DISCRETIZING NEURAL TURING MACHINES

James Gleeson & Grant Watson
NTMs: a fully differentiable computer!

Training input = program input
Training output = desired program output

Learn the locations of memory to read/write

\[ r_t \leftarrow \sum_i w_t(i)M_t(i) \]

Soft attention during reads/writes over the NTM memory cells

- NTM is built from fully differentiable computations; learnable via backprop:

As we train over many input/output pairs...

Simple memory copy task: Learning to read/write sequentially
NTMS (2): LEARNING TO ADDRESS MEMORY

*Soft attention* during reads/writes over the NTM memory cells

- NTM is built from **fully differentiable computations**; learnable via backprop:

E.g. **Learning linear addressing**

\[
\begin{align*}
\text{Previous State} & \quad [0.1, 0.8, 0.1] \\
M_t & \\
\text{Controller Outputs} & \\
\text{Content Addressing} & \quad \text{Interpolation} & \quad \text{Convolutional Shift} & \quad \text{Sharpening} & \quad w_t \\
\text{Controller Outputs} & \\
\text{Reads memory at the next location} \\
\end{align*}
\]

\[
\begin{align*}
M_t & = r_t \\
& \leftarrow \sum_i w_t(i) M_t(i) \\
& [0.1, 0.8, 0.1] \gg 1 \\
& [0.1, 0.1, 0.8] \quad w_t \\
& [0.1, 0.8, 0.1] \quad w_t \\
\end{align*}
\]
NTMS (2): LEARNING TO ADDRESS MEMORY

- Each timestep/clock-cycle, the NTM must read/write all of memory! 

**KEY PROJECT IDEA:**
Allow NTM to read/write only a single memory cell at each timestep/clock-cycle

Just like a real computer!
*Scales better*

⇒ More suitable to low-level hardware implementations (e.g. FPGAs, TPUs, etc.)
RELATED WORK

D-NTM / dynamic-NTM: (Gulcehre et al., 2017):
Instead of a weighted combination of memory cells, sample which memory cell to read/write

\[
\begin{align*}
   w_t &= [0.1, 0.8, 0.1] \\
   i &\sim \text{Categorical}(w_t) \\
   r_t &\leftarrow M_t[i]
\end{align*}
\]

Limitations:
Sampling makes D-NTM not fully differentiable;

Work-around: they use REINFORCE to train D-NTM.
+ many tweaks to combat this high variance gradient estimator

Other work: TARDIS (Gulcehre et al., 2017), RL-NTM (Zaremba & Sutskever, 2015)
DISCRETIZING NEURAL TURING MACHINES

Figure 1)

\[
\hat{\nabla}_{RELAX} = \left[ f(b) - c_\phi (\tilde{z}) \right] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi (z) - \frac{\partial}{\partial \theta} c_\phi (\tilde{z})
\]

\[
b = H(z), \; z \sim p(z|\theta), \; \tilde{z} \sim p(z|b, \theta)
\]
DISCRETIZING NEURAL TURING MACHINES

Gradient estimators to try:
- REINFORCE
- REBAR
- RELAX

Figure 1)
PROGRESS RT!

Successfully reproduced prior work:
(1) NTM: copy task performance
(2) Gradient estimators: applying REINFORCE/RELAX/REBAR to a toy Bernoulli(θ) problem

Training convergence:
Memory copy task

Bit-wise cross-entropy loss

Training time (seconds)

Target output

NTM Output
Figure 2a) Memory copy task

Training convergence:
Memory copy task

Bit-wise cross-entropy loss

Sanity check.
Simple task that the NTM should be able to learn quickly.
Figure 2b) bAbI question/answer task

Training convergence:

bAbI question/answer task

Task 15: Basic Deduction
Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? A: wolves

Can NTM be trained on difficult tasks with better accuracy than REINFORCE?