Solutions to Exercise 4

Neural Turing Machines

Reliable and Interpretable Artificial Intelligence 2017
ETH Zürich

October 30, 2017

1 Description of the NTM

The NTM consists of a controller neural network that manipulates a memory tape via one read head and one write head. For simplicity we omit the conntent-based addressing from the original paper [1]. Execution at timestep t is described by the following variables:

		$\boldsymbol{M}_t$ – tape state	
$\boldsymbol{x}_t$ – input	$\boldsymbol{q}_t$ – controller state	$\boldsymbol{r}_t$ – read head position	$\boldsymbol{e}_t$ – erase mask
$\boldsymbol{y}_t$ – output	$\boldsymbol{m}_t$ – read head value	$\boldsymbol{w}_t$ – write head position	$\boldsymbol{a}_t$ – add vector

The input and the output range in respectively the vector spaces $X = \mathbb{R}^{d_x}$ and $Y = \mathbb{R}^{d_Y}$ whose dimensions depend on the task the NTM needs to solve. The tape contains c memory cells with each cell containing a vector in $A = \mathbb{R}^n$, that is, $\boldsymbol{M}_t \in \mathbb{R}^{c \times n}$. Here, the number c can change from one run of the NTM to another, while n is part of the model and thus fixed accross all runs. The controller state is a vector in $Q = \mathbb{R}^{d_Q}$, whose dimension is another parameter of the model. The head positions are probability vectors $\boldsymbol{r}_t, \boldsymbol{w}_t \in [0,1]^{\{1..c\}}$ over the possible cell addresses. The erase mask and the add vector range in A and are used to modify individual cells.

The controller network consists of six subnetworks (each with their own weights):

1. $\varphi\colon X \times A \times Q \to Q$ — a recurrent neural network unit (e.g., an LSTM unit)
2. $\psi_y\colon Q \to Y$ — produces outputs
3. $\psi_e, \psi_a\colon Q \to A$ — set the erase mask and the add vector (where $\psi_e(q) \in [0,1]^n$)
4. $\chi_r, \chi_w\colon Q \to [0,1]^{\{-1,0,+1\}}$ — indicate the change head position

The NTM uses the following components to read and write, and to move the heads:

1. $\texttt{RD}(\boldsymbol{M}, \boldsymbol{r})$ equals the expected value of $\boldsymbol{M}[\xi]$ (the row-vector of $\boldsymbol{M}$ at position ξ), where ξ is a random variable with probability density function $\boldsymbol{r}$.

2. $\texttt{WR}(\boldsymbol{M}, \boldsymbol{w}, \boldsymbol{e}, \boldsymbol{a})$ equals the expected value of the following row-vector modification:

$$\boldsymbol{M}[\xi \mapsto \boldsymbol{M}[\xi] \odot (\mathbf{1} - \boldsymbol{e}) + \boldsymbol{a}],$$

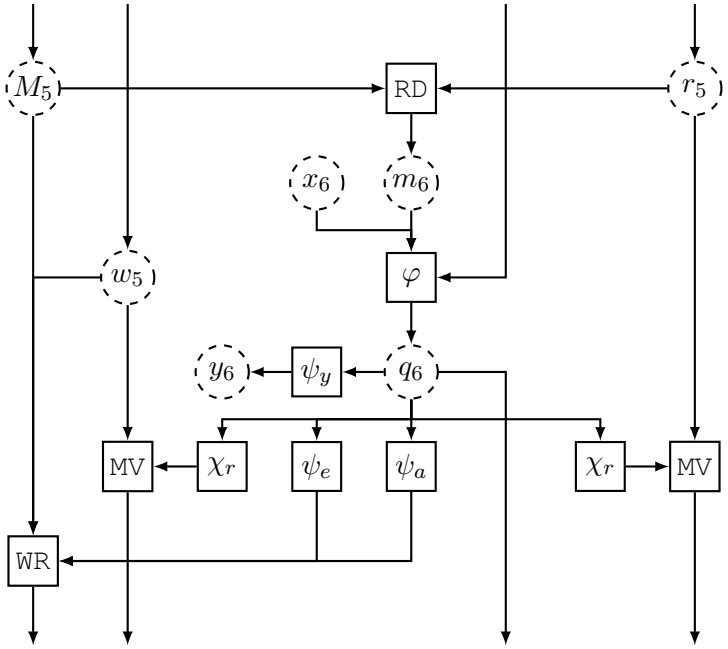
   where ξ is a random variable with a probability density function $\boldsymbol{w}$, $\mathbf{1}$ is the all ones vector $(1, 1, 1, \ldots, 1) \in \mathbb{R}^n$ and $\odot$ is pointwise multiplication.

3. $\texttt{MV}(\boldsymbol{l}, \boldsymbol{d})$ equals the pdf of the sum $\xi + \eta$ of a random variable ξ with distribution $\boldsymbol{l} \in [0, 1]^{\{1..c\}}$ and a random variable η with distribution $\boldsymbol{d} \in [0, 1]^{\{-1,0,+1\}}$.

An execution of the NTM on a given input $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$ is described by the equations:

$$\begin{array}{lll}
\boldsymbol{m}_t = \texttt{RD}(\boldsymbol{M}_{t-1}, \boldsymbol{r}_{t-1}) & & \boldsymbol{M}_t = \texttt{WR}(\boldsymbol{M}_{t-1}, \boldsymbol{w}_{t-1}, \boldsymbol{e}_t, \boldsymbol{a}_t) \\
\boldsymbol{q}_t = \varphi(\boldsymbol{x}_t, \boldsymbol{m}_t, \boldsymbol{q}_{t-1}) & \boldsymbol{e}_t = \psi_e(\boldsymbol{q}_t) & \boldsymbol{r}_t = \texttt{MV}(\boldsymbol{r}_{t-1}, \chi_r(\boldsymbol{q}_t)) \\
\boldsymbol{y}_t = \psi_y(\boldsymbol{q}_t) & \boldsymbol{a}_t = \psi_a(\boldsymbol{q}_t) & \boldsymbol{w}_t = \texttt{MV}(\boldsymbol{w}_{t-1}, \chi_w(\boldsymbol{q}_t))
\end{array}$$

Figure 1: A slice of an unfolded NTM (timestep 6).



2 Solutions to the problems

Problem 1.

1. What parts of RNNs are reused in NTMs, and what is new in the NTMs?

   The NTM reuses as a module an RNN unit network, for example an LSTM unit. In our description of the NTM, this unit is denoted φ. This way we can think of the core NTM wiring (Figure 1) as an RNN unit that wraps an existing RNN unit.

   The new thing in the NTM is the ability to manipulate a variable-sized external memory, in contrast to classic RNNs, where the extra memory is a fixed-width vector. Thus in theory, the NTM can capture arbitrarily long input-output dependencies, as long as they fit into the memory.

2. On what tasks the NTMs would perform better than vanilla RNNs?

   Tasks that involve many long-term dependencies.

3. What makes the NTMs more interpetable than RNNs?

   The NTMs typically learn "algorithms" that we can at least partially recognize from the way the machine interacts with its external memory. The way a classic RNN manipulates its hidden state is usually obscure.

4. What tasks would be difficult for NTMs?

   Any algorithmic tasks that are difficult for non-neural Turing Machines. An example would be the task of indexing an array with a sequence of random numbers: since indices in the sequence can be far apart, the Turing Machine would need much time in order to move its heads to the appropriate place.

Problem 2.

1. Run the copy example from the lecture's Slide 7 through an NTM. Note down the dimensions of the vectors and matrices involved. You can fabricate the values that the controller network uses to control the memory.

   Let us assume, for simplicity, that our set of symbols is $A = \{-, \rightarrow, \$, 1, 2, 3\}$. We will represent each symbol $x \in A$ by a one-hot probability vector over A:

$$oh(x)_i = \begin{cases} 1 & \text{if } i = x \\ 0 & \text{otherwise.} \end{cases}$$

For example the terminating symbol \$ is represented by the vector

$$oh(\text{`\$'}) = \begin{array}{|c|c|c|c|c|c|} \hline - & \rightarrow & \$ & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ \hline \end{array}$$

Now, recall that the NTM can simulate a classic Turing Machine by focusing its heads on a single location. Thus, the TM run in the slides corresponds to an NTM run where the heads at each timestep are fully focused.

2. Assume that initially the head locations are concentrated around location 1. Show how the head locations might become unconcentrated, and also the effect that losing concentration has on the NTM output.

   This could happen, for example, if at step 1 the controller leaves the write head at position 1 with pobability $\frac{1}{2}$ and moves it to position 2 again with probability $\frac{1}{2}$. Let us assume, for simplicity, that all the subsequent head shifts equal $+1$ with probability 1, that is, they are strictly to the right. We additionally assume that during the first phase, the controller modifies the tape by:

   a) always erasing with the null vector $\mathbf{0}$;

   b) adding the current input $oh(c)$, where $c = \text{`1'}, \text{`2'}, \text{`3'}$ for steps $i = 1, 2, 3$.

   We get the following trace during the writing phase:

$$\boldsymbol{w}_0 = \begin{array}{|c|c|c|} \hline 1.0 & 0.0 & 0.0 \\ \hline \end{array} \qquad \boldsymbol{M}_0 = \begin{bmatrix} oh(\text{`-'}) \\ oh(\text{`-'}) \\ oh(\text{`-'}) \end{bmatrix}$$

$$\boldsymbol{w}_1 = \begin{array}{|c|c|c|} \hline 0.5 & 0.5 & 0.0 \\ \hline \end{array} \qquad \boldsymbol{M}_1 = \begin{bmatrix} oh(\text{`1'}) \\ oh(\text{`-'}) \\ oh(\text{`-'}) \end{bmatrix}$$

$$\boldsymbol{w}_2 = \begin{array}{|c|c|c|} \hline 0.0 & 0.5 & 0.5 \\ \hline \end{array} \qquad \boldsymbol{M}_2 = \begin{bmatrix} \frac{1}{2}oh(\text{`1'}) + \frac{1}{2}oh(\text{`2'}) \\ \frac{1}{2}oh(\text{`-'}) + \frac{1}{2}oh(\text{`2'}) \\ 1oh(\text{`-'}) + 0oh(\text{`2'}) \end{bmatrix}$$

$$\boldsymbol{w}_3 = \begin{array}{|c|c|c|} \hline 0.0 & 0.0 & 0.5 \\ \hline \end{array} \qquad \boldsymbol{M}_3 = \begin{bmatrix} \frac{1}{2}oh(\text{`1'}) + \frac{1}{2}oh(\text{`2'}) + 0oh(\text{`3'}) \\ \frac{1}{4}oh(\text{`-'}) + \frac{1}{4}oh(\text{`2'}) + \frac{1}{2}oh(\text{`3'}) \\ \frac{1}{2}oh(\text{`-'}) + 0oh(\text{`2'}) + \frac{1}{2}oh(\text{`3'}) \end{bmatrix}$$

   Assuming that the read head stays focused throughout the execution, the NTM output will match the contents of the tape ($\boldsymbol{M}_3$), which is not a copy of the input.

Steps 2 and 3 above are non-trivial, and we will explain how step 2 is obtained. The outer product of the write head position $\boldsymbol{w}_t$ and the erase vector $\boldsymbol{e}_t$ gives us the fraction which should be erased from every memory cell:

$$\begin{bmatrix}0.5\\0.5\\0\end{bmatrix} \times \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5\\0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5\\0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Let's call the above matrix E. Then the ereasure leaves us with the matrix:

$$\begin{bmatrix}oh(\text{`1'})\\oh(\text{`-'})\\oh(\text{`-'})\end{bmatrix} - E \odot \begin{bmatrix}oh(\text{`1'})\\oh(\text{`-'})\\oh(\text{`-'})\end{bmatrix} = \begin{bmatrix}0.5oh(\text{`1'})\\0.5oh(\text{`-'})\\1.0oh(\text{`-'})\end{bmatrix}$$

To this we add the outer product of the write head position and the value we want to add, namely, the current input $oh(\text{`2'})$:

$$\begin{bmatrix}0.5\\0.5\\0\end{bmatrix} \times oh(\text{`2'}) = \begin{bmatrix}0.5oh(\text{`2'})\\0.5oh(\text{`2'})\\0.0oh(\text{`2'})\end{bmatrix}$$

The resulting tape state at this step is:

$$\begin{bmatrix}0.5oh(\text{`1'})\\0.5oh(\text{`-'})\\1.0oh(\text{`-'})\end{bmatrix} + \begin{bmatrix}0.5oh(\text{`2'})\\0.5oh(\text{`2'})\\0.0oh(\text{`2'})\end{bmatrix} = \begin{bmatrix}0.5oh(\text{`1'}) + 0.5oh(\text{`2'})\\0.5oh(\text{`-'}) + 0.5oh(\text{`2'})\\1.0oh(\text{`-'}) + 0.0oh(\text{`2'})\end{bmatrix}$$

3. Explain how the information gathered at Phase 1 (writing phase) propagates to Phase 2 (reading phase).

   It propagates via the memory tape: the input symbols in Phase 1 are stored to the tape in the correct order by the write head; then, in Phase 2, they are recalled by the read head in that same order.

Problem 3. Give explicit formulas for the expectations in the NTM equations.

Below we interpret vectors as column vectors. Thus for two vectors $\boldsymbol{u} \in \mathbb{R}^k$ and $\boldsymbol{v} \in \mathbb{R}^l$ the product $\boldsymbol{v}\boldsymbol{u}^{\mathrm{T}}$ results in a $k \times l$ real matrix. With $\boldsymbol{u}_i$ we denote $\boldsymbol{u}$'s i-th component.

$$\begin{aligned}\texttt{RD}(\boldsymbol{M}, \boldsymbol{r}) &= \boldsymbol{r}^{\mathrm{T}}\boldsymbol{M}\\ \texttt{WR}(\boldsymbol{M}, \boldsymbol{w}, \boldsymbol{e}, \boldsymbol{a}) &= \boldsymbol{M} - (\boldsymbol{w}\boldsymbol{e}^{\mathrm{T}}) \odot \boldsymbol{M} + \boldsymbol{w}\boldsymbol{a}^{\mathrm{T}}\\ \texttt{MV}(\boldsymbol{l}, \boldsymbol{d})_k &= \sum_{i+j=k} \boldsymbol{l}_i \boldsymbol{d}_j.\end{aligned}$$

References

[1] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural turing machines". In: *arXiv preprint arXiv:1410.5401* (2014).