基于可微神经计算的算法学习的研究

王李荣　　高志强

(东南大学计算机科学与工程学院, 南京, 210096)

摘　要　2010年, Richard A. Brualdi, Zejun Huang 以及Xingzhi Zhan提出了刻画极大非奇异partial矩阵的问题. 本文利用加边矩阵的思想证明了, 任一阶极大非奇异partial矩阵的容许的自由元个数最小为0, 且$n \geqslant 1$时, 存在自由元个数为1的n阶极大非奇异partial矩阵; 并猜想, 对于n阶极大非奇异partial矩阵, 其可能的自由元个数为$0, 1, 2, \cdots, \frac{n(n-1)}{2}$. 本文还给出了partial矩阵的$N_0-$填充的几个反例. 我们研究

关键词　partial矩阵; 填充; 加边矩阵; 非奇异; partial N_0-矩阵

The indeterminates number of a maximal-nonsingular partial matrix and some counter examples of partial N_0 matrix

Jie Ren　　Jianlong Chen

(Department of Mathematics, Southeast University, Nanjing 210096)

Abstract:　In 2010, Richard A. Brualdi，Zejun Huang and Xingzhi Zhan released a question about how to characterize a maximal-nonsingular partial matrix. It has been proved, useing idea of borderd matrix, there exists maiximal-nonsingular partial matrix of any order, with the number of the indeterminates being zero. When $n \geqslant 1$, there also exists maximal-nonsingular partialmatrix that has only one indeterminate. Then we guess that: for a n order maximal-nonsingular partial matrix, the probable numbers of its indeterminates are $0, 1, 2, \cdots, \frac{n(n-1)}{2}$.Morever, some counter examples about the completion of partial N_0-matrix are given.

Key words:　Partial matrix；Completion；Bordered matrix；Nonsingular; N_0-partial matrix

§1　引　言

执行一个程序，系统需要理解数值计算，if表达式，赋值，以及一系列操作的组合等许多基本操作。本文中将这些基本操作称为算法，例如加法，乘法，复制序列都被称为算法[1]。而通过输入和输出样例发现算法的任务，称为算法学习（Learning algorithms）。

实际上，算法学习不是一个新任务，它也被称为程序推理（Program induction），最早可以追溯到1967年[2, 3]，是一个非常古老但却十分有意义的研究领域。这项任务有着丰富的应用场景，例如：帮助没有计算机背景的人编写程序，帮助程序员发现问题，发现新算法，帮助教学。因而在过去几十年，有许多研究者投入到该任务的研究中[4, 5, 6, 7]，并做出了卓越的贡献。

在近些年，有研究者利用机器学习模型，尤其是循环神经网络（Recurrent Neural Network, RNN），研究该问题[8, 9, 10, 11, 12]。这些工作通过都是实验展示了利用机器学习模型可以学习到一些诸如复制，排序等简单的算法。

本文研究中使用的模型是可微神经计算机（Differentiable Neural Computer, DNC）[13]。DNC是一个变种的循环神经网络模型，类似于长短时记忆网路（Long Short Term Memory, LSTM）[14]。LSTM在很多存在长距离依赖的任务中表现出优秀的性能，例如语音识别[15]，自然语言处理[16] 等。但是在LSTM中，增加记忆单元的个数会导致模型参数个数的增加，使得模型难以训练。而DNC为了解决该问题，将记忆单元从网络中分离出来，将网络分割为可训练的控制器模块和外部记忆模块两个部分，外部记忆模块的大小将不再依赖于网络参数的个数。

本文尝试理解DNC的记忆能力，并将DNC应用于算法学习这个任务。通过实验我们发现：第一，使用课程学习，使DNC从不可收敛变成可收敛。第二，DNC在复制、加法、乘法等算法学习任务可以收敛。第三，增大外部记忆模块的大小，会加快模型的收敛速度，减少收敛时的震荡。第四，模型没有表现出泛化能力，即使用远长于训练时序列长度的样例进行测试时，模型无法正确预测输出。

§2 相关工作

算法学习这个任务最早可以追溯到1967年[2, 3]。在漫长的研究过程中，随着模型、方法不同，该任务有许多不同的名称，例如:合成程序（program synthesis）、程序归纳（program induction）、自动编程（automatic programming）等。但是其研究目标始终是为了让计算机有理解程序甚至自动编写程序的能力。[17] 和[18] 两篇论文从多个角度描述了算法学习的研究现状。

近年来，有一些研究者将机器学习方法应用于算法学习任务，并在该任务上取得了一些成果。特别是[13]提出的NTM模型。这是第一次明确提出外部记忆增强的神经网络模型。根据这篇文章的实验结果，NTM能学习到复制，重复排序等简单算法。

为了解决NTM不能并行和训练难度大等问题，[10]借鉴了Grid LSTM[19]的思想，提出了一种高度并行的神经网络结构-神经网络GPU（Neural GPUs，N-GPUs）。该文章将N-GPUs应用于算法学习任务，实验结果表明该模型能学到二进制加法和二进制乘法。

[9]结合增强学习算法和NTM，得到了增强学习神经网络图灵机（Reinforcement Learning neural turing machine, RL-NTM）。具体来说，该模型用离散外部记忆替代NTM的可微外部记忆，在使用增强学习训练该模型后，该模型能学到复制，重复复制等简单算法。

另一个对NTM做出改进的模型是[13]提出的DNC。文中讨论了NTM 的三个缺陷，并一一提出了改进方案。最后把DNC应用于一些有趣的实验中，例如在伦敦地铁中，寻找两站之间的最佳路线；学习拼图游戏等，DNC均达到98.8%以上的准确率。

记忆神经网络（Memory neural network, MNN）[20]是稍晚于NTM提出的外部记忆增强的神经网络模型。文中将MNN应用于问答（question answering, QA）任务，实验效果要好于LSTM。与NTM不同的是，MNN的记忆单元是不可写的。

另一个记忆单元不可写的模型是指针网络（Pointer Network, Ptr-Nets）[21]，文中用Ptr-Nets解决了凸包问题和旅行商问题。

除了这些模型外，还有一些研究者从外部记忆的结构，取址的效率等方面入手提出了新的模型。例如：栈增强的循环神经网络（Stack-Argmented Recurrent Neural Network）[22]，使用了可微的栈结构作为外部记忆。另一个类似的结构来自[23]，文中研究了三种不同结构的外部记忆，包括栈，双端队列和先进先出队列。值得注意的是，这些结构的取址效率都是常数时间。

[12]提出层次记忆网络（Hierarchial Attentive Memory, HAM）。HAM用二叉树作为外部记忆，叶子节点作为记忆单元。相比于使用标准的注意力机制访存的NTM，如果外部记忆的大小是n的话，NTM寻址的复杂度是$\Theta(n)$，而HAM的寻址的复杂度是$\Theta(logn)$。

§3 模型

这个部分将简要的描述DNC[13]。该模型的结构如图1所示。

§4 任务

本文考虑了四种不同的任务，分别是：复制，加法，三个数的加法和乘法。具体的例子可以见图2。需要强调的是，1）对于冯诺依曼体系结构下的计算机，程序员通过编码操作来CPU和寄存器可以很轻松的完成这些任务。但是对于该任务来说，不需要针对问题编码，而是使用大量数据训练DNC，使DNC学会这些算法。2）训练模型时采用的是序列到序列（sequence-to-sequence）的框架[24]，即模型每个时间步读入一个字符，当读到终止符后，模型便不在接收输入，而是转为每个时间步输出一个字符。在全部实验中，都以“.”作为终止符。

复制 这个任务的目的是复制输入字符。输入序列是任意长度的字符串，并以终止符结尾。这个任务可以测试DNC是否有能力存取任意长度的信息。

加法 这个任务的目的是对输入序列求和。这个任务要求模型：1）有能力存取任意长的信息。2）发现加法中进位的概念。3）理解一位数加法的概念。本文考虑了以下两种表示方式：

(1) 1 2 3 4 5 + 5 4

(2) 1 2 3 4 5 + 0 0 0 5 4

在实际的实验中发现，第一种表示方式处理比较困难，尤其是增加了课程学习的设计难度，而且第二种情况是包含第一种情况的。因此本文的实验采用的是第二种表示方式，即相加的两个数都有相同的长度。如果两个数的位数不同，则位数少的补零至两个数位数相同。之后的两个任务也是采用这种方式。

三个数加法 与上一个任务的不同点在于有三个数参与运算。相比于两个数的加法，难度在于进位的状态有三个（0,1,2）。

一位数乘法 这个任务是用一个一位数乘以一个多位数。这个任务的复杂度与加法类似，不过进位的状态可以是0到8里的任意一个。

§5 实验

§5.1 网络参数

DNC是可微的神经网络模型，在训练模型的过程中，本文首先用一小部分数据集对比了随机梯度下降算法以及其他两种改进的随机梯度下降算法RMSprop和Adam，并针对学习率、Mini-batch大小、权重初始化策略等超参数进行调优。最终在训练时采用RMSP算法和表1中展示的各项超参数。在各项超参数中需要说明的是DNC的控制器使用的是只有一层隐藏层的LSTM，隐藏层的记忆单元的个数是128。

§5.2 课程学习

课程学习是在训练模型的过程中一项重要的技术，本文使用的课程学习策略类似于[1]中的策略。具体来说，本文实验中使用的数据集将根据输入序列的长度

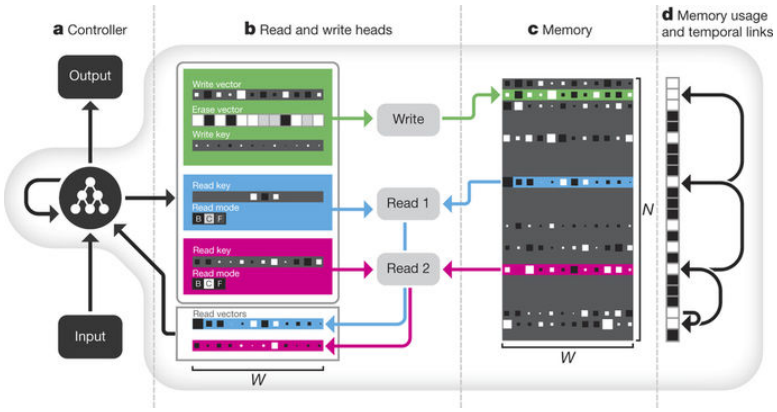

图 1 DNC模型，图片来自[13]

图 2 四个任务的例子，从左到右分别是复制，加法，三个数的加法和乘法。

表 1 网络超参数

超参数	优化策略
学习率	0.0001
momentum	0
Mini-batch大小	32
训练次数	100000
LSTM隐藏层单元的个数	128
权重初始化策略	$Uniform(-r, r)$

划分为若干组，输入序列的长度越大，则任务的难度越大。在训练时，根据数据集的难度从易到难依次用于训练。在下面的描述中，MIN表示训练时最小的序列长度，MAX表示训练时最大的序列长度。

不用课程学习 该策略作为一个比较基准，不使用课程学习策略。即实验中每一个batch的数据都从随机的从MIN和MAX生存。

单纯的课程学习 开始时，随机生成长度属于$[1, MIN]$的序列作为训练数据集，当准确率到达给定的阈值之后（本文的实验中阈值是0.9），区间的右端点值加一，即随机生成长度属于$[1, MIN+1]$的序列作为训练数据集。重复该过程，直到右端点的值等于MAX。在实验中，使用该策略的效果要好于不用课程学习，但是并不理想。因此，下一个策略是在该策略的基础上进行改进。

改进的课程学习 该策略的过程与单纯的课程学习的策略类似，唯一的不同点在于：当区间是$[1, a]$时，不再随机从整个区间生成数据，而是按照一定的比例从$[1, a-1]$和$[a, a]$生成数据，并且大部分数据来自$[a, a]$。这个策略类似于在人类学习新知识的时候，应该可以新知识为主，并需要复制旧的知识。在实验中，该策略的表现要优于其他两个策略。

本文对三种策略进行了对比实验。

§5.3 实验结果

这部分将介绍实验结果。在所有的实验中，用的模型是DNC，控制器的模型是LSTM，各项超参数如表1所示。所有的实验都对比了三种课程学习策略的影响。

在实验结果方面，本文比较了实验结果的准确率。需要注意的是，对于一个样例，模型会有一个预测序列，只有整个序列都正确，才认为这个预测是正确的。只要有一个时间步的预测是不正确的，那么这个预测就是错误的。

§6 结论

文献[?, ?]中提到: 结论

参考文献:

[1] Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

[2] E Mark Gold. Language identification in the limit. *Information and control*, 10(5):447–474, 1967.

[3] Dana Angluin and Carl H Smith. Inductive inference: Theory and methods. *ACM Computing Surveys (CSUR)*, 15(3):237–269, 1983.

[4] Peter Nordin. *Evolutionary program induction of binary machine code and its applications*. Krehl Munster, 1997.

[5] Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.

[6] Mark Wineberg and Franz Oppacher. A representation scheme to perform program induction in a canonical genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 291–301. Springer, 1994.

[7] Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22, 1964.

[8] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[9] Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521*, 419, 2015.

[10] Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

[11] Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural random-access machines. *arXiv preprint arXiv:1511.06392*, 2015.

[12] Marcin Andrychowicz and Karol Kurach. Learning efficient algorithms with hierarchical attentive memory. *arXiv preprint arXiv:1602.03218*, 2016.

[13] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[17] Sumit Gulwani. Dimensions in program synthesis. In *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming*, pages 13–24. ACM, 2010.

[18] Emanuel Kitzelmann. Inductive programming: A survey of program synthesis techniques. In *AAIP*, pages 50–73. Springer, 2009.

[19] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[20] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[21] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[22] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems*, pages 190–198, 2015.

[23] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015.

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.