本科毕业论文

论文题目：基于迁移强化学习的量子多体系统求解

姓　　名：陈骜　　学　　号：15307110151

院　　系：物理学系

专　　业：物理学

指导教师：戚扬　　职　　称：研究员

单　　位：复旦大学物理学系

完成日期：20 19 年 5 月 17

目 录

摘　要

近年来，人工神经网络已成为数值求解量子多体问题的重要工具，但该方法求解具有复杂符号结构的波函数的能力仍待进一步的发展。本文中，我们在前人的基础上提出了一个新的网络结构，该网络具有通过监督学习、强化学习和迁移学习等诸多方式构造与求解波函数的能力。为了求解符号结构较为复杂的波函数，我们先在小体系中用已知波函数监督学习得到初步参数，然后以该参数作为初始参数在需要求解的大体系中进行强化学习训练。最终求解得到的复杂体系波函数具有目前该类体系的神经网络结果中最高的精度，表明这种迁移强化学习的思路有希望在未来成为神经网络求解量子多体系统的主流方法。

关键词：量子多体系统；人工神经网络；迁移强化学习

Abstract

Recently, artificial neural networks have been an important tool for numerically solving quantum many-body problems, but its ability in dealing with wave functions with complex sign structures needs further development. Based on previous researches, we propose a new network structure that is able to construct and solve wave functions using supervised learning, reinforcement learning as well as transfer learning. For solving wave functions with complex sign structures, we firstly perform supervised learning in a known small system for primary parameters, and then the parameters are transferred to the target system for reinforcement learning. The results of the complex sign structure wave function show the best accuracy among neural network researches until now, suggesting that the proposed transfer reinforcement learning scheme will hopefully become a main method for solving complex quantum many-body systems through neural networks in the future.

Key words: quantum many-body system; artificial neural network; transfer reinforcement learning

第 1 章 引言

量子多体系统中，希尔伯特空间的维度随着系统尺度的增大而指数增大，这使得数值求解所需要的参数数量以及时间也指数增大。但是，这些系统的基态中往往存在一些可简化的结构，使得我们能够用较少的参数进行表征，同时也减少求解所需的时间。对于量子多体物理的研究，一个核心的挑战就在于如何寻找一个有效的算法，使得我们可以数值求解较大的强关联体系。

现在已有许多数值手段可以有效地求解一些量子基态问题，例如基于随机抽样的量子蒙特卡洛（QMC）以及基于量子态压缩的矩阵乘积态（MPS）[1]等。然而，这些数值方法在求解一些更为复杂的量子多体问题中的表现仍待改进。这类量子体系一般具有复杂的符号结构，使得利用 QMC 进行的数值模拟遭受符号问题[2]的严重困扰。而包括 MPS 在内的基于量子态压缩的数值方法，则在高维的复杂体系中表现不佳。

这些量子体系中，一个典型的案例即为反铁磁自旋 1/2 的二维J_1-J_2模型。这是一个目前尚未有严格解的自旋阻挫系统。尽管已有许多的相关研究，目前对于该模型在特定参数附近是否会出现自旋液体相（spin liquid phase）仍无定论 [3,4]。

作为机器学习领域的一个重要分支，使用神经网络的深度学习已在图像识别、声音识别[5]与围棋[6]等诸多领域展现出了极富创造性的应用。作为一个拟合未知的复杂函数结构的有力工具，深度学习近年来也在计算量子物理中崭露头角。在该领域的探索伊始，研究发现限制玻尔兹曼机能够精确并快速地求解一些较简单的自旋模型，其表现追平甚至超越了以往的其他数值方法[7]。该研究基于以神经网络为拟设的变分蒙特卡洛（VMC），并在此基础上进行强化学习训练。受此启发，一些跟进的研究探索了许多其他的网络结构[8-11]，这些结构也各自具有独特的优势。现在，一个重要的问题摆在我们面前：神经网络是否有能力对J_1-J_2模型这样目前研究中尚无确切定论的复杂量子体系给出超越其他数值方法的结果？

事实上，这些量子体系的复杂符号结构对于擅长拟合平滑函数的神经网络方法也提出了巨大的挑战[12]。例如，对于前述的J_1-J_2模型，目前已有许多基于神经网络的求解尝试[8,10]。这些研究仍是基于强化学习的能量变分，而且给出了一些接近其他数值方法的结果，但并没有得到完全超越以往的数值方法的新结果。

在本文中，我们首先展示了一个在量子多体问题中具有监督学习、强化学习与迁移学习能力的网络结构，然后借鉴机器学习领域内迁移强化学习[13]的思路提出了一个基于该网络结构求解复杂量子多体问题的新方案。在该方案中，我们

先在较小的体系内用数值对角化方式求出波函数严格解并对神经网络进行监督学习的训练，然后用训练得到的参数作为较大系统中的初始参数进行强化学习。通过数值测试，我们发现这种基于迁移强化学习的方案可以有效地提升神经网络在求解一维J_1-J_2模型时的表现，因此有望在未来成为求解该类问题的有力工具。

第 2 章 神经网络表示波函数

2.1 波函数的参数表示

求解一个量子系统的基态波函数是量子力学的核心问题之一。对于希尔伯特空间中的一套基矢$\{|n\rangle\}$，量子态可以写为$|\Psi\rangle=\sum_n \psi(n)|n\rangle$。波函数$\psi(n)$可视为一个函数，其输入值为系统的一个基矢$|n\rangle$，输出值为量子态在该基矢下的分量$\psi(n)=\langle n|\Psi\rangle$。在实践中，我们令波函数$\psi(n)$的函数形式由一组参数$\{W_k\}$控制。只要适当地优化$\{W_k\}$的值，就可以对波函数$\psi(n)$进行调整，从而拟合基态波函数$\psi_{\mathrm{GS}}(n)$，以得到系统的近似解。

例如，考虑包含 3 个格点的一维横场伊辛模型

$$H=-J\sum_{\langle ij\rangle}\sigma_i^z\sigma_j^z-h\sum_{i=1}^{3}\sigma_i^x \tag{2.1}$$

其中$\langle ij\rangle$代表最近邻格点。选取S_z基矢，定义自旋方向朝上为 1，朝下为-1（例如基矢$|\uparrow\uparrow\downarrow\rangle$对应的构型即为$s_1=1,s_2=1,s_3=-1$）。我们可以假设如下的波函数形式

$$\psi(s_1,s_2,s_3)=W_1(s_1+s_2+s_3)+W_2s_1s_2s_3 \tag{2.2}$$

其对应的量子态即为

$$\begin{aligned}|\Psi\rangle=&(+3W_1+W_2)|\uparrow\uparrow\uparrow\rangle+(+W_1-W_2)(|\uparrow\uparrow\downarrow\rangle+|\uparrow\downarrow\uparrow\rangle+|\downarrow\uparrow\uparrow\rangle)\\&+(-3W_1-W_2)|\downarrow\downarrow\downarrow\rangle+(-W_1+W_2)(|\uparrow\downarrow\downarrow\rangle+|\downarrow\uparrow\downarrow\rangle+|\downarrow\downarrow\uparrow\rangle)\end{aligned} \tag{2.3}$$

其中$\{W_1,W_2\}$为波函数的参数，由此即可通过改变参数拟合基态波函数。

这样假设的量子态一般是非归一的，归一的量子态为

$$|\tilde{\Psi}\rangle=\frac{|\Psi\rangle}{\sqrt{\langle\Psi|\Psi\rangle}} \tag{2.4}$$

一般使用非归一量子态$|\Psi\rangle$即可，可参见附件 B、C、D 中的详细数值方法。

2.2 神经网络量子态

如前文所述，求解量子多体系统基态的一种可行手段是用一组参数$\{W_k\}$对复杂的波函数ψ进行拟合。只要参数的数量足够多，就可以很好地逼近系统的真实基态。这种用有限参数拟合一个复杂函数的方案，正是神经网络的优势所在。神

经网络能够对任意的一个输入构型$|\mathbf{s}=(s_1,s_2,\ldots,s_N)\rangle$输出对应的分量$\psi(\mathbf{s})$，恰好能够对量子态进行有效的表示。这种表示称为神经网络量子态（neural-network quantum states）[7]。

在本文中，我们采用深度学习领域中极为普遍的前馈卷积神经网络的结构，其具体数学表示详见附录 A。网络的构造如图 2.1 所示。

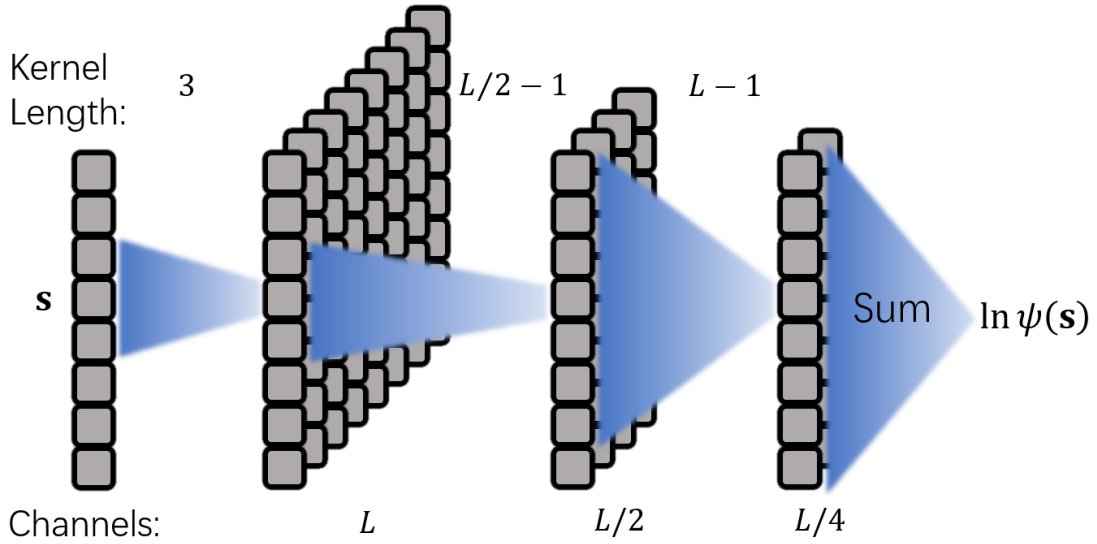

图 2.1 网络结构

图中每个格点代表一个神经网络单元，每一列代表卷积网络一层中的一个通道，蓝色箭头代表卷积核，每层中仅展示一个；输入的构型$\mathbf{s}$长度为L，其后有三层卷积层与一层求和层；第 1 层无偏置，激活函数为$\ln\cosh(z)$；第 2、3 层有偏置，激活函数为$\mathrm{ReLU}(z)$；第 4 层为无参数的直接求和；卷积层采用周期性边界条件填充，卷积核长度分别为 3、$L/2-1$、$L-1$，通道数分别为L、$L/2$、$L/4$；以图中$L=8$的情况为例，卷积核长度为 3、3、7，通道数为 8、4、2，参数总数为 182

网络结构的配置及相关原因如下：

1. 量子力学中的波函数为复数，因此网络中的所有参数均使用复数，最常用的 ReLU 激活函数也改写为复数域上拓展的形式[14]

$$\mathrm{ReLU}(z)=\begin{cases} z & -\frac{\pi}{4}\le \arg z<\frac{3\pi}{4} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

   该激活函数在复杂符号结构的量子多体问题中有较好表现[10]。

2. 一般常用的卷积网络为小卷积核的深层网络。但是根据量子多体系统中强化学习[7]与监督学习[15]的实践，浅层网络的效果较好。因此本文采用浅层网络，并对应地增长卷积核。但是测试发现扩大网络第一层卷积核的长度会显著降低迁移学习的精度，因此第一层仍使用小卷积核。这一改动会对强化学习的精度造成负面影响，但是通过第 4 章的迁移强化学习可以基本消除这一影响。

3. 首层使用$\ln\cosh(z)$激活函数，这种形式的激活函数能有效地提取波函数

的符号特征[10,15]。

4. 量子多体系统往往具有平移对称性与$\mathbb{Z}_2$对称性。在本文研究的系统中，经过平移与翻转变换波函数保持不变（更一般的情况下变换后波函数会附加一个相位，仍有相应的处理办法[16]）。在图 2.1 的网络中，卷积结构与周期性边界条件填充[8]使得平移不变，第一层无偏置使得翻转不变。
5. 最后输出结果为$\ln\psi$而不是ψ，这在以往实践中有很好的效果[7]。

总而言之，图 2.1 中的网络使用了卷积神经网络结构，并根据量子多体问题的需求进行了一些专门改造，此外并没有为特定的量子体系进行结构上的设计，其求解量子多体系统的能力具有普适性。

2.3 波函数的符号结构

在一些较为一般的量子多体问题中，符号结构极为简单。例如横场伊辛模型，其哈密顿量为

$$H = -J\sum_{\langle ij\rangle}\sigma_i^z\sigma_j^z - h\sum_i \sigma_i^x \tag{2.6}$$

其中$\langle ij\rangle$代表最近邻，$J,h>0$。仍选取S_z基矢并记为$\{|n\rangle\}$。此时哈密顿量在该基矢下的每个分量都为负数或 0，即$\langle n|H|m\rangle \le 0$。因此所有的基态波函数分量$\psi(n)=\langle n|\Psi\rangle$可直接取正实数，不用考虑其符号。在使用神经网络处理这类问题的过程中可直接令神经网络的参数为实数。即使如本文中假设参数为复数，也能通过对网络进行训练很容易地求解这类系统。

与之相对，一些其他系统有着复杂的符号结构。例如J_1-J_2模型，其哈密顿量为

$$H = J_1\sum_{\langle ij\rangle}\boldsymbol{\sigma}_i\cdot\boldsymbol{\sigma}_j + J_2\sum_{\langle\langle ij\rangle\rangle}\boldsymbol{\sigma}_i\cdot\boldsymbol{\sigma}_j \tag{2.7}$$

其中$\langle\langle ij\rangle\rangle$代表次近邻，$J_1,J_2>0$。一般仍选取$S_z$基矢，但实际上无论如何选取基矢，都不能使基态波函数的所有分量$\psi(n)$都为正实数，只是仍可以使其都为有正有负的实数。

在利用神经网络求解这类问题的过程中，通常有两种做法。第一种仍采取实参数，最后求解得到的波函数为实数。第二种采用复参数，最后求解得到的神经网络量子态既是能量基态也是动量本征态。前一种方法求解得到的基态相当于多个动量的叠加，这会增大波函数结构的复杂性与拟合的难度，因此在本文中我们采取后一种做法。

第3章　神经网络构造与求解波函数

3.1 监督学习构造已知波函数

在利用神经网络求解未知波函数之前，我们可以先测试神经网络表达已知波函数的能力。一方面，这可以展示我们选取的神经网络结构是否有足够的表达能力；另一方面，这也是之后第 4 章中迁移强化学习所需的必要步骤。

利用包括直接对角化（exact diagonalization）在内的多种非神经网络的数值方法，我们可以直接求得较小量子体系的基态波函数理论值。在已知波函数之后，就可以用它训练神经网络，观察神经网络是否能够通过改变网络中的参数有效地表示目标波函数[12]。具体的算法见附录 C。这种有已知样本而尝试通过神经网络构造目标函数的做法称为监督学习，是深度学习领域内比较成熟的训练方法。

我们在一些常见的一维量子多体系统中对监督学习的效果进行了测试。这些模型包括之前所述的横场伊辛模型以及J_1-J_2模型。当$J_2=0$时，J_1-J_2模型即为反铁磁海森堡模型；当$J_2=0.5J_1$时，即为 Majumdar-Ghosh 模型[17]。

约定$|\Psi\rangle=\sum_n \psi_n|n\rangle$为神经网络量子态，$|\Phi\rangle=\sum_n \phi_n|n\rangle$为已知基态，则$1-\frac{\langle\Psi|\Phi\rangle\langle\Phi|\Psi\rangle}{\langle\Psi|\Psi\rangle\langle\Phi|\Phi\rangle}$可以表征神经网络量子态相对于基态的误差。图 3.1 展示了不同模型中这一误差随监督学习迭代次数变化的学习曲线，选取的系统大小都为$L=12$。

从图中可以看出由于横场伊辛模型波函数结构较为简单，其训练较快，精度较高。而海森堡模型与 Majumdar-Ghosh 模型基态波函数具有复杂的符号结构，因而训练较慢且容易停滞，对参数的选择也较敏感。但总体上该网络可以通过监督学习构造出较为准确的已知波函数。我们使用的网络结构为迁移学习的性能牺牲了一部分监督学习的性能，因此监督学习效果会略差于以往的研究结果[15]。

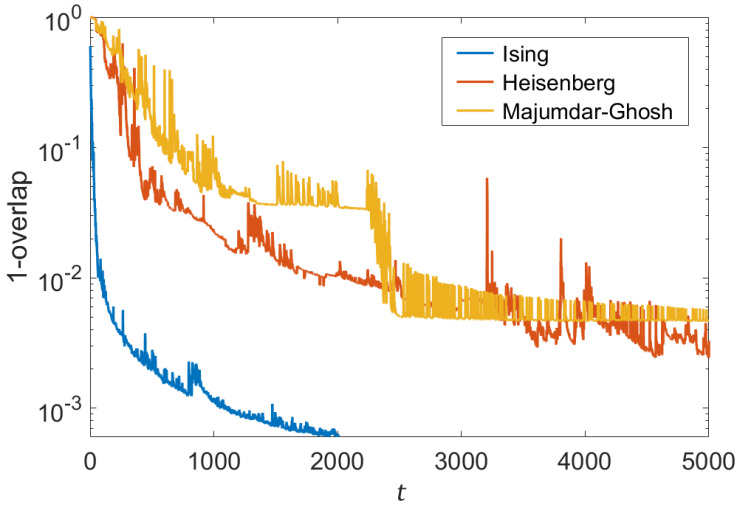

图 3.1 监督学习的学习曲线

图中横轴为监督学习的迭代次数，纵轴为量子态的误差，三条曲线分别对应于$J=h$的横场伊辛模型，反铁磁海森堡模型以及 Majumdar-Ghosh 模型，都是一维的$L=12$的系统

3.2 强化学习求解未知波函数

引入神经网络这一数值方法的核心目的是求解一个未知的波函数。我们可以将量子系统的能量作为参数调整的判据（即深度学习中的损失函数），通过数值变分方法对神经网络中的参数进行优化，使得波函数对应的能量不断降低，最终求得基态波函数的数值解。这种通过变分方法减小损失函数来尝试求解一个未知的目标函数的方法称为强化学习，是目前机器学习领域内的一个极富挑战性的方向。

实践中，我们采用的是变分蒙特卡洛[18]数值变分方法与随机重构（stochastic reconfiguration）[19]参数优化方法的结合，这也是神经网络求解量子多体系统这一领域内的通用做法。具体的算法详见附录 B、D。

图 3.2 是通过强化学习求解横场伊辛模型的学习曲线及最终能量误差。我们选取的系统大小为$L = 28$，在这一系统尺度下，直接对角化方法的计算量与内存消耗已经十分巨大，超过了普通计算机的承受能力。但通过神经网络的方法，我们顺利得到了较精确的变分波函数。即使得到的能量误差在相变点$h = J$最大，总体上仍较为准确。由于该模型的符号结构较为简单，强化学习的过程也比较容易。

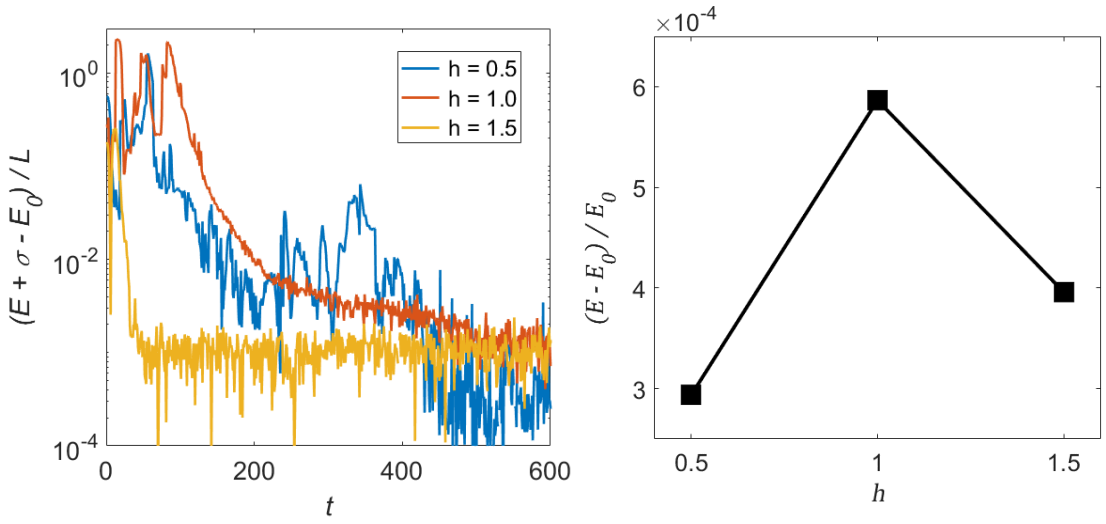

图 3.2 强化学习求解一维横场伊辛模型

左图为不同横场h下强化学习的学习曲线，横轴为参数优化的迭代次数，纵轴为变分能量的误差$(E+\sigma-E_0)/L$，其中E, E_0为变分能量与基态能量，σ为变分能量在变分蒙特卡洛抽样中的标准差（详见附录 B），由于包括了标准差该误差只是保守估计，会比实际结果偏大；右图为精确数值分析后得到的不同横场h下能量的相对误差$(E-E_0)/E_0$；两图中均取$J = 1$

图 3.3 是强化学习求解$J_1 - J_2$模型的学习曲线。由于该模型的复杂符号结构，训练过程较为困难，我们选取较小的$L = 12$系统。通过仔细的学习参数调整，仍可得到海森堡模型的较精确变分能量，最后的能量误差$(E-E_0)/E_0 = 3 \times 10^{-4}$。但对于更为复杂的 Majumdar-Ghosh 模型，纯粹的强化学习就显得无能为力了。

这种复杂模型下的困境正是目前神经网络方法乃至大多数量子多体数值方法的所面临的首要难题。第 4 章中提出的迁移强化学习，就致力于解决这一问题。

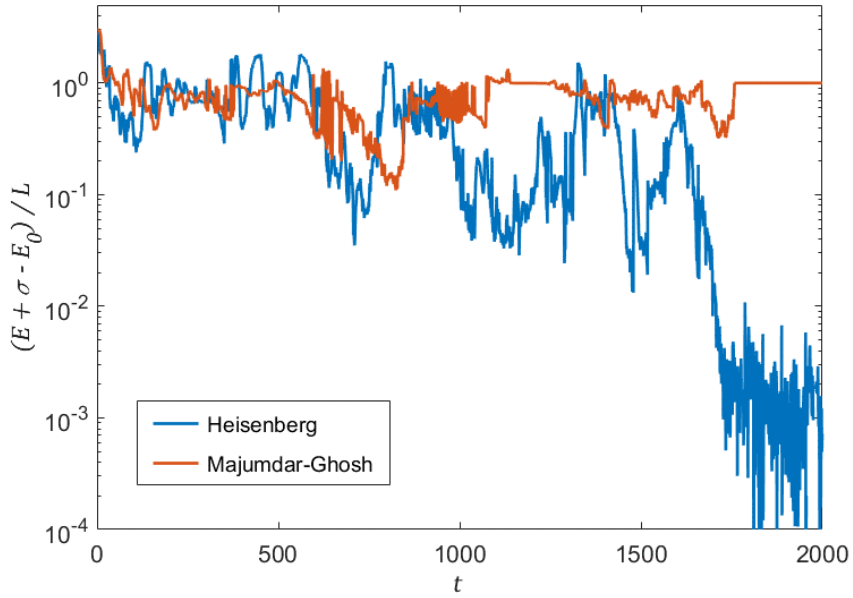

图 3.3 强化学习求解一维J_1-J_2模型

为了第 4 章中迁移学习的效果，我们牺牲了部分强化学习的性能，使得强化学习的变分能量结果会略差于以往的研究[7,15]。但通过之后迁移学习的辅助，我们能够达到极高的变分精度。

第4章　迁移学习求解复杂波函数

4.1 参数迁移

正如在 3.2 节中所展现的那样，一般的神经网络方法对于求解具有复杂符号结构的波函数仍然缺乏精度。而通过接下来 4.2 节中的结果我们可以看到，神经网络对于这类系统其实有足够的表达能力，只是由于强化学习难度过高而无法通过单纯的能量变分得出正确的波函数结构。而我们希望神经网络能够求解一些未知的复杂体系，自然不可能用已知的波函数进行监督学习训练。因此，无论是使用监督学习还是强化学习，都无法得到这类系统的基态波函数。

为了解决这一问题，我们引入了参数迁移的方案。我们采用的网络结构是前馈卷积神经网络，它的同一套参数可适用于不同大小的输入系统，如图 4.1 所示。因此，我们先在较小的系统中进行监督学习的训练得到一套初始参数，再从这套初始参数开始在较大的系统中进行强化学习。

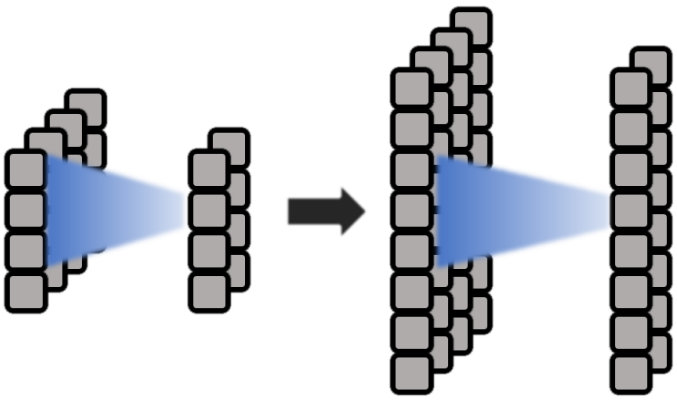

图 4.1 参数迁移

参数迁移前后卷积核尺寸以及通道数目保持不变，因此参数的数量与在网络中的结构也保持不变；图中展示的是$L = 4$到$L = 8$的参数迁移，只显示了一层卷积网络

4.2 复杂波函数的求解

基于迁移强化学习的思路，我们对 Majumdar-Ghosh 模型进行了求解的尝试。在求解过程中，我们先在$L = 20$的系统中监督学习，然后将参数迁移到$L = 24$与$L = 28$的系统中作为强化学习的初始参数。强化学习的学习曲线如图 4.2 所示。

由图可知，在不同大小的系统中，迁移学习都成功得到了极好的变分能量。更精确的数据分析得到$L = 24$与$L = 28$的系统中变分能量与基态能量的相对误差$(E - E_0)/E_0$分别为2×10^{-5}与4×10^{-5}，是目前神经网络在该类自旋阻挫模型中达到的最好结果。该结果表明这一方法能够精确地求解具有复杂符号结构的量子多体系统。

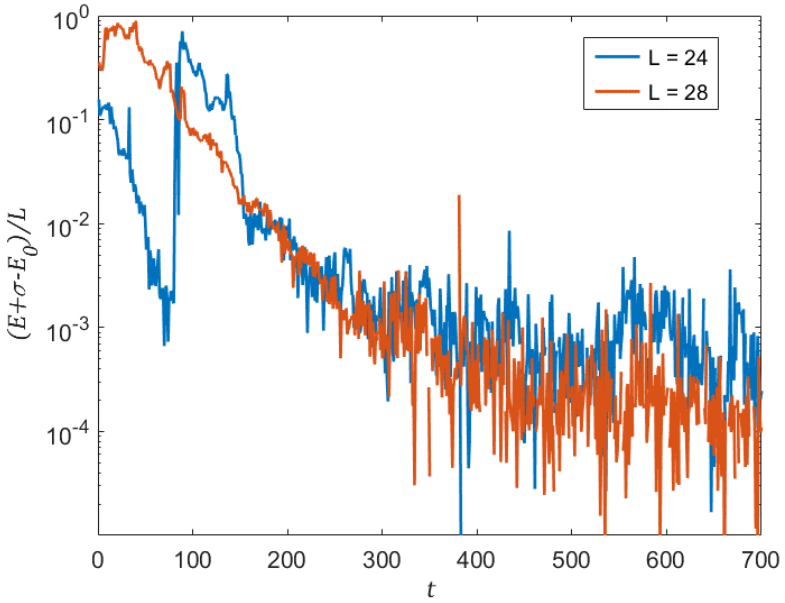

图 4.2 迁移强化学习求解 Majumdar-Ghosh 模型

图中展示了$L = 24$与$L = 28$时迁移强化学习的效果；表明上$L = 24$中的训练效果较$L = 28$时差，但这是由于前者学习参数设置不如后者恰当，而且前者的能量标准差σ较大；实际上仍是$L = 24$的情况由于接近监督学习体系大小 20 而展现出更精确的结果

此外，以往通过直接能量变分的强化学习有概率只得到能量与基态接近的亚稳态，而二者的波函数特征可能大有不同，这也是二维J_1-J_2模型的基态结构存在长期争论的重要原因。由于监督学习保证了变分开始时波函数的特征与小体系的基态波函数一致，这种迁移强化学习的做法能够有效地避免前述的困境，寻得更为可靠的变分结果。

第 5 章 结论

在本文中，我们提出了一个具有监督学习、强化学习与迁移学习能力的人工神经网络结构，并展示了迁移强化学习的框架在具有复杂符号结构的量子多体问题中的极高求解精度。当前结果还存在以下几个问题。

首先，网络的参数数量$N_W \propto L^3$，而以往的网络结构都是$N_W \propto L^2$。参数数量在量级上大于之前的网络，这会导致大系统中的计算量暴增。可以通过更加细致的网络结构设计将N_W降低到L^2的量级。

其次，目前的测试只涉及J_1-J_2模型中$J_2=0.5J_1$时的迁移学习，在其他参数区间以及其他模型中的应用潜力还需要进一步的数值实验验证。

另外，我们的数值计算只涉及参数在尺度相近的两个系统之间的迁移。不难想象，当尺度相差较大时，直接参数迁移的效果会变差。而且，若是如本文中使用小系统对应的较小网络进行监督学习，迁移之后参数数量较少，可能导致网络的表达能力不足。然而若是在大系统对应的大网络中用监督学习训练小系统，则可能导致过拟合（overfitting）。一个可行的解决方案是多步迁移逐渐增大系统尺度，在此过程中合理地调整网络结构并增加参数。这种方案的可行性还需要进一步的研究。

即便存在上述留待继续探索的问题，本文仍展示了一个求解复杂量子多体系统的有力工具，并且该方法也有望在未来成为神经网络求解复杂量子体系的主要手段。利用这一技术，我们可能让神经网络在将来的计算量子物理中发挥其他数值方法无法取代的作用，真正求解一些其他方法无法解决的重要问题。而人们对自旋阻挫系统这类复杂量子系统的理解，也可能通过这一计算手段的发展得到提升。

参考文献

[1] S. R. White. Density matrix formulation for quantum renormalization groups. Phys. Rev. Lett., 1992, 69: 2863–2866.

[2] M. Troyer and U.-J. Wiese. Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations. Phys. Rev. Lett., 2005, 94.

[3] L. Capriotti and S. Sorella. Spontaneous Plaquette Dimerization in the J1-J2 Heisenberg Model. Phys. Rev. Lett., 2000, 84: 3173.

[4] S. S. Gong, W. Zhu, D. N. Sheng, et al. Plaquette Ordered Phase and Quantum Phase Diagram in the Spin-1/2 J1-J2 Square Heisenberg Model. Phys. Rev. Lett. 113: 027201.

[5] Y. LeCun, Y. Bengio and G. Hinton. Deep learning. Nature, 2015, 521: 436–444.

[6] D. Silver, A. Huang, C. J. Maddison, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529: 484–489.

[7] G. Carleo and M. Troyer. Solving the Quantum Many-Body Problem with Artificial Neural Networks. Science, 2017, 355: 602-606.

[8] X. Liang, W. Liu, P. Lin, et al. Solving frustrated quantum many-particle medels with convolutional neural networks. Phys. Rev. B, 2018, 98: 104426.

[9] G. Carleo, Y. Nomura and M. Imada. Constructing exact representations of quantum many-body systems with deep neural networks. Nature Communications, 2018, 9: 5322.

[10]K. Choo, T. Neupert and G. Carleo. Study of the Two-Dimensional Frustrated J1-J2 Model with Neural Network Quantum States. arXiv, 2019, 1903.06713.

[11]O. Sharir, Y. Levine, N. Wies, et al. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. arXiv, 2019, 1902.04057.

[12]X. Gao and L. M. Duan. Efficient representation of quantum many-body states with deep neural networks. Nature Communications, 2017, 8: 662.

[13]J. Yosinski, J. Clune, Y. Bengio, et al. How transferable are features in deep neural networks? Advances in Neural Information Processing Systems, 2014, 27.

[14]C. Trabelsi, O. Bilaniuk, Y. Zhang, et al. Deep Complex Networks. arXiv, 2017, 1705.09792.

[15]Z. Cai and J. Liu. Approximating quantum many-body wave functions using artificial neural networks. Phys. Rev. B, 2018, 97: 035116.

[16]K. Choo, G. Carleo, N. Regnault, et al. Symmetries and Many-Body Excitations with Neural-Network Quantum States. Phys. Rev. Lett., 2018, 121: 167204.
[17]C. K. Majumdar and D. K. Ghosh. On Next-Nearest-Neighbor Interaction in Linear Chain. II. Journal of Mathematical Physics, 1969, 10: 1399.
[18]McMillan, W. L. Ground State of Liquid He4. Phys. Rev., 1965, 138: A442-A451.
[19]M. Casula, C. Attaccalite and S. Sorella. Correlated geminal wave function for molecules: an efficient resonating valence bond approach. J. Chem. Phys., 2004, 121:7110.
[20]G. Carleo, K. Choo, D. Hofmann, et al. Netket: A Machine Learning Toolkit for Many-Body Quantum Systems. arXiv, 2019, 1904.00031.
[21]K. Kreutz-Delgado. The Complex Gradient Operator and the CR-Calculus. arXiv, 2009, 0906.4835.
[22]S. C. T. Choi, C. C. Paige and M. A. Saunders. MINRES-QLP: A Krylov Subspace Method for Indefinite or Singular Symmetric Systems. SIAM J. Sci. Comput., 2011, 33(4): 1810-1836.
[23]S. Sorella, M. Casula and D. Rocca. Weak binding between two aromatic rings: feeling the van der Waals attraction by quantum Monte Carlo methods. J. Chem. Phys., 2007, 127: 014105.
[24]M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. arXiv, 2012, 1212.5701.
[25]S. J. Reddi, S. Kale and S. Kumar. On the Convergence of Adam and Beyond. ICLR 2018 Conference.
[26]K. He, X.-Y. Zhang, S.-Q. Ren, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv, 2015, 1502.01852.

致　谢

在研究过程中，戚扬老师为我确定了神经网络求解量子多体问题的研究方向，提供了迁移强化学习的研究思路以及进行了多次的具体理论指导。我在此对戚扬老师表示衷心的感谢。同样感谢我的女朋友高恬，她在繁忙的毕业季给予我巨大的帮助，使我有时间能够完成这篇高质量的论文。

本文中神经网络的计算基于 NetKet 程序包[20]。

附　录

A 前馈卷积神经网络

一个前馈网络由许多层组成，第 l 层第 i 个单元的值$v_i^{(l)}$是第$l-1$层部分单元的值的线性组合再作用一个非线性的激活函数g，即

$$v_i^{(l)} = g\left(\sum_j W_{ij}^{(l)} v_j^{(l-1)} + b_i^{(l)}\right) \tag{A.1}$$

其中$W_{ij}^{(l)}$与$b_i^{(l)}$为网络的参数，前者称为权重，后者称为偏置。每个$W_{ij}^{(l)}$都是$v_j^{(l-1)}$与$v_i^{(l)}$之间的连结，这使得后者是前者的函数，最终的输出值也是输入值经多层传播之后的函数。这样的构造可以使神经网络能够较好地拟合一个输入值到输出值之间的非线性函数。

前馈卷积网络在一般的前馈网络的基础上进行了优化，后一层每个单元的值只与前一层少数几个单元有关，而且大部分的连结参数$W_{ij}^{(l)}$的值是共享的。对于一个一维单通道卷积神经网络，其具体数学表示为

$$v_i^{(l)} = g\left(\sum_{j=i-r}^{i+r} W_{j-i}^{(l)} v_j^{(l)} + b^{(l)}\right) \tag{A.2}$$

$W_k^{(l)}(-r \le k \le r)$称为一组卷积核，$r$ 为卷积核半径。这样的结构可以有效地减少参数的个数，并且也能更好地反映体系的空间结构并抽取信息。

一般来说，前后两个卷积层都会含有多个通道（channel），此时表达式为

$$v_{n,i}^{(l)} = g\left(\sum_m \sum_{j=i-r}^{i+r} W_{n,m,j-i}^{(l)} v_{m,j}^{(l)} + b_n^{(l)}\right) \tag{A.3}$$

B 变分蒙特卡洛

变分蒙特卡洛方法是一种十分常用的数值变分方法，它通过蒙特卡洛重要性抽样的方式求得系统的各个参数，并通过变分法降低系统能量以求得量子体系的数值解。以下我们用能量的计算来举例。

对于一个给定的变分波函数$\psi_n = \psi(n)$，体系的能量为

$$E = \langle\Psi|H|\Psi\rangle = \sum_{n,m} \psi_n^* \psi_m \langle n|H|m\rangle \tag{B.1}$$

该式不能直接计算，这是因为其中对基矢n, m的求和项数都随着系统的尺度增大指数上升。但我们可进行如下的改写

$$E = \langle\Psi|H|\Psi\rangle = \sum_n |\psi_n|^2 \sum_m \frac{\psi_m}{\psi_n}\langle n|H|m\rangle = \left\langle \sum_m \frac{\psi_m}{\psi_n}\langle n|H|m\rangle \right\rangle = \langle E_{loc,n}\rangle \tag{B.2}$$

其中$\langle\ldots\rangle$为$|n\rangle$~$|\psi_n|^2$的分布，$E_{loc,n} \equiv \sum_m (\psi_m/\psi_n)\langle n|H|m\rangle$，$\langle E_{loc,n}\rangle = \sum_n |\psi_n|^2 E_{loc,n}$。改写之后，对 n 的求和可以用对基矢$|n\rangle$的以$|\psi_n|^2$为概率的蒙特卡洛抽样代替。另一方面，由于哈密顿量 H 稀疏矩阵的特性，对于给定的$|n\rangle$，使得$\langle n|H|m\rangle \neq 0$的$|m\rangle$的数量远少于$|m\rangle$的总数，一般正比于体系的大小，因此对 m 的求和也具有可行性。由此就可以求出体系的能量。

用类似的方式，我们也可以通过蒙特卡洛抽样的方式得到体系能量随参数的变化情况，从而进行数值变分，这也是变分蒙特卡洛名称的由来。我们使用的具体参数优化手段为随机重构（stochastic reconfiguration），详见附录 D。

C 监督学习算法

在通过数值方法得到体系的波函数之后，我们就可以用已知的波函数对神经网络进行监督训练。

与正文中一致，约定$|\Psi\rangle=\sum_n \psi_n|n\rangle$为神经网络量子态，$|\Phi\rangle=\sum_n \phi_n|n\rangle$为已知基态，选择损失函数（loss function）为

$$\mathcal{L}=-\ln\frac{\langle\Psi|\Phi\rangle\langle\Phi|\Psi\rangle}{\langle\Psi|\Psi\rangle\langle\Phi|\Phi\rangle} \tag{C.1}$$

对于某一个参数W_k，损失函数的梯度方向为

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial W_k^*}&=\sum_n\frac{\partial\mathcal{L}}{\partial\psi_n^*}\frac{\partial\psi_n^*}{\partial W_k^*}=\frac{\sum_n\frac{\partial\psi_n^*}{\partial W_k^*}\psi_n}{\sum_n|\psi_n|^2}+\frac{\sum_n\frac{\partial\psi_n^*}{\partial W_k^*}\phi_n}{\sum_n\psi_n^*\phi_n}\\&=\frac{\langle O_{n,k}^*|\psi_n|^2/|\phi_n|^2\rangle}{\langle|\psi_n|^2/|\phi_n|^2\rangle}-\frac{\langle O_{n,k}^*\psi_n^*/\phi_n^*\rangle}{\langle\psi_n^*/\phi_n^*\rangle}\end{aligned} \tag{C.2}$$

其中$\langle\dots\rangle$代表$|n\rangle\sim|\phi_n|^2$的分布，$O_{n,k}\equiv(1/\psi_n)\,\partial\psi_n/\partial W_k$。这里需要注意由于$\mathcal{L}$不是关于$W_k$与$\psi_n$的解析函数，正确的梯度下降方向应是对复共轭$W_k^*$与$\psi_n^*$求偏导[21]。

从 C.2 式出发，我们可以通过与附录 B 中类似的方法多次取样计算损失函数的梯度，进而通过一些现有的数值方法优化参数。在本文中，我们选取的优化方法包括 AdaDelta 与 AmsGrad，具体的选取详见附录 E。

D 随机重构

强化学习求解未知波函数时，一种较为直接的参数优化思路是让参数在能量减小的方向上变动一小步，步长正比于能量梯度，即

$$\Delta W_k = -\gamma \frac{\partial E}{\partial W_k^*} \tag{D.1}$$

其中γ是一个很小的正数，用来控制参数变化率。这种做法称为随机梯度下降（stochastic gradient descent），是机器学习领域中最基础的参数优化方法。

但是该方法等效于一个在参数空间中的数值变分。参数空间中的一段距离可能对应于希尔伯特空间中的一段很大或很小的距离，造成最终结果不收敛或是困于局域最优解。

因此，我们选择在希尔伯特空间中进行变分，这种做法称为随机重构（stochastic reconfiguration）。以下是大致的数学推导。

当我们量子态变动$|\Psi\rangle \to |\Psi\rangle + |\Delta\Psi\rangle$一小步时，实际归一量子态的变动量为

$$\left|\Delta\tilde{\Psi}\right\rangle = \frac{|\Psi\rangle + |\Delta\Psi\rangle}{\||\Psi\rangle + |\Delta\Psi\rangle\|} - \frac{|\Psi\rangle}{\||\Psi\rangle\|} = |\Delta\Psi\rangle - \mathrm{Re}\langle\Psi|\Delta\Psi\rangle \cdot |\Psi\rangle \tag{D.2}$$

在希尔伯特空间中变分，希望得到的量子态变动量为

$$|\Delta\Psi\rangle_H = -\gamma \sum_n \frac{\partial E}{\partial \psi_n^*}|n\rangle = -\gamma \sum_n \psi_n E_{loc,n}|n\rangle \tag{D.3a}$$

$$\left|\Delta\tilde{\Psi}\right\rangle_H = -\gamma \sum_n \left(E_{loc,n} - \mathrm{Re}\langle E_{loc,n}\rangle\right)|n\rangle \tag{D.3b}$$

其中$\langle\ldots\rangle$以及$E_{loc,n}$与 B.2 式中的定义一致。由参数变化引起的量子态变动量为

$$|\Delta\Psi\rangle_W = \sum_{n,k} \Delta W_k \frac{\partial \psi_n}{\partial W_k}|n\rangle = \sum_{n,k} \psi_n \Delta W_k O_{n,k}|n\rangle \tag{D.4a}$$

$$\left|\Delta\tilde{\Psi}\right\rangle_W = \sum_{n,k} \psi_n \left[\Delta W_k O_{n,k} - \mathrm{Re}\left(\Delta W_k \langle O_{n,k}\rangle\right)\right]|n\rangle \tag{D.4b}$$

其中$\langle\ldots\rangle$的定义与之前一致，$O_{n,k} \equiv (1/\psi_n)\,\partial\psi_n/\partial W_k$与 C.2 式一致。

我们希望可以调整$\{\Delta W_k\}$的值，使得$\left|\Delta\tilde{\Psi}\right\rangle_H = \left|\Delta\tilde{\Psi}\right\rangle_W$。但由于参数$\{W_k\}$的数量远少于希尔伯特空间的维度，该等式无法完全实现。但是可以寻找一组$\{\Delta W_k\}$使得两个变动量在希尔伯特空间中的距离$\left\|\left|\Delta\tilde{\Psi}\right\rangle_W - \left|\Delta\tilde{\Psi}\right\rangle_H\right\|$最小，由此解得

$$\Delta\mathbf{W} = -\gamma S^{-1}\mathbf{F} \tag{D.5}$$

其中$S_{kk'} \equiv \langle O_{n,k}O_{n,k'}^*\rangle - \langle O_{n,k}\rangle\langle O_{n,k'}^*\rangle$，$F_k \equiv \langle E_{loc,n}O_{n,k}^*\rangle - \langle E_{loc,n}\rangle\langle O_{n,k}^*\rangle$。直接用数值方法解该矩阵方程复杂度较大，可用 MINRES-QLP 算法[22]降低复杂度。此外，可以将 S 矩阵的对角元做一个提升，使得修改后的 S 矩阵为$S_{kk'}^{\mathrm{reg}} = S_{kk'}(1 + \lambda\delta_{kk'})$，防止参数空间中的变动量过大[23]。本文中取$\lambda = 0.01\sim0.1$。

此外，还可以利用其他数值优化手段与随机重构结合在希尔伯特空间中进行变分，如 AmsGrad，具体的选取详见附录 E。

E 算法与参数选取

表 E.1 优化算法及其参数

学习方式	模型	优化算法	学习率η	β_1 or ρ	β_2	ϵ
监督学习*	Ising J_1-J_2	AdaDelta**		0.95		1e-7
		AmsGrad**	0.001	0.9	0.999	
强化学习***	Ising	AmsGrad	0.005	0.8	0.999	
	J_1-J_2	AmsGrad	0.0008	0.8	0.999	
迁移后的强化学习	J_1-J_2	AmsGrad	0.0005	0.85	0.999	

* 先使用 AdaDelta 优化 10 到 100 步，再在之后的优化中使用 AmsGrad

** AdaDelta 详见[24]，AmsGrad 详见[25]

*** 强化学习都基于附录 D 中随机重构（stochastic reconfiguration）与特定优化算法的组合

表 E.2 参数初始化

学习方式	第 1 层	第 2、3 层
监督学习	标准差为 0.1 的正态分布	Kaiming initialization*
强化学习	标准差为 0.01 的正态分布	
迁移学习	监督学习中得到的参数结果	

* 详见[26]

F 监督学习程序

```
# FILENAME: Ising_supervised.ipynb
# PROGRAMMER: CHEN Ao
# COMPILER: Jupyter Notebook, Python 3.6.7
# REQUIRED: MPI, netket, numpy
# DESCRIPTION: use supervised learning to construct neural-network
#                   quantum states for 1D transverse field Ising model

from mpi4py import MPI
import netket as nk
import numpy as np
import math
from netket.machine import FFNN
from netket.layer import ConvolutionalHypercube
from netket.layer import Relu
from netket.layer import Lncosh
from netket.layer import SumOutput

def Conv(input_channels, output_channels, kernel_length, use_bias = True):
    return ConvolutionalHypercube(length = L,
                                  n_dim = 1,
                                  input_channels = input_channels,
                                  output_channels = output_channels,
                                  kernel_length = kernel_length,
                                  use_bias = use_bias)

L = 12
h_list = [1.]
g = nk.graph.Hypercube(length = L, n_dim = 1)
hi = nk.hilbert.Spin(s=0.5, graph=g)

for h in h_list:
    op = nk.operator.Ising(h = h, hilbert=hi)
    res = nk.exact.lanczos_ed(op, first_n=1, compute_eigenvectors=True)
    ttargets = []
    tsamples = []
    hind = nk.hilbert.HilbertIndex(hi)

    for i in range(hind.n_states):
        visible = hind.number_to_state(i)
        tsamples.append(visible.tolist())
        target = np.log(res.eigenvectors[0][i])
        if np.real(target) < -L/2:
        target = 1j*np.imag(target) - L/2
        ttargets.append([target])

    layers = ( Conv(1, L, 3, use_bias = False), Lncosh(L*L),
        Conv(L, L//2, L//2-1), Relu(L//2*L),
        Conv(L//2, L//4, L-1), Relu(L//4*L),
        SumOutput(L//4*L) )
    layers[0].init_random_parameters(sigma = 0.1)
    layers[2].init_random_parameters(sigma = math.sqrt(2/(L*(L//2-1))))
    layers[4].init_random_parameters(sigma = math.sqrt(2/(L//2*(L-1))))
    ma = FFNN(hi, layers)
```

```
optm_AdaDelta = nk.optimizer.AdaDelta()
optm_AmsGrad = nk.optimizer.AmsGrad()
spvsd_AdaDelta = nk.supervised.Supervised(
    machine=ma,
    optimizer=optm_AdaDelta,
    batch_size=400,
    samples=tsamples,
    targets=ttargets)
spvsd_AmsGrad = nk.supervised.Supervised(
    machine=ma,
    optimizer=optm_AmsGrad,
    batch_size=400,
    samples=tsamples,
    targets=ttargets)

spvsd_AdaDelta.run(n_iter=10,
    loss_function="Overlap_phi",
    output_prefix='Ising_AdaDelta%d'%int(10*h+0.5))
spvsd_AmsGrad.run(n_iter=2000,
    loss_function="Overlap_phi",
    output_prefix='Ising_AmsGrad%d'%int(10*h+0.5))
```

G 强化学习程序

```
# FILENAME: Ising_reinforcement.ipynb
# PROGRAMMER: CHEN Ao
# COMPILER: Jupyter Notebook, Python 3.6.7
# REQUIRED: MPI, netket, numpy
# DESCRIPTION: use reinforcement learning to solve 1D transverse field Ising model

from mpi4py import MPI
import netket as nk
import numpy as np
import math
from netket.machine import FFNN
from netket.layer import ConvolutionalHypercube
from netket.layer import Relu
from netket.layer import Lncosh
from netket.layer import SumOutput

def Conv(input_channels, output_channels, kernel_length, use_bias = True):
    return ConvolutionalHypercube(length = L,
                                  n_dim = 1,
                                  input_channels = input_channels,
                                  output_channels = output_channels,
                                  kernel_length = kernel_length,
                                  use_bias = use_bias)

L = 28
h_list = [1.]
g = nk.graph.Hypercube(length = L, n_dim = 1)
hi = nk.hilbert.Spin(s=0.5, graph=g)

for h in h_list:
    op = nk.operator.Ising(h = h, hilbert=hi)
    layers = ( Conv(1, L, 3, use_bias = False), Lncosh(L*L),
        Conv(L, L//2, L//2-1), Relu(L//2*L),
        Conv(L//2, L//4, L-1), Relu(L//4*L),
        SumOutput(L//4*L) )
    layers[0].init_random_parameters(sigma = 0.01)
    layers[2].init_random_parameters(sigma = math.sqrt(2/(L*(L//2-1))))
    layers[4].init_random_parameters(sigma = math.sqrt(2/(L//2*(L-1))))
    ma = FFNN(hi, layers)
    sa = nk.sampler.MetropolisLocal(machine = ma)
    opt = nk.optimizer.AmsGrad(learning_rate = 0.005, beta1 = 0.8)
    gs = nk.variational.Vmc(hamiltonian=op,
        sampler=sa,
        optimizer=opt,
        n_samples=1000,
        use_iterative=True,
        method='Sr')
    gs.run(output_prefix='Ising%d'%int(10*h+0.5), n_iter=1000)
```