Advanced Recurrent Architectures

Davide Bacciu
Dipartimento di Informatica
Università di Pisa

Intelligent Systems for Pattern Recognition (ISPR)
Lecture Outline

• Dealing with structured/compound data
  • Sequence-to-sequence
  • Attention models

• Dealing with very long-term dependencies
  • Multiscale networks
  • Adding memory components

• Neural reasoning
  • Neural Turing machines
Gated RNN Refresher

**LSTM Cell**

- \( h_t \)
- \( c_t \)
- \( g_t \)

**GRU Cell**

- \( h_t \)
- \( z_t \)
- \( r_t \)
Use a simplified (and overloaded) graphical notation to represent GRNN layers and stacks of GRNN
Basic Gated RNN (GRNN) Limitations

• GRNN are excellent to handle size/topology varying data in input
  • How can we handle size/topology varying outputs?
  • Sequence-to-sequence
• Structured data is compound information
  • Efficient processing needs the ability to focus on certain parts of such information
  • Attention mechanism
• GRNN have troubles dealing with very long-range dependencies
  • Introduce multiscale representation explicitly in the architecture
  • Introduce external memory components
The idea of an unfolded RNN with blank inputs-outputs does not really work well.

The approach is based on an encoder-decoder scheme.
Encoder

Produce a compressed and fixed length representation $c$ of all the input sequence $x_1, \ldots, x_n$

Originally $c = h_n$

Activations of an LSTM/GRU layer of K cells
Decoder

A LSTM/GRU layer of K cells seeded by the context vector \( c \)

Different approaches to realize this in practice
If we share the parameters between encoder and decoder we can take \( s_1 = c \)

Or, at least, assume \( c \) and \( s_1 \) have compatible size.

We risk to lose memory of \( c \) soon.
Decoder

$c$ is contextual information kept throughout output generation.
It is better to work on a **one-step-ahead scheme**

Remember **teacher forcing** (only) at training time

$c = f(c, s_{i-1}, y_{i-1})$
Sequence-To-Sequence Learning

Encoder-Decoder can share parameters (but it is uncommon)

Encoder-Decoder can be trained end-to-end or independently

Reversing the input sequence in encoding can increase performance
On the Need of Paying Attention

- Encoder-Decoder scheme assumes the hidden activation of the last input element summarizes sufficient information to generate the output
  - Bias toward most recent past
- Other parts of the input sequence might be very informative for the task
  - Possibly elements appearing very far from sequence end
On the Need of Paying Attention

- Attention mechanism select which part of the sequence to focus on to obtain a good $c$
Attention Mechanisms – Blackbox View

What’s inside of the box?

Attention Module

\[ \text{Aggregated seed} \]
\[ C \]

\[ S \]
\[ h_1 \quad h_2 \quad h_3 \quad \ldots \quad h_n \]

Encodings

Context info

Introduction
Advanced RNN
Wrap-up
Structured Output and Attention
Extending Memory
Neural Reasoning
What’s inside of the box?

The Revenge of the Gates!
Opening the Box

Introduction
Advanced RNN
Structured Output and Attention
Extending Memory
Neural Reasoning

\[
\begin{align*}
\alpha_1 \times e_1 & \quad \alpha_2 \times e_2 & \quad \alpha_3 \times e_3 & \quad \cdots & \quad \alpha_n \times e_n \\
h_1 & \quad h_2 & \quad h_3 & \quad \cdots & \quad h_n \\
S & \quad \text{SOFTMAX} & \quad C
\end{align*}
\]
Opening the Box – Relevance

Tanh layer fusing each encoding with current context $s$

$e_i = a(s, h_i)$
Opening the Box – Softmax

A differentiable max selector operator

\[ \alpha_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)} \]

\[ c \]

\[ S \]

\[ h_1 \quad h_2 \quad h_3 \quad \ldots \quad h_n \]

\[ + \quad + \quad + \quad \ldots \quad + \]
Opening the Box – Voting

**Aggregated seed by** (soft) attention voting

\[ c = \sum_{i} \alpha_i h_i \]
Attention in Seq2Seq

Context is past output state

Seed takes into account (subset of) the input states
Learning to Translate with Attention

Bahdanau et al, Show, Neural machine translation by jointly learning to align and translate, ICLR 2015
Seq-to-Seq on Steroids

Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation",
This component determines how much each $h$ is correlated/associated with current context $s$. 
Sample a single encoding using probability $\alpha_i$
Advanced Attention – Self Attention

For each element of an input sequence $X_i$ project into 3 vectors: query, key and value

For each element, compute attention over all other vectors

$$SA(Q_i, K, V) = \sum_j \text{softmax}_j \left( \frac{Q_i \cdot K^T}{\sqrt{d_k}} \right) V_j$$

Vaswani et al., Attention Is All You Need, NIPS 2017
Self Attention – K,V,Q Generation

Figure credit to this article

Self-attention

<table>
<thead>
<tr>
<th>input #1</th>
<th>input #2</th>
<th>input #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0</td>
<td>0 2 0 2</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>
Self Attention – Compute Attention Score
Self Attention – Produce Output
Self Attention – MultiHead

Introduction
Advanced RNN
Wrap-up
Structured Output and Attention
Extending Memory
Neural Reasoning

Strubell et al, Linguistically-Informed Self-Attention for Semantic Role Labeling, EMNLP 2018
Attention-Based Captioning – Focus Shifting

Soft Attention

A bird flying over a body of water

Hard Attention

Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015
Attention-Based Captioning - Generation

Learns to **correlate textual and visual concepts**

- A woman is throwing a **frisbee** in a park.
- A **dog** is standing on a hardwood floor.
- A **stop** sign is on a road with a mountain in the background.

Helps understanding why the **model fails**

- A large white **bird** standing in a forest.
- A woman holding a **clock** in her hand.

Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015
Attention-Based Captioning – The Model

Encodings associated to *n* image regions

From convolutional layers rather than from fully connected

Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015
RNN and Memory – Issue 1

- Gated RNN claim to solve the problem of learning long-range dependencies
- In practice it is still difficult to learn on longer range
- Architectures trying to optimize dynamic memory usage
  - Clockwork RNN
  - Skip RNN
  - Multiscale RNN
  - Zoneout
Clockwork RNN

**Modular** recurrent layer where each module is updated at **different clock**

Modules interconnected only when destination clock time is larger

Koutnik et al, A Clockwork RNN, ICML 2014
Skip RNN

Binary state update gate determining if RNN state is updated or copied.

Replacing gated update by copying increases network memory (LSTM has an exponential fading effect due to the multiplicative gate).
Skip RNN and Attention

Campos et al, Skip RNN: Skipping State Updates in Recurrent Neural Networks, ICLR 2018
RNN and Memory – Issue 2

A motivating example:

<table>
<thead>
<tr>
<th>Task 3: Three Supporting Facts</th>
<th>Task 15: Basic Deduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>John picked up the apple.</td>
<td>Sheep are afraid of wolves.</td>
</tr>
<tr>
<td>John went to the office.</td>
<td>Cats are afraid of dogs.</td>
</tr>
<tr>
<td>John went to the kitchen.</td>
<td>Mice are afraid of cats.</td>
</tr>
<tr>
<td>John dropped the apple.</td>
<td>Gertrude is a sheep.</td>
</tr>
<tr>
<td>Where was the apple before the kitchen? A:office</td>
<td>What is Gertrude afraid of? A:wolves</td>
</tr>
</tbody>
</table>

• In order to solve the task need to memorize
  • Facts
  • Question
  • Answers

• A bit too much for the dynamical RNN memory

• Try to address it through an external memory
Memory Networks - General Idea

Neural network

writing

Memory module

reading
Memory Network Components

(I) Input feature map: Encodes the input in a feature vector

(G) Generalization: decide what input (or function of it) to write to memory

(O) Output feature map: reads the relevant memory slots

(R) Response: returns the prediction given the retrieved memories

Weston et al, Memory Networks, ICLR 2015
End-to-End Memory Networks

- Combine output memories through soft attention
- Search for memories matching the query
- Query driven soft-attention
- Generate prediction

**Facts** $x_i$

**Embedding A**

**Embedding C**

**Weighted Sum**

**Softmax**

**Input**

**Output**

**Expected Answer** $\hat{a}$

**Predicted Answer**

**Sukhbaatar et al., End-to-end Memory Networks, NIPS 2015**
Memory Network Extensions

Often with tied weights

Use more complex output components, e.g. RNN to generate response sequences

Stack multiple memory network layers

Several iterations of reasoning to produce a better answer
Memory Nets for Visual QA with Attention

Yang et al, Stacked Attention Networks for Image Question Answering, CVPR 2016
Memory Nets for Visual QA with Attention

(a) What are pulling a man on a wagon down on dirt road?  
   Answer: horses  Prediction: horses

(b) What is the color of the box?  
   Answer: red  Prediction: red

(c) What next to the large umbrella attached to a table?  
   Answer: trees  Prediction: tree

(d) How many people are going up the mountain with walking sticks?  
   Answer: four  Prediction: four

(e) What is sitting on the handle bar of a bicycle?  
   Answer: bird  Prediction: bird

(f) What is the color of the horns?  
   Answer: red  Prediction: red
Neural Turing Machines

- Memory networks that can **read and write memories** at both training and test
- **End-to-end** differentiable
Typically an RNN emitting vectors to control read and write from the memory.

The key to differentiability is to always read and write the whole memory.

Use soft-attention to determine how much to read/write from each point.
Memory Read

Interest in the i-th memory

Attention distribution vector $\alpha$ from the RNN

Memories $M_i$

Retrieved memory is a weighted mean of all memories

$$r = \sum_{i} a_i M_i$$
**Memory Write**

**Value to write** 

\[ w \]

**Attention distribution** vector \( a \) describing how much we change each memory

\[ M_i = a_i w + (1 - a_i) M_i \]

**Write operation is actually performed by composing erasing and adding operations**
1. Generate content-based memory indexing
NTM Attention Focusing

2. Interpolate with attention from previous time

A

B

interpolate

Previous attention vector

Interpolation amount
NTM Attention Focusing

3. Generate location-based indexing

B

```
convolve
```

Shift distribution filter

Determines how we move between the locations in memory

Sharpen the distribution for final memory access
Practical Use?

• Not yet..
• Not straightforward to train
• Advantages over GRNN when it comes to learn to program
  • Copy task
  • Repeat copy
  • Associative recall
  • Sorting
• Foundations for neural reasoning
  • Pondering networks
Software

- Complete sequence-to-sequence tutorial (including attention) on **Tensorflow**
  - A shorter version in **Keras**
- **Github project** collecting several memory augmented networks
- **Pytorch** implementation of stacked attention for visual QA (originally Theano-based)
- Many implementations of the NTM (Keras, Pytorch, Lasagne,...): none seemingly official, but **this recent one** is supposedly stable (enough for TF to list it as official)
Take Home Messages

• Attention.. Attention.. and, again, attention
  • Soft attention is nice because makes everything fully differentiable
  • Hard attention is stochastic hence cannot Backprop
  • Empirical evidences of them being sensitive to different things

• Encoder-Decoder scheme
  • A general architecture to compose heterogeneous models and data
  • Decoding allows sampling complex predictions from an encoding conditioned distribution

• Memory and RNN
  • Efficient use of dynamic memory
  • External memory for search and recall tasks
  • Read/write memory for neural reasoning