

重み予測を用いたMLPの初期化による少数例からのダイナミクス 深層学習

Learning dynamics from limited sample with multilayer perceptron initialized by weight prediction

ポア インジュン^{*1} 井上 克巳^{*1*2}
Yin Jun Phua Katsumi Inoue

^{*1}東京工業大学 ^{*2}国立情報学研究所
Tokyo Institute of Technology National Institute of Informatics

Real world data are often difficult to obtain. Logical machine learning methods can produce perfect explanations for dynamics of systems when the full state transitions can be observed, but such scenario is often impossible. Statistical machine learning methods also usually require a huge amount of data. In this work, we propose a method that predicts the initial weight of an MLP to learn a model that can predict future state of a delayed system even when only a limited amount of observation is provided. We also show the effectiveness of the method applied to systems with particularly a large number of variables.

1. はじめに

論理的機械学習は、学習した結果が解釈可能であり、さらには人間による認証も可能である。ダイナミクス学習では、論理的機械学習が応用されており、調べたいダイナミックシステムが取りうる全ての状態遷移が完璧に得られれば、論理的機械学習手法を用いて完璧なモデルが得られる [7, 12, 13]。しかし実世界では、観測データや状態遷移を完璧に得られることが珍しい。データを観測するためには、実験を行わなければならない。実験を行うには、資金が必要であり、特殊な条件の下で行うことなど、様々な問題がある。それ以外に、現実世界で観測されたデータは通常ノイズや誤りがある。論理的機械学習はノイズや誤りに対して頑健でないため、現実応用が非常に困難である。一方、深層学習はノイズや誤りに対して頑健である。従って、論理的機械学習の利点および深層学習のノイズに対して頑健な部分を組み合わせる方法が求められている [4]。

近年機械学習では過剰適合を防ぐために、様々な技術が開発されており、ドロップアウトや正規化など、深層学習を行う際に過学習をうまく避けられるような技術が提案された。一方、深層学習においては重みの初期化は依然として学習がうまくいくための重要な要素として見られている [14, 15]。

本研究では、機械学習を用いたニューラルネットワークの重みの予測による初期化の方法を提案する。対象とするネットワーク構造は多層パーセプトロン (multi-layer perceptron; **MLP**)。重みを機械学習で予測することにより、学習データが少ない場合でも過学習を避け、モデルの学習がうまくいくことを示す。

本論文ではまず、対象とするダイナミックシステムについて説明する。次に、提案する重み予測モデルについて説明をする。そして、本研究で行われた実験をいくつか示し、最後に考察および関連研究について述べる。

2. 背景

本研究では主に標準論理プログラム (normal logic program; **NLP**) として表現できるダイナミックなシステムを対象とする。**NLP** は

$$A \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg A_{m+1} \wedge \cdots \wedge \neg A_n \quad (1)$$

の形式をとったルールの集合である。ここで A 及び A_i はアトムである ($n \geq m \geq 0$)。任意の (1) の形式をとるルール R において、 A は R のヘッドであり $h(R)$ で表す。そして、 $\leftarrow$ の右辺の連言は R のボディであり R のボディで現れるリテラルの集合を $b(R) = \{A_1, \dots, A_m, \neg A_{m+1}, \dots, \neg A_n\}$ で表し、 R のボディで正リテラルとして現れるアトムを $b^+(R) = \{A_1, \dots, A_m\}$ 、負リテラルを $b^-(R) = \{A_{m+1}, \dots, A_n\}$ として表す。

エルブラン解釈 I はエルブラン基底 B の部分集合である。論理プログラム P 及びエルブラン解釈 I を考える時、 T_P オペレータは $T_P : 2^B \rightarrow 2^B$ の対応である。この時 T_P オペレータは

$$T_P(I) = \{h(R) \mid R \in P, b^+(R) \subseteq I, b^-(R) \cap I = \emptyset\} \quad (2)$$

として定義できる。

状態が同期的に変化していくシステムを表現するには、(1) を

$$A^{t+1} \leftarrow A_1^t \wedge \cdots \wedge A_m^t \wedge \neg A_{m+1}^t \wedge \cdots \wedge \neg A_n^t \quad (3)$$

のようにダイナミックな形式にできる。ここで、 t は時間ステップであり、 $t+1$ は次の時間ステップとして考える。つまり、(3) では、時間 t で $A_1, \dots, A_m$ が真、そして $A_{m+1}, \dots, A_n$ が偽である時、時間 $t+1$ で A が真となることを表している。

状態遷移の集合 E が与えられた時、その全ての遷移 $(I, J) \in E$ について $J = T_P(I)$ を満たす論理プログラム P を学習するアルゴリズムを解釈遷移からの学習 (learning from interpretation transition; **LFIT**) と呼ぶ [7]。いくつかの **LFIT** アルゴリズムが提案されている [5, 12, 13]。

このような記述を、Markov(k) システムへ拡張する。 k ステップ前までの遷移を考慮した時、論理プログラム P のエルブラン基底を

$$B_k = \bigcup_{i=1}^k \{v^{t-i} \mid v \in B\} \quad (4)$$

で表す。ある Markov(k) システム S を考えた時、 S の全ての変数を B として、 S のルール R を $h(R) \in B$ 、 $b(R) \in B_k$ とすれば S は論理プログラムとして表すことができる。 S の実行トレース T は S の状態の有限系列として考える。この時、実行トレースは $T := (S_0, \dots, S_n), n \geq 1, S_i \in 2^B$ として書ける。ステップ k の解釈遷移はエルブラン解釈 (I, J) として表し、 $I \subseteq B_k, J \subseteq B_{k+1}$ である。

単一のパーセプトロンは次のように定義できる

$$y = \sigma(\mathbf{w}^\top \mathbf{x} + b) \quad (5)$$

$\mathbf{w}$ は重みであり, $\mathbf{x}$ は入力ベクトル, b はバイアス, そして $\sigma(x)$ は活性化関数である. 複数のパーセプトロンを合わせると一つの層となり, 次のように書ける

$$\mathbf{y} = \Phi(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (6)$$

ここで $\mathbf{W}$ は重みの行列であり, $\mathbf{b}$ はバイアスのベクトルである. $\Phi(\mathbf{x})$ は要素ごとに活性化関数を適応する非線形関数である. **MLP** は二つ以上のパーセプトロン層により構成される. 2 層の **MLP** は数式として以下のようにあらわせる

$$\mathbf{y} = \mathbf{W}_2 \Phi(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \quad (7)$$

$\mathbf{W}_i$ は i 層目の重み行列であり, $\mathbf{b}_i$ は i 層目のバイアスベクトルである.

次に本研究で主に使用するニューラルネットワークモデルの構造について説明する. 再帰型ニューラルネットワーク (Recurrent Neural Network; **RNN**) はフィードフォワードニューラルネットワークを系列データ扱うために拡張したものである. 系列データの入力 $(x_1, \dots, x_T)$ が与えられた時, 3 つの重み行列 W^{hx}, W^{hh}, W^{yh} 及び 3 つのバイアス項ベクトル b_h, b_y, h_0 を用いて, 標準 **RNN** は

$$h_t = \sigma(W^{hx} x_t + W^{hh} h_{t-1} + b_h)$$

$$y_t = W^{yh} h_t + b_y$$

を時間ステップごとに計算し $(y_1, \dots, y_T)$ を出力するものである. h_t は各時間ステップの隠れ状態である. ここで σ はシグモイド関数である.

標準 **RNN** は長期的な依存性が存在する系列データの学習においてはとても困難である [2]. また勾配降下法を用いて学習する時に伝搬消失問題もある [2], これに対して, Long Short-Term Memory (**LSTM**) は長期的な依存性が存在する問題も扱うことができるので, **LSTM** を用いて重み予測を行うことにする.

本論文では [6] で提案された **LSTM** モデルを用いる. 系列データの入力 $X = \{x_1, x_2, \dots, x_{n_X}\}$ が与えられた時, **LSTM** は各時間ステップに入力ゲート i_t , 出力ゲート o_t 及び忘却ゲート f_t を与え, 一つのメモリセルを形成する. [6] の記号に従い, 各メモリセルの出力を h_t , 各メモリセルへの入力を l_t , 隠れ状態を c_t として表すと, その出力 h_t は

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ l_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \cdot \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t$$

$$h_t = o_t \cdot c_t$$

として表すことができる.

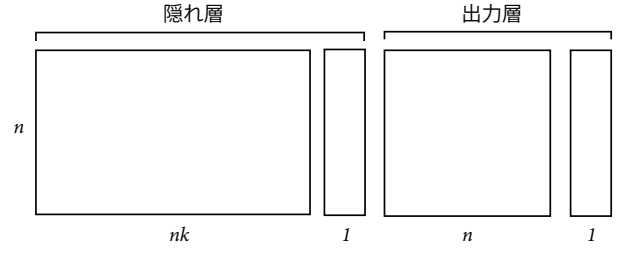


図 1: **MLP** の各パラメータの次元

3. 重み予測モデル

Gentet らは [5] で, **MLP** を用いて **NLP** をモデル化をした. ここでは [5] とは異なる **MLP** の構成方を考える.

ある時間 k の遅延まで考慮するダイナミックシステムのエルブラン基底 $\mathcal{B}$ の要素数を $|\mathcal{B}| = n$ とする. この時, $n \times k$ 個の入力を持ち, n 個の出力を持つ 2 層の **MLP** を構成する. **MLP** の入力エルブラン基底 $\mathcal{B}_k$ の各要素に対応し, 出力は $\mathcal{B}$ の各要素に対応する. この **MLP** の隠れ層のノード数は, 出力の数 n と一緒とする. そうすることにより, 隠れ層の重み行列が $(nk \times n)$ 次元となり, 出力層の重み行列が $(n \times n)$ 次元となる. これらの重み行列およびバイアスベクトルを組み合わせ, 図 1 のように合わせる. そして列を中心に展開をすれば, $nk \times n + n \times n + 2n$ 次元のベクトルが得られる. このベクトルを重みベクトルと呼び, この重みベクトルが, 重み予測モデルの予測対象とする.

任意のシステム S から得られる全ての解釈遷移 (I, J) について, I を重み予測モデルに入力したとき, ある重みベクトル $\mathbf{w}$ が得られる. ここで, $I \subseteq \mathcal{B}_k$ であることに注意したい. この重みベクトル $\mathbf{w}$ によって初期化された **MLP** が, システム S のモデルとしてみる. つまり, このパーセプトロンに I を入力したとき, 出力が J となる. 本研究では, 重み予測モデルを **LSTM** で実装した. 入力 **LSTM** の時間ステップに従い, 各ステップではその時間ステップでの解釈を入力とする. **LSTM** モデルの出力は最後の出力が遷移先の J とする. 重みベクトルは **LSTM** の隠れ状態から得られる. **LSTM** の隠れ状態はノード数の 2 倍の次元となっている. よって, 重みベクトルを得るためには **LSTM** モデルの隠れ状態を入力とし, その学習したい **MLP** の重み出力であるようなデコーダー **MLP** を通す. これらの構造を図 2 に示す.

4. 実験

本研究ではいくつかの実験を行った. 実験では提案手法の有効性を確かめるために行われた. 実験で用いられたのは Tensorflow r1.6[1] のフレームワーク. ハイパーパラメータや学習に関する詳細は下記の通りである.

- **LSTM** の活性化関数は $\tanh$ を用いた;
- **LSTM** は 2 層重ね, 各層には 128 ノード;
- **LSTM** の学習には 0.2 のドロップアウトを用いた;
- デコーダー **MLP** は 3 層, 各隠れ層は 64 ノード;
- バッチサイズは 50;
- 学習したい対象の **MLP** の活性化関数は隠れ層では Rectified Linear Unit (**ReLU**)[10] を用い, 出力層ではシグモイド関数を用いた;

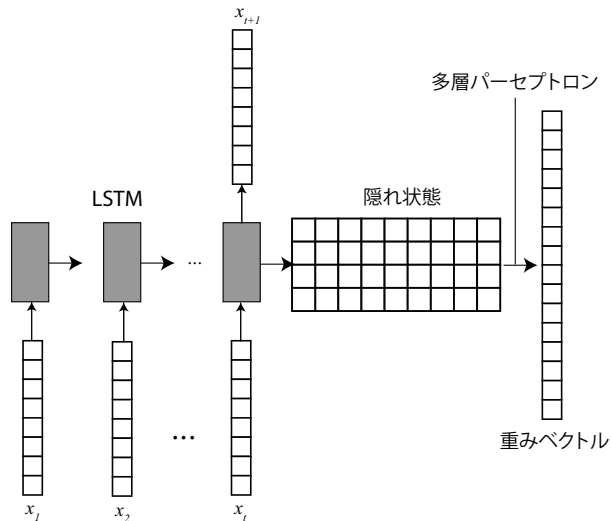


図 2: 重み予測モデル

- 最適化アルゴリズムは Adam 最適化 [8] で、パラメータは Tensorflow のデフォルトの値を用いた。

重み予測モデルの学習データはすべてランダムで生成されたものである。各実験では、それぞれ 50 個の **NLP** をランダムに生成し、45 個を学習に使い、5 個がテストに用いられた。各 **NLP** からは 10,000 個のデータを生成した。ここでデータ 1 個が解釈遷移 (I, J) であり、実験では $k = 10$ の遅延を伴うシステムを想定し、 n が変数の数ならデータ 1 個が $n \times k + n$ 個の値である。このシステムをモデル化する **MLP** の入力と出力のペアであることも注意したい。**NLP** を生成する際、その得られる遷移が十分分散されていない場合は新しい **NLP** を生成した。**NLP** から得られた遷移がほぼ常に真またはほぼ常に偽である場合は、何も学習ができないため学習データとして使用しない。重み予測モデルの学習データを生成するために、生成した 10,000 個のデータを 8,000 個学習データに、そして 2,000 個テストデータに **MLP** の学習を各 **NLP** ごとに行った。したがって、重み予測モデルの学習データは 10,000 個の遷移データおよび対応する **NLP** をモデル化した **MLP** の重みのペアである。

まず、変数の数が 10, 50, 100 および 200 に対して、重み予測モデルを用いて初期化した **MLP** とランダムに初期化された **MLP** の予測精度を比べた。各モデルの予測精度は、モデルの出力および真の状態遷移との 2 乗平均誤差で評価した。それぞれの結果を図 3 に示している。重み予測モデルを用いたモデルとランダム初期化モデルにそれぞれ 10 個, 20 個, ..., 100 個の学習データから学習を 100 エポック行わせ、10,000 個から学習用に使われたものを除き、残りからなるテストデータに対して予測精度を測った。すべての実験において、重み予測モデルを用いて初期化された **MLP** の予測精度がランダム初期化された **MLP** の予測精度を上回った。特に、重み予測モデルを用いて初期化したモデルは、学習データ 10 個与えられただけでも、誤差が 10% 以下となっている。変数が増えるほど差が広がり、効果がより発揮されている。

5. 考察

学習データが増えるのに従い、ランダムで初期化された **MLP** の予測精度が安定しない一方、重み予測モデルの予測精度は

安定している。それ以外に、システムの変数の数が増えると共に、ランダムで初期化された **MLP** は重み予測モデルで初期化されたものよりも悪くなっている。

ランダムに初期化された **MLP** を学習させるとき、予測精度がいい場合もあるが、非常に悪くなる場合もある。そして学習途中で、局所解に陥り予測精度が変わらなくなる場合もある。一方、重み予測モデルを用いた場合、学習させた後の予測精度は毎回同じである。

200 変数以上のシステムでは、ランダムに生成する学習データが大量のメモリを必要とする。そのため、学習データを生成しながら、重み予測モデルの学習を行えるような構造が必要。今回の実験ではそれを実装していないため、今後の展開の一つとする。

6. 関連研究

ニューラルネットワークの重みを予測する研究は、著者の知る限り、あまり行われていない。Denil ら [3] は行列の分解に注目し、学習に必要なパラメータを大幅に削減して予測を行なっている。対象のシステムから特徴量を考慮し、重み行列を分解する。本研究と違うのは、[3] 深層学習を対象とし、学習したいニューラルネットワークは隠れ層が複数ある場合のみである。それと違って本研究では隠れ層が 1 つの **MLP** のみを対象としている。

限られたデータから学習するものは [9] で研究されている。その研究では深層学習に注目し、特にデータが高度な非線形特徴を持つときに対応している。それと異なり、本研究ではダイナミックなシステムから得られる観測データを主に対象としている。

遅延の伴うダイナミクス学習では [11] がある。様々なダイナミクスへ対応できるモデルを構成し、ダイナミクスの規則の表現の学習を行なっている。一方、本研究では、**MLP** を用いてダイナミクス学習を行なっている。

7. おわりに

本研究では非常に限られたデータからそのシステムをモデル化する方法を提案した。**MLP** の初期重み行列を予測することにより、ランダム初期化よりも予測精度が上がったことを示した。特に、ダイナミックシステム中の変数が多い中でもその有効性を示した。変数が多いが得られるデータが少ない環境、例えば遺伝子制御ネットワークなどでの応用が可能である。今後の展開として、データに雑音や曖昧性がある場合などについて調べ、実データへの適用についても調べる。学習できた **MLP** を用いて論理プログラムの抽出についても調べている。そして、変数が増えた時のデータ量への対応についても調べる。

参考文献

- [1] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [3] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, et al. Predicting parameters in deep learning. In *Advances*

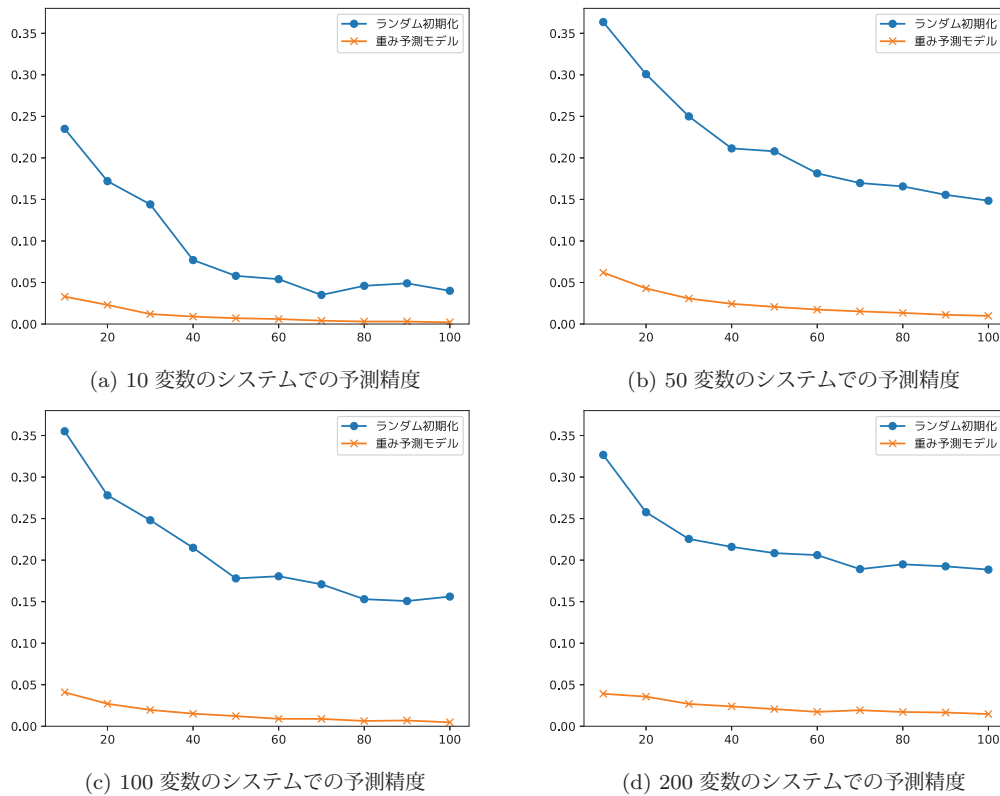


図 3: 実験結果, 横軸は学習データの数を表し, 縦軸は 2 乗平均誤差で評価される予測精度

in neural information processing systems, pages 2148–2156, 2013.

- [4] A. Garcez, T. R. Besold, L. De Raedt, P. Földiák, P. Hitzler, T. Icard, K.-U. Kühnberger, L. C. Lamb, R. Miikkulainen, and D. L. Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches, Stanford*, 2015.
- [5] E. Gentet, S. Tourret, and K. Inoue. Learning from interpretation transition using feed-forward neural network. In *Proceedings of ILP 2016, CEUR Proc. 1865*, pages 27–33, 2016.
- [6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [7] K. Inoue, T. Ribeiro, and C. Sakama. Learning from interpretation transition. *Machine Learning*, 94(1):51–79, 2014.
- [8] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [9] B. Liu, Y. Wei, Y. Zhang, and Q. Yang. Deep neural networks for high dimension, low sample size data. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2287–2293, 2017.
- [10] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [11] Y. J. Phua, T. Ribeiro, S. Tourret, and K. Inoue. Learning logic program representation for delayed systems with limited training data. In *Late Breaking Papers of ILP 2017*, 2017.
- [12] T. Ribeiro and K. Inoue. Learning prime implicant conditions from interpretation transition. In *ILP 2015*, pages 108–125. Springer, 2015.
- [13] T. Ribeiro, K. Inoue, and C. Sakama. A BDD-based algorithm for learning from interpretation transition. In *Proc. ILP 2013, LNAI 8812*, pages 47–63. Springer, 2014.
- [14] G. Thimm and E. Fiesler. Neural network initialization. In J. Mira and F. Sandoval, editors, *From Natural to Artificial Neural Computation*, pages 535–542, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
- [15] J. Yam and T. Chow. A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing*, 30(1-4):219–232, 1 2000.