NTM

Atef Chaudhury and Chris Cremer

Motivation

Memory is good

Working memory is key to many tasks

- Humans use it everyday
- Essential to computers (core to Von Neumann architecture/Turing Machine)

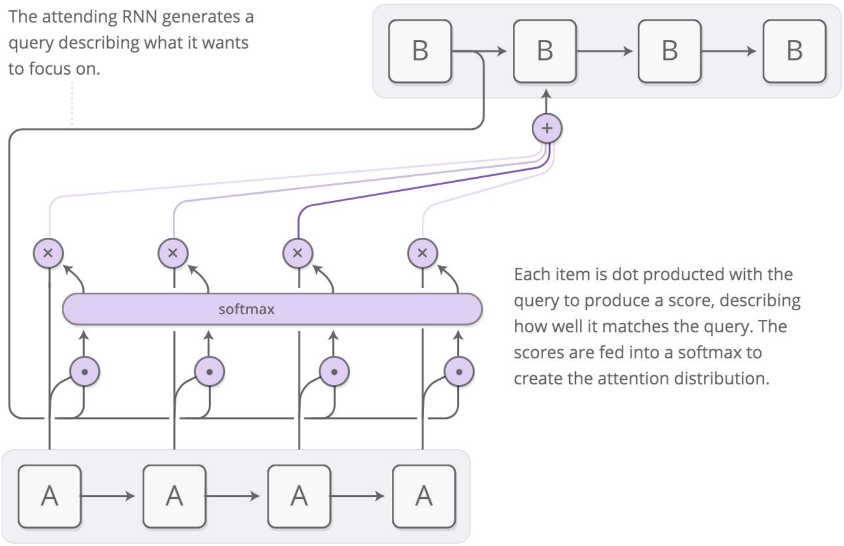
Why not incorporate it into NNs which would let us do cool things

What about RNNs?

Shown to be Turing-Complete

Practically not always the case hence there are ways to improve

- (e.g. attention for translation)



https://distill.pub/2016/augmented-rnns/

Core idea

Similar to attention, external memory could help for some tasks

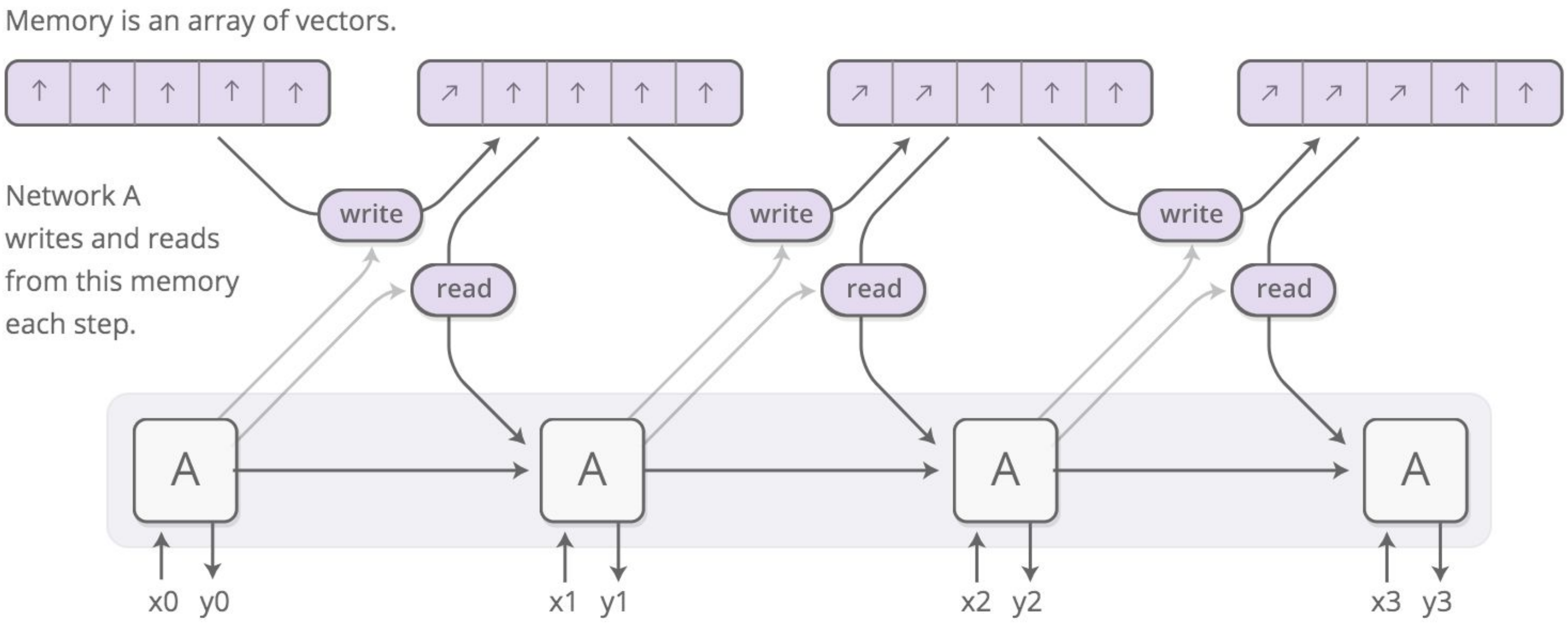
- e.g. copy sequences with lengths longer than seen at training

One module does not have to both store data and learn logic (the architecture introduces a bias towards separation of tasks)

- hope is that one module learns generic logic while other tracks values

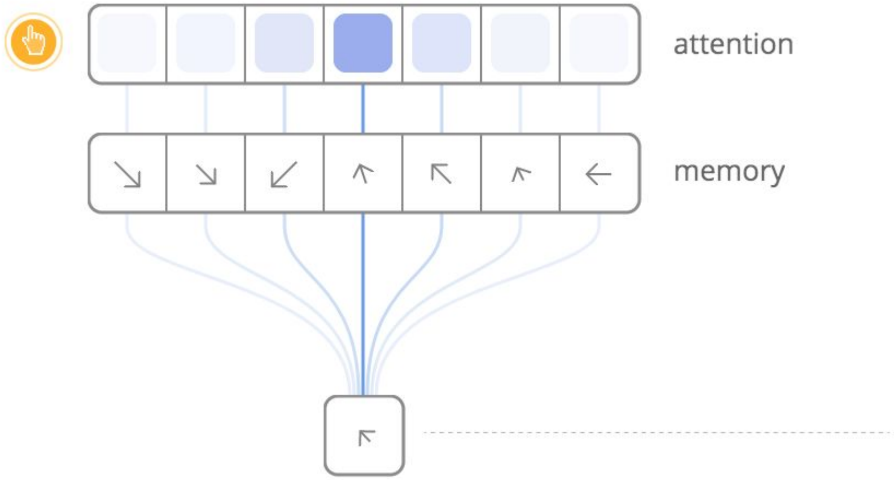
Architecture

Overview



https://distill.pub/2016/augmented-rnns/

Soft-attention reading



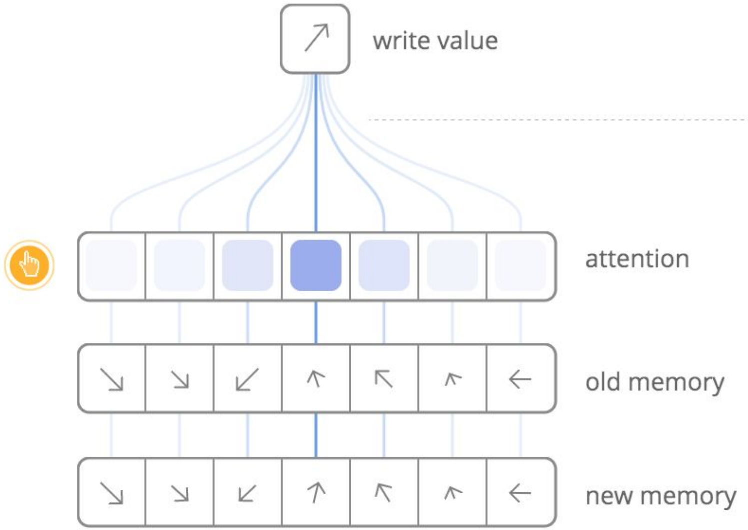
The RNN gives an attention distribution which describe how we spread out the amount we care about different memory positions.

The read result is a weighted sum.

$$r \leftarrow \sum_i a_i M_i$$

https://distill.pub/2016/augmented-rnns/

Soft-attention writing



Instead of writing to one location, we write everywhere, just to different extents.

The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

$$M_i \leftarrow a_i w + (1 - a_i) M_i$$

https://distill.pub/2016/augmented-rnns/

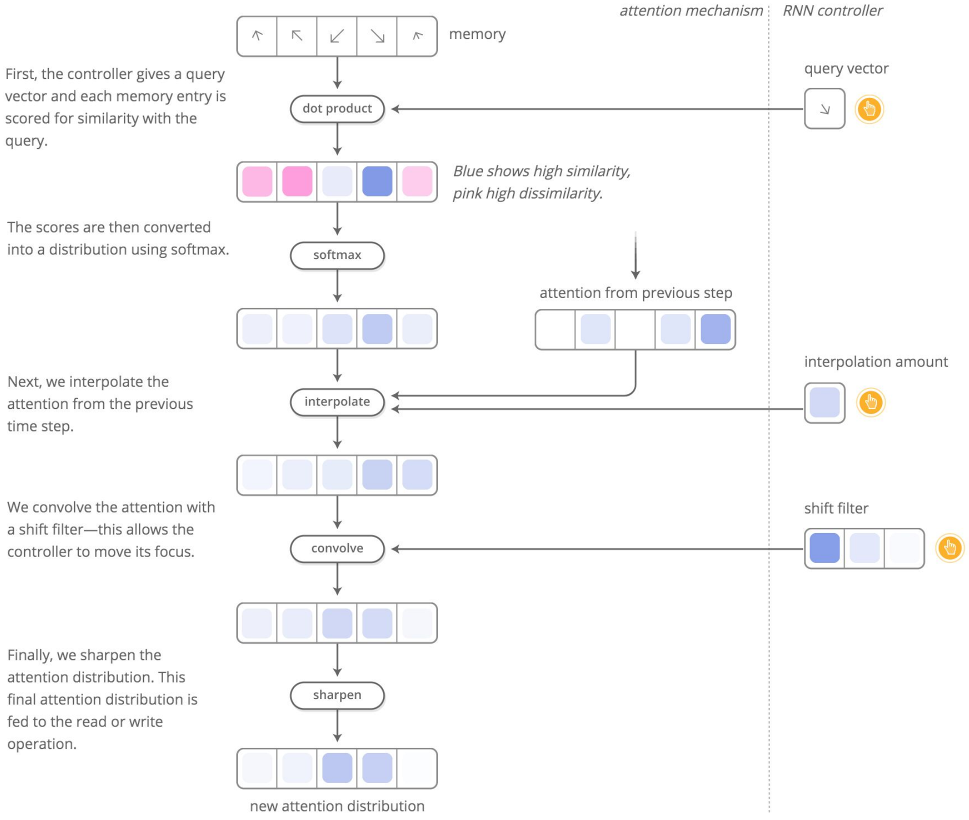
Addressing

Content-based

- (cosine similarity + softmax between key vector and memory)

Location based

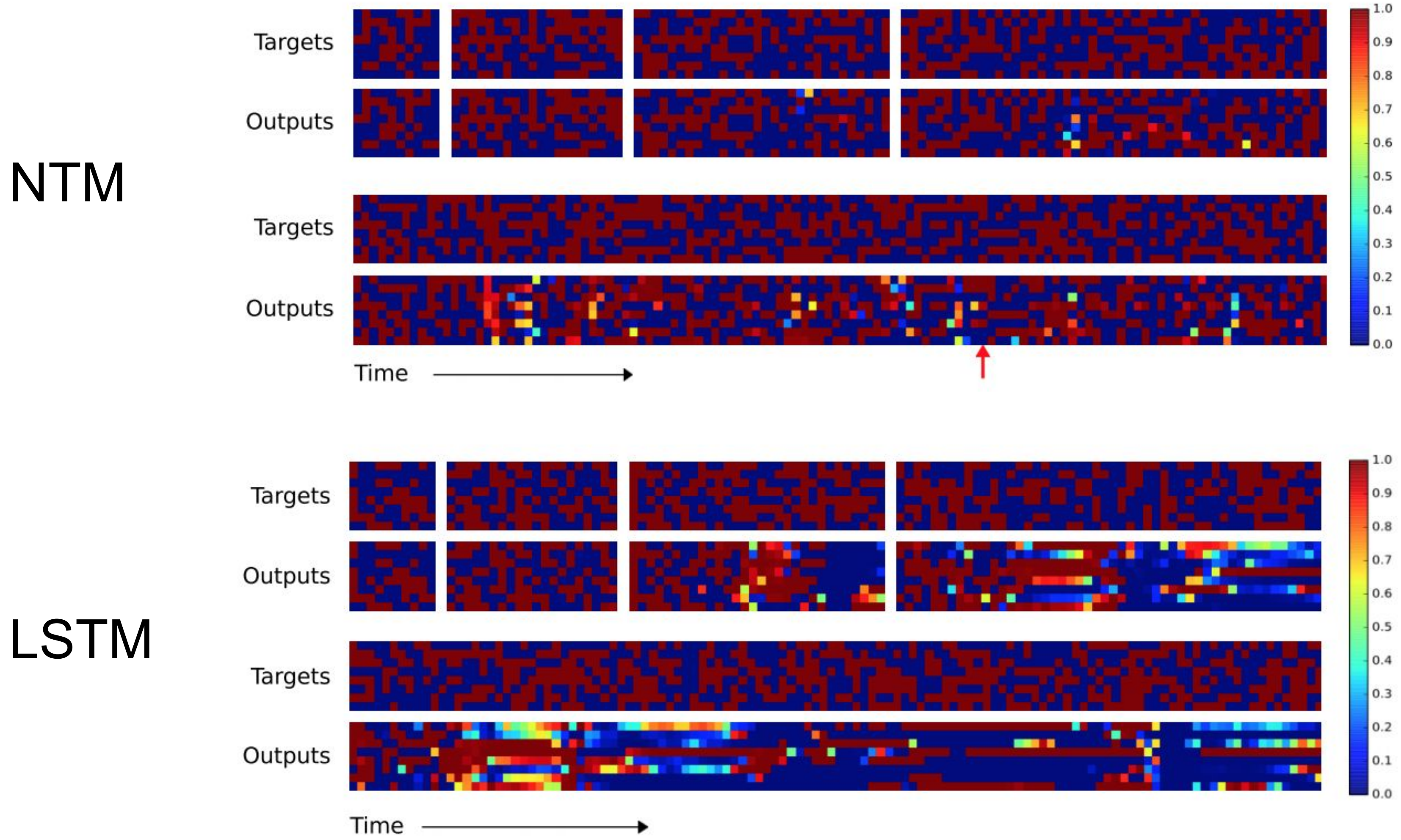
- Interpolation with last weight vector + shift operation



Results

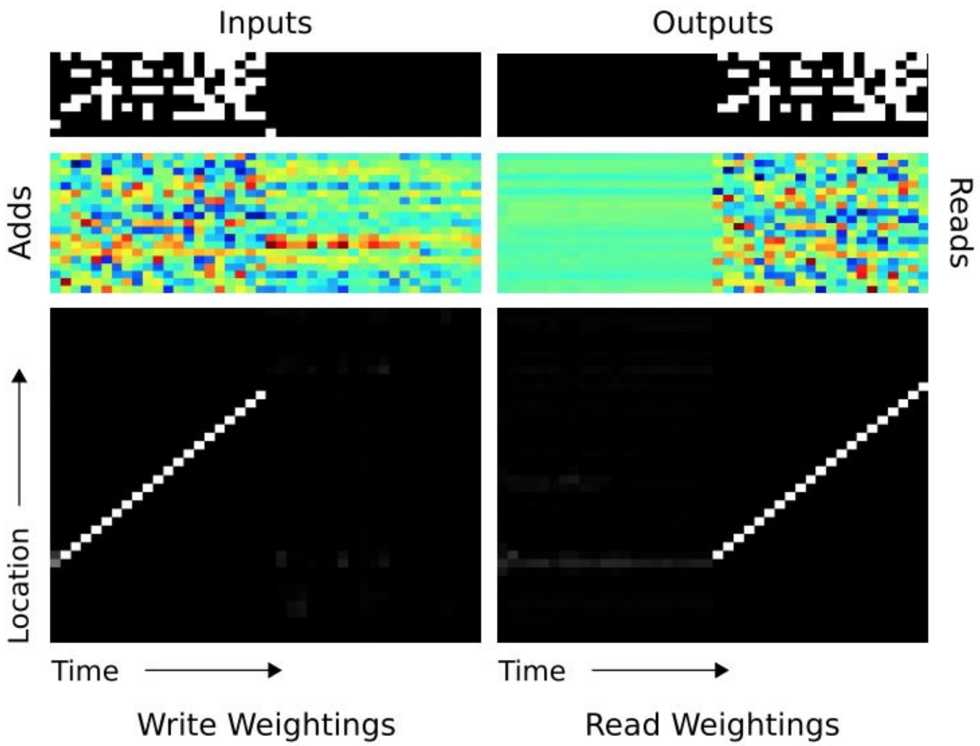
Copying

Feed an input sequence of binary vectors, and then expected result is same sequence (output after the entire sequence has been fed in)



NTM
Targets
Outputs
Targets
Outputs
Time
LSTM
Targets
Outputs
Targets
Outputs
Time
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

What's going on?



Other tasks

Repeated copy (for-loop), Adjacent elements in sequence (associative memory), Dynamic N-grams (counting), Sorting

Memory accesses work as you would expect indicating that algorithms are being learned

Generalizes to longer sequences when the LSTM on its own does not

- All with less parameters as well

Final notes

Influenced several models: Neural Stacks/Queues, MemNets, MANNs

Extensions

- Neural GPU to reduce sequential memory access
- DNC for more efficient memory usage

Discrete Read/Write

Sample distribution over memory addresses instead of weighted sum

Why?

- Constant time addressing
- Sharp retrieval

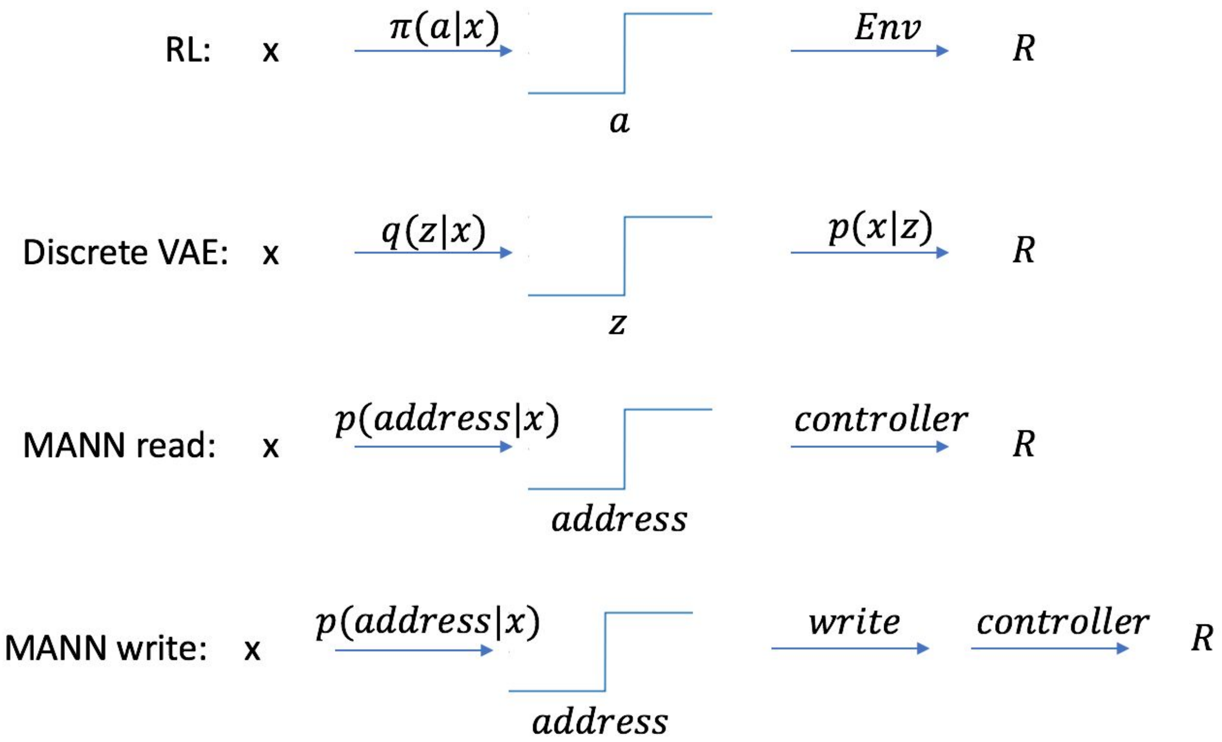
Papers: RL-NTM (2015), Dynamic-NTM (2016)

Unifying Discrete Models

	Sample	Gradient
RL:	$x \xrightarrow{\pi(a\|x)} a \xrightarrow{Env} R$	$\nabla_\pi \mathrm{E}_\pi[R] = \mathrm{E}_\pi[R\ \nabla_\pi \log \pi(a\|x)]$
Discrete VAE:	$x \xrightarrow{q(z\|x)} z \xrightarrow{p(x\|z)} R$	$\nabla_q \mathrm{E}_q[R] = \mathrm{E}_q\left[R\ \nabla_q \log q(z\|x)\right]$
MANN read:	$x \xrightarrow{p(address\|x)} address \xrightarrow{controller} R$	$\nabla_p \mathrm{E}_p[R] = \mathrm{E}_p\left[R\ \nabla_p \log p(address\|x)\right]$

Unifying Discrete Models

Sample



Gradient

$$\nabla_\pi \mathrm{E}_\pi[R] = \mathrm{E}_\pi[R \, \nabla_\pi \log \pi(a|x)]$$

$$\nabla_q \mathrm{E}_q[R] = \mathrm{E}_q\left[R \, \nabla_q \log q(z|x)\right]$$

$$\nabla_p \mathrm{E}_p[R] = \mathrm{E}_p\left[R \, \nabla_p \log p(address|x)\right]$$

RL-NTM

Variance Reduction

- future rewards back-propagation
- online baseline prediction
- offline baseline prediction

Curriculum learning

Direct access controller

RL-NTM - Variance Reduction

Future rewards back-propagation

- Use sum of rewards starting from the current time step

$$R_t = \sum_{i=t}^{T} r_i$$

- Instead of the sum of rewards over the entire episode

$$R_t = \sum_{i=1}^{T} r_i$$

RL-NTM - Variance Reduction

Online baseline prediction

$$R_t = \sum_{i=t}^{T} r_i - b_i$$

where $b_t = E[R_t]$

RL-NTM - Variance Reduction

Offline baseline prediction

- Use baseline LSTM to minimize $(r_i - b_i)^2$
- Biased

$$R_t = \sum_{i=t}^{T} r_i - b_i - b_i(a_{1:T})$$

RL-NTM - Direct Access

- All the tasks considered involved rearranging the input symbols in some way
  - For example: reverse a sequence, copy a sequence
- Controller benefits from a built-in mechanism that can directly copy an input to memory or to the output
- Drawback: domain specific

Difficulty Curriculum

RL–NTM unable to solve tasks when trained on difficult problem instances

- Complexity of problem instance measured by the maximal length of the desired output

To succeed, it required a curriculum of tasks of increasing complexity

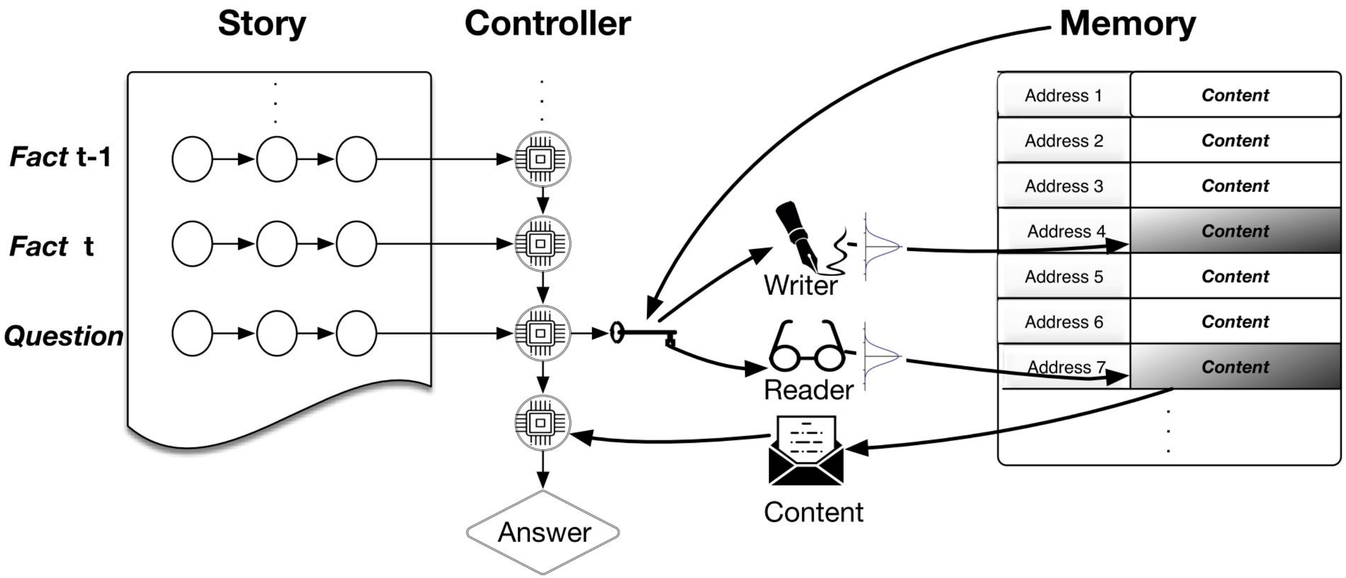
- During training, maintain a distribution over the task complexity
- Shift the distribution over the task complexities whenever the performance of the RL–NTM exceeds a threshold

RL-NTM - Results

Task \ Controller	LSTM	Direct Access
Copy	✓	✓
DuplicatedInput	✓	✓
Reverse	×	✓
ForwardReverse	×	✓
RepeatCopy	×	✓

Table 4: Success of training on various task for a given controller.

Dynamic-NTM



Story
Controller
Memory
Fact t-1
Fact t
Question
Writer
Reader
Content
Answer
Address 1
Content
Address 2
Content
Address 3
Content
Address 4
Content
Address 5
Content
Address 6
Content
Address 7
Content

Dynamic-NTM

Transition from soft/continuous to hard/discrete addressing

- For each minibatch, the controller stochastically decides to choose either to use the discrete or continuous weights
- Have hyperparameter determine the probability of discrete vs continuous
- Hyperparameter is annealed during training

$$\mathbf{w}_t = \pi_n \bar{\mathbf{w}}_t + (1 - \pi_n)\tilde{\mathbf{w}}_t$$

D-NTM

Variance Reduction

- Global baseline + variance normalization

$$\tilde{R}(\mathbf{x}) = \frac{R(\mathbf{x}) - b}{\sqrt{\sigma^2 + \epsilon}}$$

where b is the running average and σ is the standard deviation of R

- Input-dependent baseline

$$\bar{R}(\mathbf{x}) = \tilde{R}(\mathbf{x}) - b(\mathbf{x}).$$

D-NTM - Results

bAbI Question answering - reads a sequence of factual sentences followed by a question, all of which are given as natural language sentences.

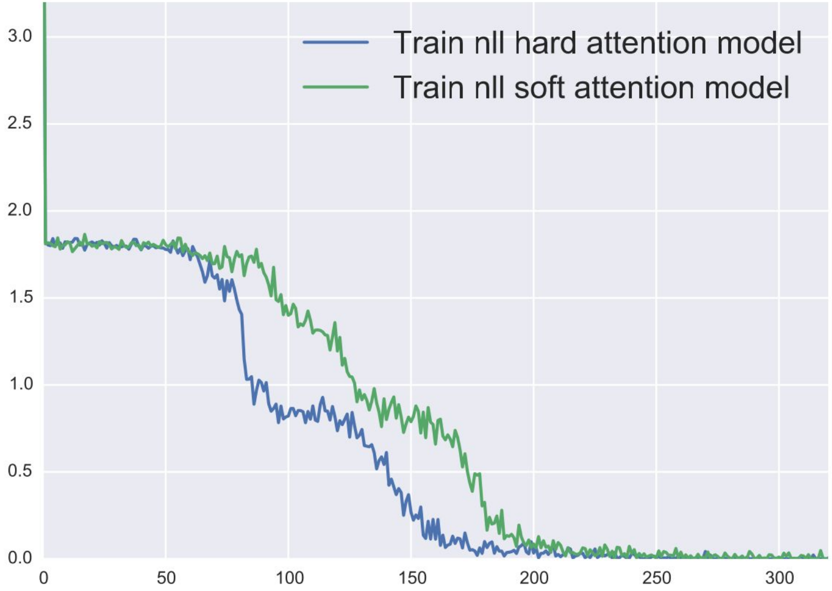
LSTM controller

Task	LSTM	MemN2N	DMN+	1-step LBA* NTM	1-step CBA NTM	1-step Soft D-NTM	1-step Discrete D-NTM	3-steps LBA* NTM	3-steps CBA NTM	3-steps Soft D-NTM	3-steps Discrete D-NTM
1	0.00	0.00	0.00	16.30	16.88	5.41	6.66	0.00	0.00	0.00	0.00
2	81.90	0.30	0.30	57.08	55.70	58.54	56.04	61.67	59.38	46.66	62.29
3	83.10	2.10	1.10	74.16	55.00	74.58	72.08	83.54	65.21	47.08	41.45
4	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	1.20	0.80	0.50	1.46	20.41	1.66	1.04	0.83	1.46	1.25	1.45
6	51.80	0.10	0.00	23.33	21.04	40.20	44.79	48.13	54.80	20.62	11.04
7	24.90	2.00	2.40	21.67	21.67	19.16	19.58	7.92	37.70	7.29	5.62
8	34.10	0.90	0.00	25.76	21.05	12.58	18.46	25.38	8.82	11.02	0.74
9	20.20	0.30	0.00	24.79	24.17	36.66	34.37	37.80	0.00	39.37	32.50
10	30.10	0.00	0.00	41.46	33.13	52.29	50.83	56.25	23.75	20.00	20.83
11	10.30	0.10	0.00	18.96	31.88	31.45	4.16	3.96	0.28	30.62	16.87
12	23.40	0.00	0.00	25.83	30.00	7.70	6.66	28.75	23.75	5.41	4.58
13	6.10	0.00	0.00	6.67	5.63	5.62	2.29	5.83	83.13	7.91	5.00
14	81.00	0.10	0.20	58.54	59.17	60.00	63.75	61.88	57.71	58.12	60.20
15	78.70	0.00	0.00	36.46	42.30	36.87	39.27	35.62	21.88	36.04	40.26
16	51.90	51.80	45.30	71.15	71.15	49.16	51.35	46.15	50.00	46.04	45.41
17	50.10	18.60	4.20	43.75	43.75	17.91	16.04	43.75	56.25	21.25	9.16
18	6.80	5.30	2.10	3.96	47.50	3.95	3.54	47.50	47.50	6.87	1.66
19	90.30	2.30	0.00	75.89	71.51	73.74	64.63	61.56	63.65	75.88	76.66
20	2.10	0.00	0.00	1.25	0.00	2.70	3.12	0.40	0.00	3.33	0.00
Avg.Err.	36.41	4.24	**2.81**	31.42	33.60	29.51	**27.93**	32.85	32.76	24.24	**21.79**

FF controller

Task	FF Soft D-NTM	FF Discrete D-NTM	FF Discrete* D-NTM
1	4.38	81.67	14.79
2	27.5	76.67	76.67
3	71.25	79.38	70.83
4	0.00	78.65	44.06
5	1.67	83.13	17.71
6	1.46	48.76	48.13
7	6.04	54.79	23.54
8	1.70	69.75	35.62
9	0.63	39.17	14.38
10	19.80	56.25	56.25
11	0.00	78.96	39.58
12	6.25	82.5	32.08
13	7.5	75.0	18.54
14	17.5	78.75	24.79
15	0.0	71.42	39.73
16	49.65	71.46	71.15
17	1.25	43.75	43.75
18	0.24	48.13	2.92
19	39.47	71.46	71.56
20	0.0	76.56	9.79
Avg.Err.	**12.81**	68.30	37.79

Learning Curves



The discrete attention D-NTM converges faster than the continuous-attention model

- Difficulty of learning continuous-attention is due to the fact that learning to write with soft addressing can be challenging.

TARDIS (2017)

Wormhole-Connections help with vanishing gradient

Uses Gumbel-Softmax

Improved results

Takeaways

Learning memory-augmented models with discrete addressing is challenging

Especially writing to memory

Improved variance reduction techniques are required

Thanks