Lecture IV: Recurrent Neural Networks
Basics of Recurrent Neural Networks (RNNs)
Limitations of Feedforward Networks

- Size of input vector $\mathbf{x}$ is fixed
- Temporal relations of elements in sequence of inputs lost
  - Restricted work-around: Windowing via convolution
- Can learn mathematical functions but not Turing Complete
Recurrent Neural Networks (RNNs) (1)

- Assume a sample is a sequence of length $T$ with $D$ features at each timestep $t$.
  - Each sample represented by matrix $X$ of shape $T \times D$.
  - $T$ may vary between samples but $D$ is constant.
Recurrent Neural Networks (RNNs) (1)

- Assume a sample is a sequence of length $T$ with $D$ features at each timestep $t$.
  - Each sample represented by matrix $X$ of shape $T \times D$
  - $T$ may vary between samples but $D$ is constant

- To address the mentioned limitations of feedforward networks, our network needs to:
  1. be able to handle variable sequence lengths $T$,
  2. remember previous inputs
Recurrent Neural Networks (RNNs) (2)

- Solution: Recurrent Neural Networks (RNNs)
Solution: Recurrent Neural Networks (RNNs)

- Feed input sequence $X$ timestep by timestep into network ($= \text{vector } x_t \text{ of length } D$)
- Add previous output $h_{t-1}$ ($= \text{hidden state}$) to current input $x_t$
  
  $x_t$ and $h_{t-1}$ are inputs to compute new $h_t$: $h_t = f(x_t, h_{t-1})$

RNN layer

$h_t$

$h_{t-1}$

$X_t$

...
Recurrent Neural Networks (RNNs) (3)

- Typically only single layers recursively connected
- Layer weight matrix $W$ reused (=shared) for all timesteps
- Computation of $h_t$ similar to feedforward networks:

$$h_t = W^T \cdot \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b$$
Power of RNNs

- RNNs are in essence a state-space model: \( y_t = f(x, y_{t-1}) \)
- RNNs are used for sequence learning, control systems, …
- RNNs are theoretically **Turing Complete**
  - For each program in a turing complete programming language, you could find an RNN that executes the code correctly
  - Like the Universal Approximation Theorem, this is mostly useless in practice
  - The hard problem is not “what can we represent?” but finding a good representation
Power of RNNs

- RNNs are in essence a state-space model: $y_t = f(x, y_{t-1})$
- RNNs are used for sequence learning, control systems, …
- RNNs are theoretically Turing Complete
  - For each program in a turing complete programming language, you could find an RNN that executes the code correctly
  - Like the Universal Approximation Theorem, this is mostly useless in practice
  - The hard problem is not “what can we represent?” but finding a good representation
Sequence Learning
Settings
Sequence Learning Settings (1)

Alex Graves distinguishes 3 types of classification tasks:

**Sequence Classification:** 1 label per sequence
**Segment Classification:** 1 label per part of sequence
**Temporal Classification:** sequence of labels per sequence

[Supervised Sequence Labelling with Recurrent Neural Networks, A. Graves, 2012]
Sequence Learning Settings (2)

[The Unreasonable Effectiveness of Recurrent Neural Networks, A. Karpathy, 2015]
RNN Training
Unrolling an RNN

- RNN can be viewed as feed forward network with shared weights = unrolled over time
Unrolling an RNN

- RNN can be viewed as feed forward network with shared weights = unrolled over time
Most common way to train RNNs: **Back-Propagation Through Time (BPTT)**

- **Complexity:** $O(N^2 T)$
  - $N$: number of hidden units
  - $T$: Length of sequence

**Truncated BPTT:** only unfold $n$ timesteps into the past
Most common way to train RNNs: **Back-Propagation Through Time (BPTT)**

**Complexity:** $O(N^2T)$
- $N$: number of hidden units
- $T$: Length of sequence
Most common way to train RNNs: **Back-Propagation Through Time (BPTT)**

- **Complexity:** $O(N^2 T)$
  - $N$: number of hidden units
  - $T$: Length of sequence

- **Truncated BPTT**: only unfold $n$ timesteps into the past
Real-Time Recurrent Learning (RTRL)

- Alternative to BPTT
- Computes all gradient information during forward pass
- Complexity $\mathcal{O}(N^4) \Rightarrow$ Independent of sequence length
- Very rarely used today
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx$ depth)

→ Vanishing or Exploding Gradients (Hochreiter, 1991)
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx$ depth)

$\rightarrow$ Vanishing or Exploding Gradients (Hochreiter, 1991)
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx$ depth)
  $\rightarrow$ Vanishing or Exploding Gradients (Hochreiter, 1991)
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx \text{depth}$)
  \[ \Rightarrow \text{Vanishing or Exploding Gradients} \quad \text{(Hochreiter, 1991)} \]
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx \text{depth}$)
  $\rightarrow$ Vanishing or Exploding Gradients (Hochreiter, 1991)

- Derivative of sigmoid function

- RNN layer diagram
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx \text{depth}$)
  $\rightarrow$ Vanishing or Exploding Gradients (Hochreiter, 1991)
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx$ depth)
- \[\rightarrow\] Vanishing or Exploding Gradients \((\text{Hochreiter, 1991})\)
Vanishing Gradients Problem

- BPTT generates very deep networks ($T \approx \text{depth}$)

→ Vanishing or Exploding Gradients (Hochreiter, 1991)
Vanishing Gradients Problem - Consequences

- RNNs tend to forget events that happened a long time ago
- Learning long-term dependencies depends on the recurrent weights
  - If $|F'| < 1$, we will forget things over time
  - If $|F'| > 1$, our system is unstable
  - $\Rightarrow$ we would need $|F'| = 1$
Long Short-Term Memory (LSTM)
Long Short-Term Memory (LSTM)

- **Idea**: Store information indefinitely but be selective about what to store

- **Solution**:
  1. **Integrator**
     - add up information over time
     - store information indefinitely
  2. **Gates**
     - hidden units
     - activations multiplied with input
     - write-access (remembering)
     - read-access (communicating)
     - reset (forgetting)

This system is called Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997)
Long Short-Term Memory (LSTM)

- **Idea:** Store information indefinitely but be selective about what to store

- **Solution:**
  1. Integrator
     - add up information over time
     - store information indefinitely

This system is called Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997)
Long Short-Term Memory (LSTM)

**Idea:** Store information indefinitely but be selective about what to store

**Solution:**

1. **Integrator**
   - add up information over time
   - store information indefinitely

2. **Gates**
   - hidden units
   - activations multiplied with input
   - write-access (remembering)
   - read-access (communicating)
   - reset (forgetting)
Long Short-Term Memory (LSTM)

- **Idea:** Store information indefinitely but be selective about what to store

- **Solution:**
  1. Integrator
     - add up information over time
     - store information indefinitely
  2. Gates
     - hidden units
     - activations multiplied with input
     - write-access (remembering)
     - read-access (communicating)
     - reset (forgetting)

This system is called Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997)
Long Short-Term Memory (LSTM)

Constant Error Carousel (CEC)

New information $\text{net}_{\text{in}}$ is squashed to scalar $c$ via function $f$:

$$c_{\ast} = c_{\ast}(t-1) + c_{\text{in}}$$

This is a simple integrator, no vanishing gradients!

$\text{CEC}$: e.g. $\tanh$, linear

$LSTM$ block

$\text{cell input}$ $\text{cell output}$
Long Short-Term Memory (LSTM)

Constant Error Carousel (CEC)

- New information $net_{in}$ is squashed to scalar $c_{in}$ via function $f_{in}$

![LSTM block diagram]
Long Short-Term Memory (LSTM)

Constant Error Carousel (CEC)

- New information $net_{in}$ is squashed to scalar $c_{in}$ via function $f_{in}$
- New cell state:
  $$c_{st}^* = c_{s(t-1)}^* + c_{int}$$
- ⇒ simple integrator, no vanishing gradients!
**Long Short-Term Memory (LSTM)**

**Constant Error Carousel (CEC)**

- New information $net_{in}$ is squashed to scalar $c_{in}$ via function $f_{in}$

- New cell state:
  \[
  c_{st}^* = c_{s(t-1)}^* + c_{int}
  \]

  $\Rightarrow$ simple integrator, no vanishing gradients!

- $f_{in}$: e.g. $\tanh$

- $f_{s^*}$: e.g. $\tanh$, $linear$
Long Short-Term Memory (LSTM)

More terminology

- CEC and gates constitute LSTM block or LSTM unit
- Cell output $h$ (hidden state) is output of LSTM block
- Multiple LSTM blocks in one layer are referred to as LSTM layer
Long Short-Term Memory (LSTM)

More terminology

- CEC and gates constitute LSTM block or LSTM unit
- Cell output \( h \) (hidden state) is output of LSTM block
- Multiple LSTM blocks in one layer are referred to as LSTM layer
Long Short-Term Memory (LSTM)

**Input gate**

- Input gate serves as gating/attention mechanism
- $c_{in}$ is multiplied by input gate activation $g_{ig}$ before entering CEC
- $f_{ig}$: e.g. sigmoid
Long Short-Term Memory (LSTM)

Recurrent hidden state

- Input gate and cell input may receive old hidden state $h_{(t-1)}$ as recurrent input.
- In an LSTM layer, the hidden states of all LSTM blocks are the recurrent input per block.
- But: fully connected LSTM might not always be the best way to go!
Long Short-Term Memory (LSTM)

Output gate

- Output gate mechanism analogous to input gate
- Output gate controls if cell state $c_s$ is visible to rest of network
- $f_{og}$: e.g. sigmoid
Long Short-Term Memory (LSTM)

**Forget gate**

- Forget gate mechanism analogous to other gates
- Can reset or decrease CEC content
- $f_\varphi$: e.g. *sigmoid*
Long Short-Term Memory (LSTM)

Forget gate

- Forget gate mechanism analogous to other gates
- Can reset or decrease CEC content
- $f_\varphi$: e.g. sigmoid

⇒ Problem: this re-introduces vanishing gradients! Only use if necessary!
Long Short-Term Memory (LSTM)

Learning behavior

- LSTM core (CEC) is an integrator
- Gates introduce complex dynamics
- LSTM blocks (de)activate and complement each other dynamically
Long Short-Term Memory (LSTM)

**Tricks of the trade**

- Plot your LSTM cell- and hidden states
- Fully connected LSTM not always needed
- Negative input gate bias helps for long sequences
- Use forget gate only if necessary
LSTM example: Task description

```
class 0  class 1
input    feature 1
feature 2
```

---

![Graph showing input, feature 1, and feature 2 for class 0 and class 1.](image-url)
LSTM example: Task description

class 0

input

0.0 0.5 1.0

feature 1

feature 2

class 0

class 1

1.0

0.0

feature 1

feature 2

class 1: feature 1 and feature 2 active at same time

class 1: feature 1 and feature 2 active at same time
LSTM example: 1 LSTM (fully connected)

class 0

input

output

cellstate

cell input

input gate

output gate

class 1

input

output

cellstate

cell input

input gate

output gate
LSTM example: 1 LSTM (fully connected)
LSTM example: 2 LSTM (fully connected)
LSTM example: 4 LSTM (not fully connected)
LSTM example: 32 LSTM (fully connected)
LSTM Formulas

\[ z^t = g \left( W_z x^t + R_z y^{t-1} + b_z \right) \]
\[ i^t = \sigma \left( W_i x^t + R_i y^{t-1} + b_i \right) \]
\[ f^t = \sigma \left( W_f x^t + R_f y^{t-1} + b_f \right) \]
\[ c^t = i^t \odot z^t + f^t \odot c^{t-1} \]
\[ o^t = \sigma \left( W_o x^t + R_o y^{t-1} + b_o \right) \]
\[ y^t = o^t \odot h(c^t) \]
Gated Recurrent Units (GRUs)
Gated Recurrent Units (GRUs)

- Reduced LSTM with merged gates (Cho et al, 2014)
- Suffers from Vanishing Gradients (always forgets)
- Less parameters, easier to use, lower complexity

\[
\begin{align*}
    z_t &= \sigma \left( W_z \cdot [h_{t-1}, x_t] \right) \\
    r_t &= \sigma \left( W_r \cdot [h_{t-1}, x_t] \right) \\
    \tilde{h}_t &= \tanh \left( W \cdot [r_t \ast h_{t-1}, x_t] \right) \\
    h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]

[Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
LSTM Applications
LSTM Applications

- LSTM can effectively learn long-term dependencies
- One of the most-used models today
- State of the Art in many applications
  - Speech/Text generation and recognition
  - Amino acid sequence classification
  - Time-Series classification/generation
  - ...
Handwriting Generation

from his travels it might have been
from his travels it might have been
from his travels it might have been
from his travels it might have been
more of national temperament
more of national temperament
more of national temperament
more of national temperament
more of national temperament
more of national temperament

[Generating Sequences With Recurrent Neural Networks, A. Graves, arxiv 2013]

Online interactive example: https://www.cs.toronto.edu/~graves/handwriting.html
Source Code Generation

```c
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMPTHREAD_UNCC) + 
                   ((count & 0x00000000ffffff) & 0x000000ff) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &offset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

[The Unreasonable Effectiveness of Recurrent Neural Networks, A. Karpathy, 2015]

Many more examples: [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)
Image Captioning

[Show and Tell: A Neural Image Caption Generator, Vinyals & Toshev & Bengio & Erhan, arxiv 2015]
<table>
<thead>
<tr>
<th>Type</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our model</td>
<td>Ulrich UNK, membre du conseil d’administration du constructeur automobile Audi, affirme qu’il s’agit d’une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d’administration afin qu’ils ne soient pas utilisés comme appareils d’écoute à distance.</td>
</tr>
<tr>
<td>Truth</td>
<td>Ulrich Hackenberg, membre du conseil d’administration du constructeur automobile Audi, déclare que la collecte des téléphones portables avant les réunions du conseil, afin qu’ils ne puissent pas être utilisés comme appareils d’écoute à distance, est une pratique courante depuis des années.</td>
</tr>
<tr>
<td>Our model</td>
<td>“Les téléphones cellulaires, qui sont vraiment une question, non seulement parce qu’ils pourraient potentiellement causer des interférences avec les appareils de navigation, mais nous savons, selon la FCC, qu’ils pourraient interférer avec les tours de téléphone cellulaire lorsqu’ils sont dans l’air”, dit UNK.</td>
</tr>
<tr>
<td>Truth</td>
<td>“Les téléphones portables sont véritablement un problème, non seulement parce qu’ils pourraient éventuellement créer des interférences avec les instruments de navigation, mais parce que nous savons, d’après la FCC, qu’ils pourraient perturber les antennes-relais de téléphonie mobile s’ils sont utilisés à bord”, a déclaré Rosenker.</td>
</tr>
<tr>
<td>Our model</td>
<td>Avec la cérémonie, il y a un “sentiment de violence contre le corps d’un être cher”, qui sera “réduit à une pile de cendres” en très peu de temps au lieu d’un processus de décomposition “qui accompagnera les étapes du deuil”.</td>
</tr>
<tr>
<td>Truth</td>
<td>Il y a, avec la cérémonie, “une violence faite au corps aimé”, qui va être “réduit à un tas de cendres” en très peu de temps, et non après un processus de décomposition, qui “accompagnerait les phases du deuil”.</td>
</tr>
</tbody>
</table>

Table 3: A few examples of long translations produced by the LSTM alongside the ground truth translations. The reader can verify that the translations are sensible using Google translate.
Hydrology Forecasts

[Sequence to Sequence Learning with Neural Networks, Kratzert & Herrnegger & Klotz & Hochreiter & Klambauer]
Summary
Summary

- Recurrent Neural Networks (RNNs):
  - Can handle sequence data of variable length
  - Turing Complete
  - Vanishing Gradients problem

- Long Short-Term Memory (LSTM):
  - Integrator with gating mechanisms
  - Solve Vanishing Gradients problem
  - State of the Art for sequence-like data