—論文題目—

関係推論能力を有する Neural Turing Machine

.

指導教授

萩原 将文 教授

慶應義塾大学 理工学部 情報工学科

令和 4 年度

学籍番号 61913674

東明 鴻希

目 次

あらまし

Neural Turing Machine(NTM) のような外部メモリを持つニューラルネットワークモデルは情報を長期的に保存し活用する能力を持つ一方で、メモリ内容間の複雑な推論を行う能力に課題がある。関係ネットワークはエンティティ間の関係を計算でき、高度な推論能力を必要とするタスクを解決可能である。本論文ではNTMのメモリ構造を保ったまま、メモリ内容に関係推論を適用することを提案する。これにより既存の関係推論ネットワークよりも高度な忘却方式、説明可能性および拡張性を持つモデルを実現する。 想定される課題に対処するため、二段階の順序整理モジュールも新しく提案する。一段階目のグラフアテンション構造は時間的に隣接する項目間の推論を行う。二段階目のモジュールはメモリをソートしたのちに全項目間の関係を計算し、関係情報専用のメモリに保存する。実験結果は

第1章

はじめに

ニューラルネットワークモデルの一つである Recurrent Neural Networks(RNN) は、時系列データの学習に広く用いられる。入力の順伝播を繰り返す最もシンプルな RNN では、長時間の入力にわたって逆伝播を計算する中で勾配の値が極端に小さくあるいは大きくなるという問題が発生する。Long-Short Term Memory(LSTM)[1] はゲーティング機構によりこれらの問題に対処した。また、メモリセルの存在により長時間にわたり情報を保持する能力が向上した。しかしメモリセルのサイズや複雑さの制限により、多くの情報を長期保持する能力にはまだ限界がある [要出典]。

メモリネットワークはメモリとして利用可能な行列を有し、各時間ステップでの入力情報をメモリに読み書きする能力を学習する。典型的なメモリネットワークの構造は図 1.1 End-To-End Memory Networks[2] は入力の埋め込みを外部メモリとして保持することで、入力中の長い時間を隔てた関係を解釈できる。しかし入力系列の各項目を全て保存する必要があるため、時間方向へのスケーリング能力に課題がある。

Neural Turing Machine (NTM)[3] が有するメモリは固定長であるため、入力の長期化に伴う計算量の増加を回避できる。メモリスロットの数を超えるサイズの入力系列に対応できるのは、メモリの特定の行を忘却/上書きする能力による。Differentiable Neural Computer (DNC)[4] は NTM の読み出し/書き込み重みの計算部分をより複雑にしたモデルである。追加モジュールの一つである usage vector は、各スロットの使用率を記憶する隠れ状態である。書き込み重みの算出に利用することで、空いているスロットに優先的に書き込める。別の追

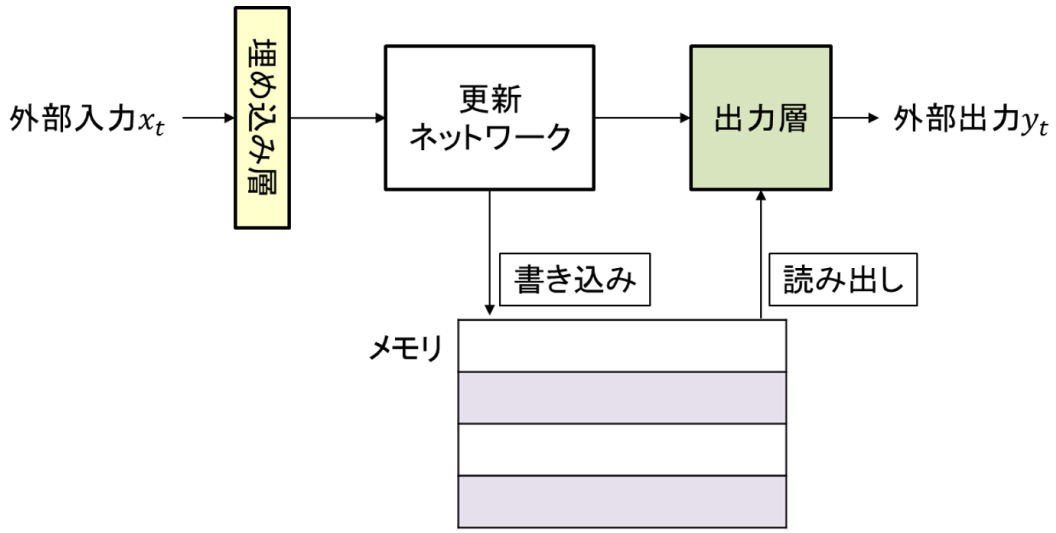

図 1.1: 典型的なメモリネットワーク

加モジュールである temporal link matrix は、スロット間の前後関係を記憶するメモリである。読み出し重みの計算に利用することで、着目するスロットの前後の時間で書き込まれたスロットも考慮できる。Convertible Short-Term and Long-Term Memory in DNC (CSLM)[5] は、DNC の長期記憶能力をさらに改善する。メモリの各スロットには重要度が割り振られており、これは学習を通して変化可能である。重要度の割り当てが低いスロットは忘却されやすく、高いスロットはその逆になる。これは DNC のメモリがスロット単位で長期記憶と短期記憶に分けられており、その割り当ては適応的かつ学習可能であることを意味する。

これら NTM ベースのモデルは学習を通して書き込みの位置や読み出す内容を最適化できる。しかしメモリの項目間の関係を計算する機構を持たないため、最短経路探索のような入力実体間の関係推論を要求するタスクでの性能に劣る。

関係ネットワークは入力や記憶内の実体間の関係推論能力を持つ。関係の計算は self-attention[6] をベースにした手法で行われる。Relational recurrent neural networks (R-RNN)[7] で提案された Relational Memory Core(RMC) は、毎時間ステップにおいて入力と全メモリスロット間での関係情報を計算し、その情報でメモリを更新する。Self-Attentive Associative Memory (SAM)[8] で提案された

SAM-based Two-memory Model(STM) は、入力を保存する項目メモリと、保存した項目間の関係情報を保存する関係メモリの 2 種類のメモリを持つ。項目を忠実に復元することを要求するタスクにおいて、関係メモリのみを持つ RMC よりも高い精度を示している。NTM のように項目がメモリスロット間で区分けされるモデルと異なり、SAM の 2 つのメモリは分散型のメモリである。項目メモリは入力の outer product を保存する自己連想記憶、関係メモリは項目メモリから抽出した項目間の outer product を保存する相互連想記憶として実装されており、各入力がメモリ全体に分散するためである。

本論文では NTM のメモリ構造を保ったまま関係推論モジュールを追加し、メモリ項目間の関係推論能力を追加することを提案する。関係メモリには RMC を用いるため、提案ネットワークは NTM を項目メモリ、RMC を関係メモリとして持つ 2 メモリモデルとなる。同じ 2 メモリモデルである SAM と比較すると、項目メモリが非分散型メモリであり各行で記憶内容が区分けされているある点が異なる。これにより各入力項目を選択的に忘却・上書きすることが可能になり、R-RNN や SAM のような分散記憶+LSTM 方式の忘却よりも高度な忘却が行えると考えられる。従って長期記憶保持が必要なタスクでより優れた性能を示すことが期待される。CSLM が各スロットに忘却強度を割り振ったように、読み書き・忘却方式に拡張性・柔軟性があることも利点である。またこれらのメモリの読み書きはアテンションに由来する説明可能性がある。アテンション係数を可視化することで読み書きした番地を追跡可能である。

この論文の貢献は以下に示す通りである

１．NTM に関係推論能力を付与する。このとき関係メモリの入力から一貫性が失われることが予想される。この課題を GAT と書き込み頻度ソートの 2 種類のモジュールにより解決する。

２．非分散型のメモリの実装により、分散型メモリよりも長期記憶が必要なタスクに適している。

３． 非分散型のメモリには CSLM に代表されるように拡張性・柔軟性がある。またアテンションを用いる読み書きにより、メモリ操作は説明可能性を持つ。

第2章

関連研究

2.1 Neural Turing Machine

NTM の構造を図 2.1 に示す。NTM の構造は 3 つのコンポーネントに分けられる。

１．情報を保存するメモリ M_t は $N \times W$ 次元の行列からなり、これは W 次元の項目を N スロット分保存できる。

２．コントローラは毎時間ステップで入力 x_t を受け取り、隠れ状態 h_t を計算する。任意の RNN がコントローラとして採用可能であるが、大抵 LSTM が用いられる。

３．読み出し/書き込みヘッドはコントローラ出力 h_t とメモリ M_t の内容から読み出し/書き込み重みを計算し、メモリへの読み書きを行う。読み出し/書き込み重みはそれぞれ各スロットへの書き込み/読み出しの強度を表す N 次元のアテンション係数である。この重みを計算する操作を ”アドレス指定 ”操作と呼ぶ。

2.1.1 節ではヘッドにおける読み書きを説明する。2.1.2 節ではアドレス指定を説明する。

2.1.1 読み出し/書き込みヘッド

NTM は読み出し用のヘッドと書き込み用のヘッドそれぞれを 1 つ以上持ち、ヘッドごとにアドレス指定及び読み書きが行われる。読み/書き重みをそれぞれ

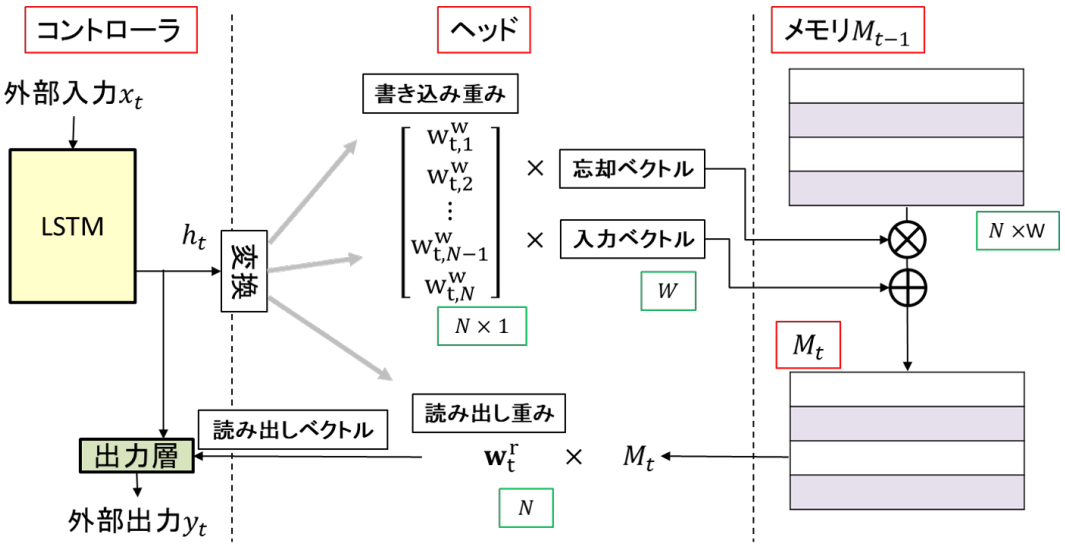

図 2.1: NTM の構造

w_t^r,w_t^w と表し、これらは 2.1.2 節で説明するアドレス指定操作を用いて計算される。w_t^r,w_t^w は式 2.1,2.2 を満たすアテンション係数である。

$$\sum_i w_t(i) = 1 \tag{2.1}$$

$$0 \leq w_t(i) \leq 1, \forall i \tag{2.2}$$

ここで $w_t(i)$ は w_t の i 番目の要素である。メモリからの読み出しは式 2.3 のように計算され、読み出しベクトル r_t を得る。

$$r_t = \sum_i w_t^r(i) M_t(i) \tag{2.3}$$

$M_t(i)$ は M_t の i 行目を表す。書き込みは式 2.4 による忘却、式 2.5 による加算の順で計算され、M_{t-1} を M_t に更新する。忘却ベクトル e_t、書き込みベクトル a_t はコントローラ出力からの変換で得られる。ただし忘却ベクトルの各要素は (0,1) の範囲にある。

$$M_t'(i) = M_{t-1}(i)[1 - w_t^w(i) e_t] \tag{2.4}$$

$$M_t(i) = M_t'(i) + w_t^w(i) a_t \tag{2.5}$$

2.1.2 アドレス指定操作

読み書きの重み w_t は2つの異なるコンセプトに基づいて算出された重みから計算される。類似するメモリ内容を参照するコンテンツベースの重み w_t^c と, メモリの各行を位置によって役割を分ける位置ベースの重み s_t である。アドレス指定操作の中で用いられる k_t,β_t,s_t,g_t,γ_t はコントローラ出力 h_t からの変換で計算される。

はじめにコンテンツベースの重み w_t^c は式2.6に示すように, キーベクトル k_t とメモリ各行の類似度に基づいて計算される。

$$w_t^c(i) = \frac{\exp(\beta_t K\left[k_t, M_t(i)\right])}{\sum_j \exp(\beta_t K\left[k_t, M_t(j)\right])} \tag{2.6}$$

キー強度 β_t は正の値を取り,$K[.,.]$ はコサイン類似度の計算を表す。w_t^c と前の時間ステップでの重み w_{t-1} を式2.7によって補間し、w_t^g を得る。

$$w_t^g(i) = g_t w_t^c + (1 - g_t) w_{t-1}^w \tag{2.7}$$

g_t は (0,1) の範囲にあるスカラー値である。

次にこの重みと位置ベースの重み s_t の畳み込み計算を式2.8行う。この畳み込みは式2.7で注目した場所からの位置のシフトを表現している。例えば前の時間ステップで書き込んだ位置の下に書き込み位置をずらしたり、類似した項目の隣の情報を読み出す機能を実現する。

$$\tilde{w}_t(i) = \sum_{j=0}^{N-1} w_t^g(j) s_t(i-j) \tag{2.8}$$

ここまでの操作で計算されたアテンション係数では重みが複数の位置に分散し, 内容が拡散してしまう可能性がある。最後に式2.9を適用して重みをシャープにし, ヘッドが作用するメモリ行を限定する。

$$w_t(i) = \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}} \tag{2.9}$$

2.2 Relational Memory Core

関係メモリのベースとして使用した、[7] で提案されたRMCを説明する。RMCの構造を図2.2に示す。

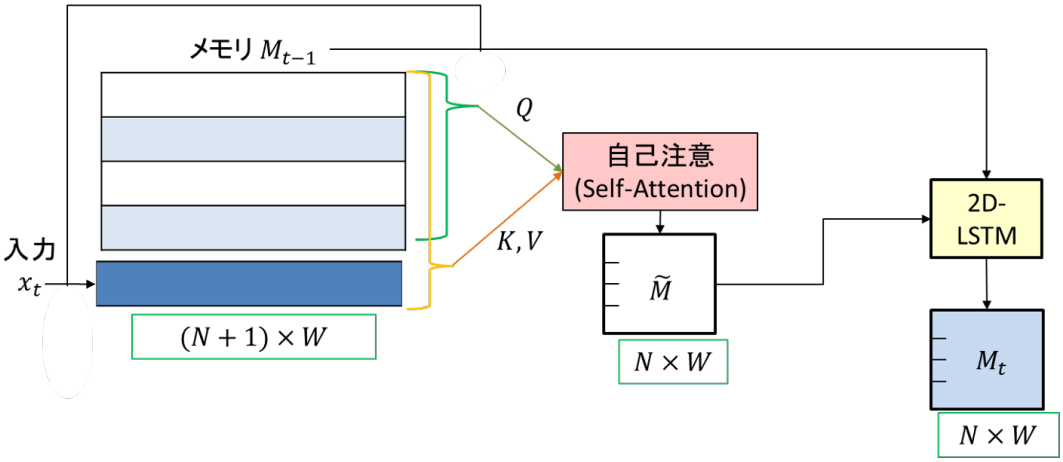

図 2.2: RMC の構造

RMC は毎ステップで入力 x_t とメモリ M_{t-1} の各項目間の関係情報を計算し、そのステップでのメモリ成分 $\tilde{M}$ を用意する。LSTM ベースのゲーティングを利用して、$\tilde{M}$ を M_{t-1} に合成し M_t を得る。$\tilde{M}$ の計算はマルチヘッドドット積アテンション [6] を用いて行われる。RMC は項目メモリ M_i と入力 x_t からアテンションを計算するために、クエリ・キー・バリューを計算する為の訓練可能な線形層を有する。それぞれを W_q,W_k,W_v と表現すると、$\tilde{M}$ は式 2.10 のようにして計算される。

$$\tilde{M} = softmax(\frac{M_{t-1}W_q([M_{t-1}; x_t]W_k)^T}{\sqrt{d_k}})[M_{t-1}; x_t]W_v \qquad (2.10)$$

ここで d_k はキーベクトルの次元　、[M;x] は M に x_t を新たな行として連結した $(N+1) \times M$ 次元の行列を表す。クエリ行列の計算では [M:x] ではなく M を入力することに注意が必要である。これは $\tilde{M}$ の次元を M_t と等しくすることを目的としている。

ヘッドが複数存在する時は、ヘッドごとに独立な線形層を用いたアテンション計算結果を結合し最終的な $\tilde{M}$ を得る。各ヘッドの計算結果を $\tilde{M}_1, \tilde{M}_2 \cdots \tilde{M}_h$ と表すとき、それぞれの次元は $N \times (M/h)$ であり、$\tilde{M}_t = [\tilde{M}_1 : \cdots : \tilde{M}_h]$ とすることで M_t と同じ次元の $\tilde{M}$ を得る。[:] は行方向の連結を表す。

$\tilde{M}$ により M_t を更新するために LSTM を利用する。M_t の各行を 2D-LSTM の各メモリセルとして実装することで、式 2.11-2.17 によって更新される。m_i,$\tilde{m}_i$

はそれぞれ M_t,$\tilde{M}$ の i 番目の行を表す。

$$s_{i,t} = (h_{i,t-1}, m_{i,t-1}) \tag{2.11}$$

$$f_{i,t} = W^f x_t + U^f h_{i,t-1} + b^f \tag{2.12}$$

$$i_{i,t} = W^i x_t + U^i h_{i,t-1} + b^i \tag{2.13}$$

$$o_{i,t} = W^o x_t + U^o h_{i,t-1} + b^o \tag{2.14}$$

$$m_{i,t} = \sigma(f_{i,t} + \bar{b}^f) \circ m_{i,t-1} + \sigma(i_{i,t}) \circ \underbrace{g_\psi(\tilde{m}_{i,t})} \tag{2.15}$$

$$h_{i,t} = \sigma(o_{i,t}) \circ \tanh(m_{i,t}) \tag{2.16}$$

$$s_{i,t+1} = (m_{i,t}, h_{i,t}) \tag{2.17}$$

式 2.15 において下線部が示す箇所は LSTM からの変更部分である。関数 g は既存研究 [7] に従い、MLP + layer normalization として実装した。パラメータ は各 m_i について共通する。

第3章

NTMへの関係推論能力の付与

提案ネットワークは2種類のメモリを持つモデルとして構成される。アーキテクチャの全体図を図3.1に示す。

ネットワークの構造は大きく3つのモジュールに分けられる。1つ目は入力の保存・忘却・読み出しを行う項目メモリ M_t^I とそのオペレータ。2つ目は項目メモリの項目間の関係推論を行い、計算した関係情報を保存する関係メモリ M_t^R とそのオペレータ。3つ目は項目メモリの内容を関係メモリに入力する前に項目メモリの項目間順序を整理する順序整理モジュールである。ネットワークはコントローラLSTMからの出力 o_t^I、項目メモリからの読み出しベクトル r_t^I、関係メモリからの読み出しベクトル r_t^R を結合したのちlogistic sigmoid活性化を行い最終的な出力とする。

項目メモリには2.1節で導入したNTMを用いる。関係メモリには2.2節で導入したRMCに変更を加えたものを利用し、この変更は3.1節で説明する。3.2節では新しく提案する順序整理モジュールを2種類説明する。

3.1 提案ネットワーク中の関係メモリ

RMCに変更を加え、項目メモリの各項目間の関係情報を保存する関係メモリとして提案ネットワークに実装する。提案ネットワーク中の関係メモリは図3.2に示すようになる。時間tにおける関係情報の計算のために、式2.10における入力 x_t とメモリ M_t^R にまたがるアテンションを項目メモリ M_t^I と関係メモリ M_t^R の連結に対するアテンションに拡張する。変更後は式3.1が示すように

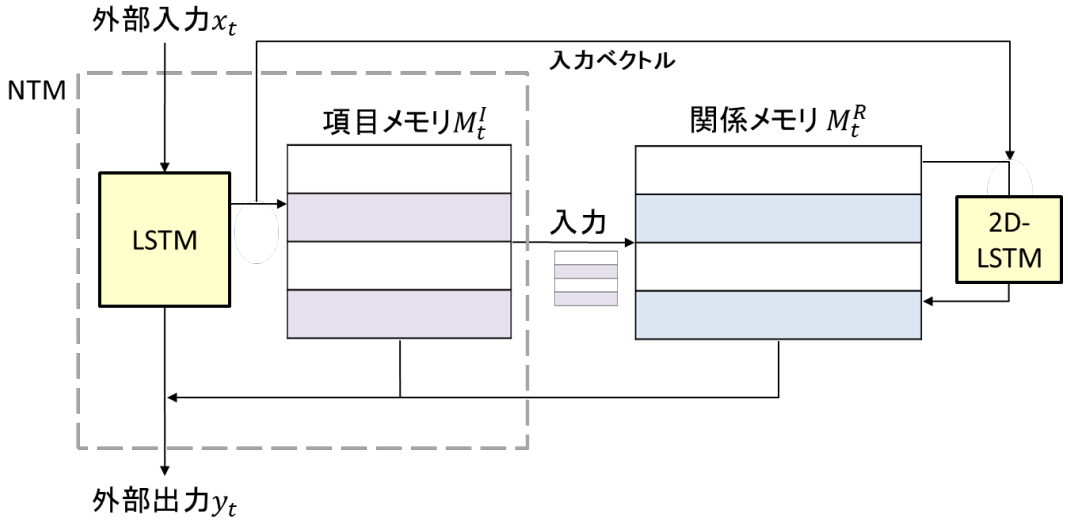

図 3.1: 提案ネットワーク

して $\tilde{M}$ が計算される。

$$\tilde{M} = softmax(\frac{M_{t-1}W_q([M_{t-1}; M_t^I]W_k)^T}{\sqrt{d_k}})[M_{t-1}; M_t^I]W_v \tag{3.1}$$

また M_{t-1}^R の更新時には、入力 x_t の代わりに項目メモリへの書き込みベクトル w_t^w を利用する。従って式 2.12-2.14 は式 3.2-3.4 のように変更される。

$$f_{i,t} = W^f w_t^w + U^f h_{i,t-1} + b^f \tag{3.2}$$

$$i_{i,t} = W^i w_t^w + U^i h_{i,t-1} + b^i \tag{3.3}$$

$$o_{i,t} = W^o w_t^w + U^o h_{i,t-1} + b^o \tag{3.4}$$

3.2 順序整理モジュール

3.2 節では項目メモリの順序を解釈・整理する 2 種類の機構を説明する。3.1 節で述べた拡張により、NTM のメモリを関係メモリへの入力とすることが可能となる。しかしこの実装では項目メモリにおける忘却や上書きは関係メモリの更新と独立して実行されるため、関係メモリへの入力の一貫性が損なわれる問題が予想される。そこで項目メモリを整理し一貫性を持った入力を生成する順序整理モジュールを提案する。

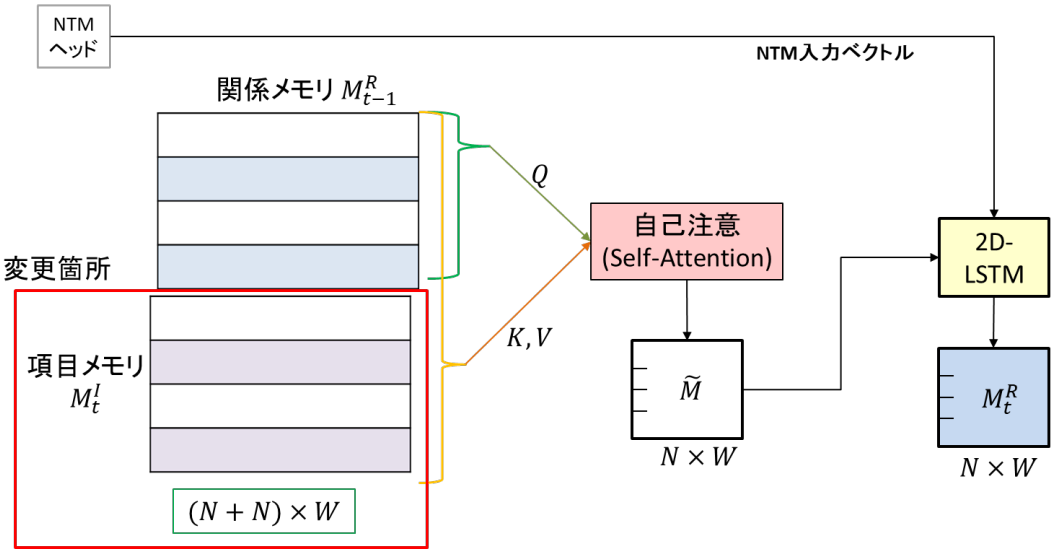

図 3.2: 提案ネットワーク中の関係メモリ

3.2.1 書き込み頻度ソート

項目メモリのヘッドによるメモリの忘却・上書きを疑似的に読み取る手段として、書き込み頻度によるソートを提案する。時間 t における各行の書き込み頻度 $q_t(N^I$ 次元ベクトル) は、式 3.5 が示すように $t = 1 \sim t$ での書き込み重みの総和とする。

$$q_t = \sum_t w_t^w \tag{3.5}$$

関係メモリは式 3.1 において M_t^I の代わりに q_t に基づいてソートされた M_t^I を入力として受け取る。

3.2.2 時間リンク行列を利用したグラフアテンション

DNC[4] では各ステップの内容を保存するだけでなく、入力系列内での前後関係の情報を保存するために時間リンク行列を実装している。時間リンク行列 L_t は式 3.6-3.8 によって示されるように、時間的に隣接する書き込み重みの outer production を保存する。

$$L_0[i, j] = 0 \forall i, j \tag{3.6}$$

$$L_t[i,i] = 0 \forall i \tag{3.7}$$

$$L_t[i,j] = (1 - w_t^w[i] - w_t^w[j])L_{t-1}[i,j] + w_t^w[i]p_{t-1}[j] \tag{3.8}$$

第一項は項目の上書きが発生した時、その位置のリンクをリセットするための項である。p_t は式 3.9,3.10 のようにして計算される。

$$p_0 = \mathbf{0} \tag{3.9}$$

$$p_t = (1 - \sigma_i w_t^w[i])p_{t-1} + w_t^w \tag{3.10}$$

提案モジュールでは時間リンク行列を各行が時間的に隣接する程度を表現するグラフと解釈して、グラフアテンション (GAT)[9] による各項目の変換を試みる。これにより関係メモリに入力される各項目の特徴量に時間的な順序情報が内包されることが期待できる。グラフアテンションによる解釈は各行の位置を変更しないため、2.4.1 節の書き込み頻度ソートと併用できる。

第4章

評価実験

全てのタスクにおいてネットワークはLSTMコントローラを採用し、ロジスティックシグモイド出力層を有する。クロスエントロピーを損失関数として訓練し、最適化手法としてはAdamを用いた。また、各入力系列の開始時にネットワークの隠れ状態はリセットされる。各タスクにおけるネットワークのパラメータ、データのバッチサイズ、学習率を付録〜に示す。実験1・2・4ではビットエラー数が1以下を記録した時点で、実験3では正答率が0.9以上を記録した時点でタスクを解決したとみなし実験を終了した。

4.1 実験1 メモリ効率性の検証

4.1.1 実験概要

NTMとSTMのメモリ操作における, メモリの容量を無駄なく使用する能力(以下メモリ効率性)を比較するためモデルのメモリサイズを揃えて性能を比較した。評価にはNTM,STMの論文中でも実験で使用されたアルゴリズムタスクであるPriority Sortタスクを用いた。NTMの論文中ではタスクの入力長が20であるのに対し行数128のメモリを用いて評価を行っている。本実験ではメモリの容量に余裕のない状態での性能を評価したいため、付録〜に示すようにメモリの行数は入力長と同じ20とした。比較するSTMのメモリサイズは提案ネットワークと同様に項目メモリ・関係メモリ共に20行とした。ただしSTMは複数の関係メモリに並列して読み書きを行うことで豊かな関係情報を計算・保存す

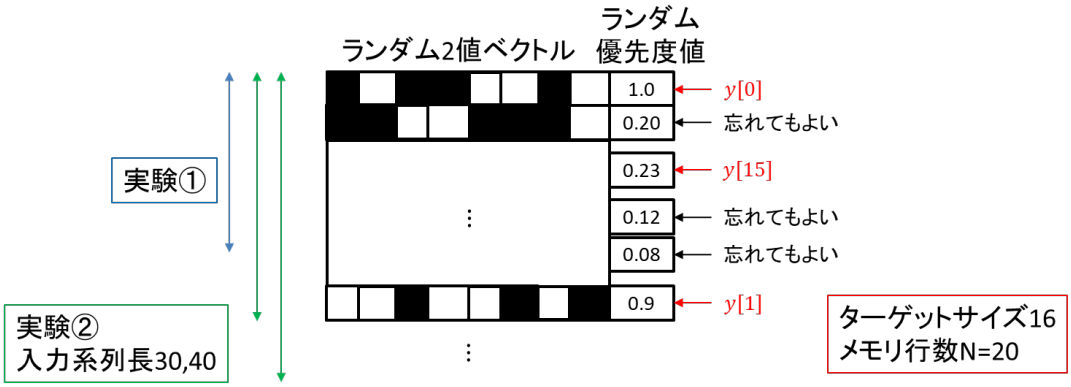

図 4.1: priority sort タスク

ることを長所にしており、論文中で評価されたモデルでも関係メモリは項目メモリの 8 倍の大きさを持つ。そこで関係メモリのサイズを 8 倍に設定した STM においても実験を行い、性能を比較した。また、実験 1・2 では各モデルにおいて 2 回ずつ独立した学習を行った。

4.1.2 タスク:priority sort

入力系列はランダムなバイナリベクトルに、[-1,1] の範囲から一様分布に従い決定した乱数を優先度として付加したものからなる。ターゲットは入力系列をこの優先度に従ってソートした系列の一部とする。NTM 中の設定に従い入力系列の長さは 20 ベクトル, ターゲットは系列の中から優先度が高い順に 16 ベクトルとする。評価指標にはビットエラー数を用いた。これは正規化されたモデルの出力を 0.5 をしきい値として 2 値の予測系列に変換した上で、ターゲット系列と異なるビットの数をカウントしたものである。図 4.1 は priority sort タスクの入力・ターゲットの構成を図示したものである。

4.1.3 実験結果

Priority Sort タスクにおける NTM および提案ネットワークの学習曲線を図 4.2 に示す。関係メモリが項目メモリと同サイズの STM の学習曲線を図 4.3 に,

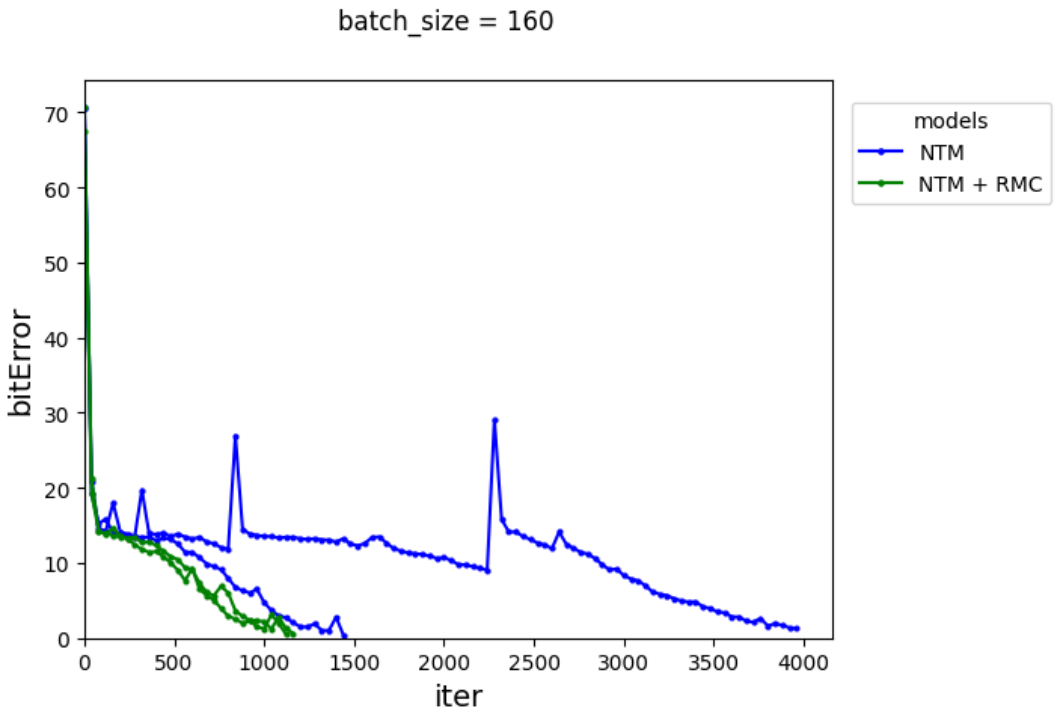

図 4.2: NTM および提案ネットワークの学習曲線 (入力長 20)

関係メモリサイズが 8 倍のときのものを図 4.4 に示す。以後、同色の学習曲線は同一のパラメータ設定下での実験を表す。

4.1.4 考察

既存研究よりも大幅にメモリサイズが削減された設定であったが、NTM および提案ネットワークはビットエラー数 1 を下回り、このタスクを解決することが出来た。STM では関係メモリのサイズに関わらず、NTM・提案ネットワークの 10 倍以上のイテレーションで学習した場合でもビットエラー数が 30 を下回ることは無かった。これらの結果から、メモリ効率性において NTM ベースのメモリネットワークの STM に対する優位性が示されたと考えられる。また NTM は 2 回の学習のうち 1 回は学習に不安定性が見られ、収束までに約 4000 ほどのイテレーションを要した。一方で提案ネットワークは安定して 1200 イテレーション未満で収束することに成功しており、これは NTM の最良の結果よ

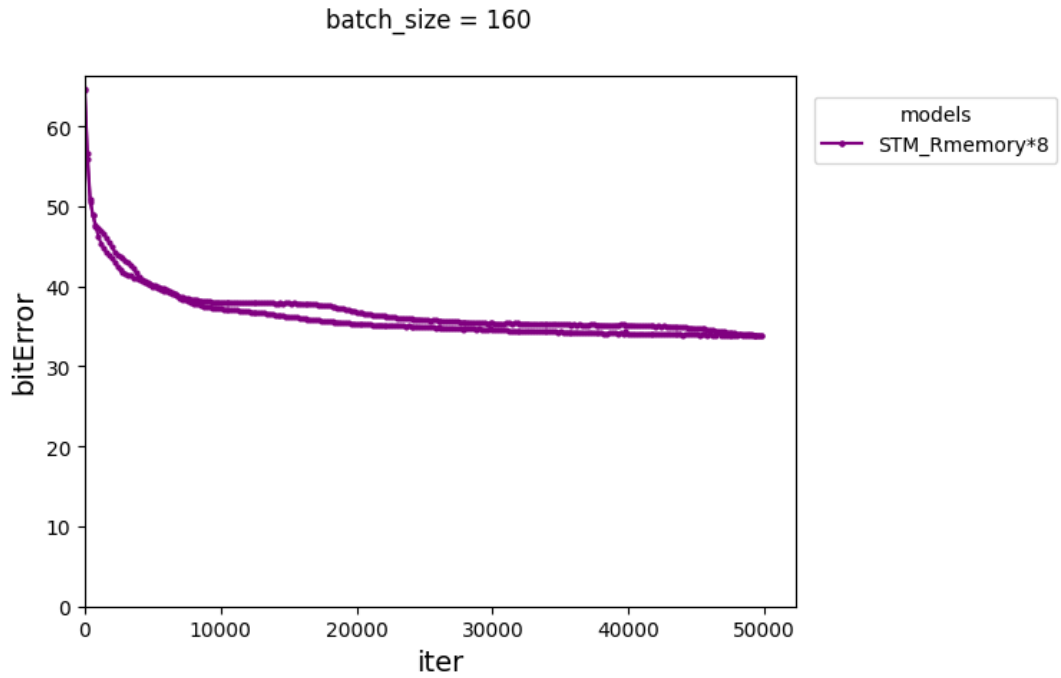

図 4.3: STM の学習曲線 (関係メモリ 1 倍・入力長 20)

りも早期の収束である。

4.2 実験 2 長期記憶・忘却能力の検証

4.2.1 実験概要

Priority Sort タスクにおいて入力系列のサイズを増加させたときの精度の推移を観察する。実験では入力長を 30 または 40 に設定した。ターゲットサイズは 16, メモリ行数は 20 から変更しないため、モデルは依然としてタスクの解決に必要な情報を全て保存する容量を有する。しかし入力の初期に提示された情報を長期間記憶する能力や優先度の低い不要な項目を選択的に忘却しメモリを再利用する能力が要求される。

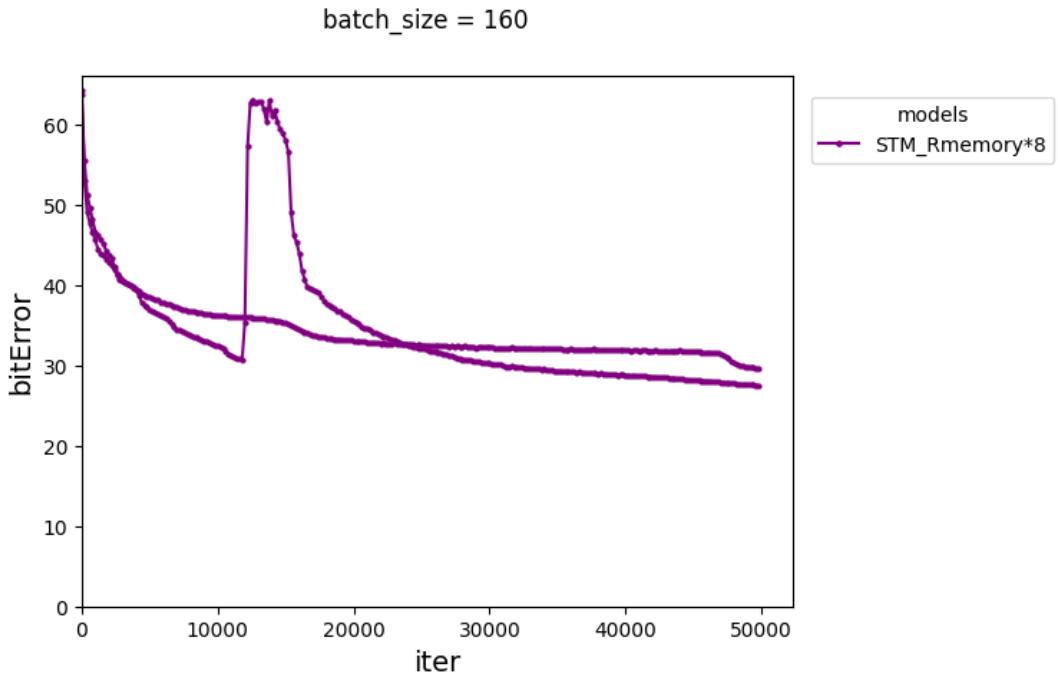

図 4.4: STM の学習曲線 (関係メモリ 8 倍・入力長 20)

4.2.2 実験結果

入力長 20,30,40 で NTM を訓練したときの学習曲線を図 4.5 に, 提案ネットワークを訓練したときのものを図 4.6 に示す。同様に関係メモリが項目メモリと同サイズの STM を訓練したときの学習曲線を図 4.7 に, 関係メモリサイズが 8 倍のときのものを図 4.8 に示す。

4.2.3 考察

関係メモリが 8 倍のサイズを持つ STM では、入力長の増加に伴いエラー率が増加することが読み取れる。関係メモリが項目メモリと同サイズの STM では、イテレーション数が 20000 以下の範囲ではエラー率に増加が見られるが 30000 以上では収束し差がなくなる様子が確認できた。NTM および提案ネットワークでは、入力長が 30 の場合は収束が大幅に遅くなりイテレーション数 4000 までに解決できない場合も確認された。ただし、学習初期の 500 イテレーション以

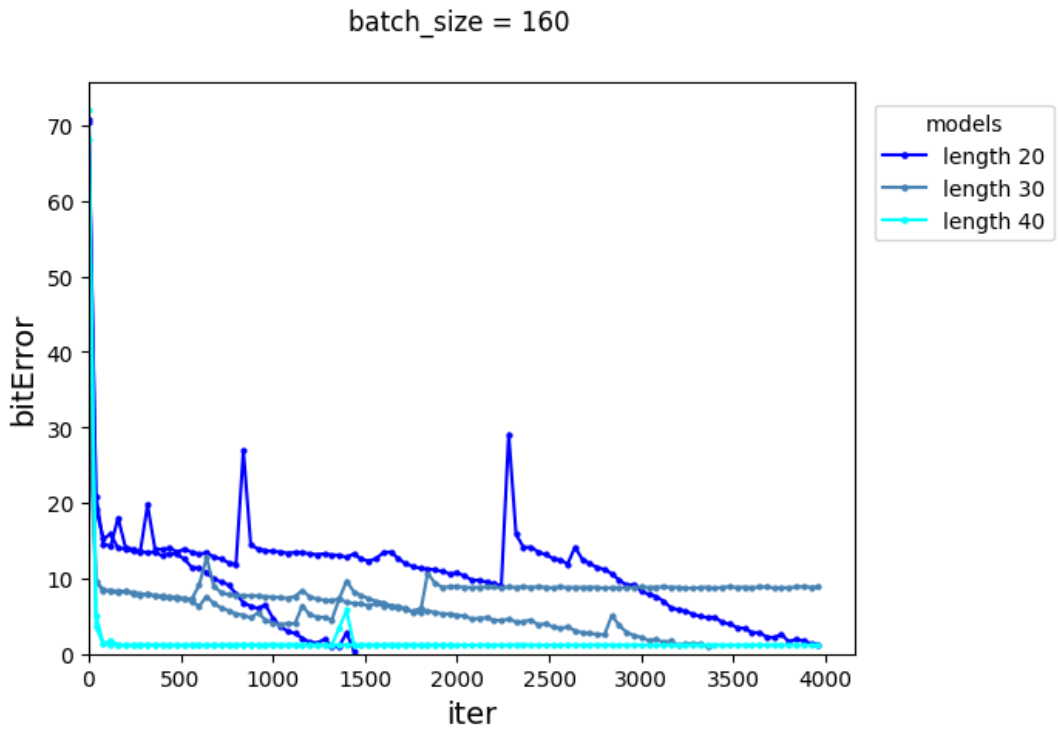

図 4.5: NTM の学習曲線 (入力長 20,30,40)

下でのエラー率は入力長 20 のケースよりも改善されており、最終的なエラー率も STM の結果に勝っている。入力長を 40 とした場合の学習では 2 回の学習共に非常に早期に収束し、モデルの解決能力は大幅に改善したと言える。

4.3 実験 3 関係推論能力の評価

4.3.1 実験概要

この実験の目的は提案手法が NTM に関係推論能力を付与出来るかを評価することである。実験で用いる Nth-farthest タスクにおいて、既存研究 [7][8] は RMC や STM のような関係推論能力を持つモデルが 90 以上の精度を記録した一方で、NTM や DNC の精度は 30 を超えないことを示している。提案ネットワークが NTM メモリ項目間の関係を計算できる場合、NTM よりも大きく改善された精度を示すことが期待される。

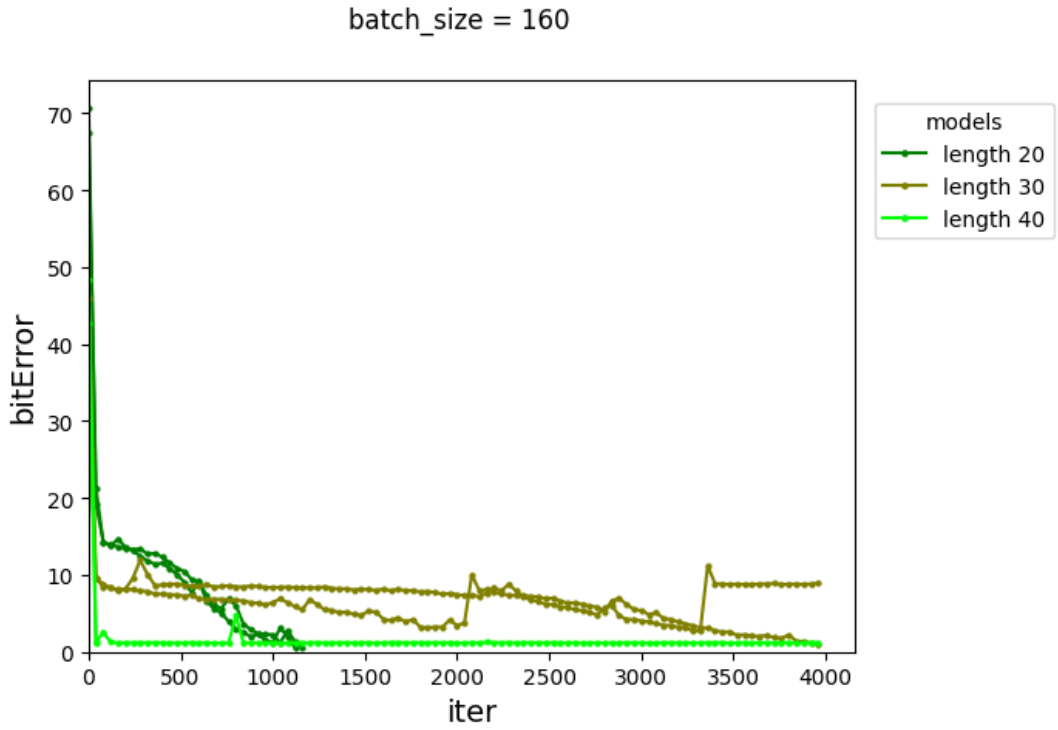

図 4.6: 提案ネットワークの学習曲線 (入力長 20,30,40)

4.3.2 タスク:Nth-farthest

入力系列はランダムに生成されたバイナリベクトルからなる。ベクトルの次元を d、系列長を l で表したとき、タスクの要求は入力系列中のあるベクトル m から n 番目に遠いベクトルを見つける事である。m, n は入力系列ごとにランダムに決定される。1 ステップあたりの入力はバイナリベクトル、ベクトルの ID、m、n を連結したベクトルからなり、$ID, m, n \in \{1, 2, \cdots, l\}$ は one-hot エンコーディングにより表現されるため、最終的な入力の次元は $d + 3l$ になる。学習に時間を要する実験であったため、入力データの長さ・次元は既存研究 [7] の半分の $d = 8, l = 4$ とした。それに伴い、メモリの大きさも既存研究の半分に設定した。モデルは正答となるベクトルの ID を出力することを要求されるため、このタスクは分類問題である。評価指標には正答率を用いる。

このタスクは入力の読み書きやソートといったタスクよりも複雑な処理をネットワークに要求する。ネットワークは m と全入力のペアの距離を計算しソート

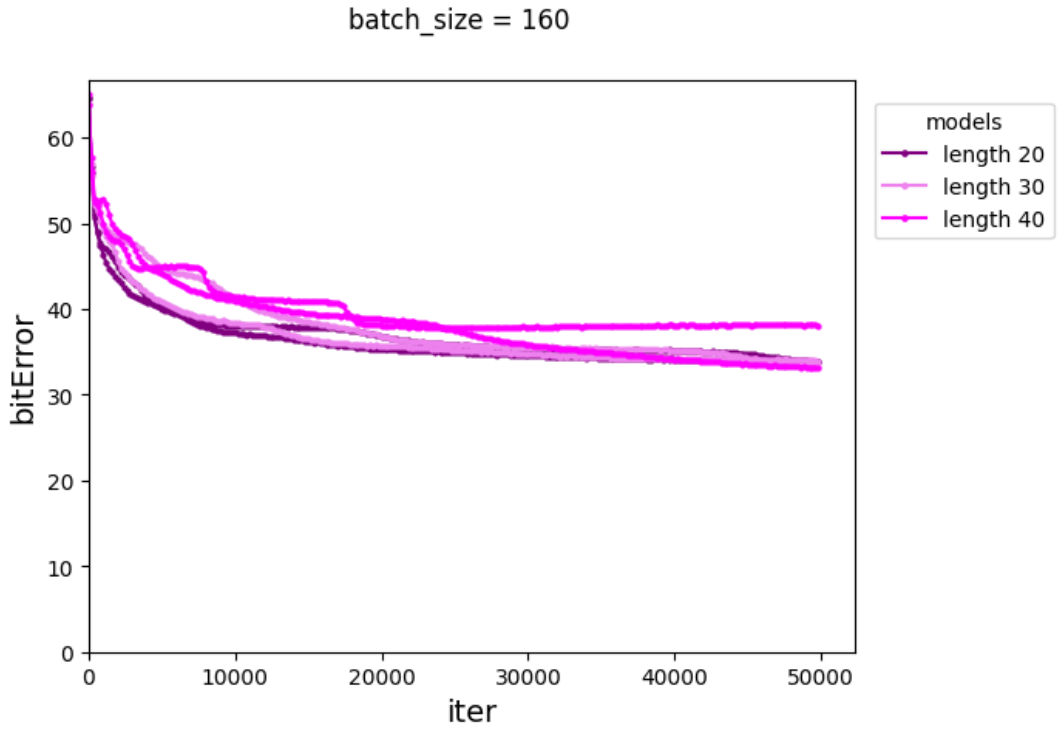

図 4.7: STM の学習曲線 (関係メモリ 1 倍・入力長 20,30,40)

を行う必要がある。距離は入力間の関係情報の一形態である。入力そのものをソートするタスクと異なり、関係情報のソートの為に関係メモリを活用する必要がある。

4.3.3 実験結果

Nth-farthest タスクにより RMC および提案ネットワークを訓練したときの学習曲線を図 4.9 に示す。

4.3.4 考察

実験結果は提案ネットワークの精度は既存手法と異なり 90%に達することはなく、提案ネットワークに関係推論能力は確認されなかった。

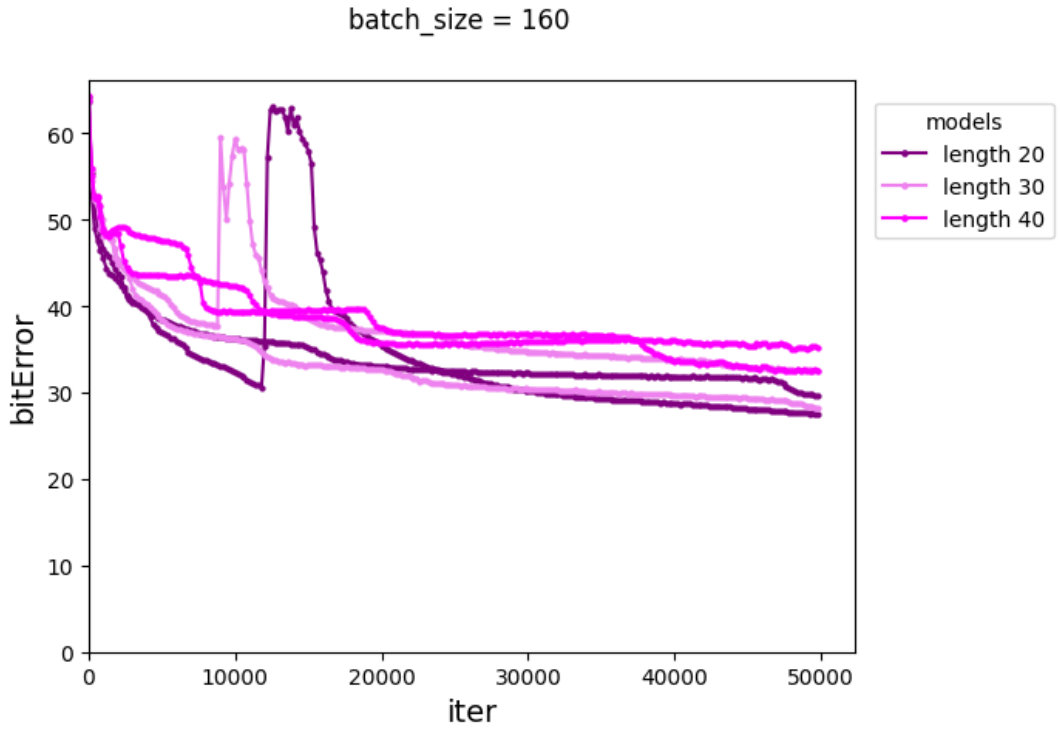

図 4.8: STM の学習曲線 (関係メモリ 8 倍・入力長 20,30,40)

原因の一つとして考えられるのは、関係メモリが受け取る情報が不足していたことである。関係メモリが受け取る入力は項目メモリの保存内容のみであり、NTM のヘッド内でメモリの更新や読み出しの判断に利用した情報を推論に用いることはない。改善としてヘッド内で計算した値をメモリ操作を介さず直接関係メモリに渡す経路の追加が考えられる。特に書き込み重みは、メモリの各行の内容を修飾するのに適した情報であると考えられる。

項目メモリ内の項目の順序に一貫性が無いことが、推論を妨げている可能性も考えられる。[10] は非分散的に項目を格納する行列に関係推論をかける既存研究の例である。この論文中のモデルは項目単位の忘却機構を持たず、入力系列に各時間ステップで埋め込みを適用したものを項目記憶として用いる (論文中では短期記憶として記載されている)。このため項目は提示された時刻に従って並ぶ一方で、提案ネットワークでは項目の上書きが発生するため、項目の並び順に一貫性がなくなる可能性がある。この問題を解決する方法としては、時

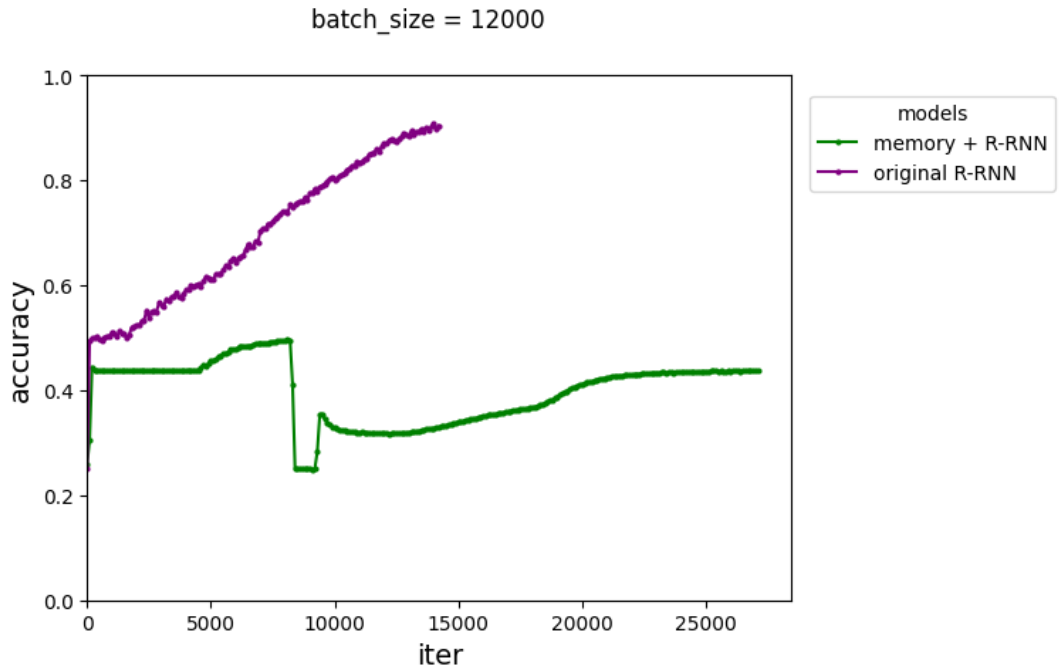

図 4.9: RMC および提案ネットワークの学習曲線

間順序に関する情報を別に保存し関係メモリに入力すること、あるいは時間とは別の基準をもってメモリを一貫性を持った並びに変えることが有効であると考えられる。

また [NTM-implement] は NTM はハイパーパラメータが共通でも、収束までの時間がばらついたり収束に失敗したりする場合があることを指摘している。提案ネットワークがこうした NTM の不安定性を引き継ぎ、実験では偶然収束しなかった可能性も考えられる。同一の設定で何度か追加で学習し、平均精度や収束した回数で評価する必要があると言える。

4.4 実験 4 関係推論能力改善の検証

4.4.1 実験概要

4.3.4 の考察を踏まえ、項目メモリへの書き込みに関する情報を用いて関係メモリの入力を改善し、項目メモリと関係メモリの連携を強化することを試みた。

実験では 4 種類の異なる方法で変更を加えたモデルの性能を NTM や提案ネットワークと比較し, 評価した。1 つ目のモデルでは式 4.1 で計算される, 書き込み重みの総和 q_t に基づき項目メモリをソートしたものを関係メモリへの入力とした。このモデルではヘッドにおける計算の内容を直接提示することではなく、関係メモリの入力を一貫性を持った並びにすることを目的としている。入力系列全体を通して頻繁に提示される項目ほどコンテンツベースの重みによる更新が発生し、q_t は大きな値になると考えられる。そのため関係メモリ内の self-attention 入力層の重みは常に同程度の出現頻度の項目を提示されることが期待される。ヘッドが複数存在する場合、全ヘッドの q_t の総和に基づいてソートを行う。実験 4 では各モデルにおいて 3 回ずつ独立した学習を行った。

$$q_t = \sum_t w_t^w \tag{4.1}$$

2 つ目から 4 つ目のモデルでは書き込みに関する情報を項目メモリに新たな列として追加し, 関係メモリへの入力とした。ヘッドが複数存在する場合、ヘッドごとに異なる列として追加される。2 つ目では書き込み重みの総和 q_t を追加した。3 つ目では式 4.2,4.3 に示すように、書き込み重みの各要素にしきい値処理を適用した値の総和 c_t を追加した。書き込み重みの総和を用いる試みでは、一つの行に一つの項目が強く書き込まれた場合と、弱い強度で数回書き込まれた場合の区別を付けることが出来ない。3 つ目の試みは 0.1 以上の強度の書き込みを全て一度の書き込みとみなし、各行の書き込み回数を保存・利用することを意図したものである。

$$c_t = \sum_t f(w_t^w) \tag{4.2}$$

$$f(w_t^w)(i) = \begin{cases} 1 & (w_t^w(i) \geq 0.1) \\ 0 & (w_t^w(i) < 0.1) \end{cases} \tag{4.3}$$

4 つ目では最後に書き込みが行われた時間ステップの情報を項目メモリに追加した。入力項目が提示された順序の情報を保持し、推論に利用することを目的としている。付与される時刻は 0-1 に正規化され、書き込みが行われたかどうかの判定は 3 つ目の試みと同じく 0.1 をしきい値とした。

4.4.2 タスク:associative recall

NTM[3] の評価で用いられたシンプルなアルゴリズムタスクの一つである。一定の長さのランダムに生成されたバイナリベクトルからなる系列を 1 アイテムとし、始めにネットワークにはランダムな数のアイテムが入力される。入力アイテム系列が提示された後、クエリとして入力アイテムのうちの一つが改めて入力される。この時ターゲットとして、入力アイテムの中でクエリアイテムの次に提示されたアイテムを取る。各アイテムの間および入力とクエリの間にはデリミタを表すベクトルが挿入される。NTM[3] 中の設定に従い、ベクトルの次元を 6,1 アイテムの長さを 3 とし、アイテムの数は一様分布を用いて 2-6 のいずれかに決定する。評価指標には Priority Sort と同様にビットエラー数を用いる。

タスクはクエリアイテムをメモリから検索する、入力ベクトルを忠実に復元するといったメモリネットワークの基礎的な能力を要求する。加えて、入力アイテムの順序の情報を保存する能力も要求する。

4.4.3 実験結果

Associative Recall タスクにより NTM および提案ネットワークを訓練した場合の学習曲線を図 4.10 に示す。

書き込み重みの総和を付与した場合, および総和に従いソートを行った場合の学習曲線を NTM のものと共に図 4.11 に示す。各行の書き込み回数, 最後に書き込んだ時刻を付与したモデルの学習曲線を図 4.12 に示す。

4.4.4 考察

結果からこのタスクにおいては、NTM と提案ネットワークの間に顕著な違いは確認されなかった。重みの総和に基づくソートでも、平均して学習曲線に大きな変化はなかった。重みの総和あるいは最後に書き込んだ時刻を付与した場合は, 平均してエラー率の収束は遅くなったことが読み取れる。書き込み回数

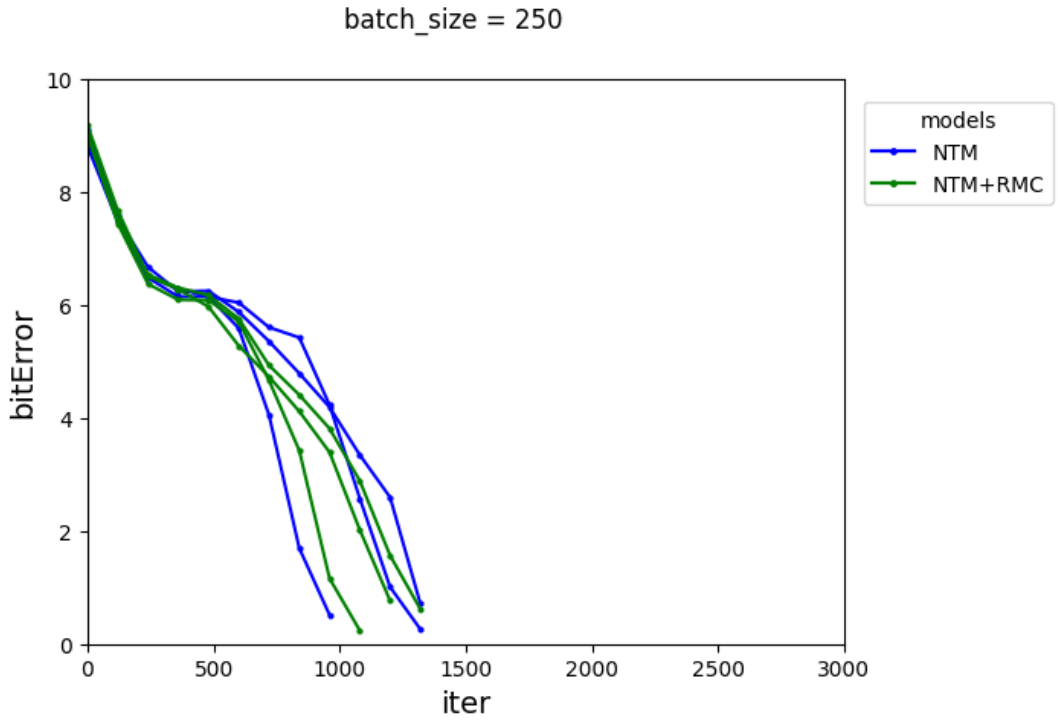

図 4.10: NTM および提案ネットワークの学習曲線

を付与した場合,3 回中 2 回の学習では NTM に大きくは劣らなかったが 1 回では収束が大幅に遅れ, モデルの安定性に悪影響を与えたと考えられる。

唯一 NTM・提案ネットワークから性能にほぼ悪化が無かったソートを適用したモデルでは、関係メモリの入力の列数に変化はなかった。性能が悪化した 3 つのモデルでは関係メモリのサイズに変化はない一方で、各行への書き込みに関する情報として項目メモリに新たな列を付加していた。これらを比較すると、付加した情報が余分な情報として他の情報が保存されるスペースを圧迫してしまったことが性能悪化の一因と考えられる。改善として付加される列数＝ヘッド数の分関係メモリのサイズを増加させることが考えられる。

最後に書き込んだ時刻および書き込み回数に改善が見られなかった一因として、しきい値処理が逆伝播不可な操作であることも考えられる。メモリ操作をスキップしてヘッドの情報を関係メモリに渡すことを目的とした操作だが、スキップした接続を通して書き込み重みの計算を最適化可能にする点に改善の余地がある。しきい値の代わりに急峻な sigmoid 関数のような微分可能関数を適

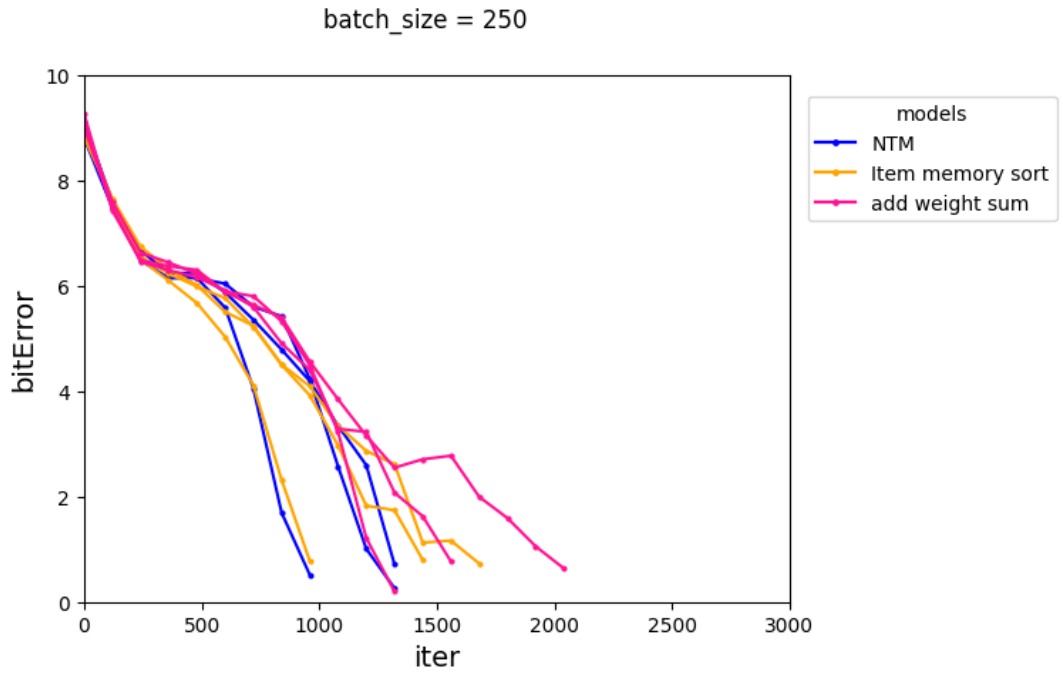

図 4.11: 書き込み重みの総和の付与, および総和に従ったソートを行ったときの学習曲線

用した場合との比較も今後行いたい。

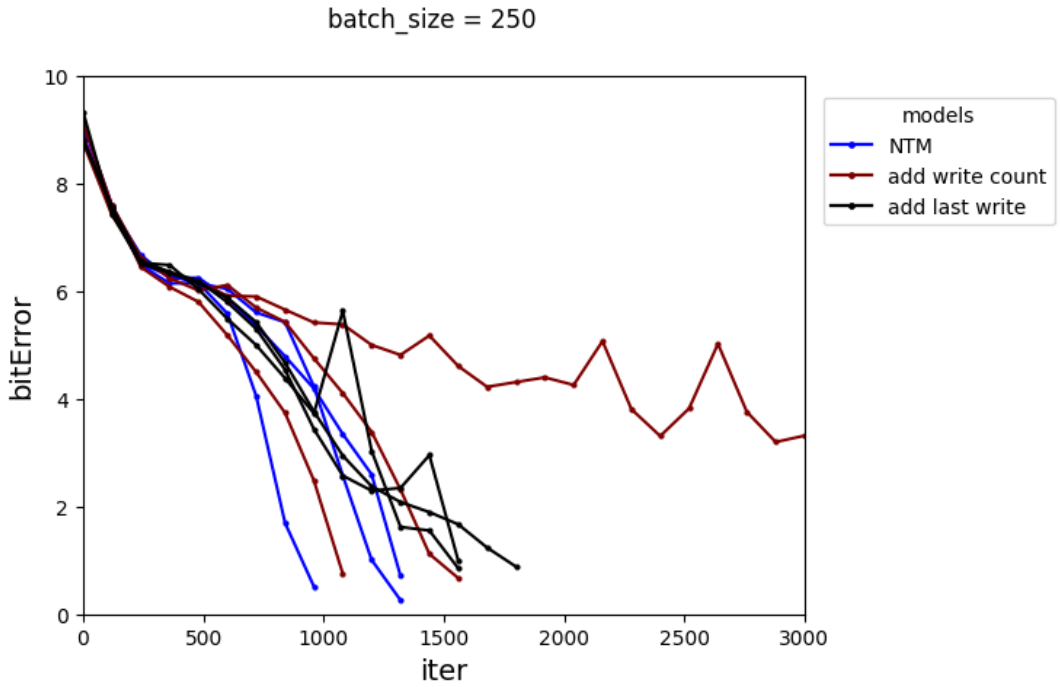

図 4.12: 各行の書き込み回数, 最後に書き込んだ時刻の付与を行ったときの学習曲線

第5章

結論

本論文では

記法メモ

test[11] [12] [3] [4] [5] [7] [13] [8] [9] [10] Wavenet[14] は音声波形を時系列データとして自己回帰モデルで学習することによって，人間の声のような自然な音声を生成することができる．時点 t における観測値を x_t，$\boldsymbol{x} = \{x_1, ..., x_T\}$ を観測値の全体集合とする．このとき，波形の同時確率は 条件付き確率の積として以下のよう表現される．

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}) \tag{5.1}$$

つまり，x_t は前時点の全てにおけるサンプルに条件づけられる．

図 5.1 に因果的畳み込み層のスタックを示す.

[15] は.

今後の課題を以下に挙げる．

- の向上

  必要がある.

- への応用

  を行いたい.

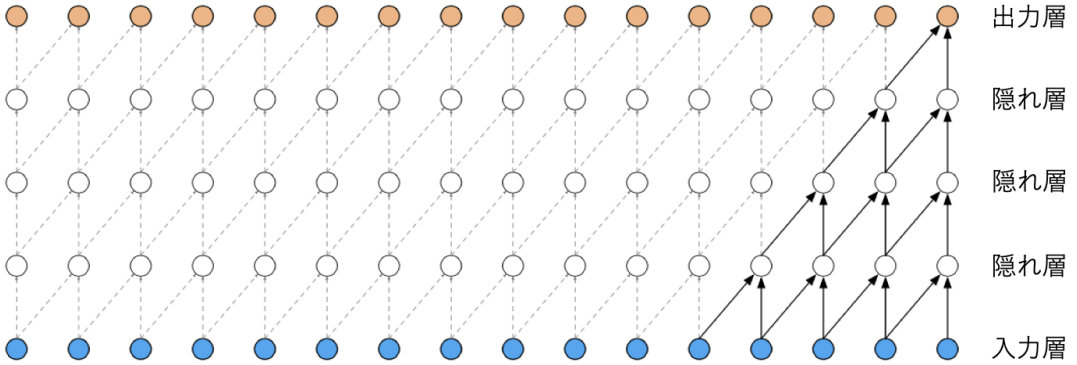

図 5.1: 因果的畳み込み層

- の改善

  今後，取り組みたい．

しかし RMC の追加によって学習にかかる時間が増加し、学習の高速化

sam は項目メモリ内容の間の関係情報を関係メモリに保存する際に、線形変換で圧縮=項目の抽出？をしている。入力が大きすぎて冗長なのが問題の場合、オートエンコーダのように圧縮・抽出したり、項目メモリ→関係メモリの部分のために読み出し操作をもう一つ実装するとよいかもでかすぎたか　RMC はもともと１ベクトルを入力　無理やり拡張した　クエリとの齟齬も

上の実験とは関係なく、実験２の考察を踏まえて、今後修士で試す予定の改善のこともいくつか考えていますがこれらは結論に書く形になると思います。ひとまず以下に箇条書きします

関係メモリを RMC ではなく SAM にしてみる。もともと RMC は１メモリモデルで入力がベクトルなので、無理に拡張するよりこちらのほうがいいかも

SAM や [10]〜のような２メモリモデルを見る限り、関係メモリから項目メモリの内容を復元する操作がある。今は項目→関係の一方向だけど逆も入れてメモリが相互作用できるようにする

sam は項目メモリ内容の間の関係情報を関係メモリに保存する際に、線形変換で圧縮=項目の抽出？をしている。入力が大きすぎて冗長なのが問題の場合、オートエンコーダのように圧縮・抽出したり、項目メモリ→関係メモリの部分のために読み出し操作をもう一つ実装するとよいかも

似た話として graph attention を利用して時間的・コサイン類似度的に近い項目に重み付けをしたセルフアテンションをかける。この結果を関係メモリに保存する (rmc,SAM とは異なる構造の関係メモリになる)

babi

また現在の提案ネットワークにおける情報の受け渡しは項目メモリから関係メモリの方向に限定されている。一方で、既存の 2 メモリモデルには関係メモリから項目メモリへと情報を渡す経路も存在する。STM には関係メモリが関連する項目を復元し、項目メモリに改めて書き込む操作が実装されている。[LTD] は長期記憶と作業記憶からなる 2 メモリモデルであるが、一方の読み出しベクトルを入力に結合しもう一方の読み書き操作で利用する。提案ネットワークにもこのような経路を導入し、2 つのメモリの連携を強化する改善が考えられる。Nth-farthest タスクにおいては関係メモリのみを持つ RMC により十分な精度を達成することが示されている。一方で [8] では、最短経路や最小全域木の探索タスクや同論文で提案された RAR タスクにおいては 2 つのメモリの存在が性能を向上させることが示されている。このようなタスクにおいて、上記の改善が提案ネットワークの性能を改善させる可能性が考えられる。

時間順序に着目すると　DNC の link matrix A

謝辞

本研究を行うにあたり親身に相談に乗っていただき，ご指導してくださった萩原将文教授，ならびに共に問題解決，議論，相談に付き合ってくださった研究室の先輩方，同期の皆様に深く感謝いたします．誠にありがとうございました．

参考文献

[1] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[2] Jason Weston Sukhbaatar Sainbayar and Rob Fergus. “End-to-end memory networks”. In: *Advances in neural information processing systems* 28 (2015).

[3] Greg Wayne Graves Alex and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[4] Reynolds M. et al Graves A. Wayne G. “Hybrid computing using a neural network with dynamic external memory”. In: *Nature* 538 (2016), pp. 471–476.

[5] Shiming Xiang and Bo Tang. “CSLM: Convertible Short-Term and Long-Term Memory in Differential Neural Computers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.9 (2020), pp. 4026–4038.

[6] et al. Vaswani Ashish. “Attention is all you need.” In: *Advances in neural information processing systems* 30 (2017).

[7] et al Santoro Adam. “Relational recurrent neural networks”. In: *Advances in neural information processing systems* 31 (2018).

[8] S. Venkatesh H. Le T. Tran. “Self-attentive associative memory”. In: *International Conference on Machine Learning,PMLR* (2020), pp. 5682–5691.

[9] P. Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

[10] Héctor Allende Pavez Juan and Héctor Allende-Cid. “Working memory networks: Augmenting memory networks with a relational reasoning module”. In: *arXiv preprint arXiv:1805.09354* (2018).

[11] J. Weston et al. “Towards ai-complete question answering: A set of prerequisite toy tasks”. In: *arXiv preprint arXiv:1502.05698.* (2015).

[12] Sumit Chopra Weston Jason and Antoine Bordes. “Memory networks”. In: *arXiv preprint arXiv:1410.3916* (2014).

[13] et al Santoro Adam. “A simple neural network module for relational reasoning”. In: *Advances in neural information processing systems* 30 (2017).

[14] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).

[15] Biing Hwang Juang and Laurence R Rabiner. “Hidden Markov models for speech recognition”. In: *Technometrics* 33.3 (1991), pp. 251–272.

付録 A

データの前処理

提案ネットワークに考えられる課題として、NTM 内部で項目の並び順に一貫性が無くなることで、関係メモリの学習を阻害することが予想される。これは入力との類似度に応じてスロットの忘却・上書きを行う機構や、DNC では空いた場所から優先的に書き込む機構が関係推論の計算と独立していることによる。この問題は入力が長期化し、忘却が頻繁に行われる場合により顕著になると考えられる。メモリ全体を常に一貫性をもって解釈するために、以下の 2 種類のモジュールを提案する。全体としてはこれらのモジュールが項目メモリの順序を整理・解釈した後に関係メモリに入力する構造となる。

１．DNC の時間リンク行列はスロット間の時間的な前後関係を表すグラフと捉えられる。これを元にメモリ項目にグラフアテンション (GAT)[9] をかけることで、時間的に局所的な情報 (文脈) を考慮した情報へと解釈できる。

２．各スロットの書き込み頻度をもとにソートする。

End-To-End Memory Networks[2] のようにメモリ内でデータが時系列で並ぶモデルや SAM[8] のように項目がメモリ全体に分散するモデルでは学習を通してメモリの解釈は一貫していると考えられるため、順序整理モジュールは新しい取り組みである。{要サーベイ}

A.1　ヒストリカルデータ

付録B

モデルのパラメータ

B.1　の予測

表??に

付録C

メモリネットワークの説明可能性

C.1　異なる

あ

付録D

のヒストグラム

D.1　異なる期間

図