nature International weekly journal of science

NATURE | ARTICLE

日本語要約

Hybrid computing using a neural network with dynamic external memory

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu & Demis Hassabis

Differential Neural Computer

Shintaro Shiba

Oct 26, 2016

DNC とは

DeepMind による3本目の Nature 論文

- “Hybrid computing using a neural network with dynamic external memory” [Graves et al., Nature, 2016]

Neural Turing Machine (2014) の改良版

- NTM については後述

推論、グラフ構造の情報からの学習の性能が劇的に向上した

Background 1/2

近年の Deep Leaning は、

感覚情報処理 (sensory processing)

系列学習 (sequence learning)

強化学習 (reinforcement learning)

において優秀な成績をおさめてきた

Background 2/2

コンピューターの処理は一般的に、計算 (processor) とデータ (RAM) を分離している

- これによって大規模なデータの保持や、同じ計算を異なるデータに適用する（変数という仕組み）ことが可能になっている

しかし、ニューラルネットにおいてはこれらが混在しており、この部分を改良できそう

- 今のニューラルネットにおいて内部に保持されているデータ（内部状態）を外部に持っていくことはできないか？

何が内部状態なのか？

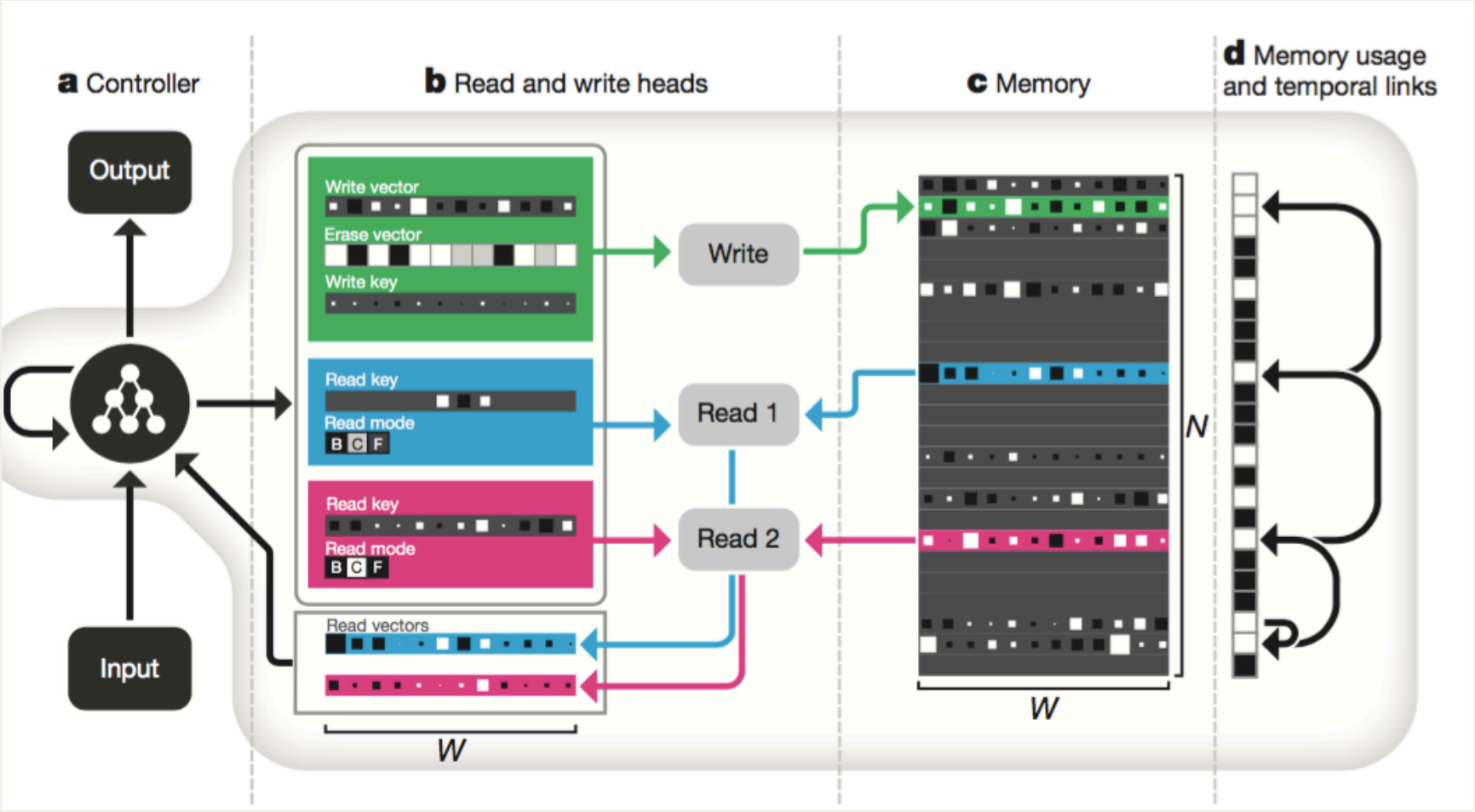
Recurrent なニューラルネットワーク (RNN, LSTM) において、以前の状態と現在の入力が出力を計算する

$$x(t+1) = x(t) + i(t)$$

それ以外のニューラルネットにおいても、学習に使用するデータの読み書きの仕方にあまり工夫がない

- 内部状態：「以前のデータをどう計算に使用するか」とも言える（？）

DNC System Overview



DNC System Overview

External memory matrix + neural network

- 論文ではLSTM
- External memory matrixとそれへのアクセスの仕方がニューラルネットの内部状態に相当し、外部入力とともにニューラルネットへ入力される

従来のニューラルネットとは、メモリが「選択的に読み書き」される点で異なる

Neural Turing Machine との違い

Neural Turing Machine の解説（わかりやすい）：

- http://distill.pub/2016/augmented-rnns/#neural-turing-machines

メモリアクセスの方法の改善

- NTM では content-based attention と location-based attention
  - 問題1. メモリブロックが重複したり干渉しあったりする可能性がある
  - 問題2. メモリ領域の再利用ができず、長い系列の問題が苦手
  - 問題3. 書き込む順番でなく場所なので、書き込み場所に跳躍があった場合に対応できない
- DNCでは content lookup, temporal link, allocation

メモリの構造

- メモリ行列 M = N x W
  - N行のWベクトル
  - N行のAttentionに相当する重みの分布
  - それぞれの場所がどのくらい読み書きされるかを示す

- read vector
  - wによって重み付けされたメモリMの和

$$r = \sum_{i=1}^{N} M[i,:] w^r[i]$$

- erase vector, write vector
  - wによって重み付けされた計算

$$M[i,j] \leftarrow M[i,j](1 - w^w[i]e[j]) + w^w[i]v[j]$$

- これらの計算ユニットのことをheadと呼ぶ
  - 細かい数式は最後で追います

メモリの読み書きの仕方

1. Content lookup
   - associative recall / 似ている情報を読み込む
   - “key vector”を読み込み、類似度計算

2. Temporal links
   - sequential recall / 書き込まれた時系列が近いものを読み込む
   - “書き込まれた順”を記録した行列L (N x N)

3. allocation
   - write head with unused locations / 使われていないものから消去して新しいデータを書き込む
   - define “usage” of each location, and unused location is delivered to the write head

Synthetic QA experiment

データセット：facebook bAbI

ex. “John is in the playground. John picked up the football.” Q: Where is the football?

Extended Data Table 1 | bAbI best and mean results

	bAbI Best Results							bAbI Mean Results			
Task	LSTM (Joint)	NTM (Joint)	DNC1 (Joint)	DNC2 (Joint)	MemN2N (Joint) [21]	MemN2N (Single) [21]	DMN (Single) [20]	LSTM	NTM	DNC1	DNC2
1: 1 supporting fact	24.5	31.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	28.4 ± 1.5	40.6 ± 6.7	**9.0 ± 12.6**	16.2 ± 13.7
2: 2 supporting facts	53.2	54.5	1.3	0.4	1.0	**0.3**	1.8	56.0 ± 1.5	56.3 ± 1.5	**39.2 ± 20.5**	47.5 ± 17.3
3: 3 supporting facts	48.3	43.9	2.4	**1.8**	6.8	2.1	4.8	51.3 ± 1.4	47.8 ± 1.7	**39.6 ± 16.4**	44.3 ± 14.5
4: 2 argument rels.	0.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.8 ± 0.5	0.9 ± 0.7	**0.4 ± 0.7**	**0.4 ± 0.3**
5: 3 argument rels.	3.5	0.8	**0.5**	0.8	6.1	0.8	0.7	3.2 ± 0.5	1.9 ± 0.8	**1.5 ± 1.0**	1.9 ± 0.6
6: yes/no questions	11.5	17.1	**0.0**	**0.0**	0.1	0.1	**0.0**	15.2 ± 1.5	18.4 ± 1.6	**6.9 ± 7.5**	11.1 ± 7.1
7: counting	15.0	17.8	**0.2**	0.6	6.6	2.0	3.1	16.4 ± 1.4	19.9 ± 2.5	**9.8 ± 7.0**	15.4 ± 7.1
8: lists/sets	16.5	13.8	**0.1**	0.3	2.7	0.9	3.5	17.7 ± 1.2	18.5 ± 4.9	**5.5 ± 5.9**	10.0 ± 6.6
9: simple negation	10.5	16.4	**0.0**	0.2	**0.0**	0.3	**0.0**	15.4 ± 1.5	17.9 ± 2.0	**7.7 ± 8.3**	11.7 ± 7.4
10: indefinite knowl.	22.9	16.6	0.2	0.2	0.5	**0.0**	**0.0**	28.7 ± 1.7	25.7 ± 7.3	**9.6 ± 11.4**	14.7 ± 10.8
11: basic coreference	6.1	15.2	**0.0**	**0.0**	**0.0**	0.1	0.1	12.2 ± 3.5	24.4 ± 7.0	**3.3 ± 5.7**	7.2 ± 8.1
12: conjunction	3.8	8.9	0.1	**0.0**	0.1	**0.0**	**0.0**	5.4 ± 0.6	21.9 ± 6.6	**5.0 ± 6.3**	10.1 ± 8.1
13: compound coref.	0.5	7.4	**0.0**	0.1	**0.0**	**0.0**	0.2	7.2 ± 2.3	8.2 ± 0.8	**3.1 ± 3.6**	5.5 ± 3.4
14: time reasoning	55.3	24.2	0.3	0.4	**0.0**	0.1	**0.0**	55.9 ± 1.2	44.9 ± 13.0	**11.0 ± 7.5**	15.0 ± 7.4
15: basic deduction	44.7	47.0	**0.0**	**0.0**	0.2	**0.0**	**0.0**	47.0 ± 1.7	46.5 ± 1.6	**27.2 ± 20.1**	40.2 ± 11.1
16: basic induction	52.6	53.6	52.4	55.1	**0.2**	51.8	0.6	**53.3 ± 1.3**	53.8 ± 1.4	53.6 ± 1.9	54.7 ± 1.3
17: positional reas.	39.2	25.5	24.1	**12.0**	41.8	18.6	40.4	34.8 ± 4.1	**29.9 ± 5.2**	32.4 ± 8.0	30.9 ± 10.1
18: size reasoning	4.8	2.2	4.0	**0.8**	8.0	5.3	4.7	5.0 ± 1.4	4.5 ± 1.3	**4.2 ± 1.8**	4.3 ± 2.1
19: path finding	89.5	4.3	**0.1**	3.9	75.7	2.3	65.5	90.9 ± 1.1	86.5 ± 19.4	**64.6 ± 37.4**	75.8 ± 30.4
20: agent motiv.	1.3	1.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.3 ± 0.4	1.4 ± 0.6	**0.0 ± 0.1**	**0.0 ± 0.0**
Mean Err. (%)	25.2	20.1	4.3	**3.8**	7.5	4.2	6.4	27.3 ± 0.8	28.5 ± 2.9	**16.7 ± 7.6**	20.8 ± 7.1
Failed (err. > 5%)	15	16	**2**	**2**	6	3	**2**	17.1 ± 1.0	17.3 ± 0.7	**11.2 ± 5.4**	14.0 ± 5.0

To compare with previous results we report error rates for the single best network across all tasks (measured on the validation set) over 20 runs. The lowest error rate for each task is shown in bold. Results for MemN2N are from ref. 21; those for DMN are from ref. 20. The mean results are reported with ±s.d. for the error rates over all 20 runs for each task. The lowest mean error rate for each task is shown in bold.

Synthetic QA タスク

一応、SoTAではないっぽい？

- こっちの方が性能がいいよ！みたいな声も
  - Gated graph sequence neural networks [Li, Zemel, Brockschmidt & Tarlow, 2016]
  - Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes [Gulcehre, Chandar, Cho, Bengio, 2016]
  - https://www.reddit.com/r/MachineLearning/comments/575hlx/research_new_nature_paper_by_deepmind_hybrid/

“Unlike previous results on this dataset, the inputs to our model were single word tokens without any preprocessing or sentence-level features.”

Graph experiment

自然言語は非明示的なグラフ構造のデータとも言える [*要出典*]

Extended Data Table 3 | Curriculum results for graph traversal

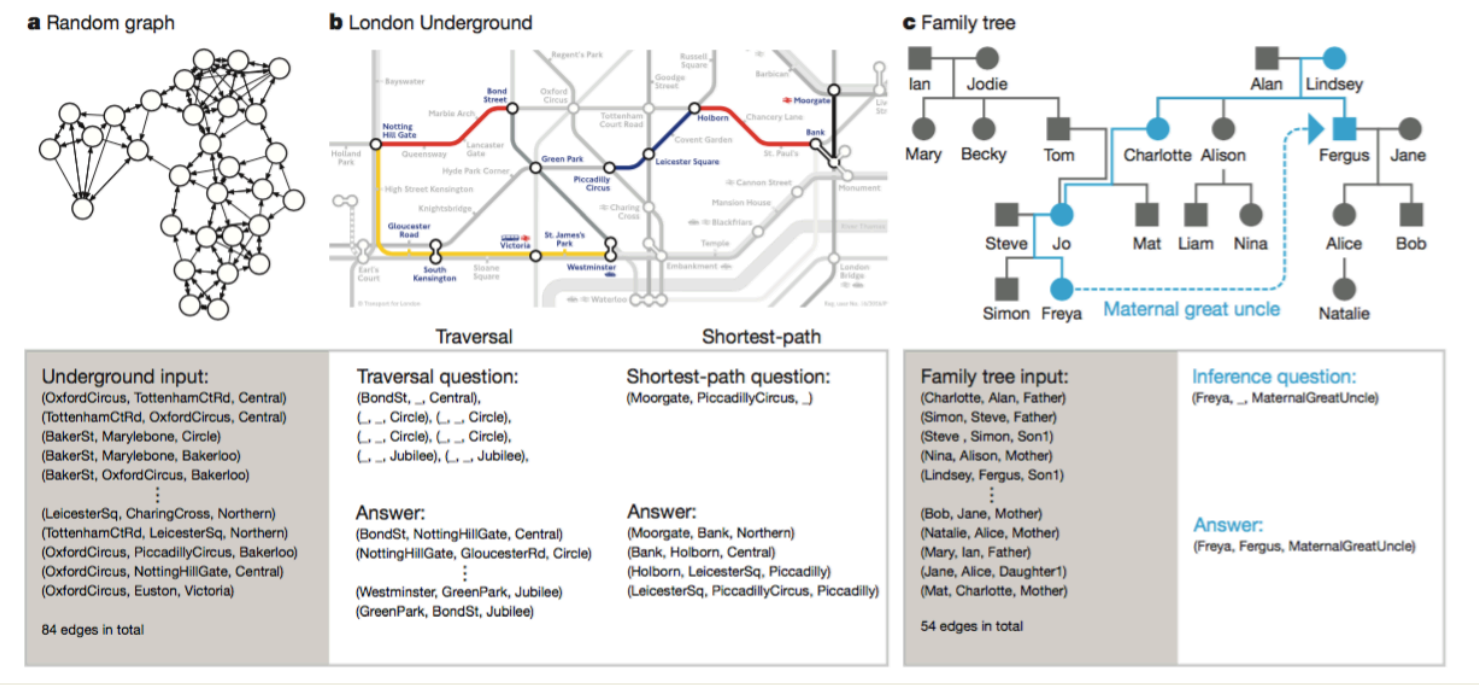
Lesson	Nodes	Out-degree	Path Length	Test	Final
1	(3, 10)	(2, 4)	(1, 1)	0.0 ± 0.0	10.7 ± 0.6
2	(3, 10)	(2, 4)	(1, 2)	0.0 ± 0.0	19.7 ± 1.3
3	(5, 10)	(2, 4)	(1, 3)	0.0 ± 0.0	27.3 ± 1.0
4	(5, 10)	(2, 4)	(1, 4)	0.0 ± 0.0	38.2 ± 2.2
5	(10, 15)	(2, 4)	(1, 4)	0.0 ± 0.0	39.7 ± 1.5
6	(10, 15)	(2, 4)	(1, 5)	0.0 ± 0.0	47.5 ± 2.2
7	(10, 20)	(2, 4)	(1, 5)	0.1 ± 0.2	48.1 ± 1.8
8	(10, 20)	(2, 4)	(1, 6)	13.6 ± 20.8	59.4 ± 5.2
9	(10, 30)	(2, 4)	(1, 6)	15.0 ± 20.3	59.0 ± 4.8
10	(10, 30)	(2, 4)	(1, 7)	72.8 ± 9.3	72.3 ± 3.5
11	(10, 30)	(2, 4)	(1, 8)	88.6 ± 5.7	81.6 ± 4.0
12	(10, 30)	(2, 4)	(1, 9)	91.8 ± 3.9	90.7 ± 3.2
13	(10, 40)	(2, 6)	(1, 10)	96.0 ± 3.7	96.8 ± 1.6
14	(10, 40)	(2, 6)	(1, 20)	98.8 ± 1.6	99.0 ± 1.1

グラフを明示的に入力する

- curriculum learning [Bengio et al. ACM, 2009; Zaremba, 2014]
  - ランダムに生成したグラフデータでトレーニング
  - クエリはtraversal, shortest path, inference の3種類

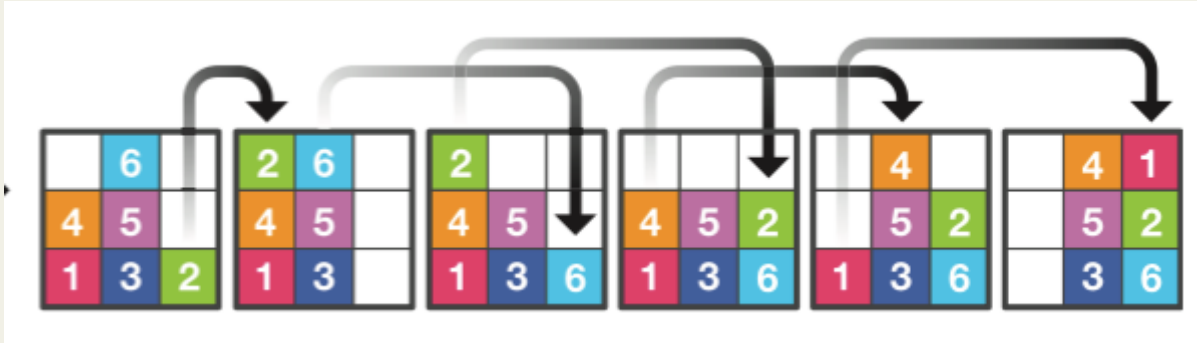
テストでは、ロンドンの地下鉄の路線図と家系図を使用

- “with no retraining” on this dataset
- 結果 https://www.youtube.com/watch?v=B9U8sI7TcMY
- Accuracy…LSTM 37%, 2000k training examples with searched hyper parameters; DNC 98.8%, 1000k examples.



Block puzzle experiment

- Logical planning task としてテスト
- ブロックを動かしてゴールの形を作る



- ゴールの表現の仕方

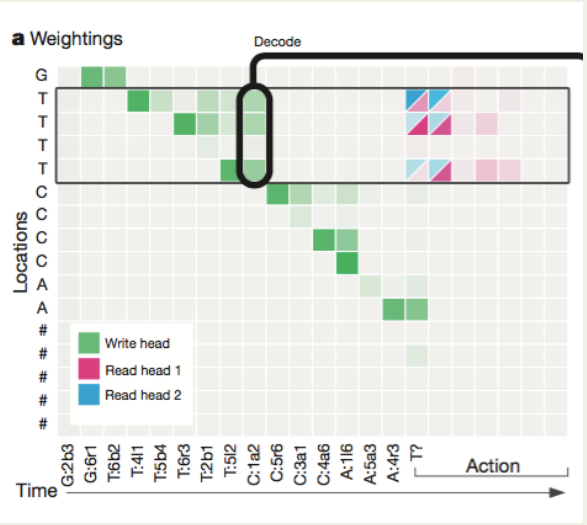
6 below 2　4 left of 1　　6 right of 3　etc…

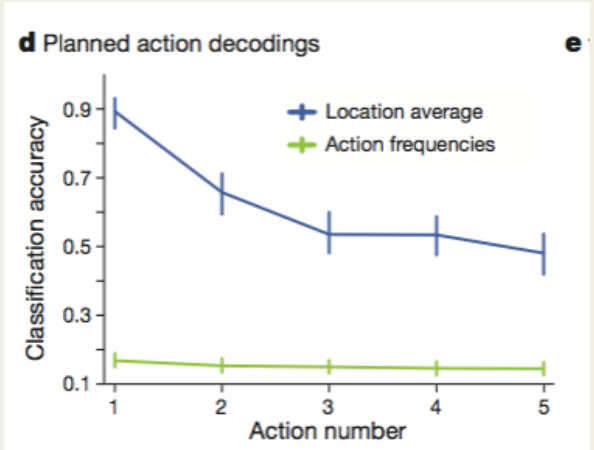
これまでは教師あり学習だったが、強化学習としてやってみた

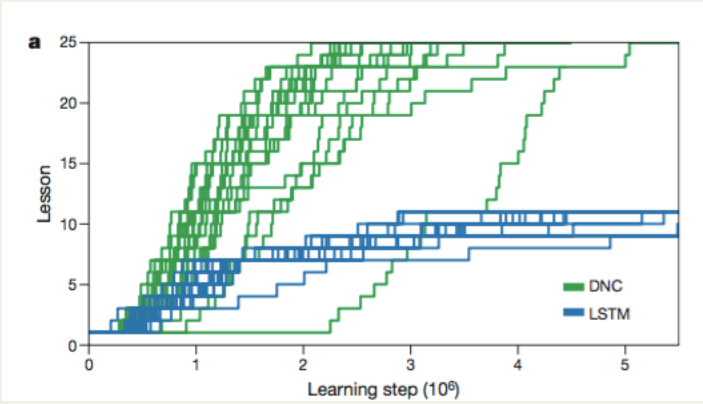
- これもcurriculum learning

大量のデータを提示された後でも、１回目の行動からきちんとゴールへの行動ができた

- 計画（？）
- もちろん比較手法 (LSTM) より良い







考察とまとめと感想

- DNC について、グラフ構造を持つデータに対する学習が可能であること、systematic にメモリを使用できることを示した
- すべてのタスクが512 location以下のメモリでなされた
- 推論が必要とされるような他のタスクにも応用可能だろう
  - （彼らがすでにしているだろう）

考察とまとめと感想

- 直感的にはわかりやすい（だからNature）
- それをやって上手くいってしまうのが恐ろしい（だからNature）
- chainer 実装（正しいか調べていない）、tensorflow 実装計画（NTMからの派生プロジェクト）などがある
- 何か間違いがあれば教えてください

以下、数式部分

アルゴリズムを追います

変数の定義

Name	Description	Domain
t	time-step	$\mathbb{N}$
N	number of memory locations	$\mathbb{N}$
W	memory word size	$\mathbb{N}$
R	number of read heads	$\mathbb{N}$
$\mathbf{x}_t$	input vector	$\mathbb{R}^X$
$\mathbf{y}_t$	output vector	$\mathbb{R}^Y$
$\mathbf{z}_t$	target vector	$\mathbb{R}^Y$
$\mathbf{M}_t$	memory matrix	$\mathbb{R}^{N\times W}$
$\mathbf{k}_t^{r,i}$	read key i $(1 \le i \le R)$	$\mathbb{R}^W$
$\mathbf{r}_t^i$	read vector i	$\mathbb{R}^W$
$\beta_t^{r,i}$	read strength i	$[1,\infty)$
$\mathbf{k}_t^w$	write key	$\mathbb{R}^W$
β_t^w	write strength	$[1,\infty)$
$\mathbf{e}_t$	erase vector	$[0,1]^W$
$\mathbf{v}_t$	write vector	$\mathbb{R}^W$
f_t^i	free gate i	$[0,1]$
g_t^a	allocation gate	$[0,1]$
g_t^w	write gate	$[0,1]$
$\boldsymbol{\psi}_t$	memory retention vector	$\mathbb{R}^N$
$\mathbf{u}_t$	memory usage vector	$\mathbb{R}^N$
$\boldsymbol{\phi}_t$	indices of slots sorted by usage	$\mathbb{N}^N$
$\mathbf{a}_t$	allocation weighting	$\Delta_N = \{\boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0,1], \sum_{i=1}^N \boldsymbol{\alpha}_i \le 1\}$
$\mathbf{c}_t^w$	write content weighting	$\mathcal{S}_N = \{\boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0,1], \sum_{i=1}^N \boldsymbol{\alpha}_i = 1\}$
$\mathbf{w}_t^w$	write weighting	Δ_N
$\mathbf{p}_t$	precedence weighting	Δ_N
$\mathbf{E}$	matrix of ones $(\mathbf{E}[i,j] = 1\ \forall i,j)$	$\mathbb{R}^{N\times W}$
$\mathbf{L}_t$	temporal link matrix	$\mathbb{R}^{N\times N}$
$\mathbf{f}_t^i$	forward weighting i	Δ_N
$\mathbf{b}_t^i$	backward weighting i	Δ_N
$\mathbf{c}_t^{r,i}$	read content weighting i	$\mathcal{S}_N$
$\mathbf{w}_t^{r,i}$	read weighting i	Δ_N
$\boldsymbol{\pi}_t^i$	read mode i	$\mathcal{S}_3$
W_r	read key weights	$\mathbb{R}^{(RW)\times Y}$
$\boldsymbol{\theta}$	controller weights	$\mathbb{R}^\Theta$
$\boldsymbol{\xi}_t$	interface vector	$\mathbb{R}^{(W\times R)+3W+5R+3}$
$\boldsymbol{\chi}_t$	controller input vector	$\mathbb{R}^{(W\times R)+X}$
$\boldsymbol{\upsilon}_t$	controller output vector	$\mathbb{R}^Y$
$\mathcal{N}(.;\boldsymbol{\theta})$	controller network	$\left[\mathbb{R}^{(W\times R)+X}\right]^* \times \mathbb{R}^\Theta \mapsto \mathbb{R}^{(W\times R)+3W+5R+3} \times \mathbb{R}^Y$

Definitions:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

[0, 1]に射影

$$\text{oneplus}(x) = 1 + \log(1+e^{x})$$

[1, ∞) に射影

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{|\mathbf{x}|} e^{x_j}}$$

正規化

$$\mathcal{C}(\mathbf{M}, \mathbf{k}, \beta)[i] = \frac{\exp\left\{\mathcal{D}(\mathbf{k}, \mathbf{M}[i,.])\beta\right\}}{\sum_j \exp\left\{\mathcal{D}(\mathbf{k}, \mathbf{M}[j,.])\beta\right\}}$$

i番目のデータ
とkの類似度

メモリの場所に関して
正規化された確率分布

$$\mathcal{D}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}||\mathbf{v}|}$$

コサイン類似度

$$(\mathbf{A}\circ\mathbf{B})[i,j] = \mathbf{A}[i,j]\mathbf{B}[i,j];\ (\mathbf{x}\circ\mathbf{y})[i] = \mathbf{x}[i]\mathbf{y}[i]$$

Initial Conditions:

$$\mathbf{u}_0 = \mathbf{0};\ \mathbf{p}_0 = \mathbf{0};\ \mathbf{L}_0 = \mathbf{0};\ \mathbf{L}_t[i,i] = 0\ \forall i$$

Controller Update:

$$\boldsymbol{\chi}_t = [\mathbf{x}_t; \mathbf{r}_{t-1}^1; \ldots; \mathbf{r}_{t-1}^R]$$

ニューラルネットへの入力
[外部入力; 内部入力]

$$(\boldsymbol{\xi}_t, \boldsymbol{\upsilon}_t) = \mathcal{N}([\boldsymbol{\chi}_1; \ldots; \boldsymbol{\chi}_t]; \boldsymbol{\theta})$$

Interface Variables:

書き

読み

$$\boldsymbol{\xi}_t = [\mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots, \hat{\beta}_t^{r,R}; \mathbf{k}_t^{w}; \hat{\beta}_t^{w}; \hat{\mathbf{e}}_t; \mathbf{v}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \hat{g}_t^a; \hat{g}_t^w; \hat{\boldsymbol{\pi}}_t^1; \ldots; \hat{\boldsymbol{\pi}}_t^R]$$

$$\beta_t^{r,i} = \text{oneplus}(\hat{\beta}_t^{r,i});\ \beta_t^w = \text{oneplus}(\hat{\beta}_t^w);\ \mathbf{e}_t = \sigma(\hat{\mathbf{e}}_t);\ f_t^i = \sigma(\hat{f}_t^i)$$

$$g_t^a = \sigma(\hat{g}_t^a);\ g_t^w = \sigma(\hat{g}_t^w);\ \boldsymbol{\pi}_t^k = \text{softmax}(\hat{\boldsymbol{\pi}}_t^k)$$

それぞれの場所の
freeにされにくさvector

使われているかのvector
1…書き込まれたばかり / 0…使われていない
場所が使われた=freeにされにくく、
すでに使用されているか直前に書き込まれた場所

Memory Updates:

$$\psi_t = \prod_{i=1}^{R} (1 - f_t^i \mathbf{w}_{t-1}^{r,i})$$

$$\mathbf{u}_t = \left(\mathbf{u}_{t-1} + \mathbf{w}_{t-1}^{w} - (\mathbf{u}_{t-1} \circ \mathbf{w}_{t-1}^{w})\right) \circ \psi_t$$

$$\phi_t = \mathrm{SortIndicesAscending}(\mathbf{u}_t)$$

$$\mathbf{a}_t[\phi_t[j]] = (1 - \mathbf{u}_t[\phi_t[j]]) \prod_{i=1}^{j-1} \mathbf{u}_t[\phi_t[i]]$$

$$\mathbf{c}_t^w = \mathcal{C}(\mathbf{M}_{t-1}, \mathbf{k}_t^w, \beta_t^w)$$

$$\mathbf{w}_t^w = g_t^w \left[g_t^a \mathbf{a}_t + (1 - g_t^a)\mathbf{c}_t^w\right]$$

$$\mathbf{M}_t = \mathbf{M}_{t-1} \circ (\mathbf{E} - \mathbf{w}_t^w \mathbf{e}_t^\top) + \mathbf{w}_t^w \mathbf{v}_t^\top$$

最も使われていない
場所を一番上に

新しく書き込む場所の
vector
usageが1なら書き込まない

write gate[0, 1]で
内挿しながら
書き込みの重みを計算

書き込み用のベクトルを準備
key vectorに似ているものを書
く

メモリを更新（消去・書き込み）

場所jのどのくらい後に
iに書き込みがあったか
各要素は[0, 1]でサイズはNxN
1なら直後

各要素=場所
それぞれの場所がどのくらい最後に書き込まれたか

$$\mathbf{p}_t = \left(1 - \sum_i \mathbf{w}_t^w[i]\right)\mathbf{p}_{t-1} + \mathbf{w}_t^w$$

$$\mathbf{L}_t[i,j] = \left(1 - \mathbf{w}_t^w[i] - \mathbf{w}_t^w[j]\right)\mathbf{L}_{t-1}[i,j] + \mathbf{w}_t^w[i]\mathbf{p}_{t-1}[j]$$

書き込まれた順

$$\mathbf{f}_t^i = \mathbf{L}_t \mathbf{w}_{t-1}^{r,i}$$

書き込まれた逆順

$$\mathbf{b}_t^i = \mathbf{L}_t^\top \mathbf{w}_{t-1}^{r,i}$$

kに似ているメモリの
場所の分布

$$\mathbf{c}_t^{r,i} = \mathcal{C}(\mathbf{M}_t, \mathbf{k}_t^{r,i}, \beta_t^{r,i})$$

$$\mathbf{w}_t^{r,i} = \boldsymbol{\pi}_t^i[1]\mathbf{b}_t^i + \boldsymbol{\pi}_t^i[2]\mathbf{c}_t^{r,i} + \boldsymbol{\pi}_t^i[3]\mathbf{f}_t^i$$

$$\mathbf{r}_t^i = \mathbf{M}_t^\top \mathbf{w}_t^{r,i}$$

読み込む重み[1]+[2]+[3] = 1
[1]…[3]の逆順を読む
[2]...kに似たものを読む
[3]…書き込まれた順に近いもの
を読む

Output:

$$\mathbf{y}_t = W_r[\mathbf{r}_t^1; \ldots; \mathbf{r}_t^R] + \boldsymbol{v}_t$$

ニューラルネットへの入力となる