Differentiable Neural Computers

Hybrid Computing using a neural network with dynamic external memory (Graves et al. 2016)

Konstantinos Kogkalidis

May 28, 2018

Logic and Computation

Overview

Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

Overview

Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

- Extension of NTMs

Overview

Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

- Extension of NTMs
  - End-to-end differentiable

Overview

Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

- Extension of NTMs
    - End-to-end differentiable
    - Auto-associative memory

Overview

Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

- Extension of NTMs
    - End-to-end differentiable
    - Auto-associative memory
    - Turing complete

Overview

Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

- Extension of NTMs
    - End-to-end differentiable
    - Auto-associative memory
    - Turing complete
    + Memory attention mechanisms

Overview

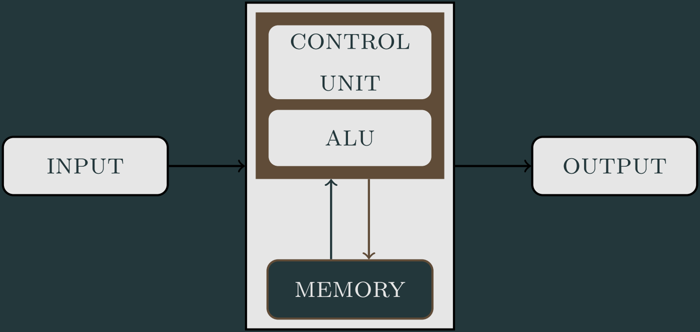
Differentiable Neural Computer

A recurrent neural network coupled with an external memory.

- Extension of NTMs
  - End-to-end differentiable
  - Auto-associative memory
  - Turing complete
  + Memory attention mechanisms
- Mimic mammalian biological memory
- Employ classical concepts of computation

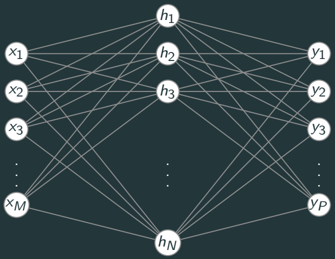
Introduction: Motivation

Von Neumann architecture



Introduction: Motivation

Simple Neural Net

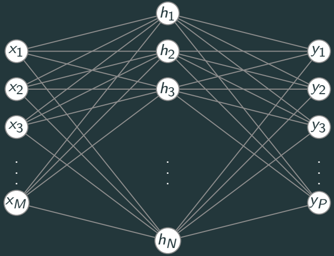


$$y = g(h),\ h = f(x)$$

No memory

Introduction: Motivation

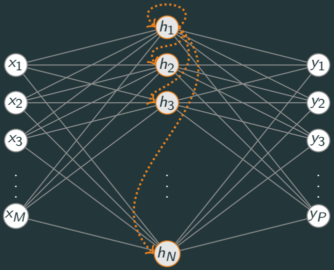
Simple Neural Net



$$y = g(h),\ h = f(x)$$

No memory

Recurrent Neural Net



$$h(t) = f([x(t); h(t-1)])$$

Finite, non-contiguous memory

Approach

Train a RNN to act as a controller to interact with a memory matrix of N (arbitrary many) addresses.

Approach

Train a RNN to act as a controller to interact with a memory matrix of N (arbitrary many) addresses.

1. Content Lookup
   - Attention over memory defined by weightings $W \in \mathbb{R}^N$
   - Compare controller output with memory objects (auto-associative memory)
   - Allow partial matches (pattern completion)

Approach

Train a RNN to act as a controller to interact with a memory matrix of N (arbitrary many) addresses.

1. Content Lookup
    - Attention over memory defined by weightings $W \in \mathbb{R}^N$
    - Compare controller output with memory objects (auto-associative memory)
    - Allow partial matches (pattern completion)
2. Sequential Retrieval
    - Fill $L \in \{0,1\}^{2N}$ indexing temporal transitions
    - Shift operations defined by $LW, \ L^T W$

Approach

Train a RNN to act as a controller to interact with a memory matrix of N (arbitrary many) addresses.

1. Content Lookup
    - Attention over memory defined by weightings $W \in \mathbb{R}^N$
    - Compare controller output with memory objects (auto-associative memory)
    - Allow partial matches (pattern completion)
2. Sequential Retrieval
    - Fill $L \in \{0,1\}^{2N}$ indexing temporal transitions
    - Shift operations defined by $LW,\ L^T W$
3. Dynamic Allocation
    - Mark memory locations with $\{0,1\}$ to signal usage
    - Manipulate signals during R/W operations to enable reallocation
    - Generalization to unbounded memory

Controller

A deep long short-term memory network receiving

$$\boldsymbol{\mathcal{X}}_t = [\boldsymbol{x}_t; \boldsymbol{r}^1_{t-1}; \ldots \boldsymbol{r}^R_{t-1}]$$

and producing

$$(\boldsymbol{v}_t, \boldsymbol{\xi}_t) = \mathcal{N}([\boldsymbol{\mathcal{X}}_1; \ldots; \boldsymbol{\mathcal{X}}_T]; \theta)$$

where $\mathcal{N}$ a set of state equations and θ their trainable parameters.

Controller

A more detailed look into $\mathcal{N}$ and LSTMs:

$$\boldsymbol{i}_t^l = \sigma(W_i^l[\boldsymbol{\mathcal{X}}_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_i^l) \qquad \text{(input gate)}$$

$$\boldsymbol{f}_t^l = \sigma(W_f^l[\boldsymbol{\mathcal{X}}_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_f^l) \qquad \text{(forget gate)}$$

$$\boldsymbol{s}_t^l = \boldsymbol{f}_t^l \boldsymbol{s}_{t-1}^l + \boldsymbol{i}_t^l tanh(W_s^l[\boldsymbol{\mathcal{X}}_t; \boldsymbol{h}_{t_1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_s^l) \qquad \text{(state)}$$

$$\boldsymbol{o}_t^l = \sigma(W_o^l[\boldsymbol{\mathcal{X}}_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_o^l) \qquad \text{(output gate)}$$

$$\boldsymbol{h}_t^l = \boldsymbol{o}_t^l tanh(\boldsymbol{s}_t^l) \qquad \text{(hidden)}$$

$$\boldsymbol{v}_t = W_y[\boldsymbol{h}_t^1; \ldots; \boldsymbol{h}_t^L] \qquad \text{(output vector)}$$

$$\boldsymbol{\xi}_t = W_\xi[\boldsymbol{h}_t^1; \ldots; \boldsymbol{h}_t^L] \qquad \text{(interface vector)}$$

Controller