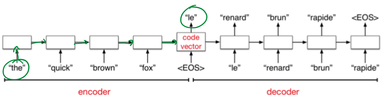
Video 33

Sunday, November 29, 2020 8:36 PM



long temporal gap

⇒ Difficult to learn long-distance dependencies

Recall backpropagation through time:

Assume $\overline{h^{(T)}}$ has already been computed (by backpropagating through the decoder)

Encoder backpropagation rules:

from video on backpropagation through time

$$\begin{cases} \overline{h^{(t)}} = \overline{z^{(t+1)}}\, w \\ \overline{z^{(t)}} = \overline{h^{(t)}}\, \phi'\left(z^{(t)}\right) \end{cases}$$

By iteration:

$$\begin{aligned} \overline{h^{(1)}} = \overline{z^{(2)}}\, w &= \overline{h^{(2)}}\, \phi'\left(z^{(2)}\right) w \\ &= \overline{z^{(3)}}\, w\, \phi'\left(z^{(2)}\right) w \\ &= \overline{h^{(3)}}\, \phi'\left(z^{(3)}\right) \phi'\left(z^{(2)}\right) w^2 \\ &\;\;\vdots \\ &= w^{T-1} \phi'\left(z^{(2)}\right) \cdots \phi'\left(z^{(T)}\right) \overline{h^{(T)}} \end{aligned}$$

T = # of time steps

$\frac{\partial h^{(T)}}{\partial h^{(1)}}$

computed by backpropagating through the decoder.

For simplicity, if ϕ is linear: $\frac{\partial h^{(T)}}{\partial h^{(1)}} = w^{T-1}$

if w is not close to 1, w^{T-1} will either explode or vanish

$|w| > 1$

if $\begin{cases} w = -1.1 \\ T = 50 \end{cases} \Rightarrow w^{T-1} = 117.4$

$0 < |w| < 1$

if $\begin{cases} w = 0.9 \\ T = 50 \end{cases} \Rightarrow w^{T-1} = 0.0515$

In the multivariate case:

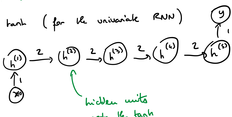
$$\frac{\partial \vec{h}^{(T)}}{\partial \vec{h}^{(1)}} = \frac{\partial \vec{h}^{(T)}}{\partial \vec{h}^{(T-1)}} \cdots \frac{\partial \vec{h}^{(2)}}{\partial \vec{h}^{(1)}} = W^{T-1}$$

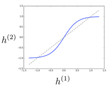
matrix

assume that activation is linear

Exploding / vanishing gradients

Example with a tanh (for the univariate RNN)





One application of the tanh

Five applications of the tanh

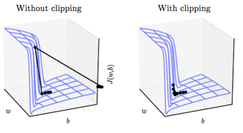
Solution 1: Gradient clipping

Prevent gradients from exploding by rescaling them:

$$\vec{g} \leftarrow \eta \frac{\vec{g}}{\|\vec{g}\|}$$

constant used to rescale gradients

Biases training procedure but can make it more stable.



— Goodfellow et al., *Deep Learning*

Solution 2: Long-Term Short-Term Memory (LSTM)

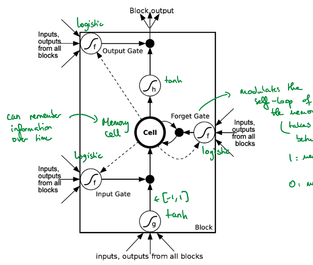
Goal: remember information over long time periods.

long-term memory: weights

short-term memory: activation

Architecture uses an LSTM unit which is called a block

Each hidden layer of an LSTM architecture is made up of many of these blocks



Input gate	Forget gate	Behavior of the memory cell
0	1	remember the previous value
1	1	add to previous value
0	0	erase the value
1	0	overwrite the value.

Implementation ⟶ values of { the memory cell / the different gates

input gates of all blocks ⟶
forget gates ⟶
output gates ⟶
weighted inputs of all blocks ⟶

$$\begin{pmatrix} \vec{i}^{(t)} \\ \vec{f}^{(t)} \\ \vec{o}^{(t)} \\ \vec{g}^{(t)} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} \vec{x}^{(t)} \\ \vec{h}^{(t-1)} \end{pmatrix}$$

input to the blocks

previous time step hidden state

each of the gates and the weighted input computations involve a set of independent weights

memory cells for all blocks

$$\vec{c}^{(t)} = \vec{f}^{(t)} \cdot \vec{c}^{(t-1)} + \vec{i}^{(t)} \cdot \vec{g}^{(t)}$$

$$\vec{h}^{(t)} = \vec{o}^{(t)} \cdot \tanh\left(\vec{c}^{(t)}\right)$$