9.1. Gated Recurrent Units

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

An early observation could be highly significant for all future observations.

Example

First observation contains a checksum and the goal is to discern whether the checksum is correct at the end of the sequence.

- First token is vital.
- We will have to assign a very large gradient to this observation, since it affects all subsequent observations.

We would like to have some mechanism for storing vital early information in a **memory cell**.

Some symbols carry no pertinent observation.

Example

When parsing a webpage, there is auxiliary HTML code that is irrelevant for assessing the page sentiment.

We would like to have a mechanism for **skipping such symbols** in the latent state representation.

A logical break between parts of a sequence.

Example

- a transition between chapters in a book
- a transition between a bear and a bull market for securities

We would like to have have a means of **resetting** our internal state representation.

These are provided by

- Long Short Term Memory (LSTM) of Hochreiter and Schmidhuber, 1997
- The Gated Recurrent Unit (GRU) of Cho et al., 2014 is a slightly more streamlined variant that often offers comparable performance and is significantly faster to compute.[1]

[1] See also Chung et al., 2014 for more details.

GRUs support gating of the hidden state.
They have mechanisms

- when the hidden state should be updated
- when the hidden state should be reset.

The mechanisms are learned

- If the first symbol is important, we will learn not to update the hidden state after the first observation.
- We will learn to skip irrelevant temporary observations.
- We will learn to reset the latent state whenever needed.

Reset and **update gates** are vectors with entries in $(0, 1)$. They enable **convex combinations**, e.g. of a hidden state and an alternative.

For instance,

- a reset variable would allow us to control how much of the previous state we might still want to remember.
- an update variable would allow us to control how much of the new state is just a copy of the old state.

The **inputs** for both reset and update gates in a GRU are

- the current input $\mathbf{X}_t$
- the hidden state of the previous step $\mathbf{H}_{t-1}$.

The **output** is given by a fully connected layer with a sigmoid activation.

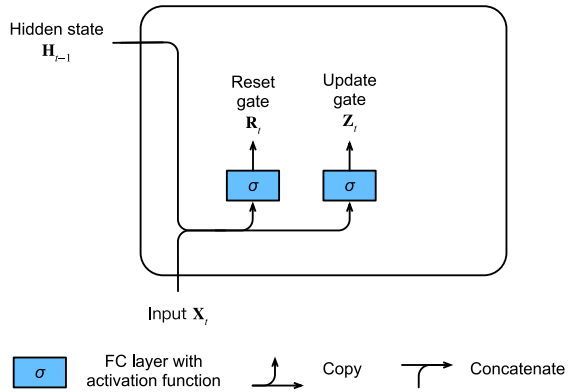

Figure: Reset and update gate in a GRU.

- the mini-batch input is $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ (examples: n, inputs: d)
- the hidden state at $t-1$ is $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$.
- The **reset** gate $\mathbf{R}_t \in \mathbb{R}^{n \times h}$:

$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \mathbf{b}_r)$$

- The **update** gate $\mathbf{Z}_t \in \mathbb{R}^{n \times h}$:

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \mathbf{b}_z)$$

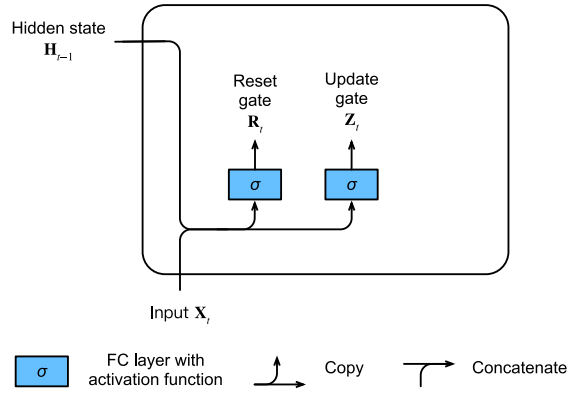

Figure: Reset and update gate in a GRU.

- $\mathbf{W}_{xr}, \mathbf{W}_{xz} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}_{hr}, \mathbf{W}_{hz} \in \mathbb{R}^{h \times h}$ are weight parameters.
- $\mathbf{b}_r, \mathbf{b}_z \in \mathbb{R}^{1 \times h}$ are biases.
- We use a sigmoid function to transform values to the interval $(0, 1)$.

In a conventional **deep RNN** we would have

$$\mathbf{H}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h).$$

In the **GRU**, the **candidate** hidden state $\tilde{\mathbf{H}}_t$ is

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h)$$

- Elementwise multiplication of $\mathbf{H}_{t-1}$ with $\mathbf{R}_t$ reduces the influence of previous states.
- If $\mathbf{R}_t \approx 1$ we recover a conventional deep RNN.
- If $\mathbf{R}_t \approx 0$, $\mathbf{H}_t$ is the result of an MLP with $\mathbf{X}_t$ as input.

  Any pre-existing hidden state is thus 'reset'.

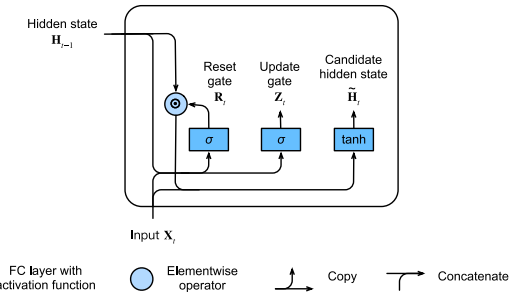

Figure: Candidate hidden state computation in a GRU. The multiplication is carried out elementwise.

$\odot$ indicates pointwise multiplication between tensors.

The **update** gate determines the extent to which the new state $\mathbf{H}_t$ is just the old state $\mathbf{H}_{t-1}$ and by how much the new candidate state $\tilde{\mathbf{H}}_t$ is used.

The gating variable $\mathbf{Z}_t$ takes the elementwise convex combinations between both candidates.

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t.$$

- If $\mathbf{Z}_t \approx 1$: we retain the old state.

  The information from $\mathbf{X}_t$ is essentially ignored, effectively skipping time step t in the dependency chain.
- If $\mathbf{Z}_t \approx 0$: the new latent state $\mathbf{H}_t$ approaches the candidate latent state $\tilde{\mathbf{H}}_t$.

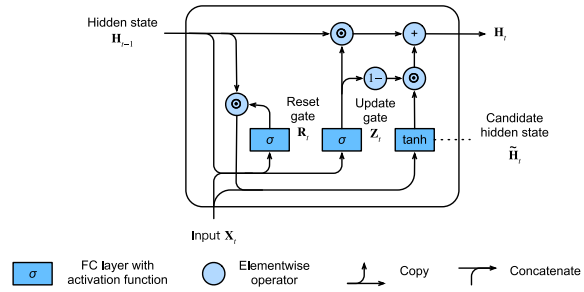

Figure: Hidden state computation in a GRU. As before, the multiplication is carried out elementwise.

$\odot$ indicates pointwise multiplication between tensors.

Summary

- Gated recurrent neural networks are better at capturing dependencies for time series with large time step distances.
- Reset gates help capture short-term dependencies in time series.
- Update gates help capture long-term dependencies in time series.
- GRUs contain basic RNNs as their extreme case whenever the reset gate is switched on.
  They can ignore sequences as needed.
- GRUs can help cope with the vanishing gradient problem in RNNs and better capture dependencies for time series with large time step distances.

[1] Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.
[2] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.