1. How Gated Units Fix Things – Backpropagation through Time

Intuitively, what happens with RNNs?

1. Measure the influence of the past on the future

\[
\frac{\partial \log p(x_{t+n} | x_{<t+n})}{\partial h_t} = \frac{\partial \log p(x_{t+n} | x_{<t+n})}{\partial g} \frac{\partial g}{\partial h_{t+n}} \frac{\partial h_{t+n}}{\partial h_{t+n-1}} \ldots \frac{\partial h_{t+1}}{\partial h_t}
\]

2. How does the perturbation at $t$ affect $p(x_{t+n} | x_{<t+n})$?

1. How Gated Units Fix Things

– Backpropagation through Time
Backpropagation through Time

Vanishing gradient is super-problematic

- When we only observe

\[
\left\| \frac{\partial h_{t+N}}{\partial h_t} \right\| = \left\| \prod_{n=1}^{N} U^\top \text{diag} \left( \frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}} \right) \right\| \to 0 ,
\]

- We cannot tell whether
  1. No dependency between \( t \) and \( t+n \) in data, or
  2. Wrong configuration of parameters (the vanishing gradient condition):

\[
e_{\text{max}}(U) < \frac{1}{\max \tanh'(x)}
\]
Gated Recurrent Unit

• Is the problem with the naïve transition function?
  \[ f(h_{t-1}, x_t) = \tanh(W [x_t] + Uh_{t-1} + b) \]

• With it, the temporal derivative is
  \[ \frac{\partial h_{t+1}}{\partial h_t} = U^\top \frac{\partial \tanh(a)}{\partial a} \]
Gated Recurrent Unit

- It implies that the error must backpropagate through all the intermediate nodes:

- Perhaps we can create shortcut connections.
Gated Recurrent Unit

• Perhaps we can create *adaptive* shortcut connections.

• Candidate Update

\[ \tilde{h}_t = \tanh(W [x_t] + U h_{t-1} + b) \]

• Update gate

\[ u_t = \sigma(W_u [x_t] + U_u h_{t-1} + b_u) \]

\( \odot \): element-wise multiplication
Gated Recurrent Unit

- Let the net prune unnecessary connections \textit{adaptively}.

\[ f(h_{t-1}, x_t) = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1} \]

- Candidate Update \quad \tilde{h}_t = \tanh(W [x_t] + U (r_t \odot h_{t-1}) + b)

- Reset gate \quad r_t = \sigma(W_r [x_t] + U_r h_{t-1} + b_r)

- Update gate \quad u_t = \sigma(W_u [x_t] + U_u h_{t-1} + b_u)
Gated Recurrent Unit

tanh-RNN ....

Execution
1. Read the whole register $h$
2. Update the whole register

$$h \leftarrow \tanh(W[x] + Uh + b)$$
Gated Recurrent Unit

**GRU** ...

Gated recurrent units are much more realistic!
Note that there is some overlap in ideas with attention
Gated Recurrent Unit

Two most widely used gated recurrent units

Gated Recurrent Unit
[Cho et al., EMNLP2014; Chung, Gulcehre, Cho, Bengio, DLUFL2014]

\[
\begin{align*}
    h_t &= u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1} \\
    \tilde{h} &= \tanh(W [x_t] + U (r_t \odot h_{t-1}) + b) \\
    u_t &= \sigma(W_u [x_t] + U_u h_{t-1} + b_u) \\
    r_t &= \sigma(W_r [x_t] + U_r h_{t-1} + b_r)
\end{align*}
\]

Long Short-Term Memory
[Hochreiter & Schmidhuber, NC1999; Gers, Thesis 2001]

\[
\begin{align*}
    h_t &= o_t \odot \tanh(c_t) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
    \tilde{c}_t &= \tanh(W_c [x_t] + U_c h_{t-1} + b_c) \\
    o_t &= \sigma(W_o [x_t] + U_o h_{t-1} + b_o) \\
    i_t &= \sigma(W_i [x_t] + U_i h_{t-1} + b_i) \\
    f_t &= \sigma(W_f [x_t] + U_f h_{t-1} + b_f)
\end{align*}
\]
The LSTM
The LSTM gates all operations so stuff can be forgotten/ignored rather than it all being crammed on top of everything else.
The non-linear update for the next time step is just like an RNN
This part is the the secret! (Of other recent things like ResNets too!) Rather than multiplying, we get \( c_t \) by adding the non-linear stuff and \( c_{t-1} \) ! There is a direct, linear connection between \( c_t \) and \( c_{t-1} \).
As a result, LSTM have a long memory
Training a (gated) RNN

1. Use an LSTM or GRU: *it makes your life so much simpler!*
2. Initialize recurrent matrices to be orthogonal
3. Initialize other matrices with a sensible (**small**) scale
4. Initialize forget gate bias to 1: *default to remembering*
5. Use adaptive learning rate algorithms: *Adam, AdaDelta, …*
6. Clip the norm of the gradient: *1–5 seems to be a reasonable threshold when used together with Adam or AdaDelta.*
7. Either only dropout vertically or learn how to do it right
8. *Be patient!*

[Saxe et al., ICLR2014; Ba, Kingma, ICLR2015; Zeiler, arXiv2012; Pascanu et al., ICML2013]