

Artificial Neural Networks

Lecture 8: Neural Networks for Algorithmic Learning

Tudor Berariu

tudor.berariu@gmail.com



Faculty of Automatic Control and Computers
University Politehnica of Bucharest

Lecture : 9th of December, 2015
Last Updated: 9th of December, 2015

Course Progress

- ① Introduction to Artificial Neural Networks
- ② Linear Discriminants; The Perceptron Algorithm
- ③ Feedforward Neural Networks; Backpropagation
- ④ Optimization Algorithms
- ⑤ Convolutional Neural Networks
- ⑥ Radial Basis Function Networks
- ⑦ Reinforcement Learning; **Recurrent Neural Networks**
- ⑧ Networks with external memory
 - **Neural Turing Machines**

Limitations of Classic Neural Models

- classic neural networks store all knowledge in the weights during training (no use of external memory resources)
- although RNNs are Turing-complete [SS95], they were not successful in modelling general algorithms (sequences of operations)
- recent models try to address these limitations:
 - Neural Turing Machines [GWD14]
 - Memory Networks [WCB14], [SSWF15]
 - Neural Networks that learn Algorithms [ZMJF15]

Today's Outline

1 Neural Turing Machines

Neural Turing Machines

Alex Graves, Greg Wayne, and Ivo Danihelka, *Neural turing machines*,
arXiv preprint arXiv:1410.5401 (2014)

<http://arxiv.org/abs/1410.5401>

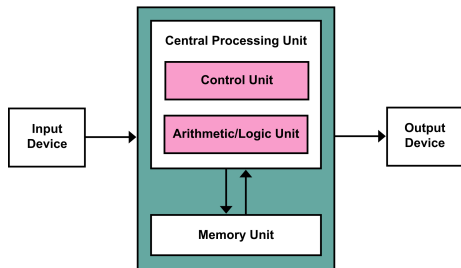
Today's Outline

- 1 Neural Turing Machines
 - The Model
 - Reading and Writing
 - Experiments and Results

Von Neumann



- elementary arithmetic operations
- logical flow control (branching)
- external memory to read from and write to

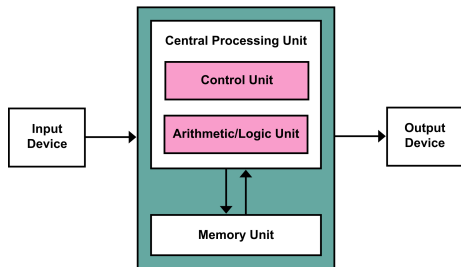


Von Neumann Architecture
(source: wikipedia.org)

Von Neumann



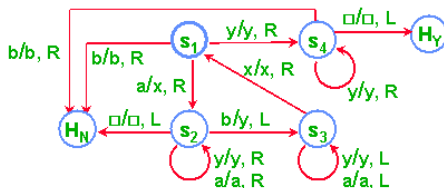
- elementary arithmetic operations
- logical flow control (branching)
not used in ML
- external memory to read from and write to
not used in ML



Von Neumann Architecture
(source: wikipedia.org)

The idea

- enrich neural networks with an external addressable memory
 - analogous to Turing's enrichment of FSA with an infinite tape



- Neural Turing Machines are differentiable end-to-end
 - they can be trained with gradient descent
- NTMs resemble the *working memory* in two aspects:
 - they solve tasks that require the application of rules to *rapidly-created variables*
 - memory is addressed by an attentional mechanism

Architecture

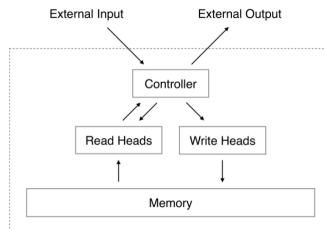


Figure 1: Neural Turing Machine Architecture. During each update cycle, the controller network receives inputs from an external environment and emits outputs in response. It also reads to and writes from a memory matrix via a set of parallel read and write heads. The dashed line indicates the division between the NTM circuit and the outside world.

Architecture of NTMs

Neural Turing Machines = network controller + memory bank

- network controller:
 - input vectors
 - output vectors
 - vectors that control read / write operations (*heads*)

Differentiable operations

- architecture needs to be differentiable
- solution: blurry reads and writes
- read and write operations interact with all memory locations
- the *heads* define a normalised weighting over all rows in memory

Today's Outline

- 1 Neural Turing Machines
 - The Model
 - Reading and Writing
 - Experiments and Results

Reading from the memory

- \mathbf{M}_t - the contents of the memory at time t
- $\mathbf{M}_t \in \mathbb{R}^{N \times M}$ - N vectors (rows) of size M
- \mathbf{w}_t - a weighting vector emitted by a head at time t
- the N elements of \mathbf{w}_t are normalised:

$$\sum_i^N w_t(i) = 1$$

$$0 \leq w_t(i) \leq 1, \forall i$$

- the read vector \mathbf{r}_t :

$$\mathbf{r}_t \leftarrow \sum_i^N w_t(i) \mathbf{M}_t(i)$$

Writing

write = erase + add

Writing

write = erase + add

- Given a weighting \mathbf{w}_t and an erase vector \mathbf{e}_t :

$$\tilde{\mathbf{M}}_t(i) \leftarrow \mathbf{M}_{t-1}(i) [\mathbf{1} - w_t(i)\mathbf{e}_t]$$

- if there are multiple erase heads, erasures can be done in any order

Writing

write = erase + add

- Given a weighting \mathbf{w}_t and an add vector \mathbf{a}_t :

$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\mathbf{a}_t$$

- if there are multiple add heads, operations can be performed in any order

Addressing Mechanisms

- How to address memory locations?

Addressing Mechanisms

- How to address memory locations?
 - content-based addressing (similar to Hopfield - next week)
 - location-based addressing
-
- content-based addressing is more general
 - both mechanisms are implemented concurrently

Addressing Mechanism

Previous
State



Controller
Outputs



Addressing by Content

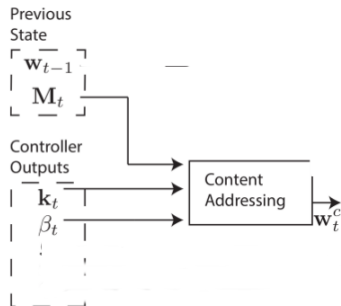
- each head produces a key vector \mathbf{k}_t
- the key is compared with all memories by a similarity measure $K[\cdot, \cdot]$

$$w_t^c(i) \leftarrow \frac{\exp(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(i)])}{\sum_j \exp(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(j)])}$$

- β_t - key strength
- K - cosine similarity:

$$K[\mathbf{u}, \mathbf{v}] = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}$$

Addressing Mechanism



Addressing by Location

- achieve:
 - simple iteration across locations
 - random access jumps
- idea: rotational shifts of a weighting
- steps:
 - 1 interpolation
 - 2 shifting
 - 3 sharpening

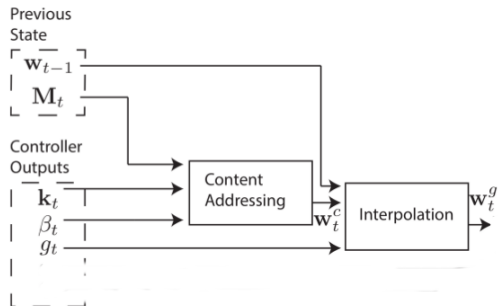
Interpolation

- each head emits a scalar interpolation gate: g_t

$$\mathbf{w}_t^g \leftarrow g_t \mathbf{w}_t^c + (1 - g_t) \mathbf{w}_{t-1}$$

- interpolates previous weighting with content-based addressing

Addressing Mechanism

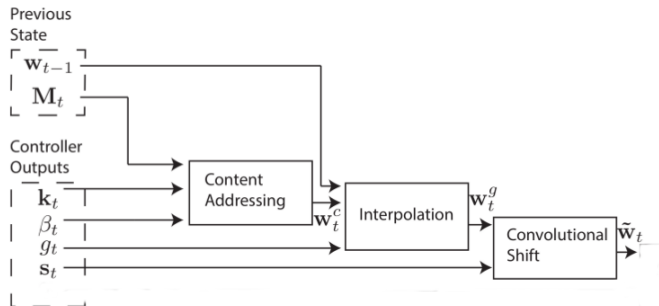


Shifting

- each head emits a *shift weighting* \mathbf{s}_t - a normalised distribution over allowed integer shifts
- approaches:
 - softmax
 - width one uniform distribution over shifts

$$\tilde{w}_t(i) \leftarrow \sum_{j=0}^{N-1} w_t^g(j) s_t(i-j)$$

Addressing Mechanism



Sharpening

- to combat dispersion of weightings, a sharpening step is applied:

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$

Addressing Mechanism

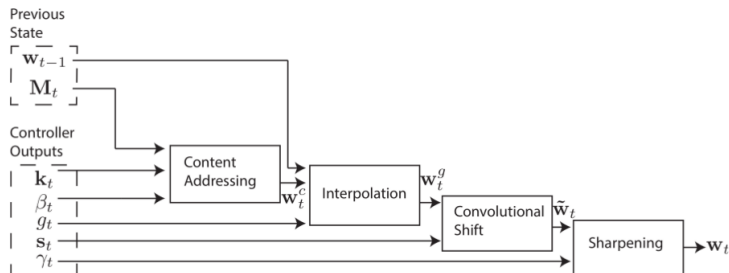


Figure 2: Flow Diagram of the Addressing Mechanism. The *key vector*, k_t , and *key strength*, β_t , are used to perform content-based addressing of the memory matrix, M_t . The resulting content-based weighting is interpolated with the weighting from the previous time step based on the value of the *interpolation gate*, g_t . The *shift weighting*, s_t , determines whether and by how much the weighting is rotated. Finally, depending on γ_t , the weighting is sharpened and used for memory access.

The Versatility of the Addressing System

- strict content-based addressing
- content addressing + shifting
- weighting from previous step can be rotated

The Controller Network

- free parameters:
 - number of memory locations
 - number of read / write heads
 - range of allowed shifts
 - feed-forward controller vs RNN controller

Today's Outline

- 1 Neural Turing Machines
 - The Model
 - Reading and Writing
 - Experiments and Results

Tasks

- tasks:
 - copy
 - repeat copy
 - associative recall
 - dynamic N-grams
 - priority-sort
- compared architectures:
 - NTM with a feedforward controller
 - NTM with a LSTM controller
 - standard LSTM network

Copy task

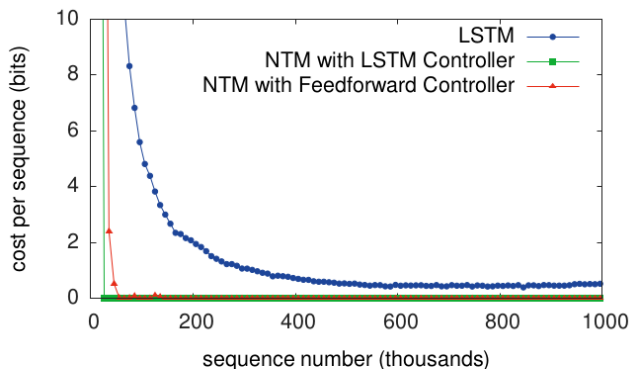
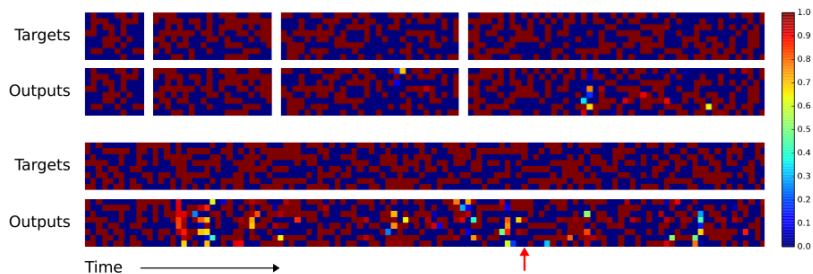


Figure 3: Copy Learning Curves.

Copy task



NTM Generalisation on the copy task

Copy task

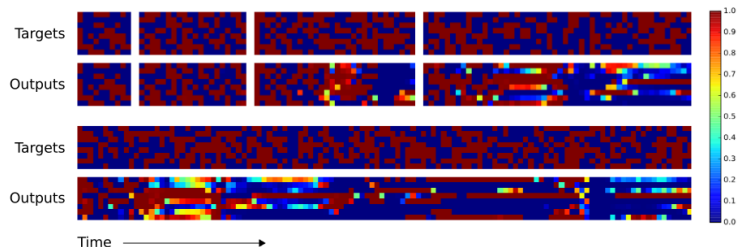
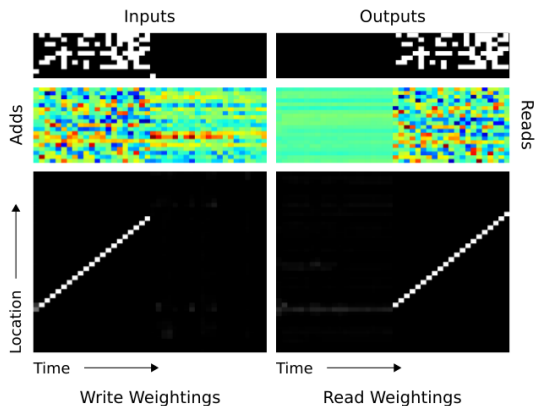


Figure 5: LSTM Generalisation on the Copy Task. The plots show inputs and outputs for the same sequence lengths as Figure 4. Like NTM, LSTM learns to reproduce sequences of up to length 20 almost perfectly. However it clearly fails to generalise to longer sequences. Also note that the length of the accurate prefix decreases as the sequence length increases, suggesting that the network has trouble retaining information for long periods.

Copy task



NTM Memory use during Copy task

Repeat Copy task

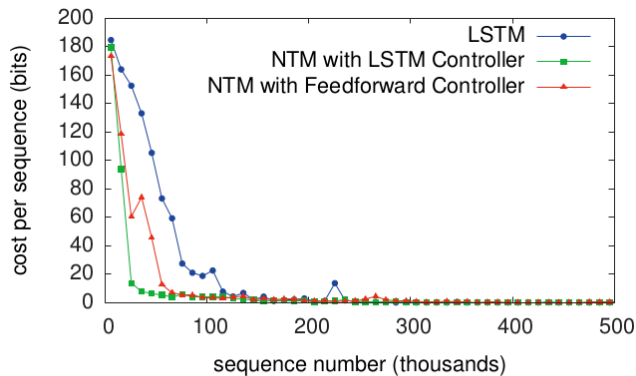
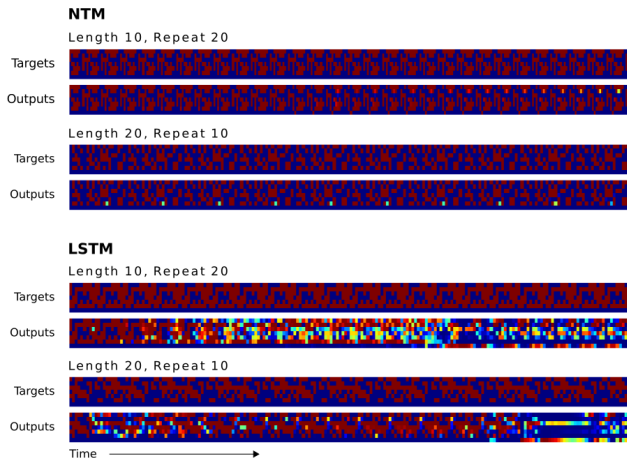


Figure 7: Repeat Copy Learning Curves.

Repeat Copy task



NTM and LSTM Generalisation for the Repeat Copy Task

Repeat Copy task

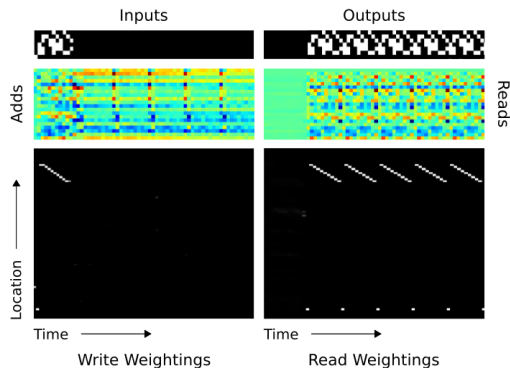


Figure 9: NTM Memory Use During the Repeat Copy Task. As with the copy task the network first writes the input vectors to memory using iterative shifts. It then reads through the sequence to replicate the input as many times as necessary (six in this case). The white dot at the bottom of the read weightings seems to correspond to an intermediate location used to redirect the head to the start of the sequence (The NTM equivalent of a *goto* statement).

Associative Recall task

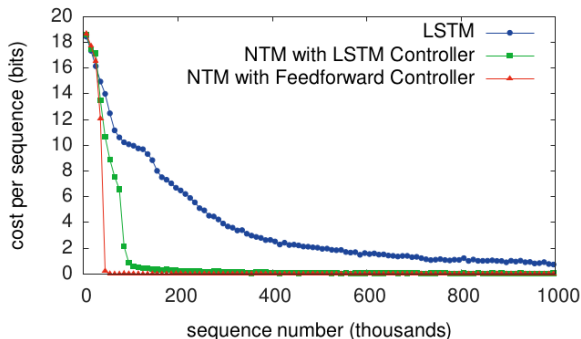


Figure 10: Associative Recall Learning Curves for NTM and LSTM.

Associative Recall task

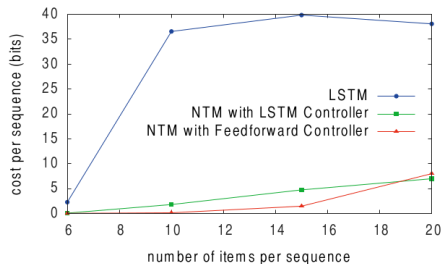
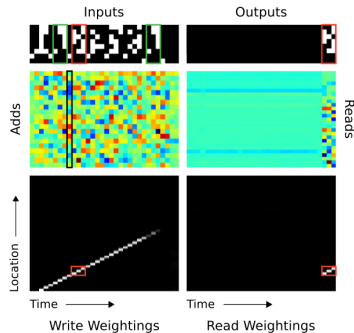


Figure 11: Generalisation Performance on Associative Recall for Longer Item Sequences.

The NTM with either a feedforward or LSTM controller generalises to much longer sequences of items than the LSTM alone. In particular, the NTM with a feedforward controller is nearly perfect for item sequences of twice the length of sequences in its training set.

Associative Recall task



NTM Memory Use During the Associative Recall Task

Priority Sorting task

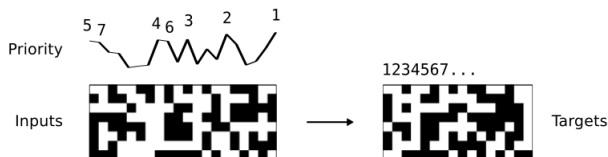


Figure 16: Example Input and Target Sequence for the Priority Sort Task. The input sequence contains random binary vectors and random scalar priorities. The target sequence is a subset of the input vectors sorted by the priorities.

Priority Sorting task

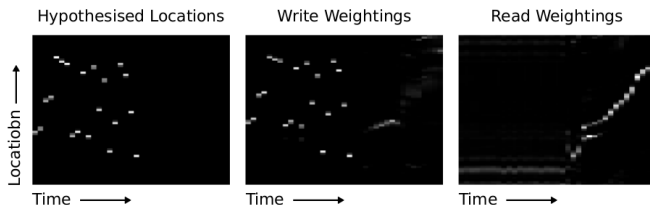


Figure 17: NTM Memory Use During the Priority Sort Task. Left: Write locations returned by fitting a linear function of the priorities to the observed write locations. Middle: Observed write locations. Right: Read locations.

Priority Sorting task

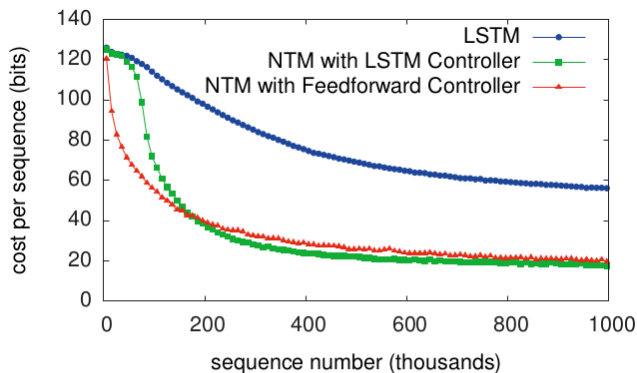


Figure 18: Priority Sort Learning Curves.

For the exam

For the exam you should be able to ...

- describe the architecture of Neural Turing Machines
- describe how a radial-basis function network works
- describe the two-stage training procedure for the RBFNs






Read ...

Alex Graves, Greg Wayne, and Ivo Danihelka, *Neural turing machines*,
arXiv preprint arXiv:1410.5401 (2014)

Today's Outline

2 References

References I

-  Alex Graves, Greg Wayne, and Ivo Danihelka, *Neural turing machines*, arXiv preprint arXiv:1410.5401 (2014).
-  H.T. Siegelmann and E.D. Sontag, *On the Computational Power of Neural Nets*, Journal of Computer and System Sciences **50** (1995), no. 1, 132–150.
-  Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus, *End-to-end memory networks*, arXiv preprint arXiv:1503.08895 (2015).
-  Jason Weston, Sumit Chopra, and Antoine Bordes, *Memory networks*, arXiv preprint arXiv:1410.3916 (2014).
-  Wojciech Zaremba, Tomas Mikolov, Armand Joulin, and Rob Fergus, *Learning simple algorithms from examples*, arXiv preprint arXiv:1511.07275 (2015).