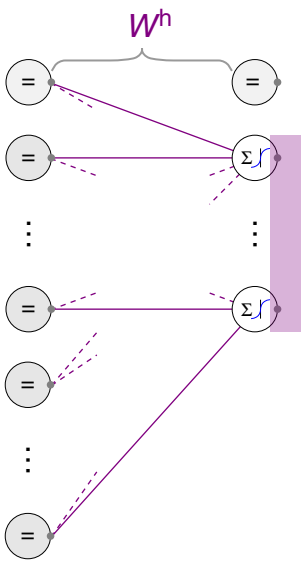
Input encoding over t.

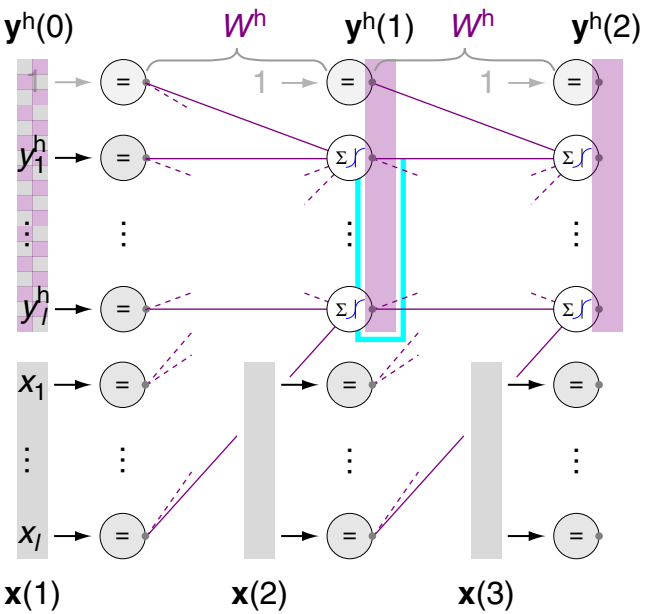
The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1





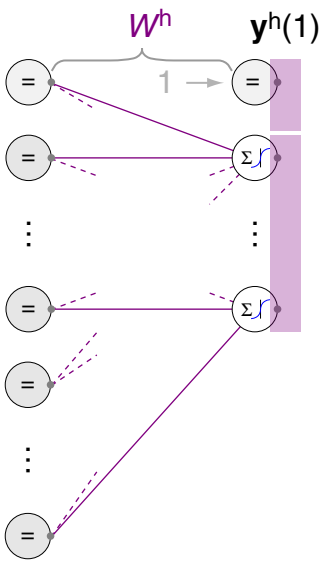
Input encoding over t.

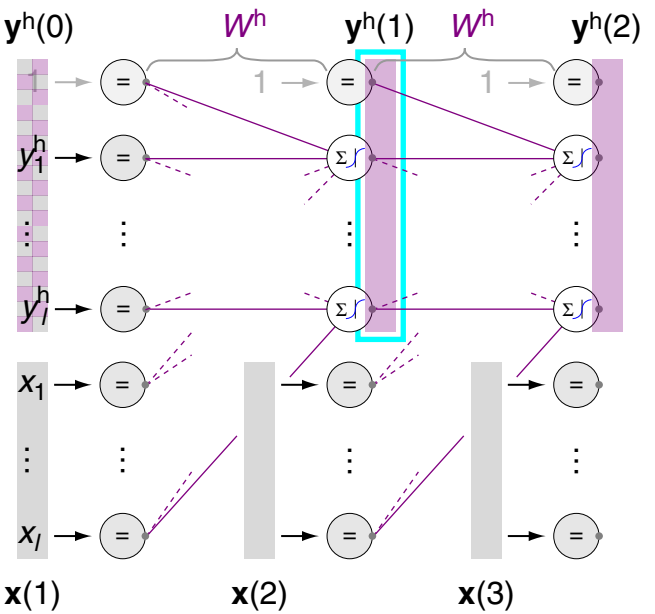
The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1





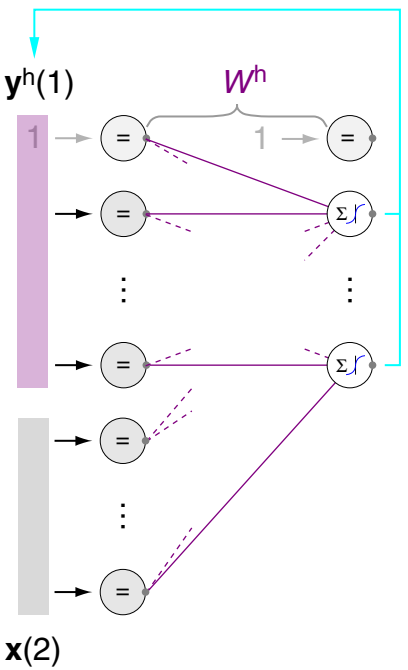
Input encoding over t.

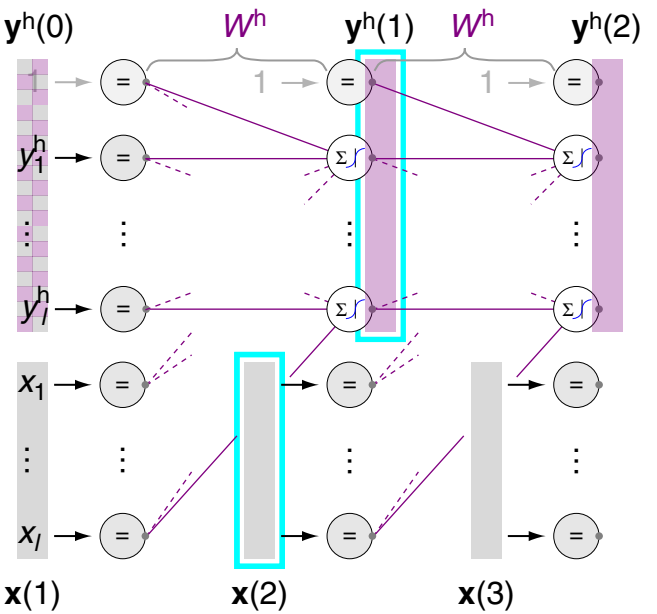
The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 0 → 1





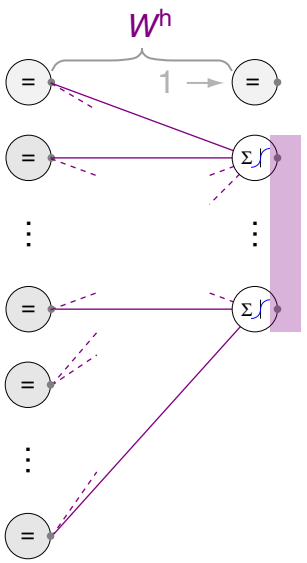
Input encoding over t.

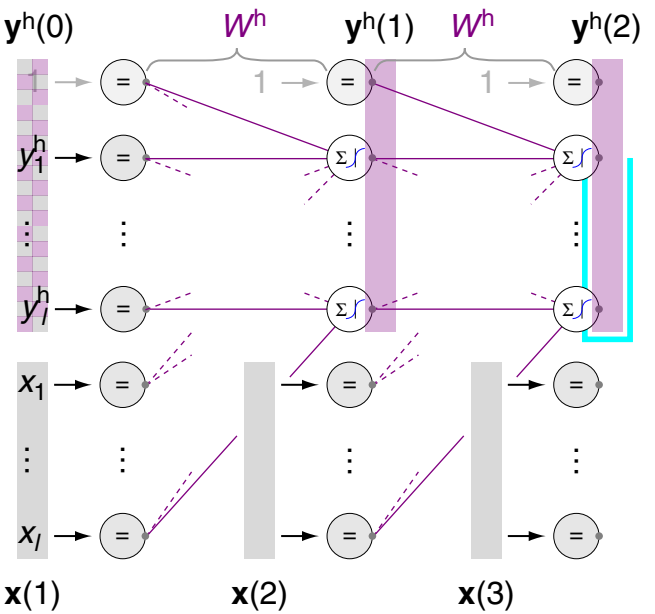
The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 1 → 2





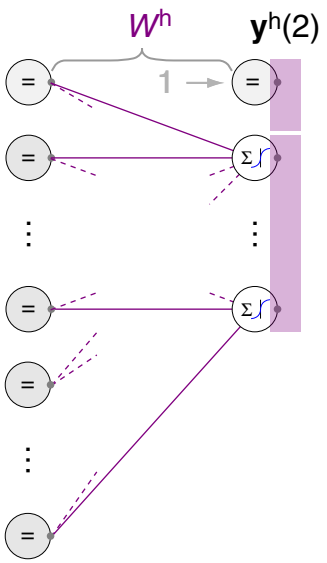
Input encoding over t.

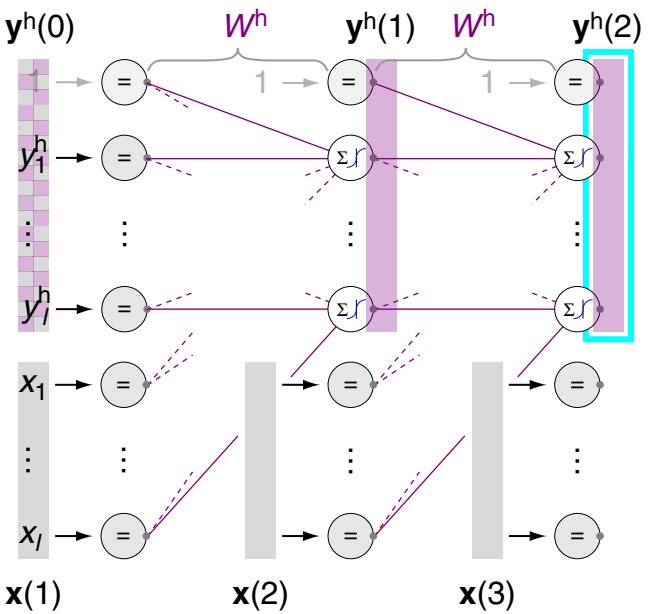
The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 1 → 2





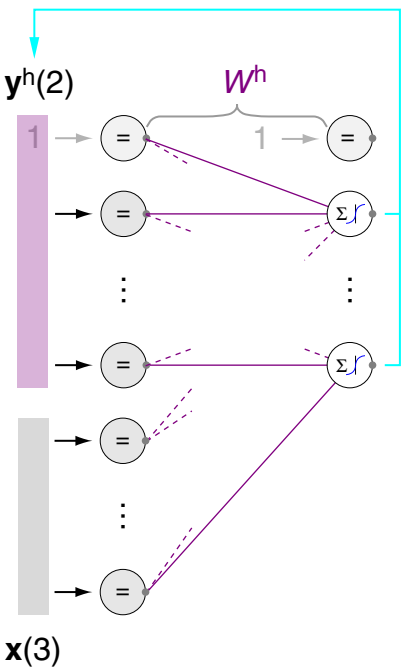
Input encoding over t.

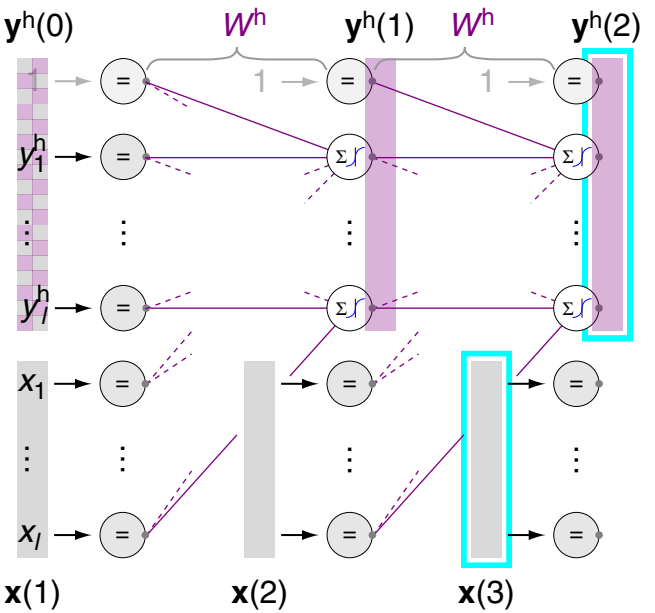
The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]

t: 1 → 2



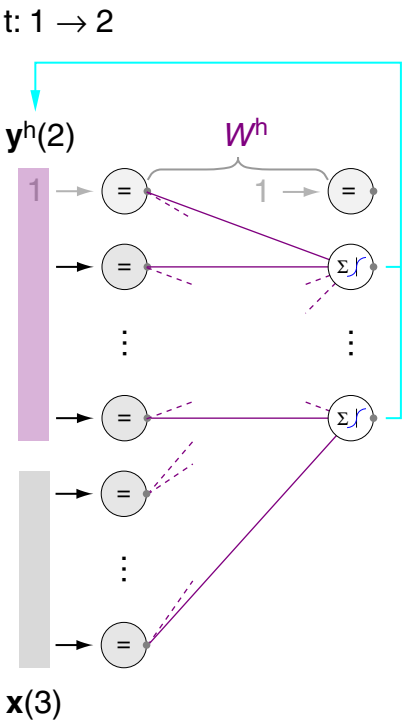


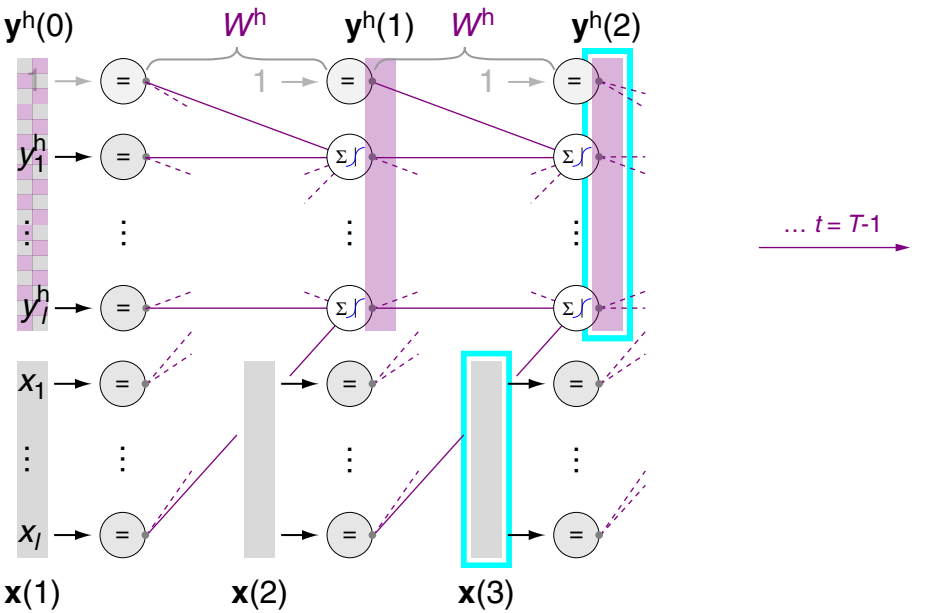
Input encoding over t.

The hidden layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Encoding (continued) [encoding overview]





Input encoding over t.

The hidden layer at subsequent time steps.

Recurrent Neural Networks

(S1) Sequence-to-Class: Sentiment Classification

- I love my cat. $\rightarrow \oplus$
- Cats and dogs lap water. $\rightarrow \oplus$
- It is raining cats and dogs. $\rightarrow \ominus$
- Cats and dogs are not allowed. $\rightarrow \ominus$
- Cats and dogs have always been natural enemies. $\rightarrow \ominus$

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

Recurrent Neural Networks

(S1) Sequence-to-Class: Sentiment Classification (continued)

- I love my cat. $\rightarrow \oplus$
- Cats and dogs lap water. $\rightarrow \oplus$
- It is raining cats and dogs. $\rightarrow \ominus$
- Cats and dogs are not allowed. $\rightarrow \ominus$
- Cats and dogs have always been natural enemies. $\rightarrow \ominus$

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

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

$\widehat{=}$ [`word_11`, `word_15`, `word_16`, `word_6`]

$\widehat{=}$ I love my cat

Recurrent Neural Networks

(S1) Sequence-to-Class: Sentiment Classification (continued)

- I love my cat. $\rightarrow \oplus$
- Cats and dogs lap water. $\rightarrow \oplus$
- It is raining cats and dogs. $\rightarrow \ominus$
- Cats and dogs are not allowed. $\rightarrow \ominus$
- Cats and dogs have always been natural enemies. $\rightarrow \ominus$

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

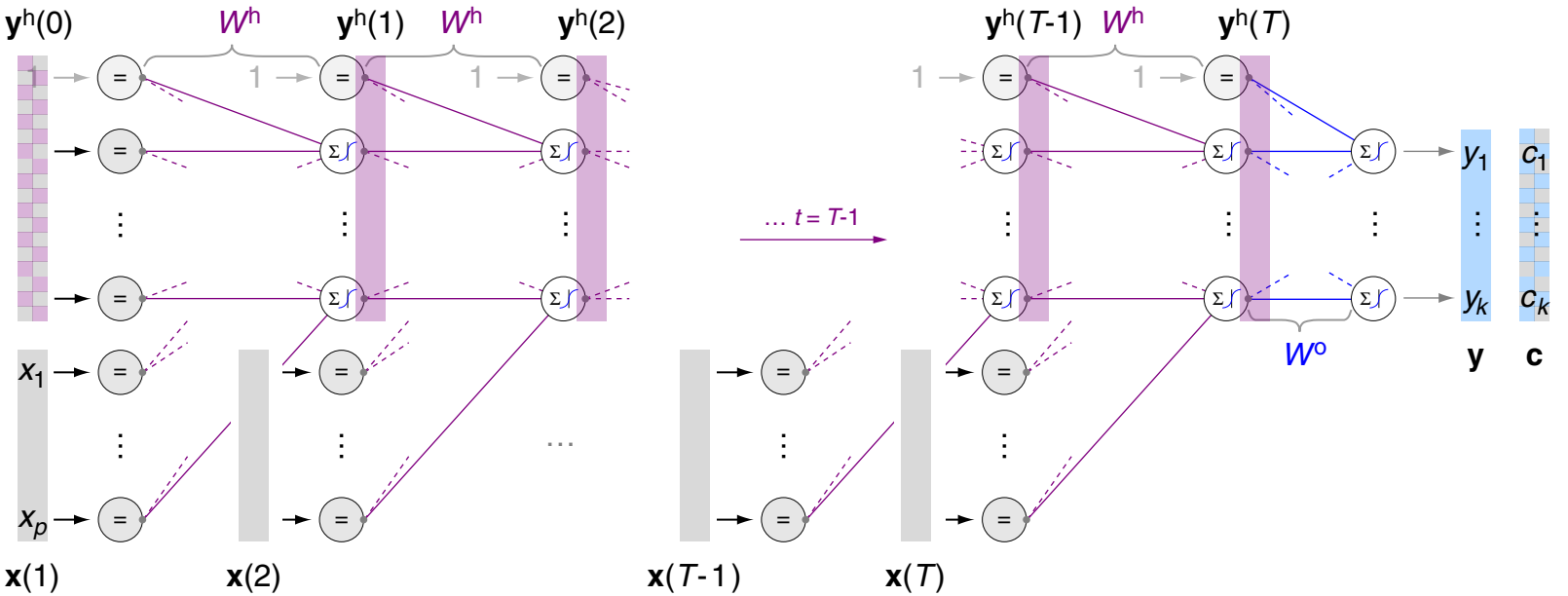
Input:
$$[\mathbf{x}(1), \ldots, \mathbf{x}(4)] = \left[\begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ \vdots \end{pmatrix} \right]$$
$$\widehat{=} \; [\texttt{word\_11}, \texttt{word\_15}, \texttt{word\_16}, \texttt{word\_6}]$$
$$\widehat{=} \; \text{I love my cat}$$

Output:
$$\mathbf{y}([\mathbf{x}(1), \ldots, \mathbf{x}(4)]) = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

Target:
$$\mathbf{c} = \begin{pmatrix} \oplus \\ \cdot \end{pmatrix}$$

Recurrent Neural Networks

(S1) Sequence-to-Class Mapping with RNNs



Input:

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

Hidden:

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

Target:

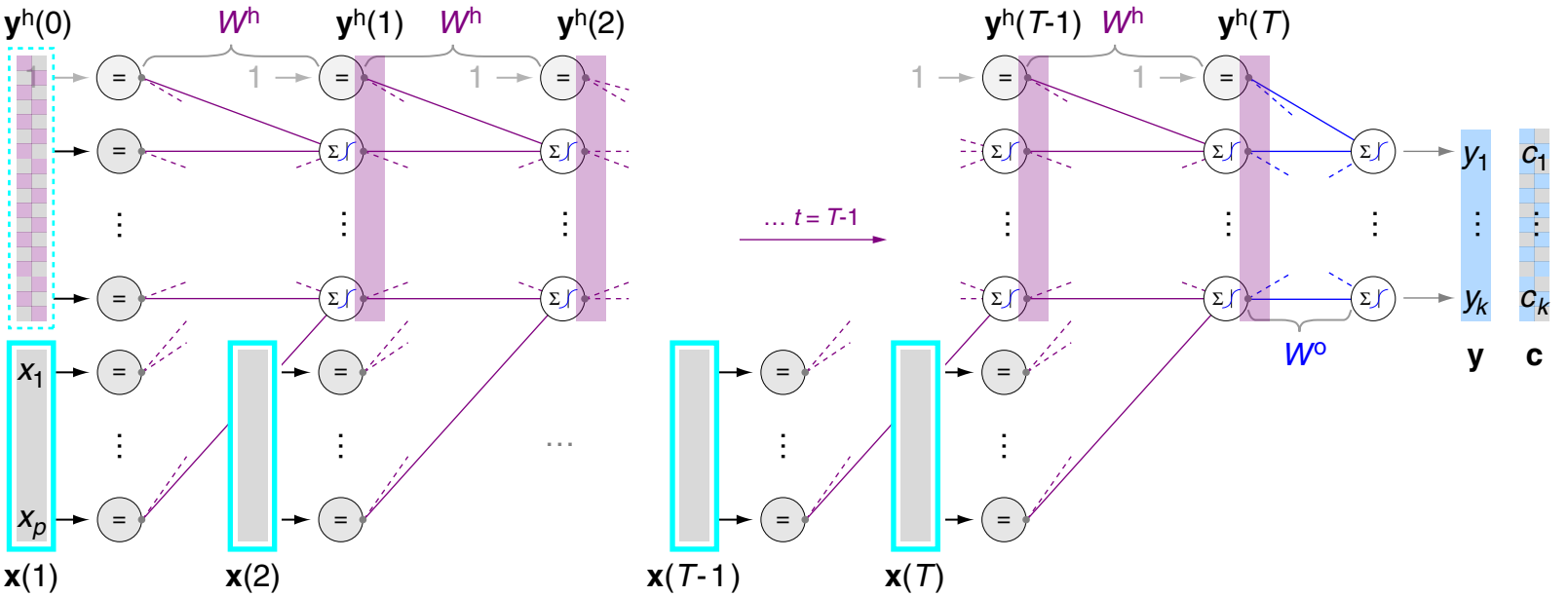
$$\mathbf{c}$$

Output:

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\,\mathbf{y}^{\mathrm{h}}(T)\right)$$

Recurrent Neural Networks

(S1) Sequence-to-Class Mapping with RNNs (continued)



Input:

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

Hidden:

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

Target:

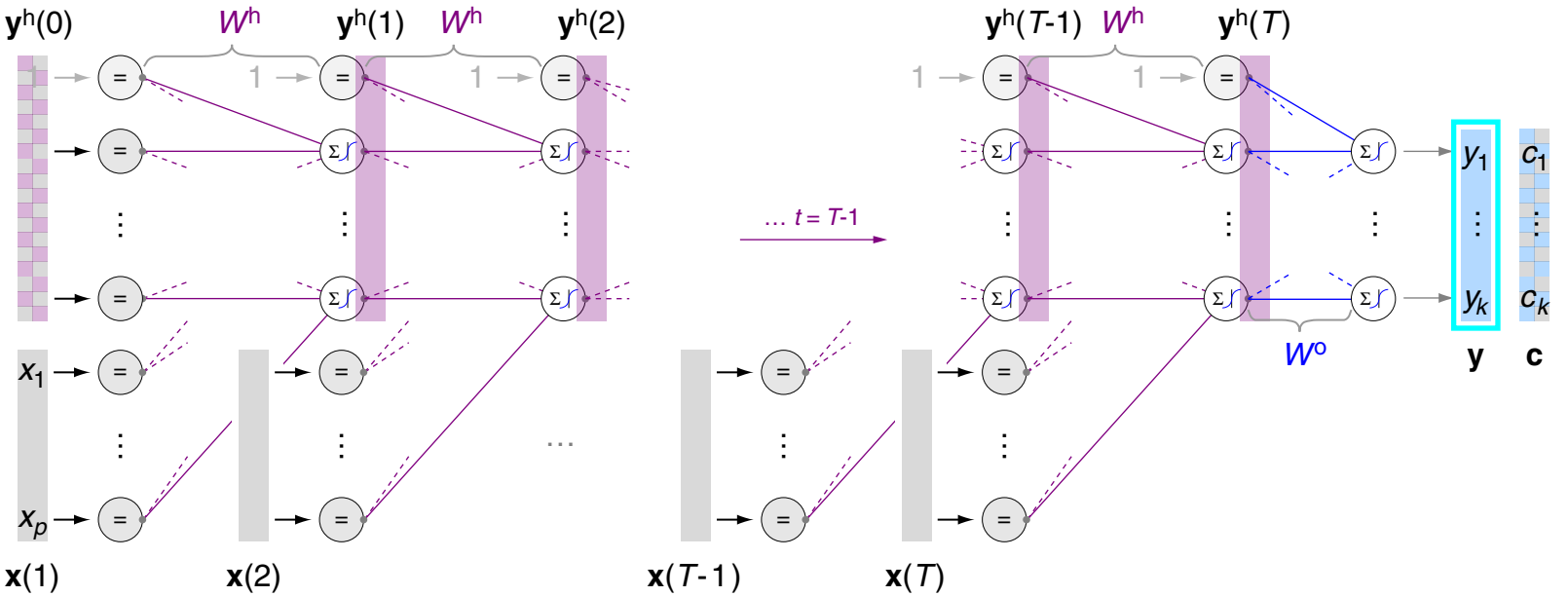
$$\mathbf{c}$$

Output:

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\,\mathbf{y}^{\mathrm{h}}(T)\right)$$

Recurrent Neural Networks

(S1) Sequence-to-Class Mapping with RNNs (continued)



Input:

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

Hidden:

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

Target:

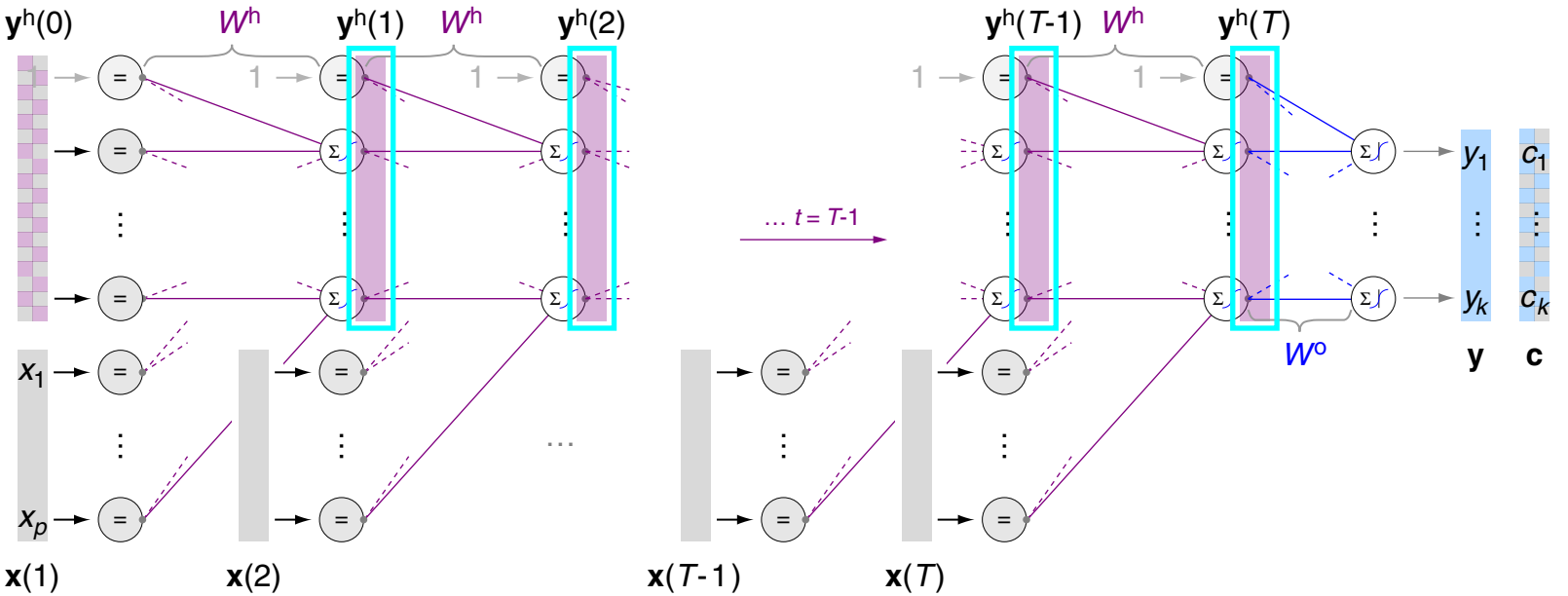
$\mathbf{c}$

Output:

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\,\mathbf{y}^{\mathrm{h}}(T)\right)$$

Recurrent Neural Networks

(S1) Sequence-to-Class Mapping with RNNs (continued)



Input:

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

Hidden:

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

Target:

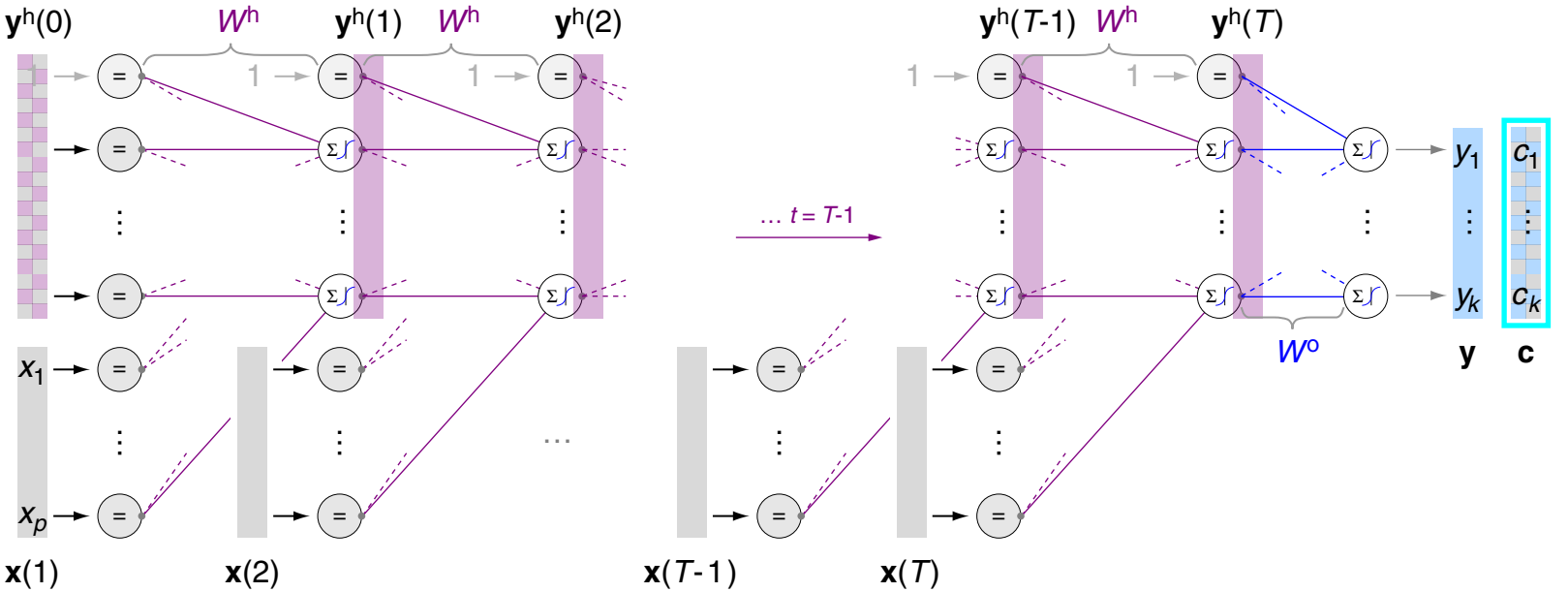
$$\mathbf{c}$$

Output:

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^o\, \mathbf{y}^h(T)\right)$$

Recurrent Neural Networks

(S1) Sequence-to-Class Mapping with RNNs (continued)



Input:

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

Hidden:

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

Target:

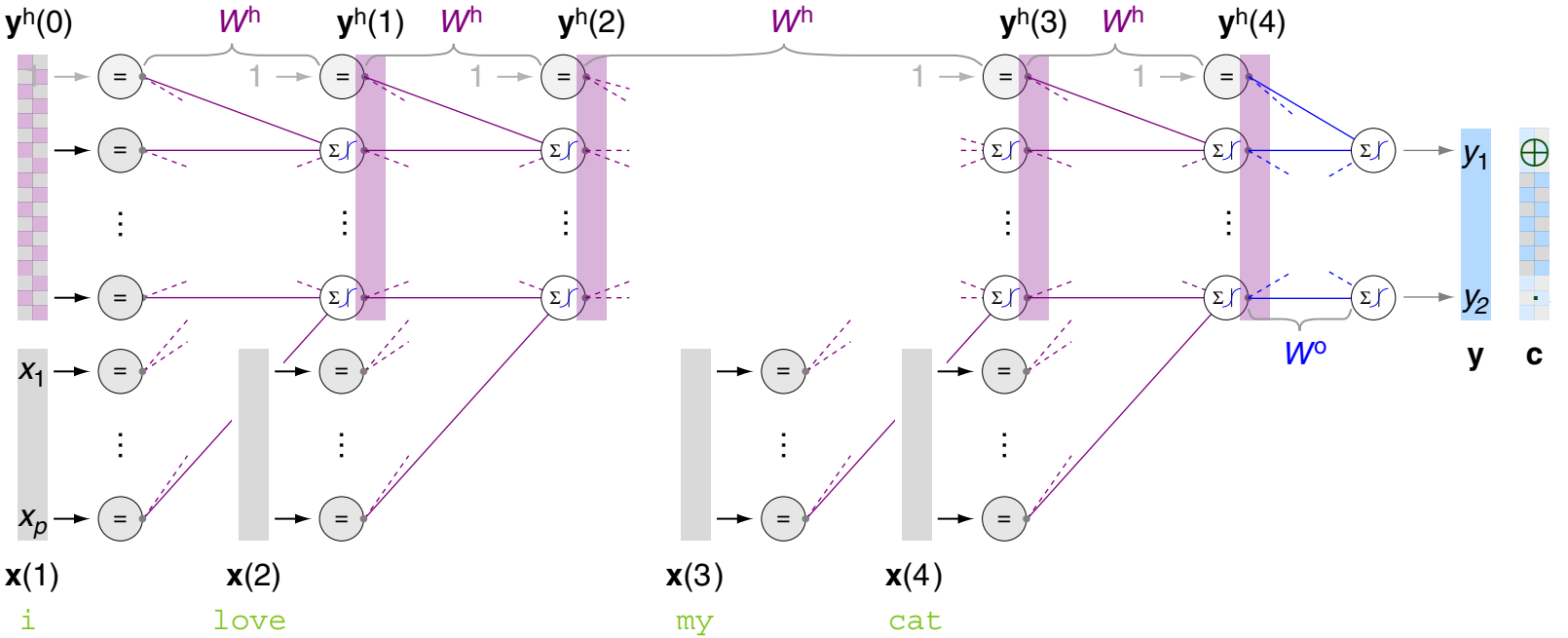
$$\mathbf{c}$$

Output:

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\,\mathbf{y}^{\mathrm{h}}(T)\right)$$

Recurrent Neural Networks

(S1) Sequence-to-Class Mapping with RNNs (continued)



Input:

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

Hidden:

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

Target:

$$\mathbf{c}$$

Output:

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^{\mathsf{o}}\,\mathbf{y}^{\mathsf{h}}(4)\right)$$

Remarks:

- We denote $\mathbf{y}^{\mathsf{h}}(0)$ not as input since this kind of predefined hidden vector does not contain any "actual knowledge", but is usually initialized as vector of zeros.
- To keep the illustrations clear we use the bag-of-words model for representing (= embedding) the words as vectors $\mathbf{x}(t)$.

  In practice, however, one considers semantically stronger (language-model-based) embeddings, which also encode information about neighborhoods and occurrence probabilities. In this regard, either a previously computed embedding can be used, or the embedding can be learned along with the task, end-to-end.
- Recap. $\boldsymbol{\sigma}_1()$ denotes the softmax function. $\boldsymbol{\sigma}_1 : \mathbf{R}^k \to \Delta^{k-1}$, generalizes the logistic (sigmoid) function to k dimensions (to k exclusive classes), where $\boldsymbol{\sigma}_1(\mathbf{z})|_i = e^{z_i} / \sum_{j=1}^{k} e^{z_j}$. [Wikipedia]

  $\Delta^{k-1} \subset \mathbf{R}^k$ denotes the standard $k-1$-simplex, which contains all k-tuples with non-negative elements that sum to 1. [Wikipedia]

Recurrent Neural Networks

The IGD Algorithm for Sequence-to-Class Tasks [IGD$_{c2seq}$]

Algorithm:	IGD$_{seq2c}$	Incremental Gradient Descent for RNNs at seq2class tasks.
Input:	D	Multiset of examples $([\mathbf{x}(1), \ldots, \mathbf{x}(T)], \mathbf{c})$ with $\mathbf{x}(t) \in \mathbf{R}^p$, $\mathbf{c} \in \{0,1\}^k$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{y}^h(0), W^{\mathsf{h}}, W^{\mathsf{o}}$	Weights of predefined hidden vector and matrices. (= hypothesis)

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

Recurrent Neural Networks

The IGD Algorithm for Sequence-to-Class Tasks (continued) [$\text{IGD}_{\text{c2seq}}$]

Algorithm:	$\text{IGD}_{\text{seq2c}}$	Incremental Gradient Descent for RNNs at seq2class tasks.
Input:	D	Multiset of examples $([\mathbf{x}(1),\ldots,\mathbf{x}(T)],\mathbf{c})$ with $\mathbf{x}(t)\in\mathbf{R}^p,\ \mathbf{c}\in\{0,1\}^k$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{y}^h(0), W^{\text{h}}, W^{\text{o}}$	Weights of predefined hidden vector and matrices. (= hypothesis)

1. *initialize_random_weights*$(\mathbf{y}^h(0), W^{\text{h}}, W^{\text{o}})$, $t_{\text{training}} = 0$
2. **REPEAT**
3. $\quad t_{\text{training}} = t_{\text{training}} + 1$
4. $\quad$ FOREACH $([\mathbf{x}(1),\ldots,\mathbf{x}(T)],\mathbf{c}) \in D$ DO
5. $\quad\quad$ FOR $t = 1$ TO T DO // forward propagation

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

$\quad\quad$ ENDDO

$$\mathbf{y} = \boldsymbol{\sigma}_1\left(W^{\text{o}}\,\mathbf{y}^{\text{h}}(T)\right)$$

6. $\quad\quad \boldsymbol{\delta} = \mathbf{c} - \mathbf{y}([\mathbf{x}(1),\ldots,\mathbf{x}(T)])$
7. $\quad\quad$ Calculate $_{\triangle}\mathbf{y}^{\text{h}}(0)$, $_{\triangle}W^{\text{h}}$, $_{\triangle}W^{\text{o}}$ // backpropagation
8. $\quad\quad \mathbf{y}^{\text{h}}(0) = \mathbf{y}^{\text{h}}(0) + {}_{\triangle}\mathbf{y}^{\text{h}}(0),\ W^{\text{h}} = W^{\text{h}} + {}_{\triangle}W^{\text{h}},\ W^{\text{o}} = W^{\text{o}} + {}_{\triangle}W^{\text{o}}$
9. $\quad$ ENDDO
10. **UNTIL**(*convergence*$(D, \mathbf{y}(\cdot), t_{\text{training}})$)
11. *return*$(\mathbf{y}^h(0), W^{\text{h}}, W^{\text{o}})$

Recurrent Neural Networks

Types of Learning Tasks [Recap]

(S1) sequence $\rightarrow$ class

sentence $\rightarrow \{\oplus, \ominus\}$

`i love my cat` $\rightarrow \oplus$

(S2) class $\rightarrow$ sequence

$\{\oplus, \ominus\} \rightarrow$ sentence

$\oplus \rightarrow$ `i love my cat`

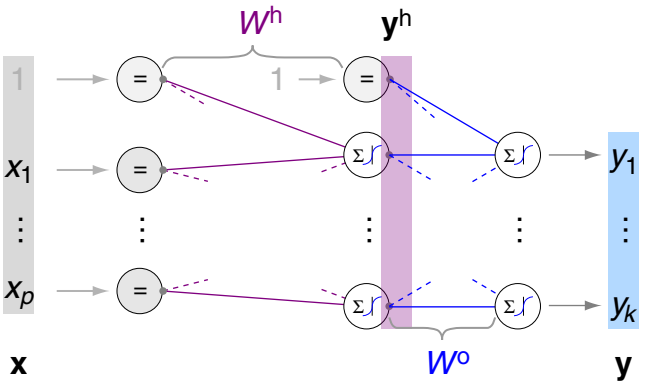
(S3) sequence $\rightarrow$ sequence

English sentence $\rightarrow$ German sentence

`i love my cat` $\rightarrow$ `ich liebe meine katze`

Recurrent Neural Networks

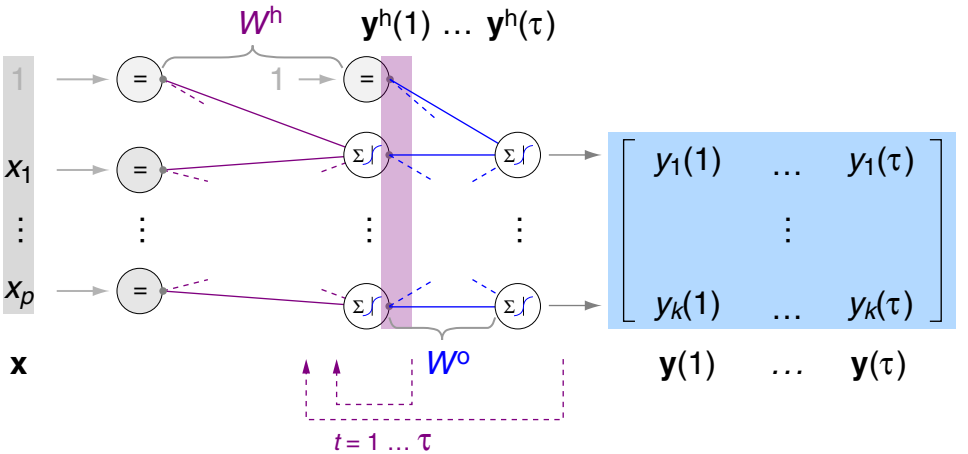
RNN Sequence Decoding



- 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 $\mathbf{y}(\mathbf{x})$.

Recurrent Neural Networks

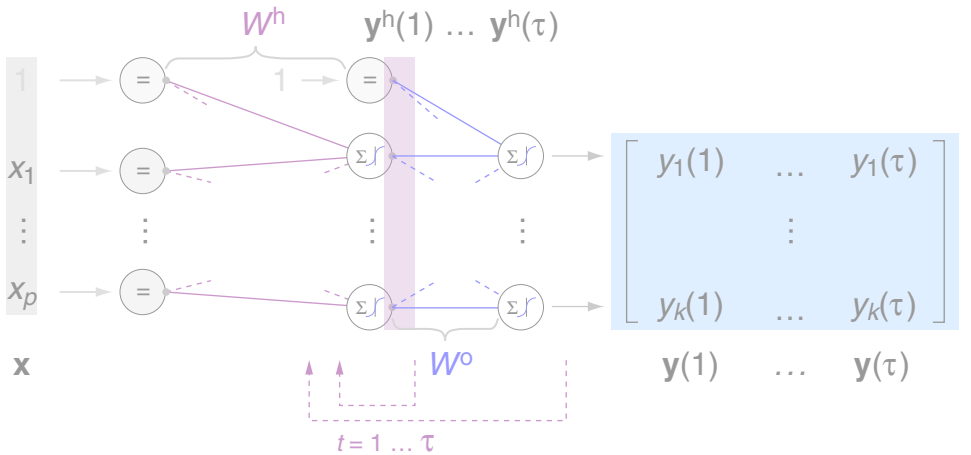
RNN Sequence Decoding (continued)



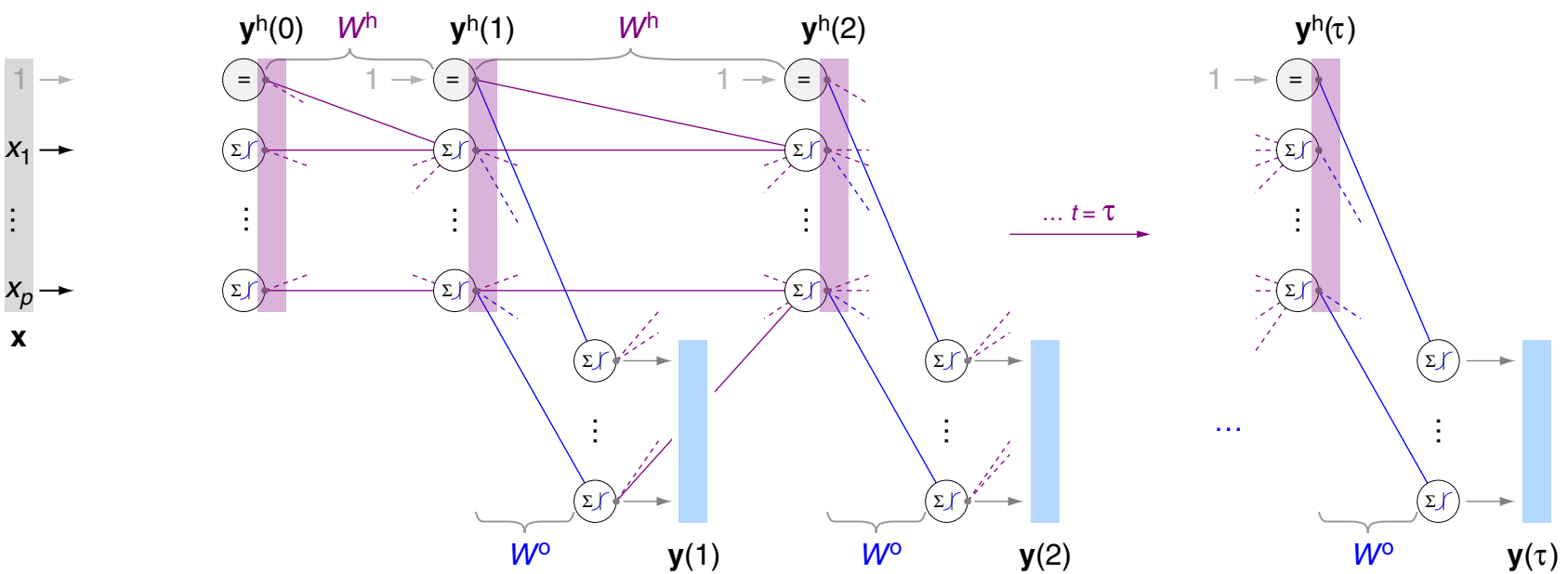
- One p-dimensional input vector $\mathbf{x}$.
- One hidden and one output layer, which are recurrently updated.
- Sequence of k-dimen. output vectors $[\mathbf{y}(\mathbf{x}, 1), \ldots, \mathbf{y}(\mathbf{x}, \tau)]$ or $[\mathbf{y}(1), \ldots, \mathbf{y}(\tau)]$.

Recurrent Neural Networks

RNN Sequence Decoding (continued)

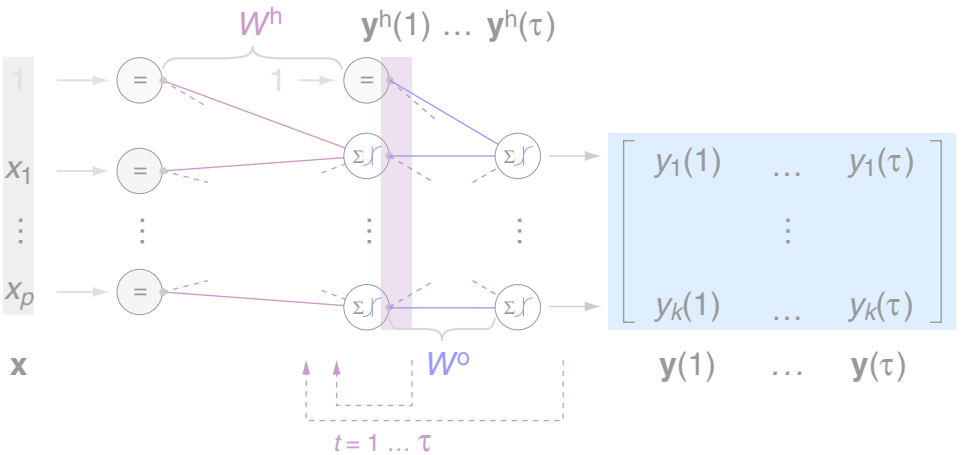


Unfold hidden and output layer

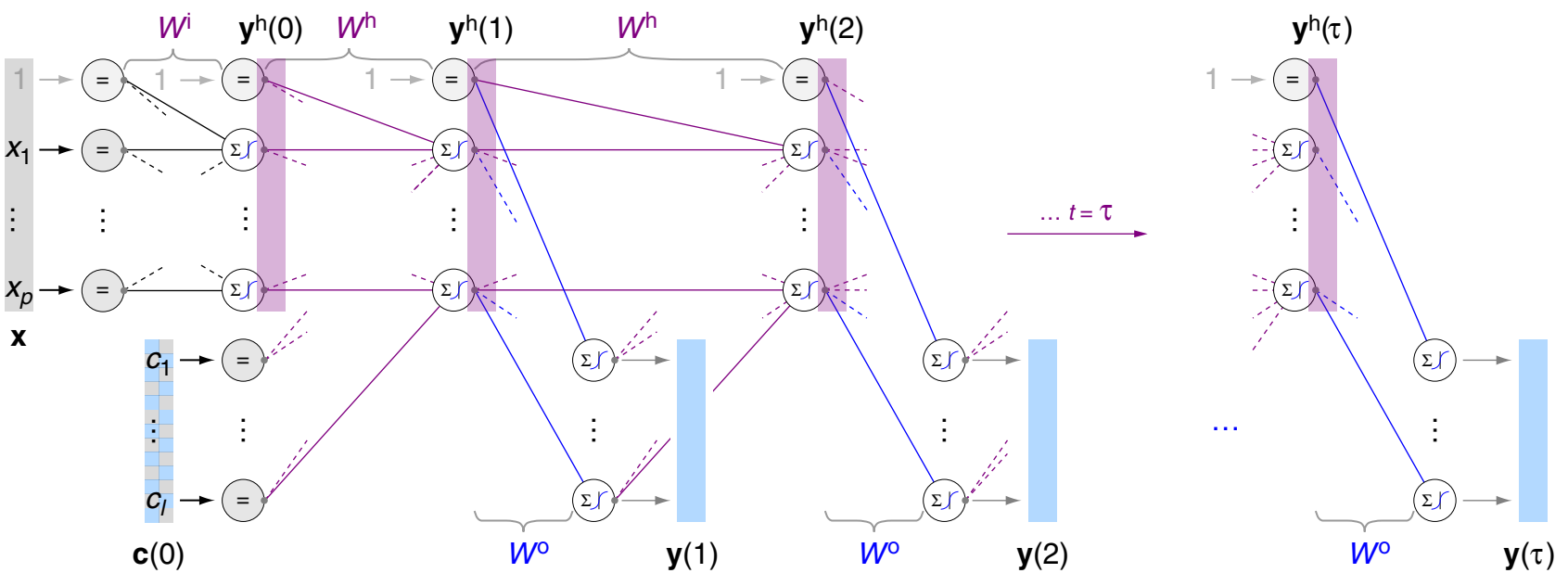


Recurrent Neural Networks

RNN Sequence Decoding (continued)



Unfold hidden and output layer



Remarks:

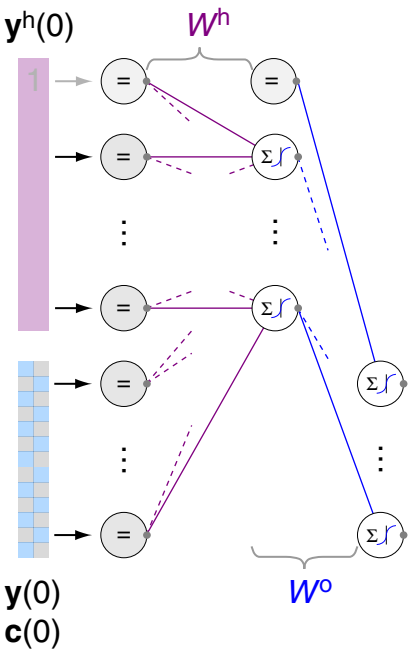
- An output sequence is written in brackets, $[\mathbf{y}(1), \ldots, \mathbf{y}(\tau)]$, where $\mathbf{y}(t), t = 1, \ldots, \tau$, denotes the output vector at time step t.
- The words in the output sequence are usually one-hot-encoded, i.e., by a k-dimensional output vector with a "1" whose position 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 does not have to be equal to T.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1

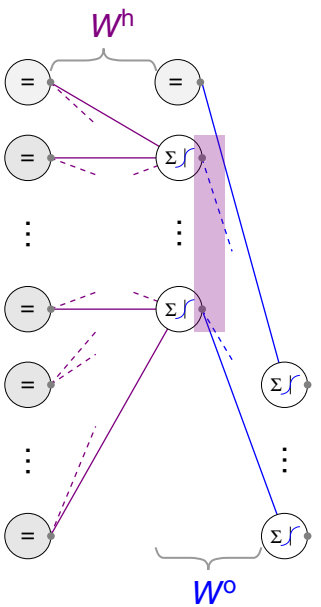


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1

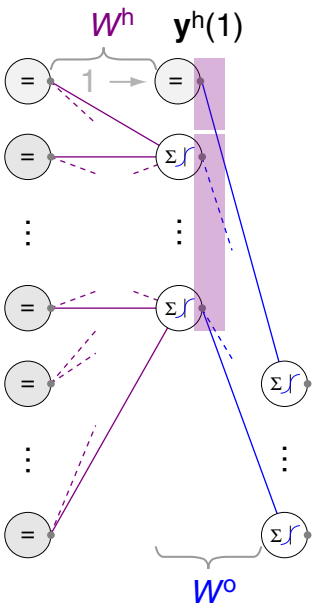


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1

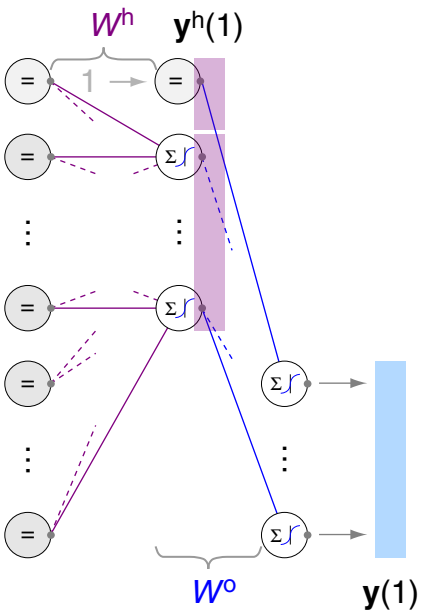


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1

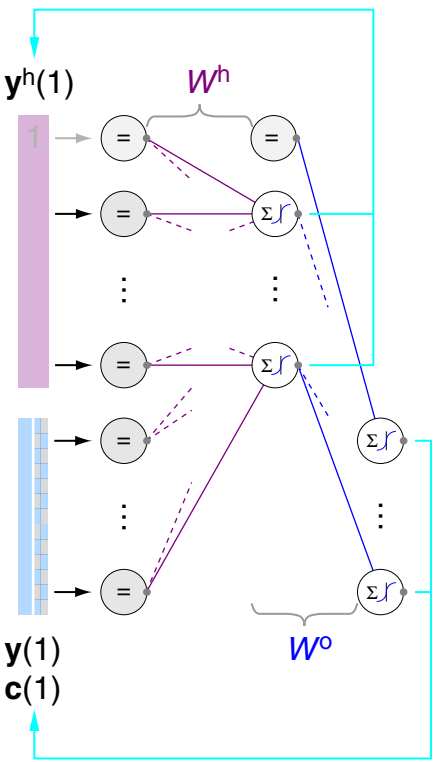


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1

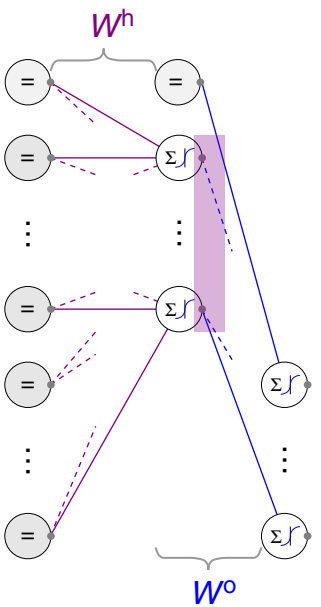


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2

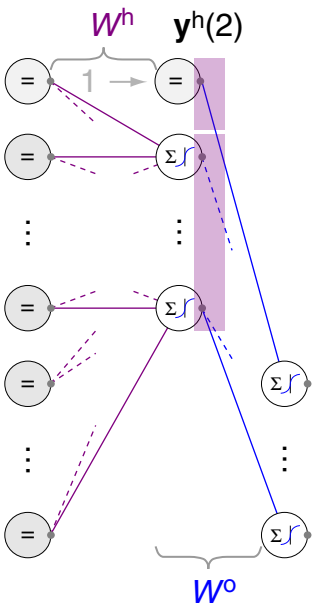


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2

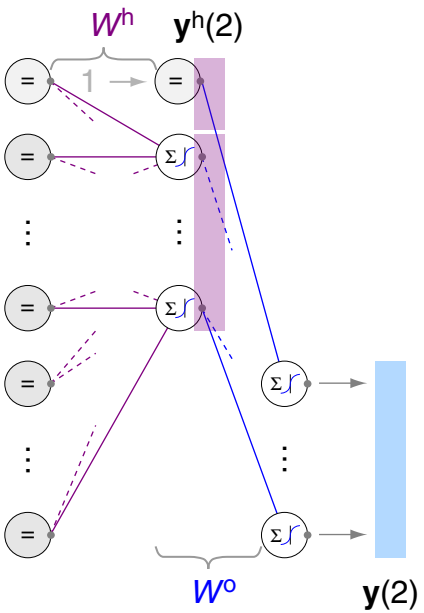


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2

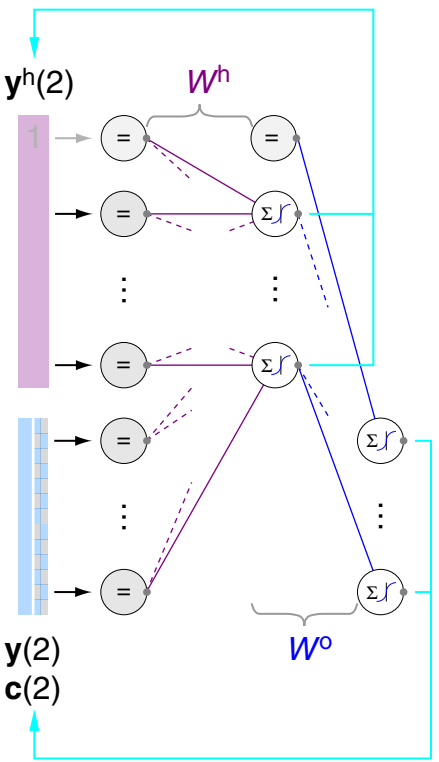


Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2

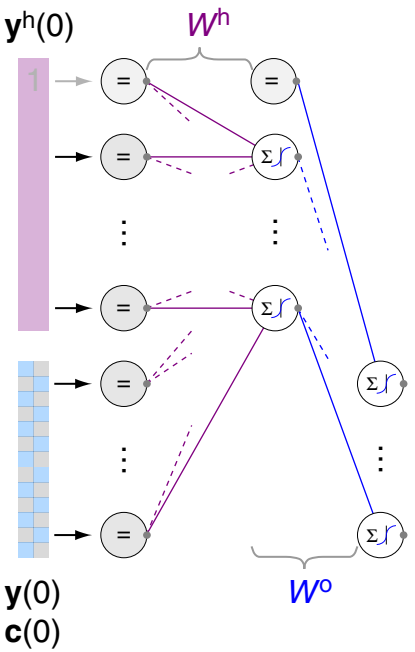


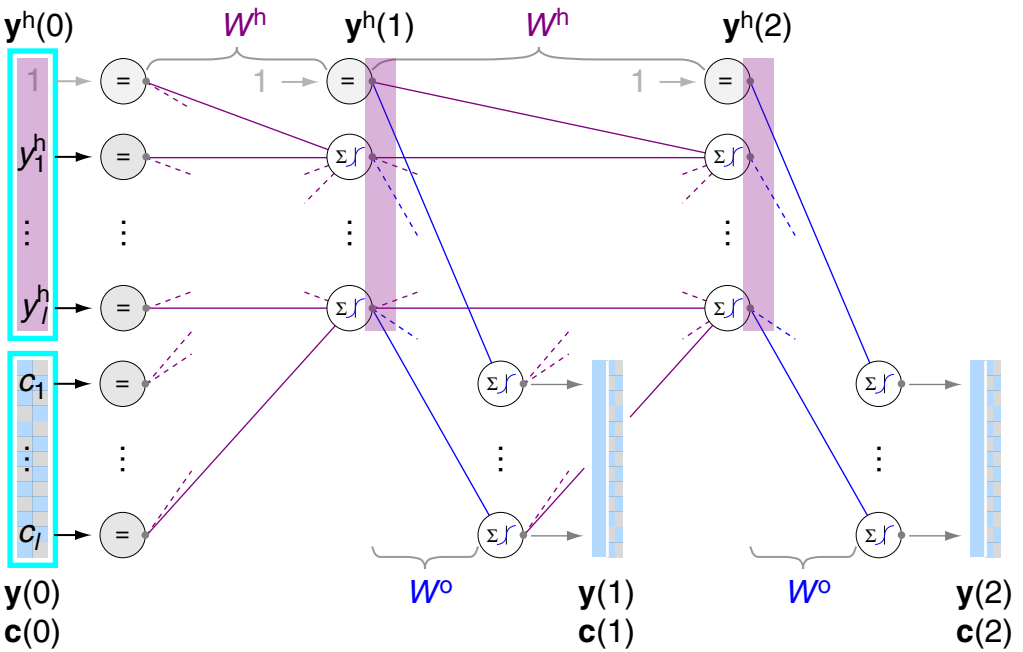
Output decoding over t.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1





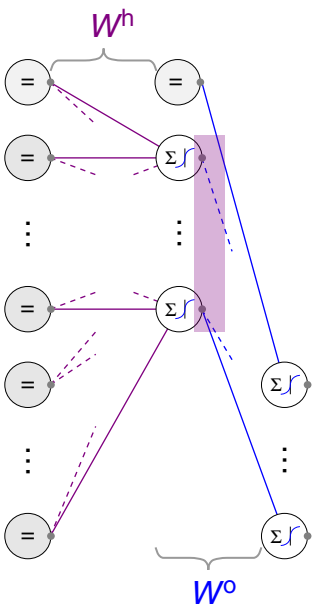
Output decoding over t.

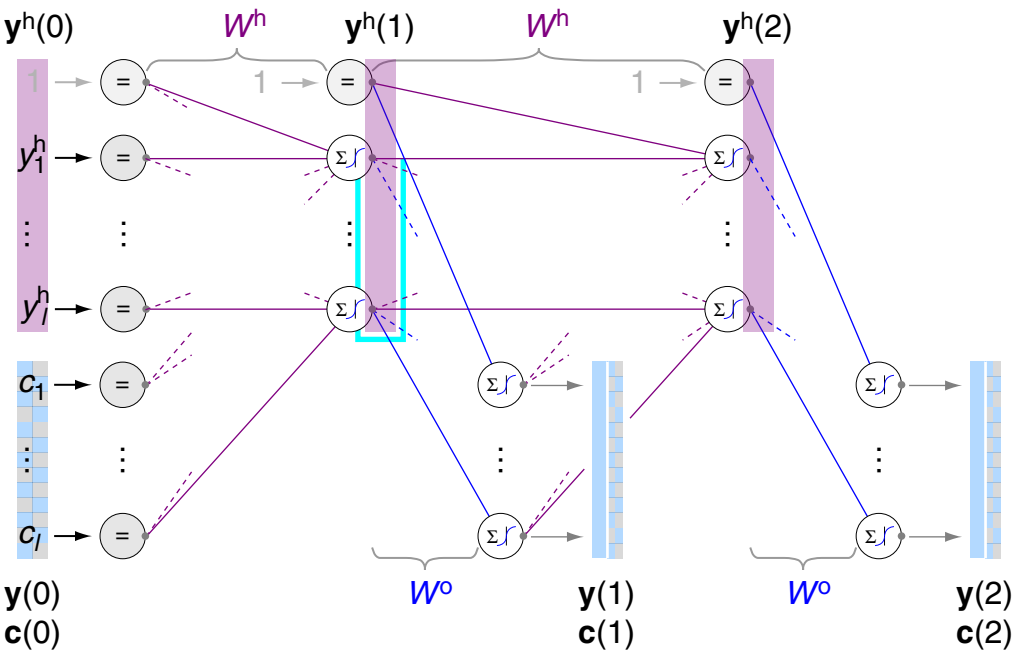
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1





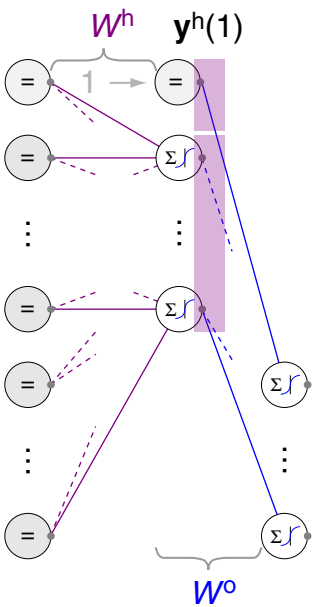
Output decoding over t.

Hidden and output layer at subsequent time steps.

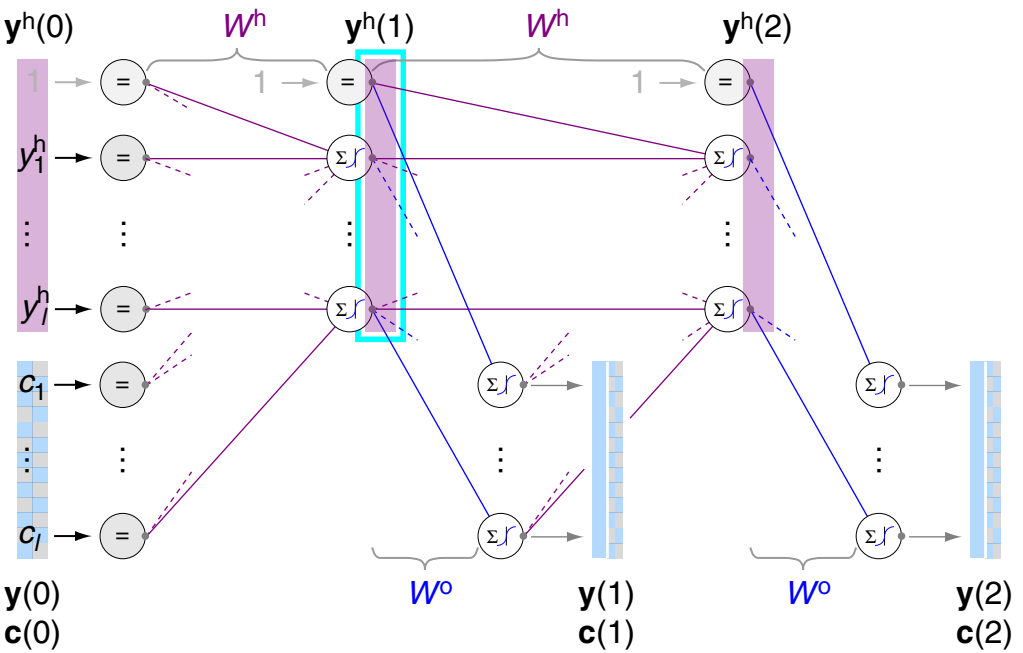
Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1



Output decoding over t.

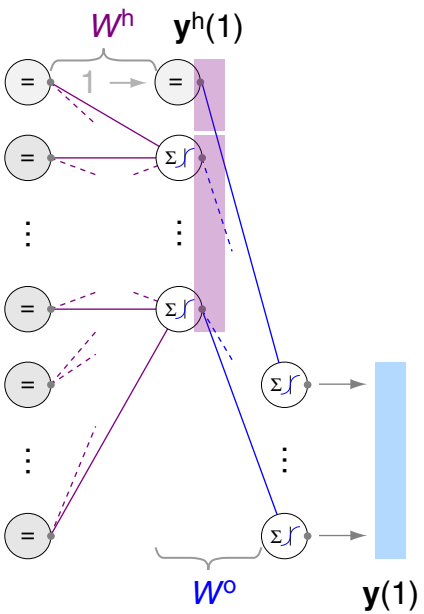


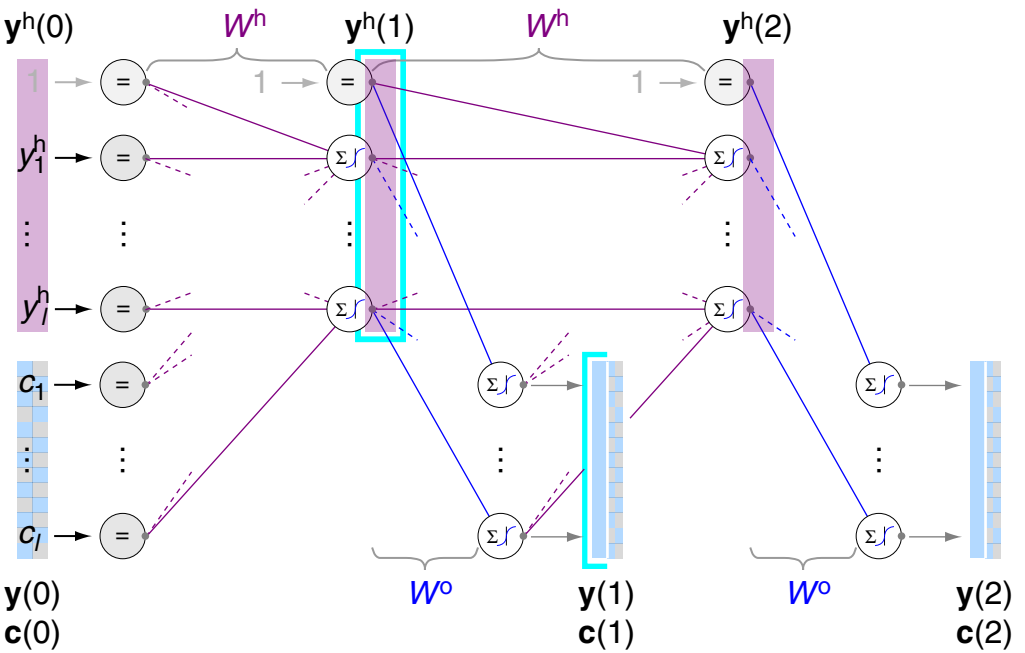
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 0 → 1



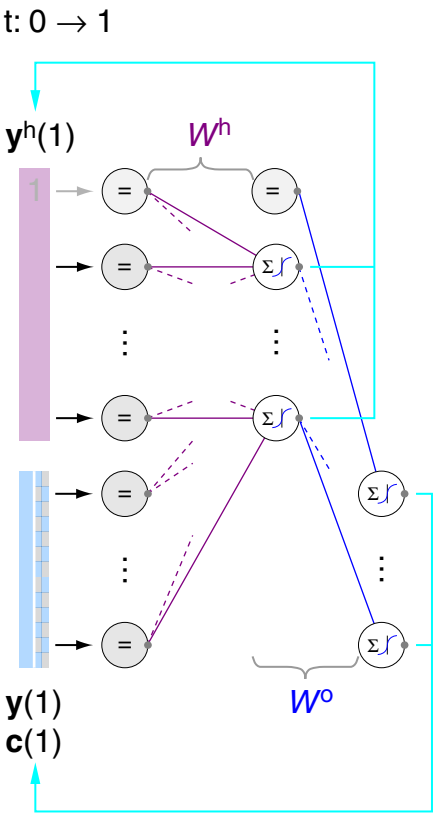


Output decoding over t.

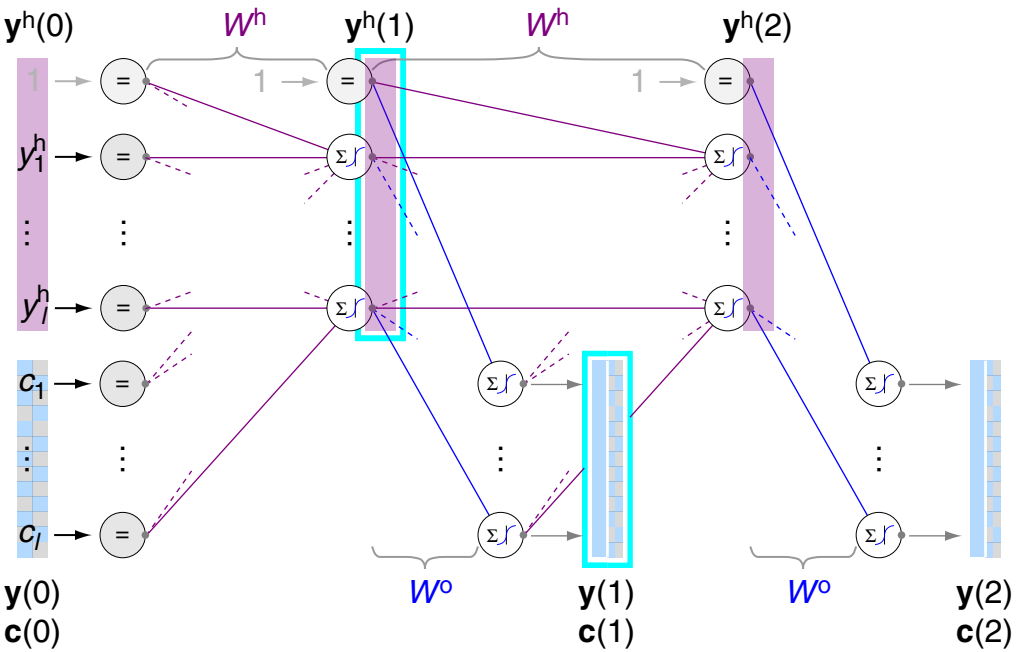
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]



Output decoding over t.

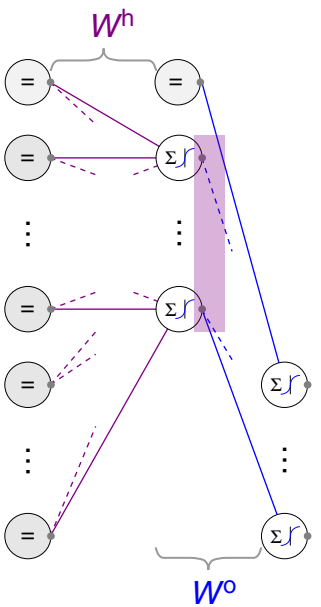


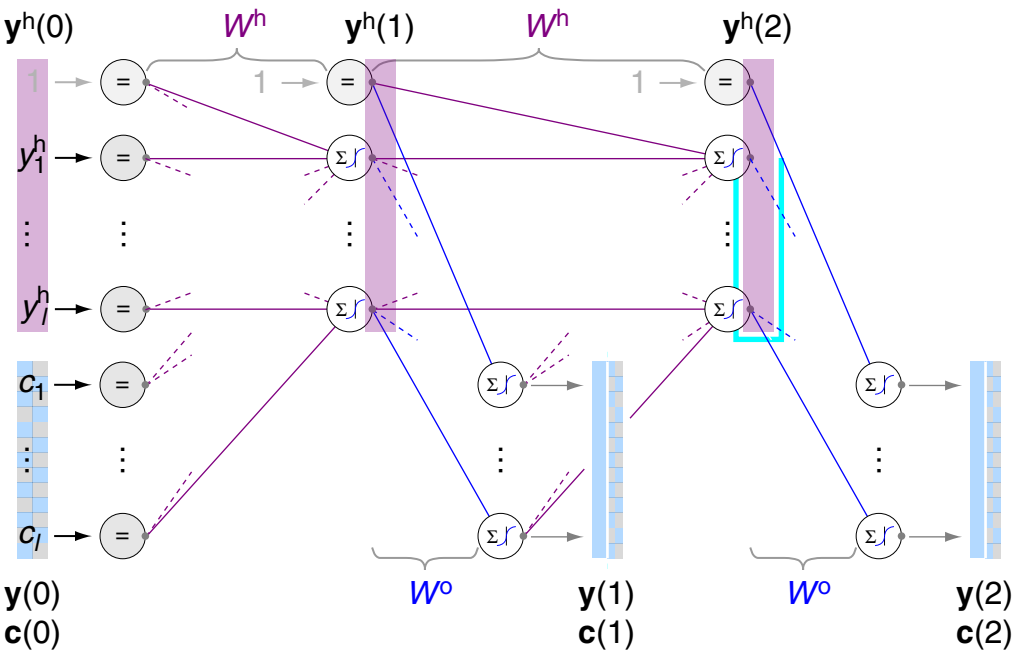
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2





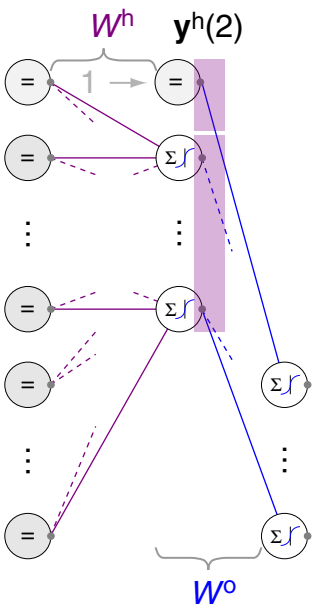
Output decoding over t.

Hidden and output layer at subsequent time steps.

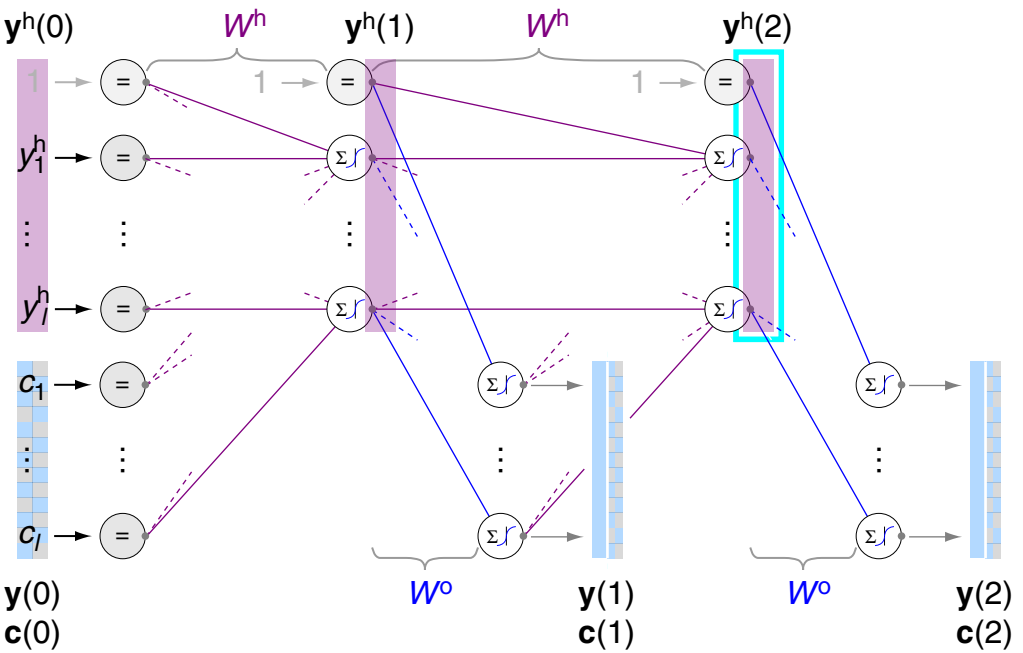
Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2



Output decoding over t.

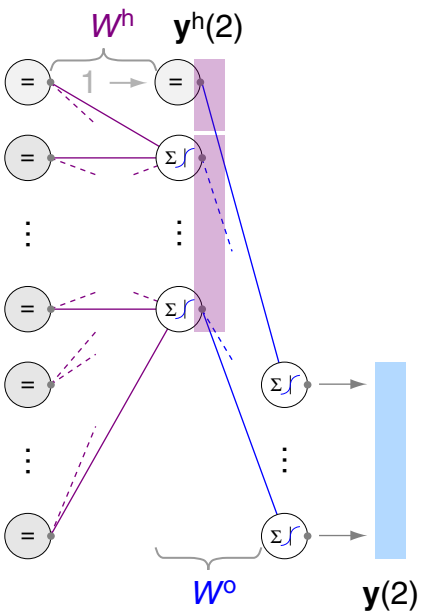


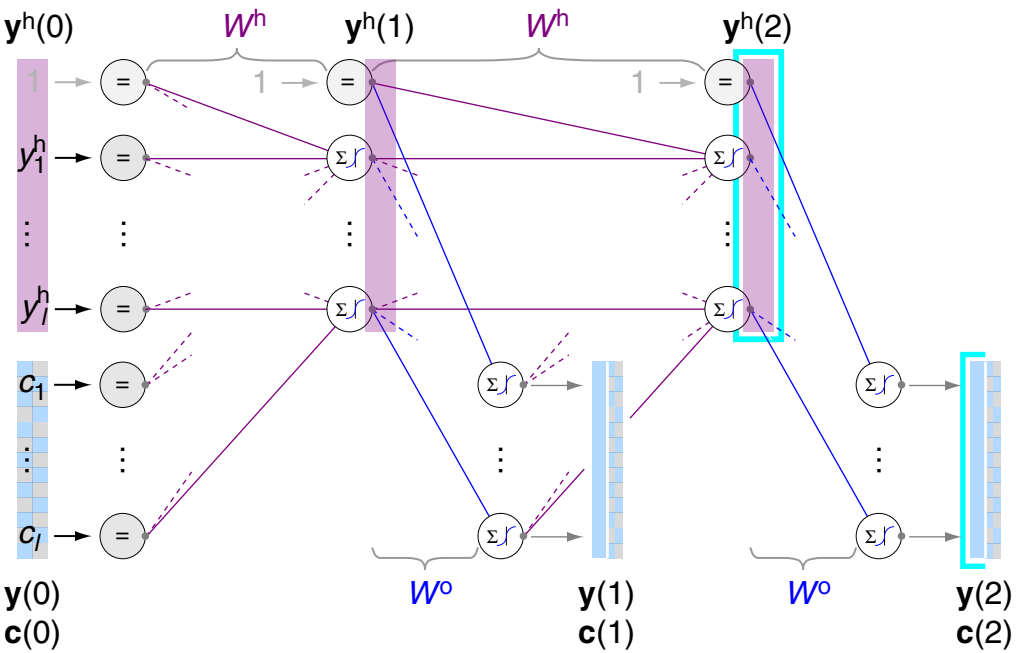
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]

t: 1 → 2



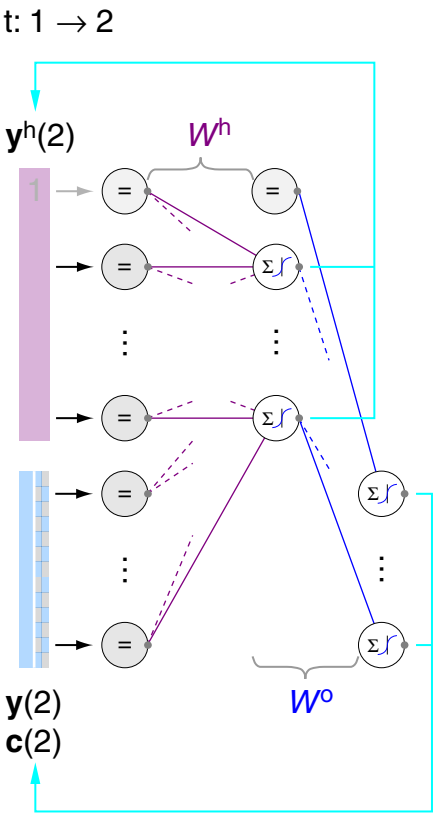


Output decoding over t.

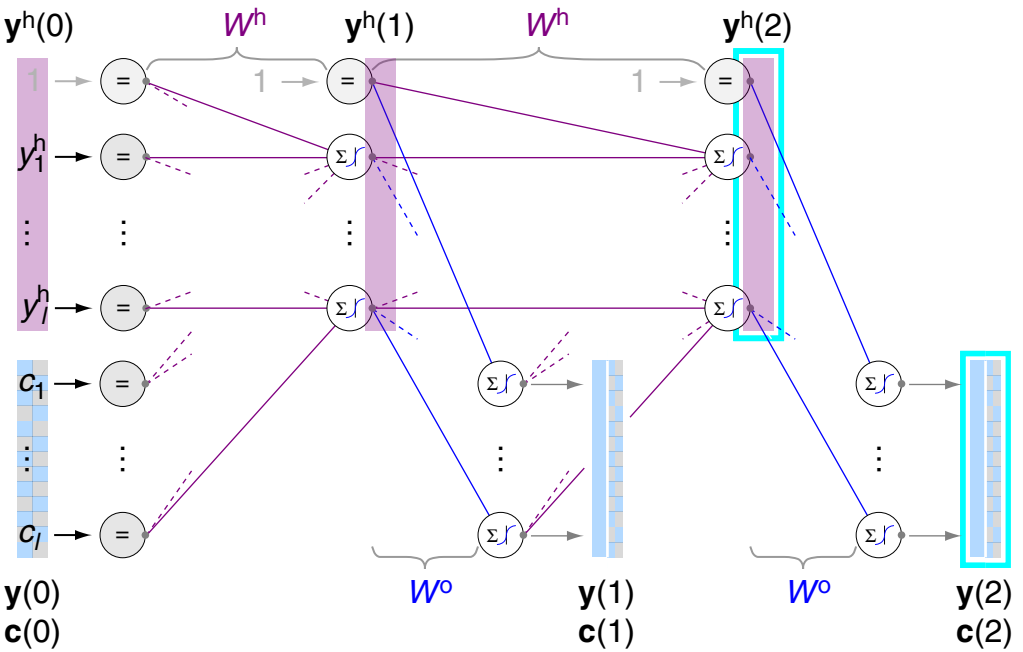
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]



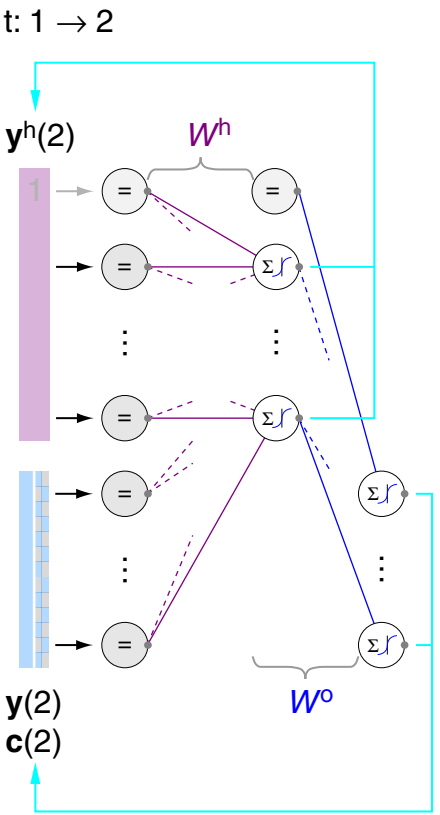
Output decoding over t.



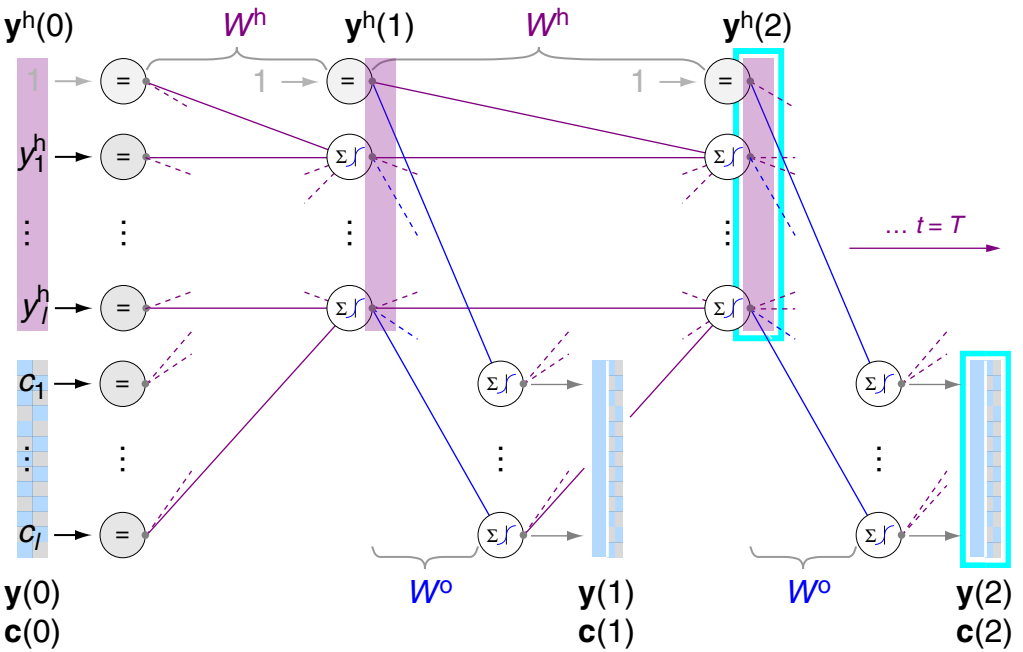
Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

RNN Sequence Decoding (continued) [decoding overview]



Output decoding over t.



Hidden and output layer at subsequent time steps.

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation

- $\oplus$ $\rightarrow$ I love my cat.
- $\oplus$ $\rightarrow$ Cats and dogs lap water.
- $\ominus$ $\rightarrow$ It is raining cats and dogs.
- $\ominus$ $\rightarrow$ Cats and dogs are not allowed.
- $\ominus$ $\rightarrow$ Cats and dogs have always been natural enemies.

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

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation (continued)

- $\oplus \rightarrow$ I love my cat.
- $\oplus \rightarrow$ Cats and dogs lap water.
- $\ominus \rightarrow$ It is raining cats and dogs.
- $\ominus \rightarrow$ Cats and dogs are not allowed.
- $\ominus \rightarrow$ Cats and dogs have always been natural enemies.

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

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

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)], \quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}} \texttt{<start>}, \quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(5) \mathrel{\widehat{=}} \texttt{<end>}$

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation (continued)

- $\oplus$ $\rightarrow$ I love my cat.
- $\oplus$ $\rightarrow$ Cats and dogs lap water.
- $\ominus$ $\rightarrow$ It is raining cats and dogs.
- $\ominus$ $\rightarrow$ Cats and dogs are not allowed.
- $\ominus$ $\rightarrow$ Cats and dogs have always been natural enemies.

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

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

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)], \quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}} \texttt{<start>}, \quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(5) \mathrel{\widehat{=}} \texttt{<end>}$

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation (continued)

- $\oplus$ $\rightarrow$ I love my cat.
- $\oplus$ $\rightarrow$ Cats and dogs lap water.
- $\ominus$ $\rightarrow$ It is raining cats and dogs.
- $\ominus$ $\rightarrow$ Cats and dogs are not allowed.
- $\ominus$ $\rightarrow$ Cats and dogs have always been natural enemies.

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

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

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)], \quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}} \texttt{<start>}, \quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(5) \mathrel{\widehat{=}} \texttt{<end>}$

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation (continued)

- $\oplus$ $\rightarrow$ I love my cat.
- $\oplus$ $\rightarrow$ Cats and dogs lap water.
- $\ominus$ $\rightarrow$ It is raining cats and dogs.
- $\ominus$ $\rightarrow$ Cats and dogs are not allowed.
- $\ominus$ $\rightarrow$ Cats and dogs have always been natural enemies.

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

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

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}} \texttt{<start>},\quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(5) \mathrel{\widehat{=}} \texttt{<end>}$

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation (continued)

- $\oplus$ $\rightarrow$ I love my cat.
- $\oplus$ $\rightarrow$ Cats and dogs lap water.
- $\ominus$ $\rightarrow$ It is raining cats and dogs.
- $\ominus$ $\rightarrow$ Cats and dogs are not allowed.
- $\ominus$ $\rightarrow$ Cats and dogs have always been natural enemies.

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

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

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}} \texttt{<start>},\quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(5) \mathrel{\widehat{=}} \texttt{<end>}$

Recurrent Neural Networks

(S2) Class-to-Sequence: Text Generation (continued)

- $\oplus$ $\rightarrow$ I love my cat.
- $\oplus$ $\rightarrow$ Cats and dogs lap water.
- $\ominus$ $\rightarrow$ It is raining cats and dogs.
- $\ominus$ $\rightarrow$ Cats and dogs are not allowed.
- $\ominus$ $\rightarrow$ Cats and dogs have always been natural enemies.

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

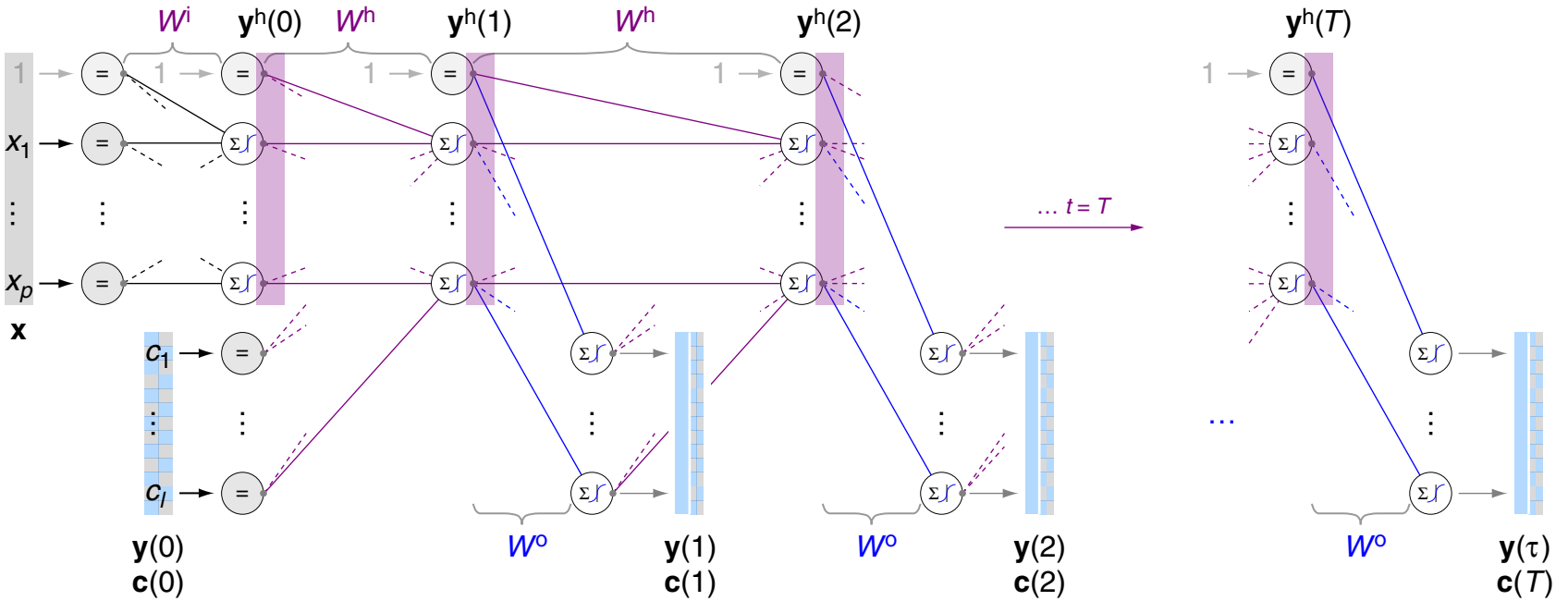
Input: $[\,[\,[\,[\,[\mathbf{x},\ \mathbf{y}(0)],\ \mathbf{y}(1)],\ \mathbf{y}(2)],\ldots],\ \mathbf{y}(\tau-1)],\quad \mathbf{x} = \begin{pmatrix}\oplus\\ \cdot\end{pmatrix}$

Output: $[\mathbf{y}(1), \mathbf{y}(2), \mathbf{y}(3), \ldots, \mathbf{y}(\tau)],\quad \mathbf{y}(0) \equiv \mathbf{c}(0) \mathrel{\widehat{=}} \texttt{<start>},\quad \mathbf{y}(\tau) \mathrel{\widehat{=}} \mathbf{c}(5) \mathrel{\widehat{=}} \texttt{<end>}$

Target:
$$[\mathbf{c}(1), \ldots, \mathbf{c}(5)] = \left[\begin{pmatrix}0\\ \vdots\\ 1\\ 0\\ \vdots\end{pmatrix}, \begin{pmatrix}0\\ \vdots\\ 1\\ 0\\ \vdots\end{pmatrix}, \begin{pmatrix}0\\ \vdots\\ 1\\ 0\end{pmatrix}, \begin{pmatrix}0\\ 1\\ 0\\ \vdots\end{pmatrix}, \begin{pmatrix}0\\ \vdots\\ 1\end{pmatrix}\right]$$
$$\mathrel{\widehat{=}} [\,\texttt{word\_11}, \texttt{word\_15}, \texttt{word\_16}, \texttt{word\_6}, \texttt{word\_22}\,]$$
$$\mathrel{\widehat{=}} \text{I love my cat}$$

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, \tau$

Hidden:

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

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

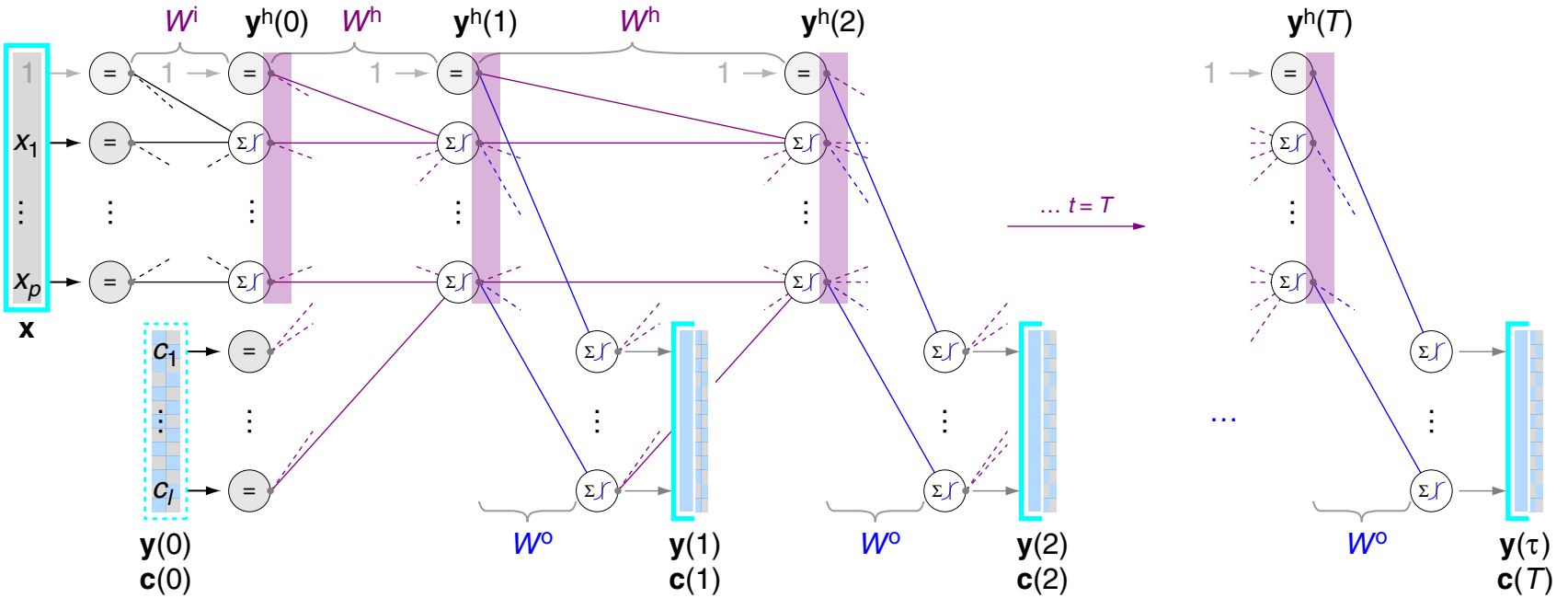
Target:

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

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

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, \tau$

Hidden:

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

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

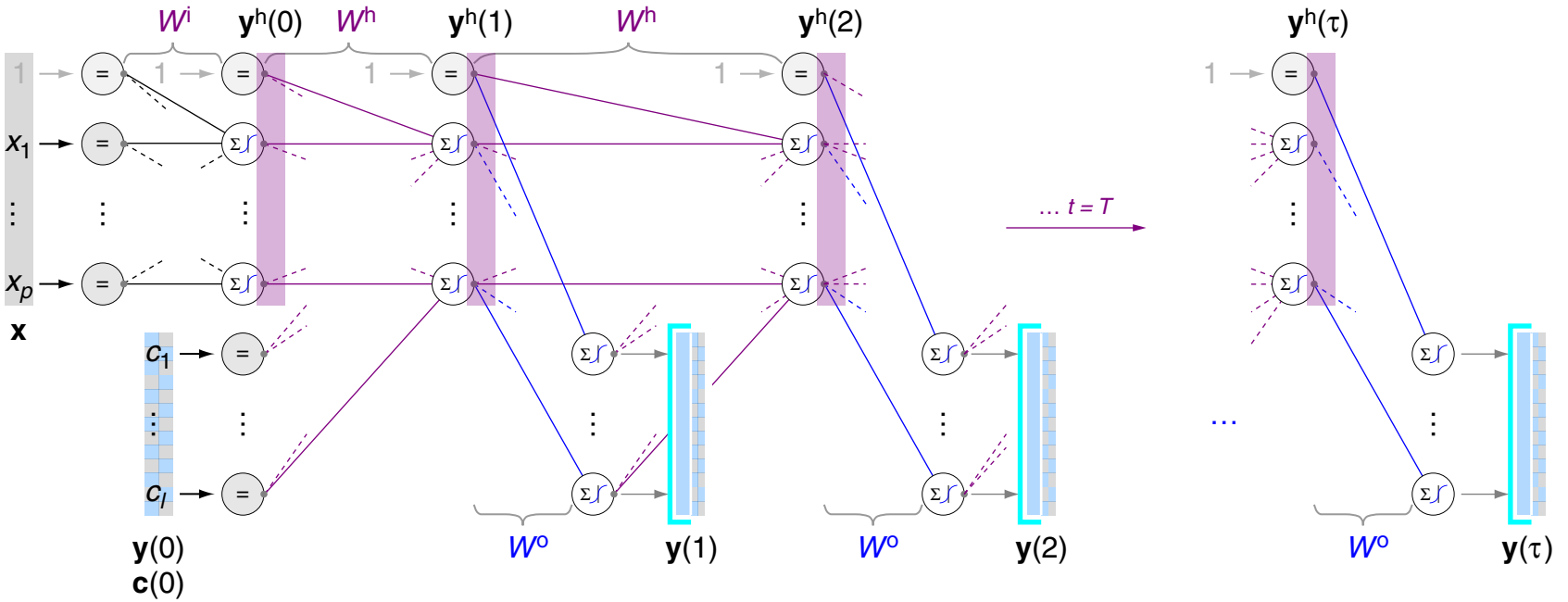
Target:

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

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

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



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:

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

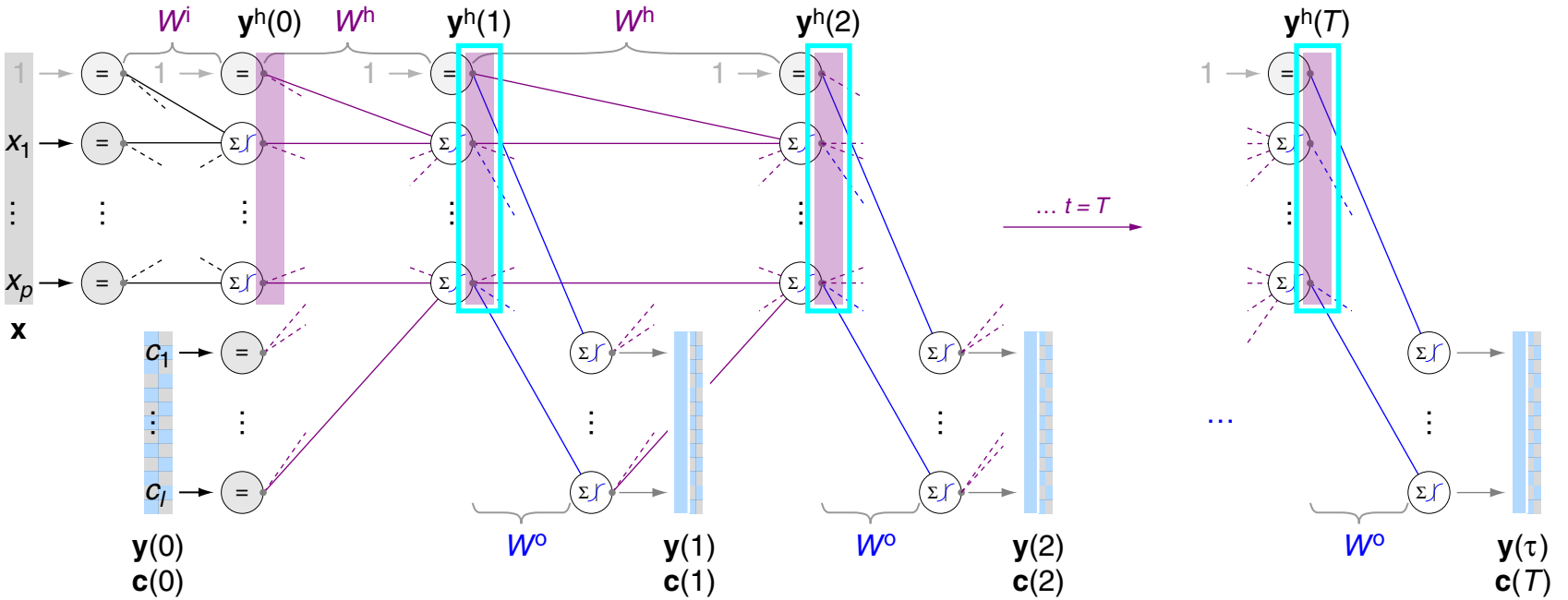
$\mathbf{c}(T) \mathrel{\widehat{=}} \texttt{<end>}$

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^o \mathbf{y}^h(t)\right), t = 1, \ldots, \tau$

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, \tau$

Hidden:

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

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

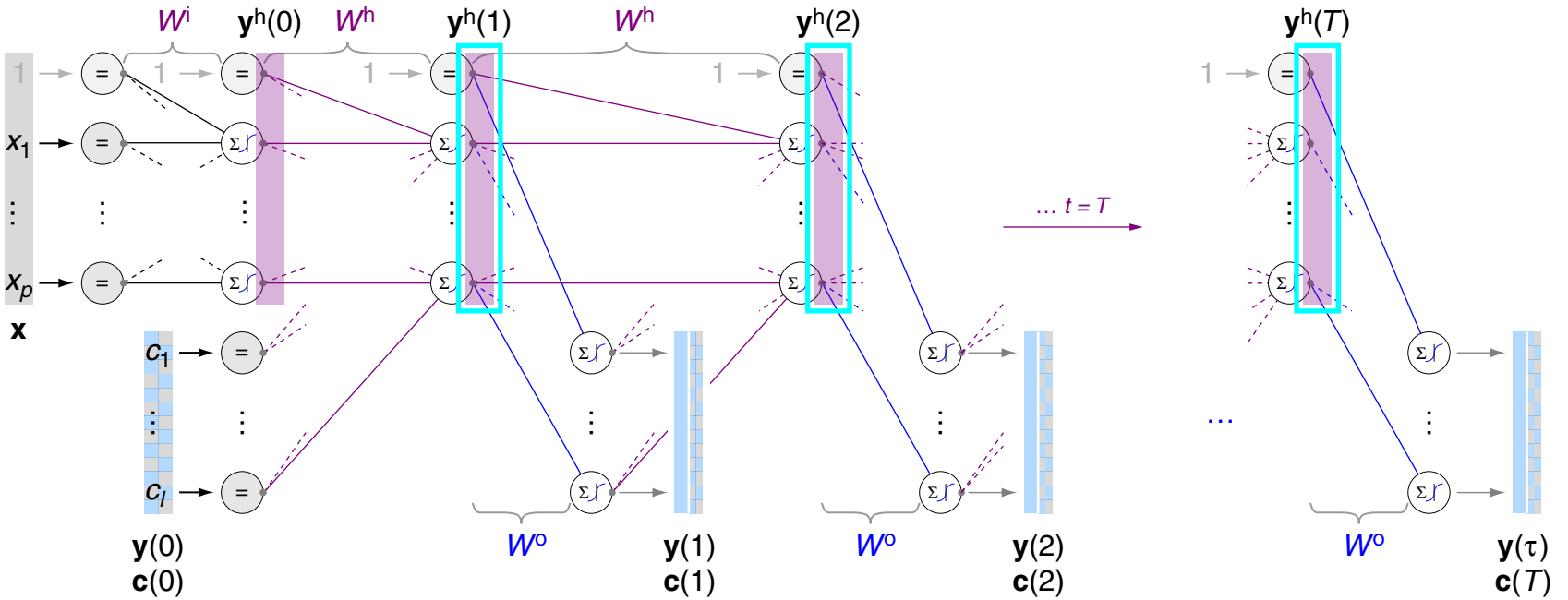
Target:

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

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

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, \tau$

Hidden:

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

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

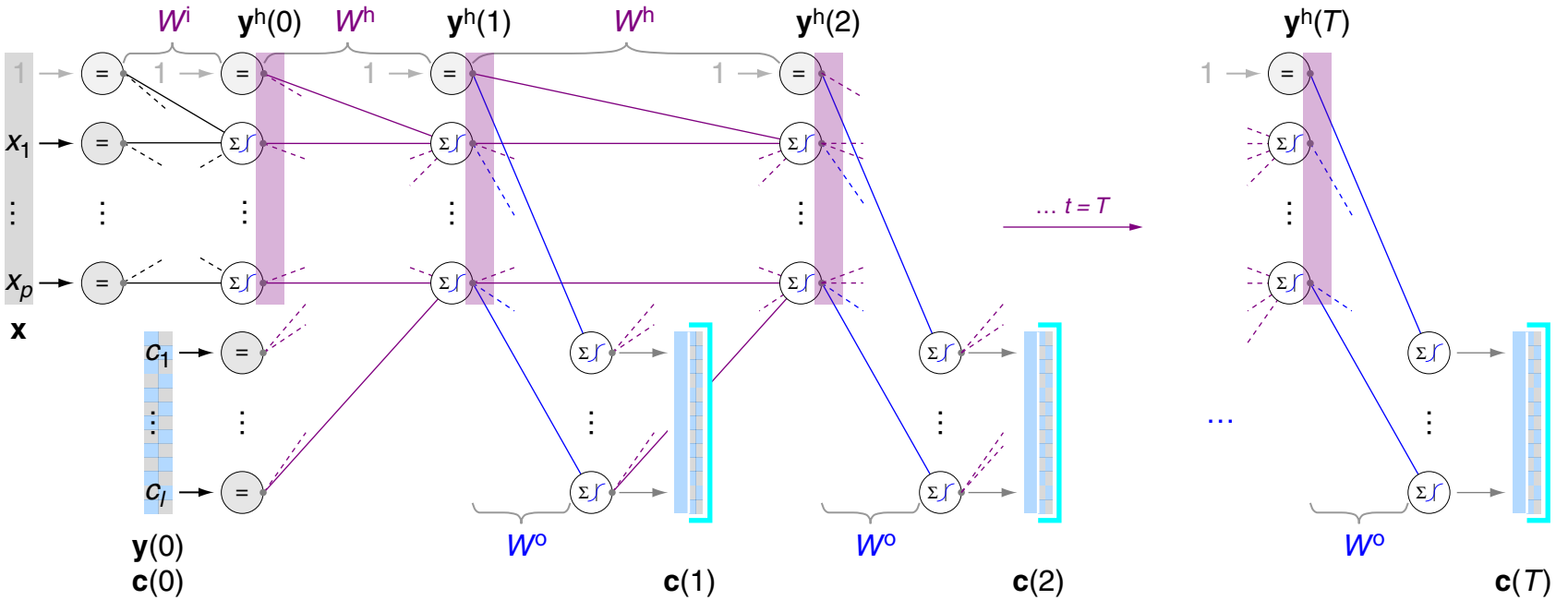
Target:

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

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

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, \tau$

Hidden:

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

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

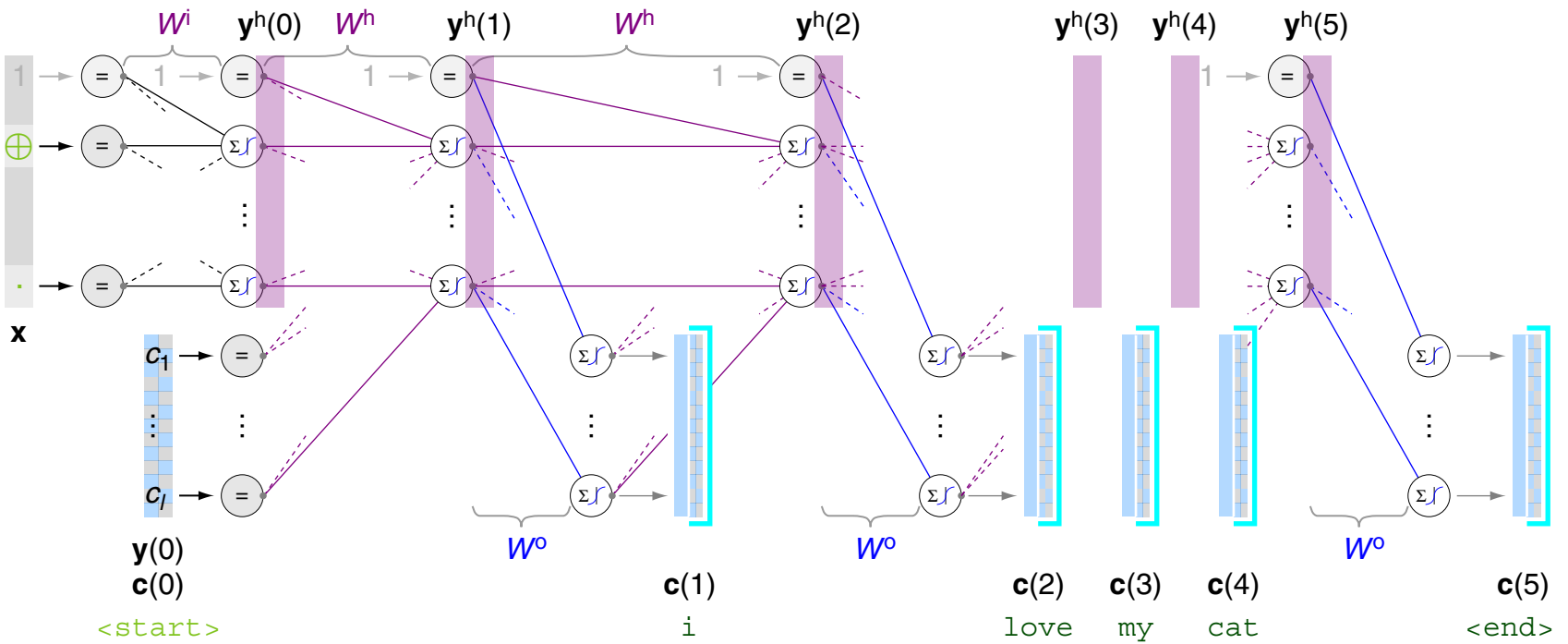
Target:

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

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

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, 5$

Hidden:

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

$\mathbf{y}^{\mathrm{h}}(t) = \boldsymbol{\sigma} \left(W^{\mathrm{h}} \begin{pmatrix} \mathbf{y}^{\mathrm{h}}(t-1) \\ \mathbf{c}(t-1) \end{pmatrix} \right), t = 1, \ldots, 5$

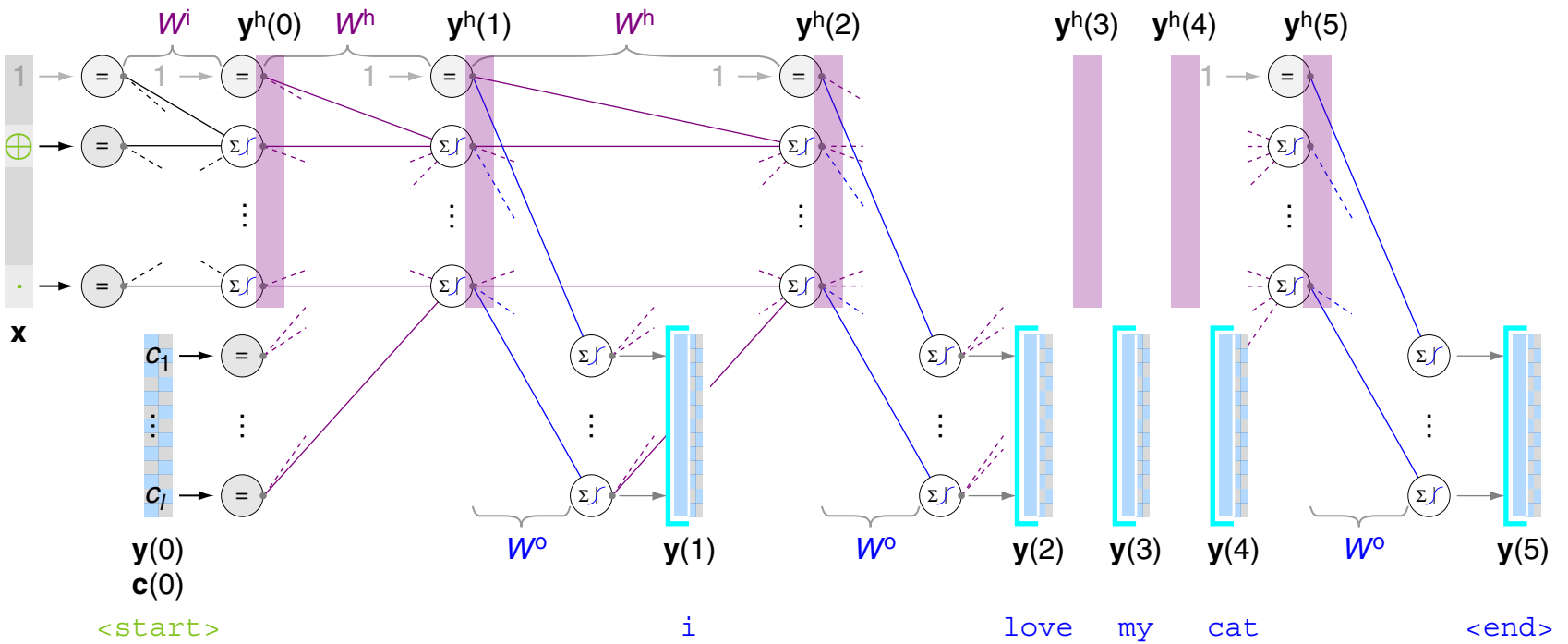
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(5)]$

$\mathbf{c}(5) \mathrel{\widehat{=}}$ <end>

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, 5$

Hidden:

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

$\mathbf{y}^{\mathrm{h}}(t) = \boldsymbol{\sigma} \left(W^{\mathrm{h}} \begin{pmatrix} \mathbf{y}^{\mathrm{h}}(t-1) \\ \mathbf{y}(t-1) \end{pmatrix} \right), t = 1, \ldots, 5$

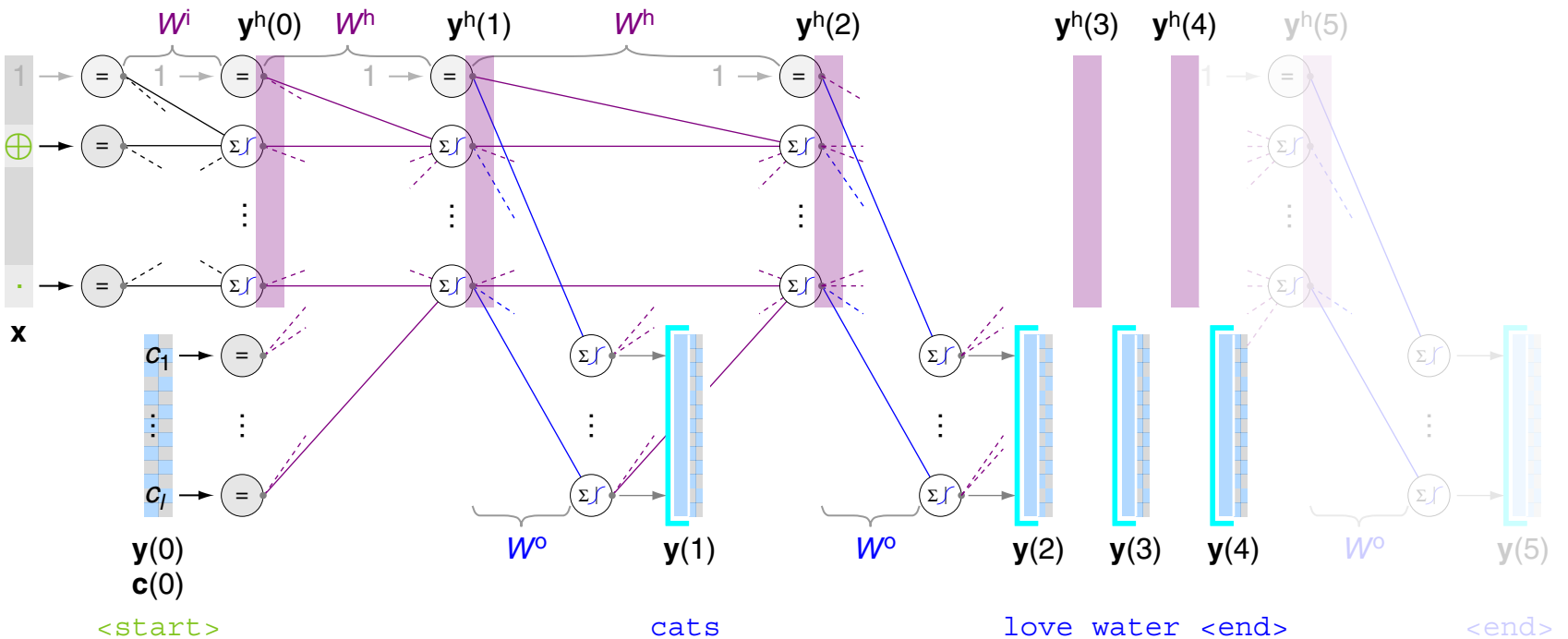
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(5)]$

$\mathbf{c}(5) \mathrel{\widehat{=}}$ <end>

Recurrent Neural Networks

(S2) Class-to-Sequence Mapping with RNNs (continued)



Input:

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

Output:

$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \, \mathbf{y}^{\mathrm{h}}(t) \right), t = 1, \ldots, 4$

Hidden:

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

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

Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(5)]$

$\mathbf{c}(5) \mathrel{\widehat{=}}$ <end>

Remarks:

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

- 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 $\mathbf{y}(t-1)$:

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

Recurrent Neural Networks

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

Recurrent Neural Networks

The IGD Algorithm for Class-to-Sequence Tasks (continued) [$\text{IGD}_{\text{seq2c}}$]

Algorithm:	$\text{IGD}_{\text{c2seq}}$	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:	$W^{\text{i}}, W^{\text{h}}, W^{\text{o}}$	Weights matrices. (= hypothesis)

1. *initialize_random_weights*$(W^{\text{i}}, W^{\text{h}}, W^{\text{o}})$, $t_{\text{training}} = 0$
2. **REPEAT**
3. $\quad t_{\text{training}} = t_{\text{training}} + 1$
4. $\quad$ FOREACH $(\mathbf{x}, [\mathbf{c}(1), \ldots, \mathbf{c}(T)]) \in D$ DO
5. $\quad\quad \mathbf{y}^{\text{h}}(0) = \boldsymbol{\sigma}\left(W^{\text{i}}\,\mathbf{x}\right)$
$\quad\quad$ FOR $t = 1$ TO T DO // forward propagation
$\quad\quad\quad \mathbf{y}^{\text{h}}(t) = \boldsymbol{\sigma}\left(W^{\text{h}}\begin{pmatrix}\mathbf{y}^{\text{h}}(t-1)\\ \mathbf{c}(t-1)\end{pmatrix}\right)$, $\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{\text{o}}\,\mathbf{y}^{\text{h}}(t)\right)$
$\quad\quad$ 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 $\neq \tau$
7. $\quad\quad$ Calculate $_{\vartriangle}W^{\text{i}}$, $_{\vartriangle}W^{\text{h}}$, $_{\vartriangle}W^{\text{o}}$ // backpropagation
8. $\quad\quad W^{\text{i}} = W^{\text{i}} +_{\vartriangle} W^{\text{i}}$, $W^{\text{h}} = W^{\text{h}} +_{\vartriangle} W^{\text{h}}$, $W^{\text{o}} = W^{\text{o}} +_{\vartriangle} W^{\text{o}}$
9. $\quad$ ENDDO
10. **UNTIL**(*convergence*$(D, \mathbf{y}(\,\cdot\,), t_{\text{training}})$)
11. *return*$(W^{\text{i}}, W^{\text{h}}, W^{\text{o}})$

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.