Neural Memories

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA’ DI PISA

DAVIDE.BACCIU@UNIPI.IT
Lecture Outline

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

- Neural reasoners
  - Neural memories
  - Differentiable memory read, write, indexing
Hierarchical and Multiscale Recurrent Networks
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
Vanishing effects are exponential w.r.t. the unrolled network’s depth.

Skipping updates entirely reduces the vanishing effect.

How can we shorten the path between long-range dependencies?

How can we skip updates? Any idea/intuitions?
Convolutional Seq2Seq

- Use convolution instead of recurrence
- Better parallelization on GPUs
- Smaller distance between long-range dependencies
Depth of different architectures

- Vanishing gradients depend on the depth of the network

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Complexity per Layer</th>
<th>Sequential Operations</th>
<th>Maximum Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(n^2 \cdot d)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Recurrent</td>
<td>$O(n \cdot d^2)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>$O(k \cdot n \cdot d^2)$</td>
<td>$O(1)$</td>
<td>$O(\log_k(n))$</td>
</tr>
<tr>
<td>Self-Attention (restricted)</td>
<td>$O(r \cdot n \cdot d)$</td>
<td>$O(1)$</td>
<td>$O(n/r)$</td>
</tr>
</tbody>
</table>
Hierarchical RNNs

- Add skip connections to the model (static skip)
- Learn when to skip updates (adaptive skip)
- Skip units, blocks of units, or entire layer
Regularization – Zoneout

- At each timestep, force some units to keep the same value
- Random sampling
- Hard gate
- Avoiding the update improves gradient propagation

Zoneout: \[ T = d_t \odot \tilde{T} + (1 - d_t) \odot 1 \]

Dropout: \[ T = d_t \odot \tilde{T} + (1 - d_t) \odot 0 \]
Clockwork RNN

Modular recurrent layer where each module is updated at different clock

Modules interconnected only when destination clock time is larger
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

Attended pixels

Ignored pixels
Hierarchical Sequential Structure

- Many sequences have a latent hierarchical structure
- Example: wikipedia, represented as a sequence of characters
- Hierarchy
  - Characters
  - Words
  - Sentences
  - Paragraphs
  - Documents
- We want to model the hierarchy explicitly
Hierarchical Multiscale RNN (HM-RNN)

- **UPDATE**: state update (LSTM cells) according to boundary detector.
- **COPY**: copies cell and hidden states from the previous timestep to the current timestep.
- **FLUSH**: sends summary to next layer and re-initializes current layer’s state.

Recap (1)

- **Recurrence**: update at each timestep, linear scan of the sequence. Path length = n
- **Convolution**: update at each timestep but look at the last k timestep. Path length = \( \log_k(n) \)
- **Attention**: update the entire sequence in parallel. Path length = 1
Recap (2)

- **Zoneout**: randomly disable unit update.
- **CW-RNN**: blocks of units with different update frequencies. Static.
- **Skip-RNN**: adaptive gates learn to skip entire updates. Also saves computation.
- **HM-RNN**: each layer models more abstract features by learning the boundaries. Adaptive.
Neural Reasoning
Neural Reasoning

- Recurrent (sequential) models as general programs
- Reasoning abilities, typical of classic algorithms or classic AI
- Why don’t we use a «classic» algorithm? Because we may not have a proper input (e.g. pathfinding graph) but only sensor information.
  - Example: robot navigation
  - The model needs to learn to encode the structure from the raw data and then solve the problem
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

Query driven soft-attention

Generate prediction

Facts $x_i$

Search for memories matching the query

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
Neural Controller

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 $a$ from the RNN

Memories $M_i$

Retrieved memory is a weighted mean of all memories

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

Attention distribution vector $\alpha$ 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
NTM Attention Focusing

1. Generate content-based memory indexing

- Memory
  - Query vector
  - dot product
  - softmax
  - A
NTM Attention Focusing

2. Interpolate with attention from previous time

A

\[\text{interpolate}\]  

B

Previous attention vector

Interpolation amount
NTM Attention Focusing

3. Generate location-based indexing

- Shift distribution filter
  - Determines how we move between the locations in memory
- Sharpen the distribution for final memory access
Practical Use?

- Not really of these models (but inspired Neural Algorithmic Reasoning)
- 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

○ Hierarchy
  ● Can be directly encoded in RNN, or learned adaptively
  ● Used to mitigate vanishing gradients and learn hierarchical encodings

○ Memory and RNN
  ● Efficient use of dynamic memory
  ● External memory for search and recall tasks
  ● Read/write memory for neural reasoning
Next Week

Generative Deep Learning module

- Explicit density models
- Implicit/sampling models
- Neural diffusion
- ...

DAVIDE BACCIU - ISPR COURSE
After Next Week

...we will continue on April 30th!

○ No lecture 23, 24 and 25th April

○ Lecture on 30th April, 02 May