Recurrent Neural Network (CNN)
Gated RNNs – Preventing Vanishing Gradient

M. R. Hasan

Deep Learning
Readings

- Geron: 15
- Zhang: 9
- Chollet: 6
What We Will Cover

• Gated RNNs for preventing the vanishing gradient problem
• Analysis: LSTM
• A variant of LSTM: Peephole Connections
Gated RNNs for Preventing the Vanishing Gradient Problem
How does the Gated RNNs solve the vanishing gradient problem?

We use the **LSTM architecture** to show how it prevents the vanishing gradient problem.
Gated RNNs: Solving Vanishing Gradient Problem

- We start by writing the expression for computing the loss gradient with respect to $\mathbf{W}$ for the \textit{vanilla RNNs}.

$$\frac{\partial L}{\partial \mathbf{W}} = \sum_{t=0}^{T} \frac{\partial l(\hat{y}_t, y_t^*)}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial \hat{z}_t} \frac{\partial \hat{z}_t}{\partial \hat{h}_t} \frac{\partial \hat{h}_t}{\partial \mathbf{W}}$$

Where

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{xh} \\ \mathbf{W}_{hh} \end{bmatrix}$$

\[ H_t = \tanh(X_t \mathbf{W}_{xh} + H_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h) \]
Gated RNNs: Solving Vanishing Gradient Problem

- In LSTMs, past hidden states $\tilde{h}_{t-1}$ do not directly pass to the current hidden state $\tilde{h}_t$.
- Instead the past hidden state $\tilde{h}_{t-1}$ and the past cell state or past memory state $c_{t-1}$ are combined to $c_t$, which produces $\tilde{h}_t$.

\[
\begin{align*}
\tilde{c}_t &= \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \\
C_t &= F_t \otimes C_{t-1} + I_t \otimes \tilde{c}_t \\
H_t &= O_t \otimes \tanh(C_t)
\end{align*}
\]
Gated RNNs: Solving Vanishing Gradient Problem

- Thus, during backward propagation, $\vec{h}_t$ passes backward through the cell state $c_t$.

\[
\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \frac{\partial l(y_t, y_t^*)}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial c_t} \frac{\partial c_t}{\partial W}
\]

\[
c_t = F_t \otimes c_{t-1} + I_t \otimes \tilde{c}_t
\]
Including the cell state \( c_t \) in the loss gradient equation, we get:

\[
\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \frac{\partial l(y_t, y_t^*)}{\partial \tilde{y}_t} \frac{\partial \tilde{y}_t}{\partial \tilde{z}_t} \frac{\partial \tilde{z}_t}{\partial \tilde{h}_t} \frac{\partial \tilde{h}_t}{\partial \tilde{c}_t} \frac{\partial \tilde{c}_t}{\partial W}
\]

\[
\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \frac{\partial l(y_t, y_t^*)}{\partial \tilde{y}_t} [\phi' \otimes (\tilde{z}_t)] W_{hy} \frac{\partial \tilde{h}_t}{\partial \tilde{c}_t} \frac{\partial \tilde{c}_t}{\partial W}
\]
Gated RNNs: Solving Vanishing Gradient Problem

Let’s focus on the last term $\frac{\partial \tilde{c}_t}{\partial W}$.

The **current memory cell state** $\tilde{c}_t$ is influenced by the previous state $\tilde{c}_{t-1}$, which is influenced by its previous state $\tilde{c}_{t-2}$, and so on until it reaches to the cell state $\tilde{c}_0$ at timestep 0 forming a chain of influence.

Including this chain effect, we obtain:

$$
\frac{\partial \tilde{c}_t}{\partial W} = \frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} \frac{\partial \tilde{c}_{t-1}}{\partial \tilde{c}_{t-1}} \cdots \frac{\partial \tilde{c}_2}{\partial \tilde{c}_1} \frac{\partial \tilde{c}_1}{\partial \tilde{c}_0} \frac{\partial \tilde{c}_0}{\partial W}
$$

$$
\Rightarrow \frac{\partial \tilde{c}_t}{\partial W} = \left( \prod_{t=1}^{T} \frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} \right) \frac{\partial \tilde{c}_0}{\partial W}
$$
Gated RNNs: Solving Vanishing Gradient Problem

\[ \frac{\partial \hat{c}_t}{\partial W} = \left( \prod_{t=1}^{t} \frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}} \right) \frac{\partial \hat{c}_0}{\partial W} \]

- This product is reminiscent of the gradient term in vanilla RNNs that led to the exploding/vanishing gradient problem.
- We now show why this product term will not cause the same issue in LSTMs.
Gated RNNs: Solving Vanishing Gradient Problem

- Let’s write the equation for computing $\tilde{c}_t$.

$$
\tilde{c}_t = \tilde{c}_{t-1} \otimes \tilde{f}_t + \tilde{c}_t \otimes \tilde{i}_t
$$

$$
\tilde{i}_t = \sigma(W_{hi}^T \tilde{x}_t + W_{h_{t-1}}^T h_{t-1} + b_i)
$$

$$
\tilde{f}_t = \sigma(W_{xf}^T \tilde{x}_t + W_{h_{t-1}}^T h_{t-1} + b_f)
$$

$$
\tilde{c}_t = \tanh(W_{xc}^T \tilde{x}_t + W_{h_{t-1}}^T h_{t-1} + \tilde{b}_c)
$$
Gated RNNs: Solving Vanishing Gradient Problem

\[ \frac{\partial \mathbf{c}_t}{\partial \mathbf{w}} = \left( \prod_{t=1}^{t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} \right) \frac{\partial \mathbf{c}_0}{\partial \mathbf{w}} \]

\[ \mathbf{c}_t = \mathbf{c}_{t-1} \otimes \mathbf{f}_t + \mathbf{\tilde{c}}_t \otimes \mathbf{i}_t \]

- Now compute \( \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} \).

\[
\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \frac{\partial [\mathbf{c}_{t-1} \otimes \mathbf{f}_t + \mathbf{\tilde{c}}_t \otimes \mathbf{i}_t]}{\partial \mathbf{c}_{t-1}}
\]

\[
\Rightarrow \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \frac{\partial [\mathbf{c}_{t-1} \otimes \mathbf{f}_t]}{\partial \mathbf{c}_{t-1}} + \frac{\partial [\mathbf{\tilde{c}}_t \otimes \mathbf{i}_t]}{\partial \mathbf{c}_{t-1}}
\]

\[
\Rightarrow \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \frac{\partial \mathbf{f}_t}{\partial \mathbf{c}_{t-1}} \mathbf{c}_{t-1} + \frac{\partial \mathbf{c}_{t-1}}{\partial \mathbf{c}_{t-1}} \mathbf{f}_t + \frac{\partial \mathbf{i}_t}{\partial \mathbf{c}_{t-1}} \mathbf{\tilde{c}}_t + \frac{\partial \mathbf{\tilde{c}}_t}{\partial \mathbf{c}_{t-1}} \mathbf{i}_t
\]

\[
\Rightarrow \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \frac{\partial \mathbf{f}_t}{\partial \mathbf{c}_{t-1}} \mathbf{c}_{t-1} + \mathbf{f}_t + \frac{\partial \mathbf{i}_t}{\partial \mathbf{c}_{t-1}} \mathbf{\tilde{c}}_t + \frac{\partial \mathbf{\tilde{c}}_t}{\partial \mathbf{c}_{t-1}} \mathbf{i}_t
\]
Gated RNNs: Solving Vanishing Gradient Problem

We compute the 1st, 3rd and 4th term as follows.

1st term: \( \frac{\partial \vec{f}_t}{\partial \vec{c}_{t-1}} \vec{c}_{t-1} = \frac{\partial [\sigma(\mathbf{W}_{xf}^T \vec{x}_t + \mathbf{W}_{hf}^T \vec{h}_{t-1} + \vec{b}_f)]}{\partial \vec{c}_{t-1}} \vec{c}_{t-1} \)

\[ \Rightarrow \frac{\partial \vec{f}_t}{\partial \vec{c}_{t-1}} \vec{c}_{t-1} = \sigma'(\mathbf{W}_{xf}^T \vec{x}_t + \mathbf{W}_{hf}^T \vec{h}_{t-1} + \vec{b}_f) \mathbf{W}_{hf} \frac{\partial \vec{h}_{t-1}}{\partial \vec{c}_{t-1}} \vec{c}_{t-1} \]

\[ \Rightarrow \frac{\partial \vec{f}_t}{\partial \vec{c}_{t-1}} \vec{c}_{t-1} = \sigma'(\mathbf{W}_{xf}^T \vec{x}_t + \mathbf{W}_{hf}^T \vec{h}_{t-1} + \vec{b}_f) \mathbf{W}_{hf} \hat{\vec{c}}_{t-1} \otimes \tanh'(\vec{c}_{t-1}) \vec{c}_{t-1} \]

\[ \Rightarrow \frac{\partial \vec{f}_t}{\partial \vec{c}_{t-1}} \vec{c}_{t-1} = \sigma'(\cdot) \mathbf{W}_{hf} \hat{\vec{c}}_{t-1} \otimes \tanh'(\vec{c}_{t-1}) \vec{c}_{t-1} \]

\[ H_t = \mathbf{O}_t \otimes \tanh(\vec{C}_t) \]
Gated RNNs: Solving Vanishing Gradient Problem

\[
\frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} = \frac{\partial \tilde{f}_t}{\partial \tilde{c}_{t-1}} \tilde{c}_{t-1} + \tilde{f}_t + \frac{\partial \tilde{i}_t}{\partial \tilde{c}_{t-1}} \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} \tilde{i}_t
\]

- The 3rd and the 4th terms can be computed in the exact same fashion.

  - 3rd term:
    \[
    \frac{\partial \tilde{i}_t}{\partial \tilde{c}_{t-1}} \tilde{c}_t = \sigma'(W_{xi} \tilde{x}_t + W_{hi} \tilde{h}_{t-1} + b_f)W_{hi} \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1})\tilde{c}_t
    \]
    \[
    => \frac{\partial \tilde{i}_t}{\partial \tilde{c}_{t-1}} \tilde{c}_t = \sigma'(.)W_{hi} \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1})\tilde{c}_t
    \]
  
  - 4th term:
    \[
    \frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} \tilde{i}_t = \sigma'(W_{xc} \tilde{x}_t + W_{hc} \tilde{h}_{t-1} + b_f)W_{hc} \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1})\tilde{i}_t
    \]
    \[
    => \frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} \tilde{i}_t = \sigma'(.)W_{hc} \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1})\tilde{i}_t
    \]
Gated RNNs: Solving Vanishing Gradient Problem

\[ \frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}} = \frac{\partial \hat{f}_t}{\partial \hat{c}_{t-1}} \hat{c}_{t-1} + \hat{f}_t + \frac{\partial \tilde{i}_t}{\partial \hat{c}_{t-1}} \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial \hat{c}_{t-1}} \tilde{i}_t \]

\[ \frac{\partial \hat{f}_t}{\partial \hat{c}_{t-1}} \hat{c}_{t-1} = \sigma'(\cdot) W^T_{hf} \hat{o}_{t-1} \otimes \tanh'(\hat{c}_{t-1}) \hat{c}_{t-1} \]

\[ \frac{\partial \tilde{i}_t}{\partial \hat{c}_{t-1}} \tilde{c}_t = \sigma'(\cdot) W^T_{hi} \hat{o}_{t-1} \otimes \tanh'(\hat{c}_{t-1}) \tilde{c}_t \]

\[ \frac{\partial \tilde{c}_t}{\partial \hat{c}_{t-1}} \tilde{i}_t = \sigma'(\cdot) W^T_{hc} \hat{o}_{t-1} \otimes \tanh'(\hat{c}_{t-1}) \tilde{i}_t \]

- Using the above 3 expressions we finally get:

\[ \frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}} = \sigma'(\cdot) W^T_{hf} \hat{o}_{t-1} \otimes \tanh'(\hat{c}_{t-1}) \hat{c}_{t-1} + \hat{f}_t + \sigma'(\cdot) W^T_{hi} \hat{o}_{t-1} \otimes \tanh'(\hat{c}_{t-1}) \tilde{c}_t + \sigma'(\cdot) W^T_{hc} \hat{o}_{t-1} \otimes \tanh'(\hat{c}_{t-1}) \tilde{i}_t \]
Gated RNNs: Solving Vanishing Gradient Problem

\[
\frac{\partial c_t}{\partial c_{t-1}} = \sigma'(\cdot) W_{hf}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_{t-1} + \tilde{f}_t + \sigma'(\cdot) W_{hi}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_t + \sigma'(\cdot) W_{hc}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{i}_t
\]

- The above expression allows us to **complete the loss gradient equation**.

\[
\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \frac{\partial l(y_t, y_{t}^*)}{\partial y_t} [\phi'(\cdot) \otimes (\tilde{z}_t)] W_{hy} \frac{\partial h_t}{\partial c_t} \left( \prod_{t=1}^{T} \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_0}{\partial W}
\]

\[
\frac{\partial L}{\partial W} = \sum_{t=0}^{T} \frac{\partial l(y_t, y_{t}^*)}{\partial y_t} [\phi'(\cdot) \otimes (\tilde{z}_t)] W_{hy} \frac{\partial h_t}{\partial c_t} \left( \prod_{t=1}^{T} \sigma'(\cdot) W_{hf}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_{t-1} + \tilde{f}_t + \sigma'(\cdot) W_{hi}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_t + \sigma'(\cdot) W_{hc}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{i}_t \right) \frac{\partial c_0}{\partial W}
\]
Let's see why $\frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}}$ will not lead to the vanishing gradient problem.

We use a shorthand notation to represent this product.

$$\prod_{t=1}^{T} \frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}} = \prod_{t=1}^{T} [X + \sigma(.) + Y + Z]$$

$$X = \sigma'(.) W_{hf}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_{t-1}$$

$$\sigma(.) = \tilde{f}_t = \sigma(W_{xf}^T \tilde{x}_t + W_{hf}^T \tilde{h}_{t-1} + \tilde{b}_f)$$

$$Y = \sigma'(.) W_{hi}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_t$$

$$Z = +\sigma'(.) W_{hc}^T \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{i}_t$$
We compare LSTM’s term with a similar expression of vanilla RNNs that was responsible for the driving the gradient to zero in vanilla RNNs.

\[
\frac{\partial \hat{h}_t}{\partial \hat{h}_{t-1}} \text{ of vanilla RNNs}
\]

\[
\frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}}
\]

Vanilla RNN: Vanishing Gradient

\[
\frac{\partial \hat{h}_t}{\partial \hat{h}_{t-1}} = \phi' \otimes (. ) W_{hh} \quad \text{tanh}
\]

\[
\phi'(.) = 0 \leq 1 \quad \text{Norm of W < 1:}
\]

\[
\prod_{t=1}^{T} \frac{\partial \hat{h}_t}{\partial \hat{h}_{t-1}} \rightarrow 0
\]

\[
\frac{\partial \hat{h}_t}{\partial \hat{h}_{t-1}} = \phi' \otimes (. ) W_{hh} \quad \text{tanh}
\]

\[
\phi'(.) < 1 \quad \text{Norm of W > 1:}
\]

\[
\prod_{t=1}^{T} \frac{\partial \hat{h}_t}{\partial \hat{h}_{t-1}} \rightarrow 0
\]

LSTM

\[
\frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}} = \sigma'( . ) W_{hf}^T \hat{o}_{t-1} \otimes tanh' (\hat{c}_{t-1}) \hat{c}_{t-1} + \hat{f}_t + \sigma'( . ) W_{hi}^T \hat{o}_{t-1} \otimes tanh' (\hat{c}_{t-1}) \hat{c}_t + \sigma'( . ) W_{hc}^T \hat{o}_{t-1} \otimes tanh' (\hat{c}_{t-1}) \hat{i}_t
\]

\[
X \quad \hat{f}_t = \sigma( . ) \quad Y \quad Z
\]

\[
\prod_{t=1}^{T} \frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}} = \prod_{t=1}^{T} [X + \sigma( . ) + Y + Z] \Rightarrow 0
\]
• Observe that in vanilla RNNs, the derivative of the activation function (assuming tanh) is always between 0 and 1 (as shown below).
  - If $W < 1$, but $\tanh'(.) \leq 1$ : successive products of these two terms will lead to zero
  - If $W > 1$, but $\tanh'(.) < 1$ : successive products of these two terms will lead to zero
• However, in LSTMs, the backpropagation through a long sequence doesn’t force the successive products of $\frac{\partial \hat{c}_t}{\partial \hat{c}_{t-1}}$ to zero.
• I.e., gradients are not bound to vanish.
Due to LSTMs gated control mechanism, the gradients are well-behaved.

Vanilla RNN: Vanishing Gradient

\[
\prod_{t=1}^{T} \frac{\partial h_t}{\partial h_{t-1}} \to 0
\]

\[
\prod_{t=1}^{T} \frac{\partial h_t}{\partial h_{t-1}} \to 0
\]

LSTM

\[
\frac{\partial c_t}{\partial c_{t-1}} = \sigma'(\cdot) W_h^{\text{T}} \tilde{c}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_{t-1} + \tilde{f}_t + \sigma'(\cdot) W_h^{\text{T}} \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{c}_t + \sigma'(\cdot) W_h^{\text{T}} \tilde{o}_{t-1} \otimes \tanh'(\tilde{c}_{t-1}) \tilde{i}_t
\]

X \quad \hat{f}_t = \sigma(\cdot) \quad Y \quad \tilde{c}_t \quad Z

\[
\prod_{t=1}^{T} \frac{\partial c_t}{\partial c_{t-1}} = \prod_{t=1}^{T} [X + \sigma(\cdot) + Y + Z] \not\to 0
\]
• There are **two main reasons**.

  - Unlike the vanilla RNNs, $\frac{\partial \tilde{c}_t}{\partial \tilde{c}_{t-1}}$ is **additive**.
  - The forget gate term (2\textsuperscript{nd} term) is always between 0 and 1.

• If the X, Y and Z terms in the product start to converge to zero, LSTM cell will set the forget gate term to 1 (highest value of sigmoid).
• This happens when the network wants to retain long-range dependencies by letting the cell state to pass through a long sequence.
• In such a case, the near 1 value of the forget gate tends to shrink at a much slower rate.

• In vanilla RNNs, the derivative values of tanh (later on during the training processes) are likely to be saturated and thus have a value close to 0.
Another reason is the LSTM cell learns to decide **when to let the gradients vanish** (i.e., $X$, $Y$, $Z$, and $\sigma(.)$ are driven to 0) and when to **preserve the gradients** by setting the gate values accordingly.

\[
\begin{align*}
\frac{\partial h_t}{\partial h_{t-1}} &= \phi' \otimes (.) W_{hh} \\
\frac{\partial h_t}{\partial h_{t-1}} &= \phi' \otimes (.) W_{hh} \\
\phi'(.) &= 0 \leq 1 & \text{Norm of } W < 1: \quad \Pi_{t=1}^{T} \frac{\partial h_t}{\partial h_{t-1}} \to 0 \\
\phi'(.) &< 1 & \text{Norm of } W > 1: \quad \Pi_{t=1}^{T} \frac{\partial h_t}{\partial h_{t-1}} \to 0 \\
\end{align*}
\]
Thus, LSTMs solve the vanishing gradient problem using a unique additive gradient structure that includes direct access to the forget gate’s activations. It enables the network to encourage desired behavior from the loss gradient using frequent gate updates on every timestep of backpropagation.
Peephole Connections
Peephole Connections

- In a regular LSTM cell, the gate controllers can look only at the input $x_t$ and the previous short-term state $h_{t-1}$.
- It may be a good idea to give them a bit more context by letting them peek at the long-term state $c_{t-1}$ as well.

This idea was proposed by Felix Gers and Jürgen Schmidhuber in 2000. [https://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf](https://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf)
Peephole Connections

- We can create an LSTM variant as follows.
  - Add the previous long-term state $c_{t-1}$ to the controllers of the forget gate and the input gate.
  - Add the current long-term state $c_t$ is added as input to the controller of the output gate.

These connections are known as **peephole connections**
Peephole Connections

- The peephole connections often improves the performance, but **not always**.
- There is **no clear pattern** for which tasks are better off with or without them.

We will have to try it on our task and see if it helps.
Peephole Connections

• In TensorFlow we can implement the peephole connection based LSTM using the experimental tf.keras.experimental.PeeepholeLSTMCell.
  https://www.tensorflow.org/api_docs/python/tf/keras/experimental/PeeepholeLSTMCell