

5. Sequence modeling with recurrent neural networks

5.2a. Training an RNN. Recursive gradients

Manel Martínez-Ramón
Meenu Ajith
Aswathy Rajendra Kurup

RNN training

The cost function

- Assume an RNN designed to classify among K classes
- A training sequence $\mathbf{x}_t \in \mathbb{R}^D$, $\mathbf{y}_t \in \mathbb{R}^K$, $1 \leq t \leq T$ is available.
- We must then maximize the cross entropy between the labels and the outputs or, equivalently, the output likelihood:

$$J_{ML}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{Y}) = - \sum_{t=1}^T \ell(\mathbf{x}_t) = - \sum_{t=1}^T \sum_{k=0}^{K-1} y_{k,t} \log \text{softmax} \left(z_{k,t}^{(o)} \right) \quad (1)$$

where

- $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_T]$ and $\mathbf{Y} = [\mathbf{y}_1 \cdots \mathbf{y}_T]$.
- $\boldsymbol{\theta} = \{\mathbf{W}_{hx}, \mathbf{W}_{hh}, \mathbf{W}_{oh}\}$ contains all trainable parameters.

RNN training

The Jacobian with respect to \mathbf{h}_t

- We must consider that the hidden state \mathbf{h}_t at every instant depends on the states $\mathbf{h}_{t'}, t' < t$.
- During the backpropagation $\nabla_{\mathbf{h}_T} J_{ML}$ appears. For $t = T$,

$$\nabla_{\mathbf{h}_T} J_{ML} = \frac{\delta \mathbf{z}_T^{(o)}}{\delta \mathbf{h}_T} \nabla_{\mathbf{z}_T^{(o)}} J_{ML} \quad (2)$$

where $\nabla_{\mathbf{z}_T^{(o)}} J_{ML}$ is the output error:

$$\frac{dJ_{ML}}{dz_{k,T}^{(o)}} = \frac{dJ_{ML}}{do_{k,T}} \frac{o_{k,t}}{dz_{k,T}^{(o)}} = \frac{dJ_{ML}}{do_{k,T}} o'_{k,T} = \delta_{k,T} \quad (3)$$

- Since the output of the RNN is a softmax, $\delta_t = \text{softmax}(\mathbf{z}_t^{(o)}) - \mathbf{y}_t$

RNN training

The Jacobian with respect to \mathbf{h}_t

- For $t < T$, the cost function in Eq. (1) contains \mathbf{h}_t in element $\ell(\mathbf{x}_t)$ and in the next one, $\ell(\mathbf{x}_{t+1})$ since

$$\mathbf{h}_{t+1} = \tanh \left(\mathbf{z}_{t+1}^{(x)} \right) = \tanh \left(\mathbf{W}_{hx}^\top \mathbf{x}_{t+1} + \mathbf{W}_{hh}^\top \mathbf{h}_t + \mathbf{b}_h \right) \quad (4)$$

- Jacobian $\frac{\delta \mathbf{h}_{t+1}}{\delta \mathbf{h}_t}$ will appear in the chain rule, with elements $\frac{\delta h_{j,t+1}}{\delta h_{i,t}}$.
By applying the chain rule of calculus to them

$$\frac{\delta h_{j,t+1}}{\delta h_{i,t}} = \frac{\delta h_{j,t+1}}{\delta z_{j,t+1}^{(x)}} \frac{\delta z_{j,t+1}^{(x)}}{\delta h_{i,t}} = \frac{\delta h_{j,t+1}}{\delta z_{j,t+1}^{(x)}} \frac{\delta \mathbf{w}_{j,hh}^\top \mathbf{h}_t}{\delta h_{i,t}} \quad (5)$$

RNN training

The Jacobian with respect to \mathbf{h}_t

- The first derivative of the right side of expression (5) is the derivative of the hyperbolic tangent. The second derivative is parameter $w_{i,j,hh}$.

- Therefore

$$\frac{\delta h_{j,t+1}}{\delta h_{i,t}} = w_{i,j,hh} \tanh' \left(z_{j,t+1}^{(x)} \right) \quad (6)$$

- Thus, Jacobian $\frac{\delta \mathbf{h}_{t+1}}{\delta \mathbf{h}_t}$ is

$$\frac{\delta \mathbf{h}_{t+1}}{\delta \mathbf{h}_t} = \mathbf{W}_{hh} \text{diag} \left(\tanh' \left(\mathbf{z}_{t+1}^{(x)} \right) \right) \quad (7)$$

RNN training

The Jacobian with respect to \mathbf{h}_t

- With all these elements, the gradient of the cost function with respect to hidden state \mathbf{h}_t is

$$\begin{aligned}\nabla_{\mathbf{h}_t} J_{ML} &= \frac{\delta \mathbf{z}_t^{(o)}}{\delta \mathbf{h}_t} \nabla_{\mathbf{z}_t^{(o)}} J_{ML} + \frac{\delta \mathbf{h}_{t+1}}{\delta \mathbf{h}_t} \nabla_{\mathbf{h}_{t+1}} J_{ML} \\ &= \mathbf{W}_{oh} \delta_t + \mathbf{W}_{hh} \text{diag} \left(\tanh' \left(\mathbf{z}_{t+1}^{(x)} \right) \right) (\nabla_{\mathbf{h}_{t+1}} J_{ML})\end{aligned}\tag{8}$$

- This is a recursive equation.
- The first term of of Eq. 8 is the the error backpropagated from the output at instant t through the output weights \mathbf{W}_{oh} .
- The error backpropagated from the next time instant is the second term, which contains the output error at instant $t + 1$ backpropagated to the network at instant t through the hidden weights \mathbf{W}_{hh} .

RNN training

Idea of the backpropagation

