Chapter ML:IX (continued)

IX. Deep Learning

- Elements of Deep Learning
- Convolutional Neural Networks
- Autoencoder Networks
- Recurrent Neural Networks
- RNNs for Machine Translation
- Vanishing Gradient Problem
- Self Attention and Transformers
- Transformer Language Models

Vanishing Gradient Problem

Vanishing Gradient Illustration

Vanishing Gradient Problem

RNN with Long Short-Term Memory (LSTM)

[*SKIPPED*]

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.

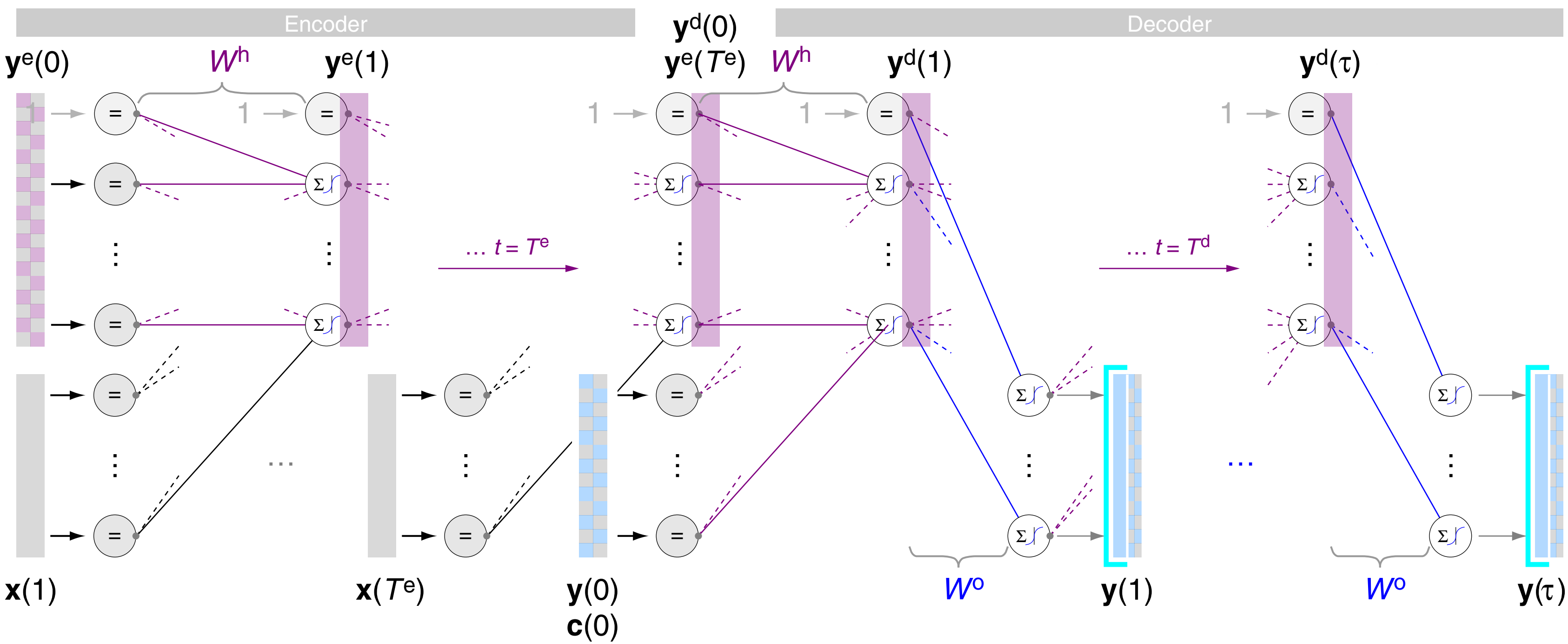
Vanishing Gradient Problem

RNN with Gated Recurrent Units (GRU)

[*SKIPPED*]

Vanishing Gradient Problem

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}\left(W^{\mathsf{o}}\,\mathbf{y}^{\mathsf{d}}(t)\right)$$

Attention:

$$\mathbf{y}^{\mathsf{a}}(t) = \left[\mathbf{y}^{\mathsf{e}}(1), \ldots, \mathbf{y}^{\mathsf{e}}(T^{\mathsf{e}})\right]\mathbf{a}(t),\ t = 1, \ldots, T^d$$

$$\mathbf{a}(t) = \boldsymbol{\sigma}_1\left(\left[\mathbf{y}^{\mathsf{e}}(1), \ldots, \mathbf{y}^{\mathsf{e}}(T^{\mathsf{e}})\right]^T \mathbf{y}^{\mathsf{d}}(t)\right)$$

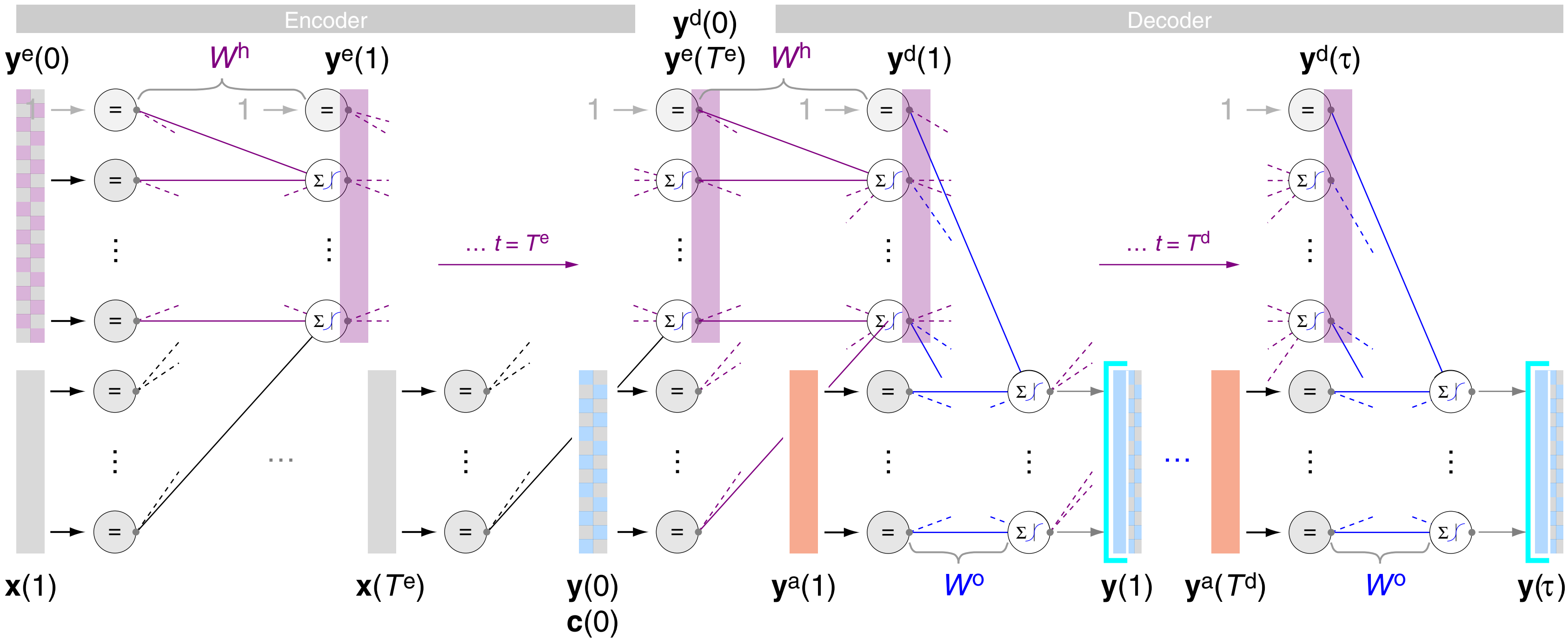
Hidden:

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

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

Vanishing Gradient Problem

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}\left(W^o\begin{pmatrix}\mathbf{y}^d(t)\\ \mathbf{y}^a(t)\end{pmatrix}\right)$$

Attention:

$$\mathbf{y}^a(t) = \left[\mathbf{y}^e(1), \ldots, \mathbf{y}^e(T^e)\right]\mathbf{a}(t),\ t = 1, \ldots, T^d$$

$$\mathbf{a}(t) = \boldsymbol{\sigma}_1\left(\left[\mathbf{y}^e(1), \ldots, \mathbf{y}^e(T^e)\right]^T \mathbf{y}^d(t)\right)$$

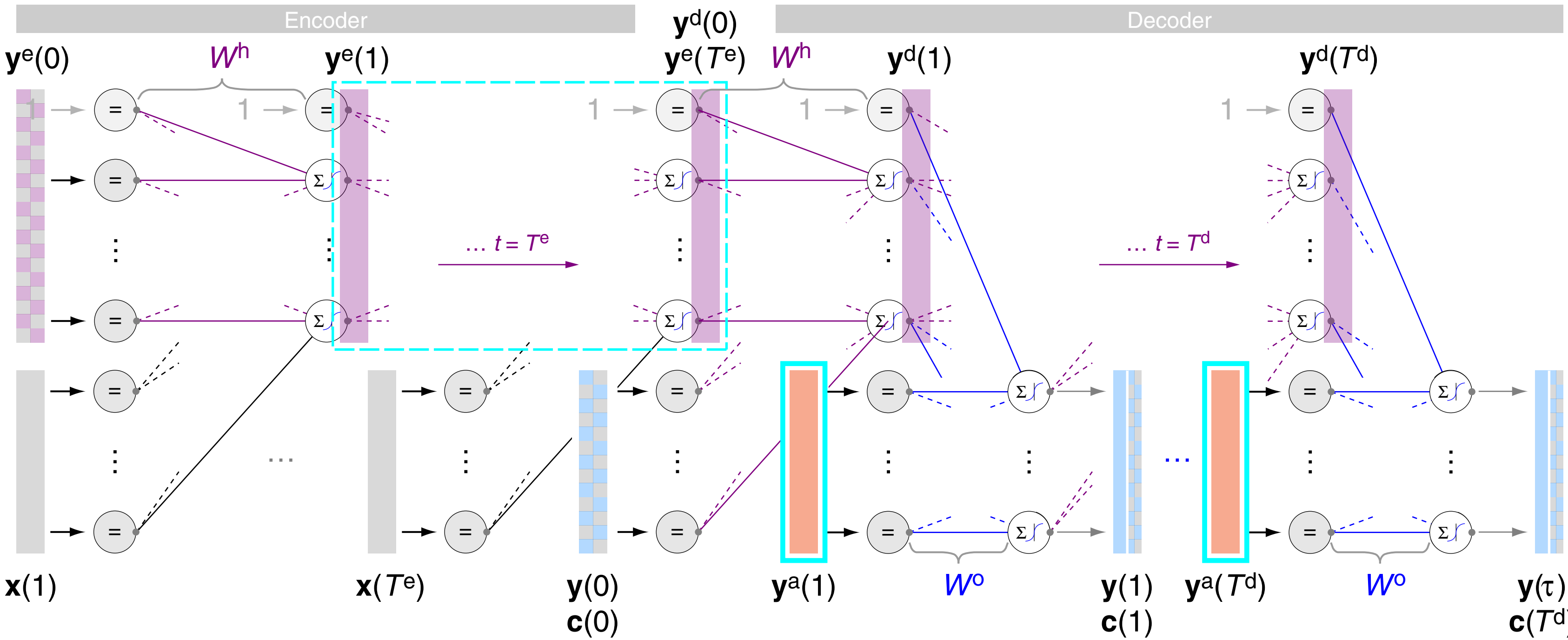
Hidden:

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

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

Vanishing Gradient Problem

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}\left(W^{\mathrm{o}}\begin{pmatrix}\mathbf{y}^{\mathrm{d}}(t)\\ \mathbf{y}^{\mathrm{a}}(t)\end{pmatrix}\right)$$

Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1),\ldots,\mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]\mathbf{a}(t),\ t = 1,\ldots,T^d$$

$$\mathbf{a}(t) = \boldsymbol{\sigma}_1\left(\left[\mathbf{y}^{\mathrm{e}}(1),\ldots,\mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]^T \mathbf{y}^{\mathrm{d}}(t)\right)$$

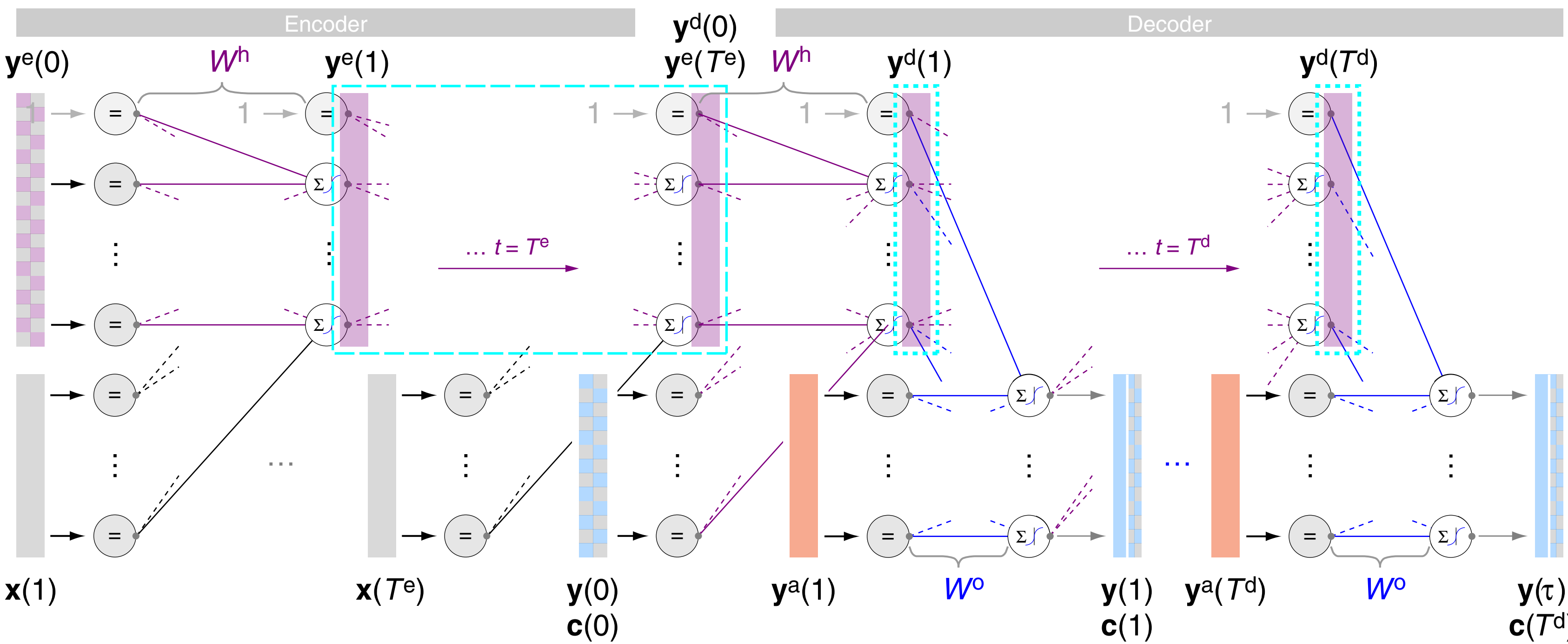
Hidden:

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

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

Vanishing Gradient Problem

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}\left(W^{\mathrm{o}}\begin{pmatrix}\mathbf{y}^{\mathrm{d}}(t)\\ \mathbf{y}^{\mathrm{a}}(t)\end{pmatrix}\right)$$

Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]\mathbf{a}(t),\ t = 1, \ldots, T^d$$

$$\mathbf{a}(t) = \boldsymbol{\sigma}_1\left(\left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]^T \mathbf{y}^{\mathrm{d}}(t)\right)$$

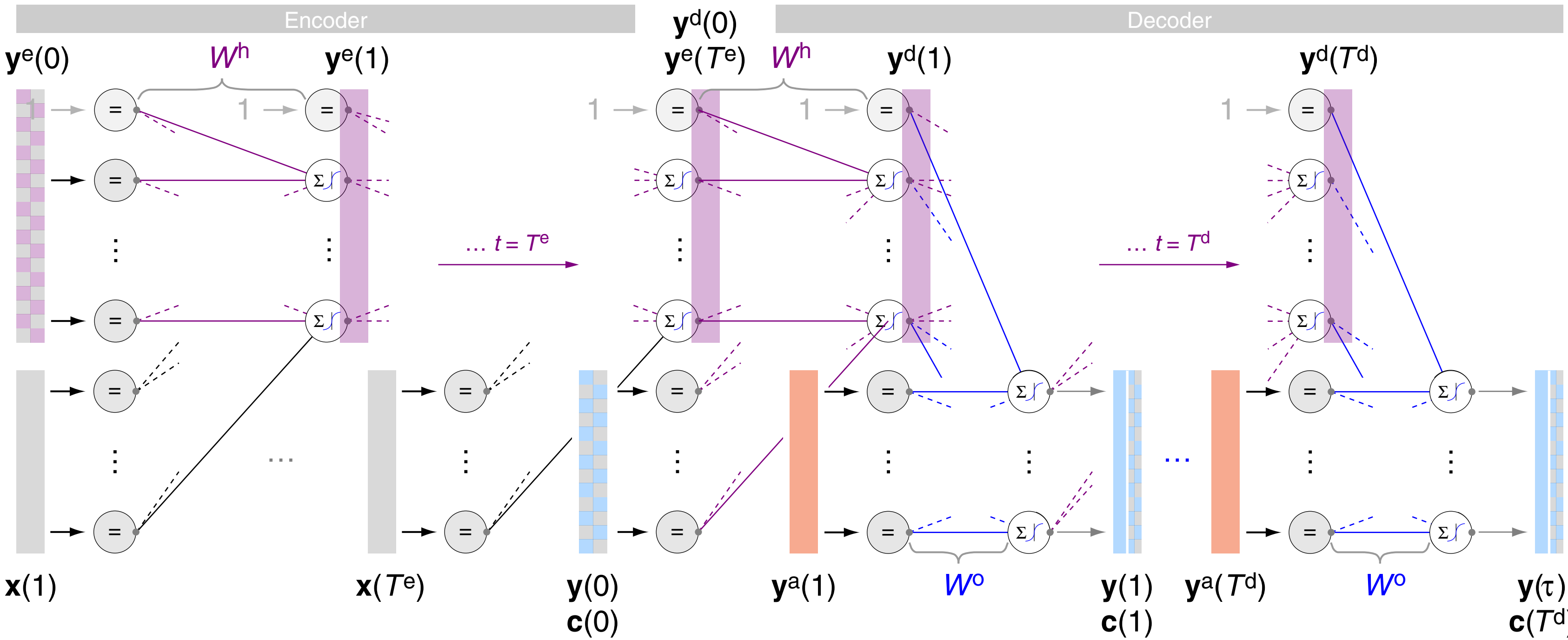
Hidden:

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

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

Vanishing Gradient Problem

RNN with Parameterized Attention



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}\left(W^o \begin{pmatrix}\mathbf{y}^d(t)\\ \mathbf{y}^a(t)\end{pmatrix}\right)$$

Attention:

$$\mathbf{y}^a(t) = \left(W^V\left[\mathbf{y}^e(1), \ldots, \mathbf{y}^e(T^e)\right]\right)\mathbf{a}(t)$$

$$\mathbf{a}(t) = \boldsymbol{\sigma}_1\left(\left(W^K\left[\mathbf{y}^e(1), \ldots, \mathbf{y}^e(T^e)\right]\right)^T \left(W^Q\mathbf{y}^d(t)\right)\right)$$

Hidden:

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

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

Remarks (attention calculus) :

- $\boldsymbol{\sigma}_{\mathbf{1}}()$ denotes the softmax function.
- The ith component $a_i(t)$ of the "attention score vector" $\mathbf{a}(t)$, $i = 1, \ldots, T^{\mathsf{e}}$, models the importance of the ith *encoder* hidden state $\mathbf{y}^{\mathsf{e}}(i)$ for the *decoder* hidden state $\mathbf{y}^{\mathsf{d}}(t)$: $a_i(t)$ is the scalar product of $\mathbf{y}^{\mathsf{e}}(i)$ and $\mathbf{y}^{\mathsf{d}}(t)$ (do not overlook the matrix transpose operation).
- $\mathbf{y}^{\mathsf{a}}(t)$ is the result of combining the encoder hidden state sequence $[\mathbf{y}^{\mathsf{e}}(1), \ldots, \mathbf{y}^{\mathsf{e}}(T^{\mathsf{e}})]$ with the attention score vector $\mathbf{a}(t)$. I.e., each vector $\mathbf{y}^{\mathsf{e}}(i)$ is considered as a "value" that is weighted with the importance stored in the respective dimension (= time step) of $\mathbf{a}(t)$. $\mathbf{y}^{\mathsf{a}}(t)$ is called attention [vector] for output vector $\mathbf{y}(t)$ since it helps to pay attention to the most influential input states for $\mathbf{y}(t)$.
- Consider $\mathbf{a}(t)$. The scalar product of $\mathbf{y}^{\mathsf{e}}(i)$ and $\mathbf{y}^{\mathsf{d}}(t)$ becomes maximum if $\mathbf{y}^{\mathsf{e}}(i)$ and $\mathbf{y}^{\mathsf{d}}(t)$ are identical. The distribution of the T^{e} weights of $\mathbf{a}(t)$ reflects the distribution of absolute values among the $\mathbf{y}^{\mathsf{e}}$.

  Consider $\mathbf{y}^{\mathsf{a}}(t)$. If some $\mathbf{y}^{\mathsf{e}}(i)$ has a high absolute value (compared to the other $\mathbf{y}^{\mathsf{e}}$) and if it has the same direction as $\mathbf{y}^{\mathsf{d}}(t)$, it will push the weight of the ith dimension of $\mathbf{y}^{\mathsf{a}}(t)$ towards 1 (and the others towards zero). In the extreme case, the ith *encoder* state, $\mathbf{y}^{\mathsf{e}}(i)$, is used along with the tth *decoder* state, $\mathbf{y}^{\mathsf{d}}(t)$, as input for W^{o}, say, $\mathbf{y}^{\mathsf{e}}(i)$ is passed directly to position t of the output sequence.

Remarks (parameterized attention) :

- $\mathbf{y}^{\mathsf{d}}(t)$ is also denoted as “query,” while the sequence of $\mathbf{y}^{\mathsf{e}}(i)$ are denoted as “keys” in this setting. Since $\mathbf{a}(t)$ is normalized with a softmax operation it represents the importance of the T^{e} time steps as probabilities.
- *Parameterized* attention introduces three weight matrices, W^{Q}, W^{K}, and W^{V}, in order to learn a more sophisticated version of the simple attention vector $\mathbf{y}^{\mathsf{a}}(t)$. In this regard the matrices are called “query projection”, “key projection”, and “value projection” [matrix] respectively.
- Inspired by nature, the structure of the model function $\mathbf{y}()$ has been developed in the form of a network that connects the matrices W^{h}, W^{o}, W^{Q}, W^{K}, and W^{V} in a particular manner. Note that the shown model function is specified completely by the set of parameters $\mathbf{w}$, which are organized in the aforementioned matrices.