Reinforcement Learning
Neural Turing Machine

by Wojciech Zaremba,
and Ilya Sutskever
Powerful model

● Need external memory
  ○ Hierarchical
  ○ Associative
  ○ Location related

● Should run for arbitrary time (so it can solve problems of arbitrary complexity > O(n))
  ○ Searching facts in memory can take arbitrary time
  ○ We should give time for contemplation

● Scalable
Contemporary models

● RNN, CNN, any feed-forward network
  ○ have constant running time
  ○ no external memory

● Stack RNN, Neural Turing Machine, Memory networks
  ○ can be seen as very similar, generic thing
  ○ very powerful
Reinforcement Learning NTM

- Needs external memory
  - Hierarchical
  - Associative
  - Location related
- Should run for arbitrary time (so it can solve problems of arbitrary complexity > O(n))
  - Searching facts in memory can take arbitrary time
  - We should give time for contemplation
- Scalable
Learn probability

Most generic form of sequence probability is (no approximation):

\[ p(x_1, x_2, \ldots, x_n | \theta) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}, \theta) \]

It assumes that \( x_i \) can be dependent on any previous time step \( x_k, \ k < i \).
RNNs objective

- RNNs learn to predict sequentially elements on right hand side of equation:

\[ p(x_1, x_2, \ldots, x_n | \theta) = \prod_{i=1}^{n} p(x_i | x_{k<i} \theta) \]

- They don’t do conditional independence assumption.
- Can model arbitrary probability distribution.
Recurrent neural networks

My name is Wojciech
Language modelling

the meaning of life is that only if an end would be of the whole supplier. widespread rules are regarded as the companies of refuses to deliver. in balance of the nation’s information and loan growth associated with the carrier thrifts are in the process of slowing the seed and commercial paper.
More depth gives more power
LSTM* - Long Short Term Memory

- Ad-hoc way of modelling long dependencies
- Many alternative ways of modelling it
- Next hidden state is modification of previous hidden state (so information doesn’t decay too fast).

*Hochreiter and Schmidhuber, 1997
RNN doesn’t have to make always predictions

It can consume input, and then start prediction.

Sutskever et. al. (2014)
Postponing predicting

- We allow model to decide if it wants to predict
- Model produces probability distribution over actions: “predict” vs “don’t predict”
- We sample from this distribution
- Trained with Reinforce
Interfaces

- Input is a tape on which you can move left, or right, or stay
- Output is a tape on which you can move only to the right or stay
  - It defines reward (which is equal to log likelihood of prediction).
  - Reward is differentiable!
- Memory is a tape, but it could be a grid etc.
Tasks

1st step

Prediction tape: $\text{[ } a \text{ } b \text{ } c \text{ ]}$
Action: Predict $a$

Input tape: $\text{[ } a \text{ } a \text{ } b \text{ } b \text{ } b \text{ } c \text{ } c \text{ ]}$
Action: Move to the right
Tasks

2nd step

Prediction tape: \[ a \quad \text{\(\bigcirc\)} \quad b \quad c \quad . \]
Action: Wait

Input tape: \[ a \quad \text{\(\bigcirc\)} \quad b \quad b \quad c \quad c \quad c \quad = \]
Action: Move to the right
Tasks

3st step

Prediction tape: $a \quad \boxed{b} \quad c \quad .$
Action: Predict $b$

Input tape: $a \quad a \quad \boxed{b} \quad b \quad c \quad c \quad =$
Action: Move to the right
Tasks

4th step

Prediction tape: $\begin{array}{c} a \\ b \\ c \end{array}$.

Action: Wait

Input tape: $\begin{array}{cccc} a & a & b & b \\ c & c \end{array}$

Action: Move to the right
Tasks
Tasks

6th step
Prediction tape: 
Action: Wait
Input tape: 
Action: Move to the right
Tasks

7th step

Prediction tape:  
Action: Wait

Input tape:  
Action: Move to the right.
Tasks

8th step

Prediction tape: \[a\ b\ c\ \circ\]
Action: Predict ".", which means end of the sample

Input tape: \[a\ a\ b\ b\ c\ c\ \equiv\]
Action: This action makes no difference
Task - RepeatInput
Training

- Trained with SGD
- Reinforce to deal with discrete decisions
- Curriculum learning is critical
- Not easy to train (due to variance coming from sampling)
  - Various techniques to decrease variance
Memory interface

- Memory is a tape with 3 actions, go to the left, stay, go to the right
- RNN always reads from previous memory location, and always saves to the next memory location
- It stores high dimensional vector through which we backpropagate
Task - Reverse
Task - Reverse, no external memory
Task - RepeatCopy, no external memory
Task. RepeatCopy with memory. Failure
RL NTM is the first model that in principle is Turing Complete

Disclaimer: it doesn’t mean that we can train it to solve hard tasks
Reinforce

Objective of Reinforce:

\[ \sum_{a_1, \ldots, a_n} p(a_1, \ldots, a_n | \theta) \sum_i r_i \]

we access it through sampling:

\[ \mathbb{E}_a \sum_i r_i \]
Reinforce

Derivative:

\[ \sum_a p'(a|\theta) \sum_i r_i + \sum_a p(a|\theta) \sum_i r'_i \]

\[ p' = p(\log p)' \]

we access it through sampling:

\[ \mathbb{E}_a \log p' \sum_i r_i + \sum_i r'_i \]
Gradient Checking for Reinforce

- We could sample actions many times and compare average gradient to average numerical gradient.
Gradient Checking for Reinforce

- We could sample actions many times and compare average gradient to average numerical gradient.
- Impractical. To get good precision we would need millions of samples.
Gradient Checking for Reinforce

- We enumerate all sequences of actions
- We make mini batch of size number of all sequences
- We overload sampler to return predefined actions. Every row of minibatch corresponds to a different action sequence.
- Sampler returns probability of enumerated actions.
- We multiply resulting probabilities by rewards
Gradient Checking for Reinforce

- It was critical to make model work.
- We can limit size of action space during gradient checking
- Gradient checking takes seconds
Variance of gradients

- Sampling of actions introduces variance into gradient estimate
- We subtract baseline reward to decrease variance
Baseline reward

\[ \sum_a p(a | \theta) = 1 \]

\[ \sum_a p'(a | \theta) = 0 \]

\[ \mathbb{E}_a \log p'(\sum_i r_i - \nu) + \sum_i r'_i \]

\[ \| \mathbb{E}_a \sum_i r_i - \nu \|_{L_2} \]
Additional baseline prediction

- We read entire input, and output once
- Next we run 2 LSTM in tandem
- One initialized with hidden state after reading input, output
- Another starting with clean state
- We use one that have seen everything to predict baseline
Modified RL NTM

new mem value

sofmax prediction

0 1

-1 0 1

-1 0 1

LSTM

current input

current memory

actions taken in prev step
Guideline

- Use Reinforcement learning to learn over small number of actions
- Reinforcement learning doesn’t work when action space is is large
- Gradient Check your Reinforce code. Probability of a bug tends to 1.
Q&A

- Powerful models
- RNNs
- Interfaces
- Solved tasks
- Gradient checking
- Variance of gradients
- Modified RL NTM

I am happy to answer any questions.