Chapter ML:IX (continued)

IX. Deep Learning

- Elements of Deep Learning
- Convolutional Neural Networks
- Autoencoder Networks

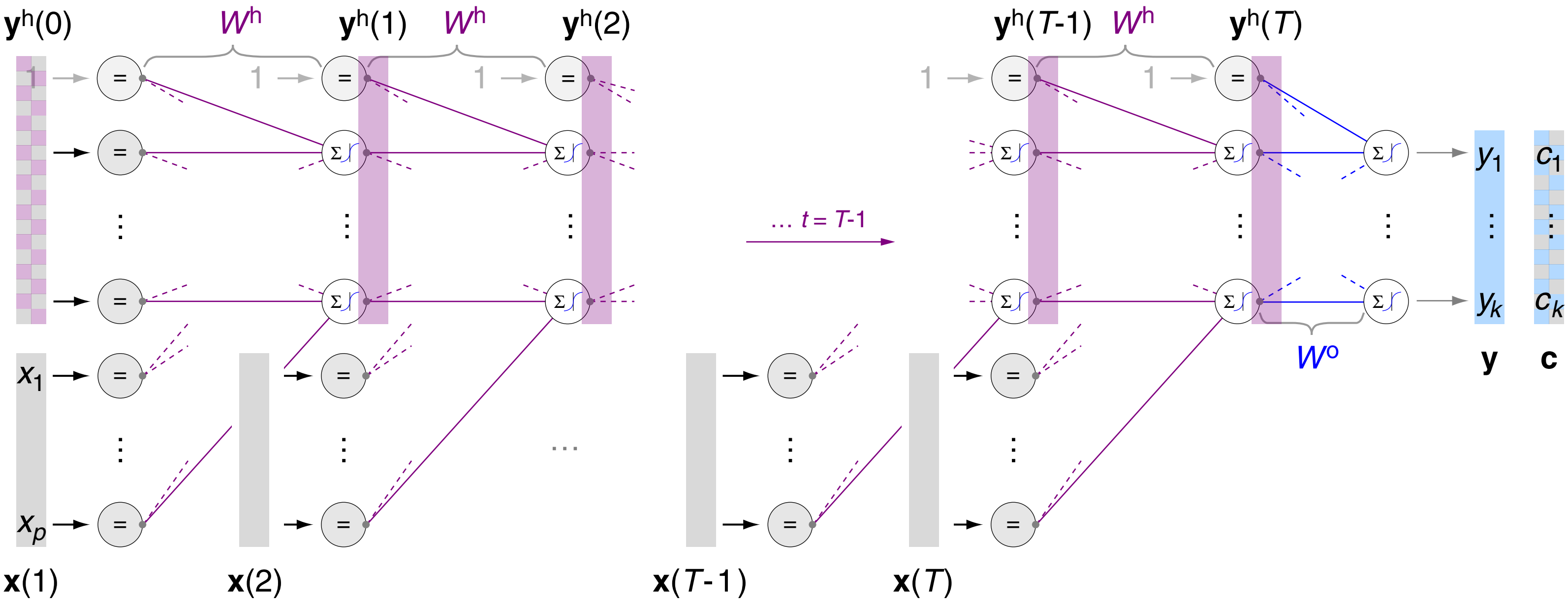
- Recurrent Neural Networks
- Long-Term Dependencies
- RNNs for Machine Translation
- Attention Mechanism
- Self Attention and Transformers

- Transformer Language Models

Long-Term Dependencies

Notation II (computational graph)

[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)

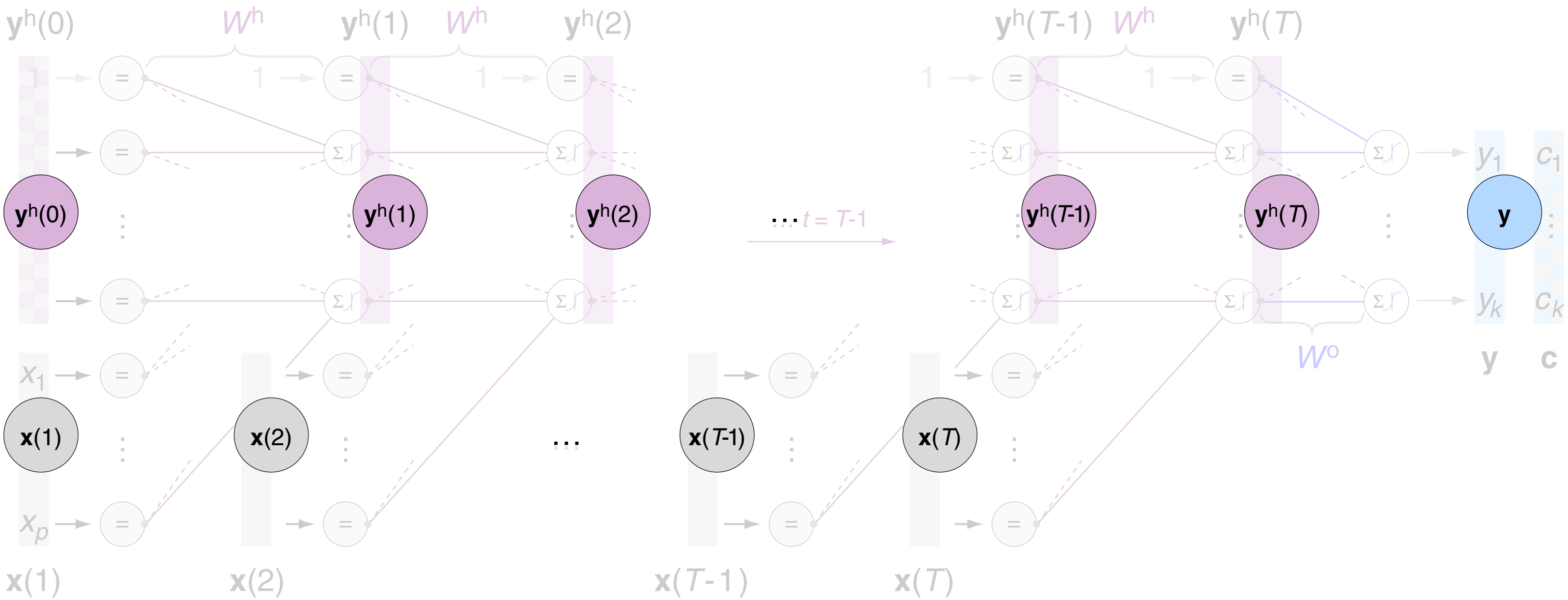
▨ $\mathbf{c}(t)$ target
`out_word` (training)

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

Long-Term Dependencies

Notation II (computational graph)

[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

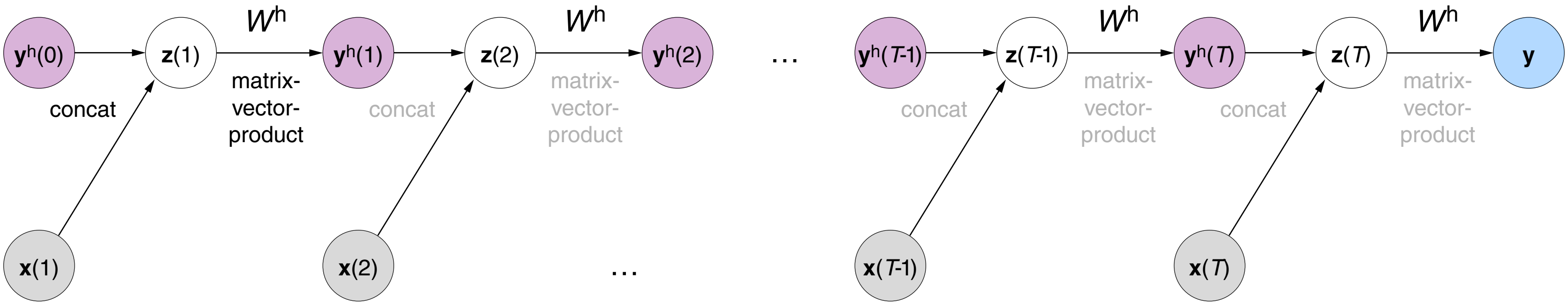
■ $\mathbf{y}(t)$ output
out_word (test)

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

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

Long-Term Dependencies

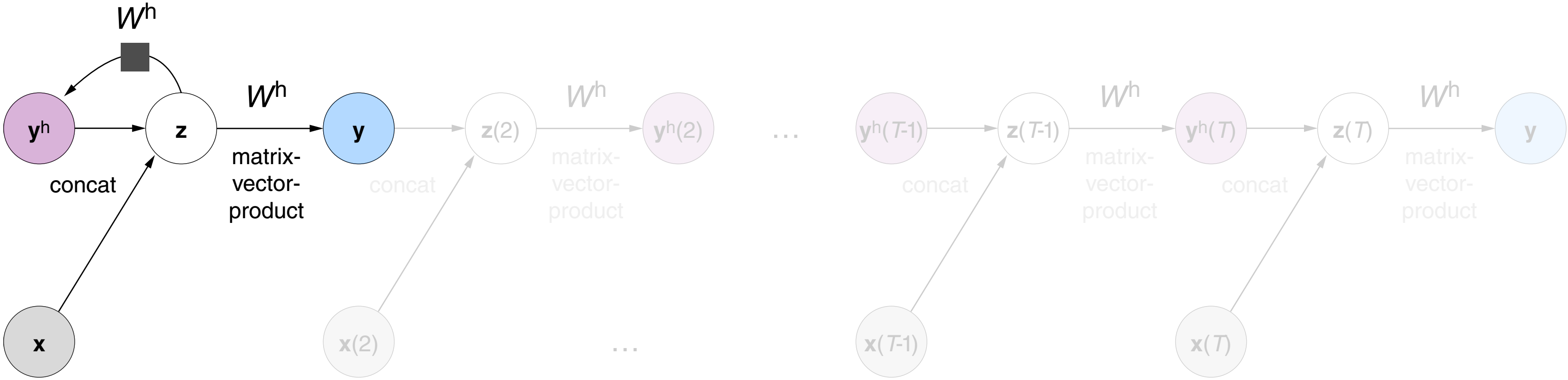
Notation II (computational graph) [notation: color, graph, language]



Long-Term Dependencies

Notation II (computational graph)

[notation: color, graph, language]



Remarks (computational graph) :

- The computational graph notation used here follows Goodfellow/Bengio/Courville 2016 :
  1. Each node in the graph indicates a variable. A variable may be a scalar, vector, matrix, tensor, or be of another type.
  2. An operation is a function of one or more variables. An operation returns a single output variable, which does not lose generality because the output variable can have multiple entries, such as a vector.

     If a variable b is computed by applying an operation to a variable a, a directed edge is drawn from a to b.

Long-Term Dependencies

Vanishing Gradient Problem

[*TODO*]

Long-Term Dependencies

RNN with Long Short-Term Memory (LSTM)

[*TODO*]

Remarks:

- LSTM is a recurrent neural network architecture that is very efficient at remembering long term dependencies and that is less vulnerable to the vanishing gradient problem.

Long-Term Dependencies

RNN with Gated Recurrent Units (GRU)

[*TODO*]