Train BNNs with Reinforcement Learning

毕业设计开题报告

唐国鑫

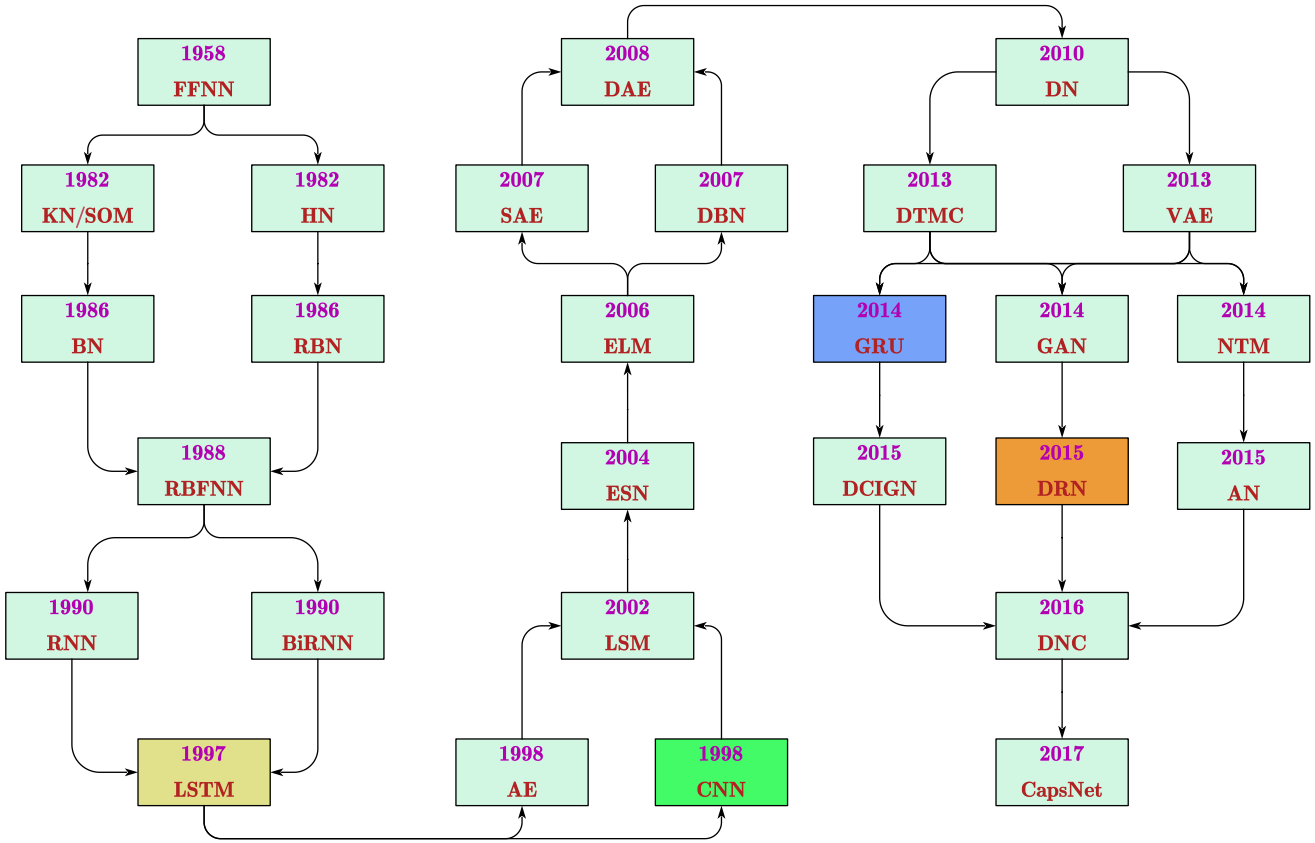
四川师范大学数学科学学院

2022 年 4 月 12 日

Such A Huge Model

人工神经网络 (ANN) 是机器学习的主要研究对象之一. 早在 1958 年, Rosenblatt 提出了感知机模型. 这是最早的 ANN 模型. 然而由于模型难以训练, 因此 ANN 并没有过多引起学者们的关注. 直到 1986 年 Hinton 提出了反向传播算法 (BP), 并且随着计算机算力的发展, ANN 再次在机器学习领域掀起了研究热潮. 并衍生出了深度学习和强化学习领域. 而真正让人们记住深度学习和强化学习的威力的时间是 2016 年 AlphaGo 击败围棋世界冠军李世石, 并且在 2017 年再次击败围棋世界冠军柯洁. ANN 强大的能力与它庞大而复杂的结构息息相关, 算法和算力的同时发展才使得训练出一个强大的 ANN 模型成为了可能.

发展史


1958
FFNN
1982
KN/SOM
1982
HN
1986
BN
1986
RBN
1988
RBFNN
1990
RNN
1990
BiRNN
1997
LSTM
2008
DAE
2007
SAE
2007
DBN
2006
ELM
2004
ESN
2002
LSM
1998
AE
1998
CNN
2010
DN
2013
DTMC
2013
VAE
2014
GRU
2014
GAN
2014
NTM
2015
DCIGN
2015
DRN
2015
AN
2016
DNC
2017
CapsNet

应用领域

经典领域

1. 时间序列分析
2. 异常检测
3. 图像及语音识别
4. 数据挖掘及推荐系统
5. 文本分类
6. 微分方程求解, 分子及蛋白质结构预测

更多探索

1. 上下文知识推理
2. 数学逻辑和定理证明
3. 根据注释进行代码补全
4. 图形风格迁移
5. 视频画质修复和补帧
6. 自动玩游戏, 写诗

Such A Huge Model

1998 年 Yann LeCun 提出了 LeNet-5[1] 模型, 并在手写数字识别上获得成功. LeNet-5 不是卷积神经网络 (CNN) 的起点, 但却是 CNN 兴起的标志. 图1是 LeNet-5 的结构. 它一共有 122 304 个参数.

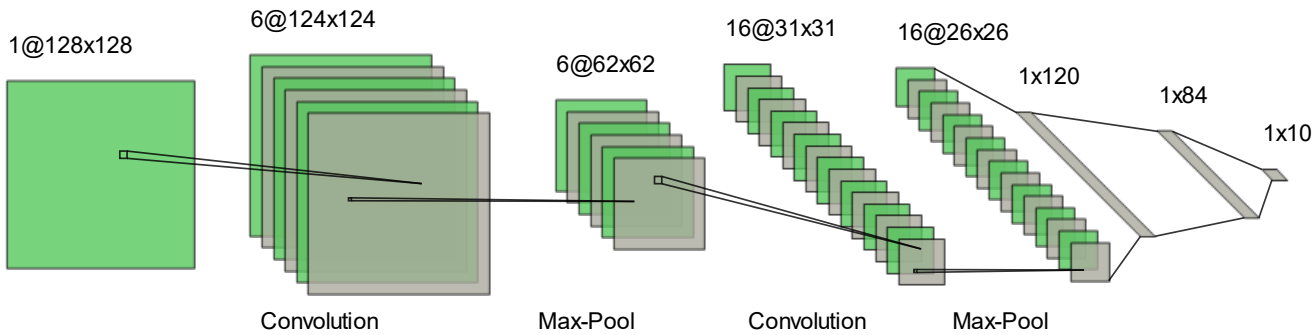


图 1: LeNet-5

Such A Huge Model

表 1: 经典的 CNN 构架

Name	Year	Author	Depth	Parameter
LeNet-5	1998	LeCun	7	122 304
AlexNet	2012	Alex & Jeffry	9	60 965 128
VGGNet	2014	Oxford university	16	138 357 544
GoogleNet	2014	Google	22	6 994 392
ResNet	2015	Microsoft(何凯明等人)	168	25 636 712
Xception	2017	Chollet	126	22 910 480
MobileNets	2017	Google	88	4 253 864
PaLM	2022	Google	118	540.35 billions

Such A Huge Model

然而, 训练一个如此庞大的模型的代价在实际应用中很多时候是难以接受的. 以 PaLM 模型为例, Google 共用了 6144 块 TPU(Tensor Processing Unit) 来进行训练. 强大的系统和算力带来了惊艳的结果, 也产生了巨大的资源消耗. 此外, 杨迪一 [2] 等人在论文中的实验证明, 在 GNN 中, 可能有 99% 的参数都是冗余的. 我们有理由怀疑是否模型中所有的参数都是必要的.

另一方面, 庞大的模型很难嵌入到我们的小型设备 (如智能手机, 智能手表等). 对模型进行瘦身是有必要的.

Such A Huge Model

图2是 $f(x) = x^2 + \mathcal{N}(0,1)$ 生成的数据.

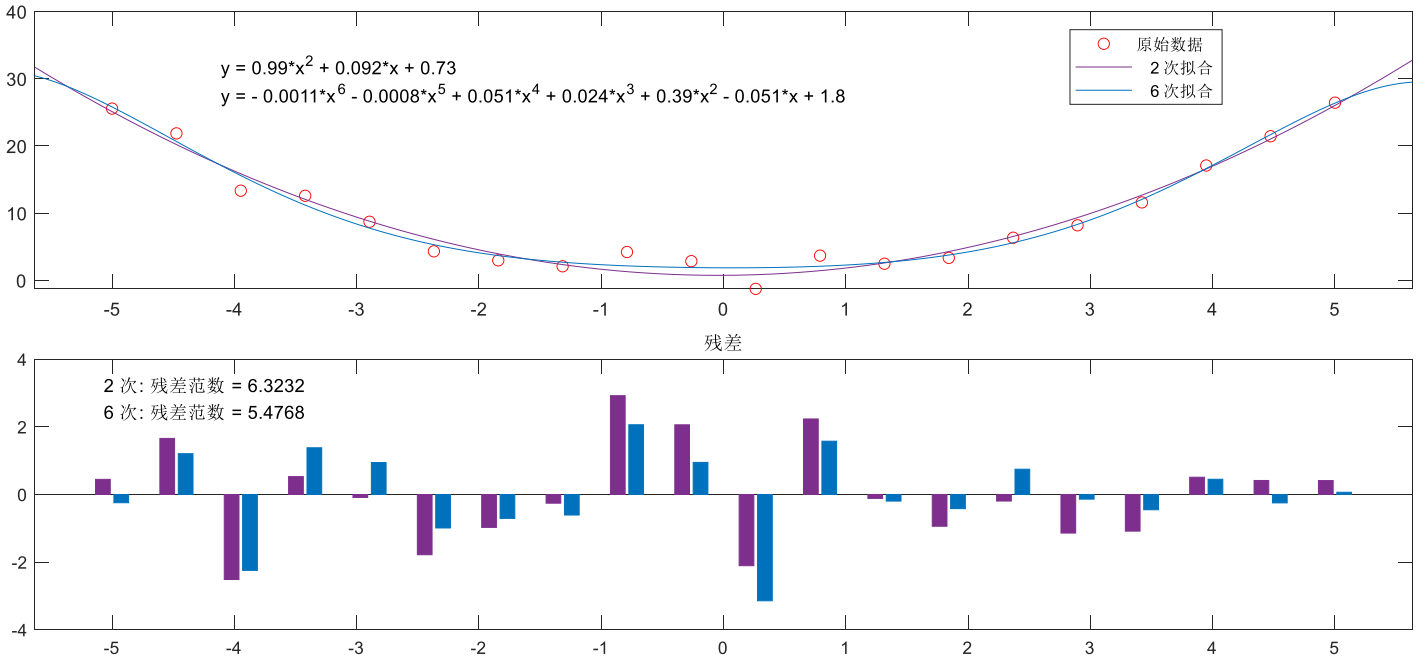


图 2: 二次多项式拟合与六次多项式拟合

知识蒸馏

知识蒸馏 [3] 最早于 2015 年由 Hinton 提出，蒸馏的目的是为了缩减模型. 此外也有蒸馏特征的数据蒸馏 [4]. 通过蒸馏的手段，使得模型变小或数据集更加易于学习，以此到达降低训练成本的目的.

- 模型剪枝
- 量化
- 参数共享

量化

我们的主要研究集中在模型的量化上, 比如用 8 位浮点数代替 32 位浮点数, 或将浮点数向上或向下取整为一个整数等. 模型量化的主要方法有:

- 二值化网络: Binary Network [5]
- 三值化网络: Ternary Network [6]
- 深度压缩: Deep Compression [7]

此外, 对模型的压缩或缩减不局限于使用一种方法, 多数时候会结合量化与剪枝等操作在一个模型上.

Binary Neural Networks(BNNs)

2015 年, Bengio[5] 等人第一次将神经网络中的参数二值化, 并在 MINST, CIFAR-10 和 SVHN 数据集上达到了 SOTA 级别的性能. Bengio 对网络量化的可行性的解析如下:

- 对参数的量化其实是对参数引入了一定程度的噪声, 并在训练过程中通过累计逐渐被平均掉.
- 类似于 DropOut 和 DropConnect, 参数引入噪声意味着给网路施加了某种正则成分, 从而提高网络的泛化性能.

通过对参数的量化, 使得原来浮点数的乘法运算变成了加法运算甚至直接使用位运算, 这不仅大大降低了模型所需要的物理内存, 也极大提高了计算效率.

值得一提的是, Hinton, LeCun 和 Bengio 三人一起获得了 2018 年国际计算机协会 (ACM) 图灵奖

阻碍

然而白璧微瑕. 二值化量化带来了低内存, 高效率的优势的同时, 也引发了一系列新的问题. 由于权重的值域不再连续, 面临的第一个困难就是 BNNs 的梯度处处为 0. 这使得 Hinton 的反向传播算法无法直接对 BNNs 生效.

2012 年, Hinton 提出直通估计器 (Straight Through Estimator: STE)[8]. 网络的前向传播使用被量化的 w_b 来计算, 但是仍然保留全精度的原始 w 用于计算梯度. 下面的公式是文中参数二值化的方法.

$$w_b = \operatorname{sign}(w) = \begin{cases} +1 & w \geqslant 0 \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

方案

为了解决参数二值化后梯度处处为 0，泛化性能降低等问题，许多学者从以下方面对 BNNs 进行了研究 [9]:

- 使用新的预定义函数对权重进行量化
- 通过最小量化误差来逼近全精度值
- 修改损失函数来限制权重
- 减小梯度误差来学习离散参数

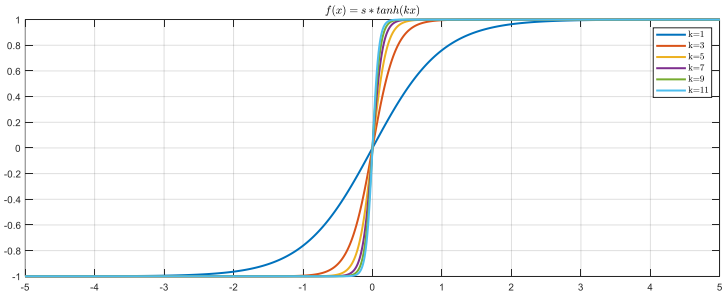


图 3: 量化逼近

方案

下面的方法分别是部分对损失函数的修改，使用量化误差逼近全精度权重以及减少梯度误差的方法.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{DL}$$
$$\mathcal{L}(x; \mathbf{w}^T, \mathbf{b}_{\mathbf{w}}^S) = \alpha \mathcal{H}(y, p^T) + \beta \mathcal{H}(y, p^S) + \gamma \mathcal{H}(z^T, p^S)$$

$$J(\alpha, \boldsymbol{b}_{\mathrm{W}}) = \min_{\alpha, \mathbf{b}_{\mathrm{W}}} \|\mathrm{W} - \alpha \boldsymbol{b}_{\mathrm{W}}\|$$
$$J(\alpha, \boldsymbol{b}_{\mathrm{W}}) = \min_{\alpha, \mathbf{b}_{\mathbf{w}}} \|\mathbf{z} - Q_a(\alpha(\boldsymbol{a} \odot \mathbf{b}_{\mathbf{w}}))\|$$
$$\alpha^*, \mathbf{b}_{\mathrm{W}} = arg\min_{\alpha, \mathbf{b}_{\mathrm{W}}} J(\mathbf{b}_{\mathrm{W}}, \alpha)$$

$$\text{ApproxSign}(x) = \begin{cases} -1, & x < -1 \\ 2x + x^2, & -1 \le x < 0 \\ 2x - x^2, & 0 \le x < 1 \\ 1, & \text{otherwise} \end{cases}$$

$$\frac{\partial \text{ApproxSign}(x)}{\partial x} = \begin{cases} 2 + 2x, & -1 \le x < 0 \\ 2 - 2x, & 0 \le x < 1 \\ 0, & \text{otherwise} \end{cases}$$

图 4: 已有的部分方案

方案

自 2015 年以来, 已查阅到的 BNNs 新的研究已多达 36 种. 下表给出了部分已有的对 BNNs 的研究, 它们都是基于以上思想对 BNNs 新的探索.

表 2: 部分基于这些方法所提出的新模型

Type	Method	Key Tech.
朴素 BNNs	BinaryConnect Bitwise Neural Networks Binarized Neural Networks	STE
最小化量化误差	Binary Weight Networks XNOR-Net DoReFa-Net	$\alpha^*, \mathbf{b}_\mathrm{W} = \underset{\alpha, \mathbf{b}_\mathrm{W}}{arg\min} J\left(\mathbf{b}_\mathrm{W}, \alpha\right)$
改进损失函数	BNN-DL CI-BCNN Loss-Aware Binarization	$\mathcal{L}_\mathrm{total} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{DL}$
减少梯度误差	BNN+ BCGD IR-NET	$\varphi\left(x\right) = s\tanh\left(kx\right)$

无法回避的问题

总的来说, 几乎所有对 BNNs 的探索都集中如何处理梯度处处为 0 的问题. 目的就是为了使得误差的反向传播能够重新工作起来. STE 尽管可以一定程度上可以让我们重新使用 SGD, Adam 等算法进行训练. 然而, 二值化参数对梯度的大小是不敏感的. 这可能导致下面的情况产生:

$$\begin{aligned} \operatorname{sign}(+1-\nabla_w) &= +1 \ (\nabla_w = -1, -10) \\ \operatorname{sign}(-1-\nabla_w) &= -1 \ (\nabla_w = -1, -10). \end{aligned} \tag{2}$$

换句话说, 达到一定大小值的梯度对参数的改变不再起决定性作用. 我们应该更加关心的是梯度的方向或梯度的动作, 而不是梯度的值.

优化方法

在梯度可求且不处处为 0 的情况下, SGD, Adam, L-BFGS 等算法是训练 BNNs 的不错选择. 如果将 BNNs 模型写作下面的形式:

$$\begin{aligned} \mathcal{L} &= \max -\mathcal{L}\left(\mathrm{W}_b\right) \\ s.t.\ & w_{b(i,j)} = -1 \text{ or } 1. \end{aligned} \tag{3}$$

我们可以将其看做一个非线性整数规划问题 (当参数为 0 和 1 时, 也可以看做是 0, 1 整数规划或指派问题). 然而, 常规的割平面法和分支定界法无法使用. 遗传算法 (GA) 是解决这一问题的一种可行方案. 但是庞大的参数会给 GA 的搜索进化带来困难, 甚至难以收敛到一个令人满意的局部最优值. 并且迭代时间可能也是难以让人接受的.

非梯度不可吗?

如果我们一定要用基于梯度的算法来训练 BNNs, 我们就不得不向寻找梯度的代替方案屈服. 如果要用基于无梯度方法的算法 (如 GA), 我们就要付出更多的时间代价甚至牺牲模型的精度.

已有的对 BNNs 梯度为 0 的研究都是在想办法如何使得反向传播重新工作. 在我们的研究中, 我们不再将梯度当做梯度, 也不再需要梯度来调整我们的参数. 而是将参数的调整看作是一个动作的调整. 这是就是强化学习的基本观点之一. 强化学习是最接近人的学习行为方式的算法.

动机

考虑一个如下的走迷宫问题: 机器人从左上角出发, 且只能上下左右移动, 需要从绿色的出口离开. 机器人每走一步, 就会得到 -1 分的奖励, 掉入红色的陷阱中则游戏结束, 需要重新从起点出发. 而到达出口则可以获得 100 分的奖励. 那么我们需要考虑的就是如何最大化自己的奖励.

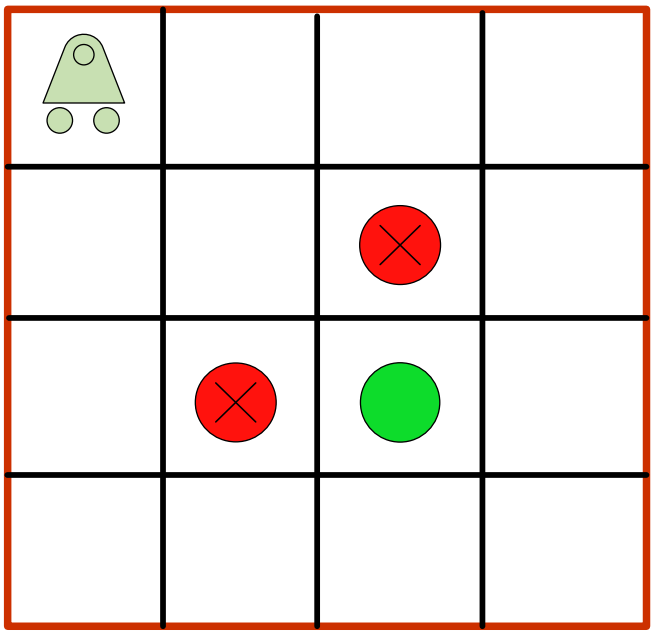


图 5: Maze

动机

比较传统的方法如 DFS 和 BFS 算法，我们都可以直接找到到达出口的最优方案或策略. 然而在强化学习中，我们会让机器人找到在每一个位置 (或状态) 价值最大的动作. 在图6中，通过观察可以发现想要最快到达，应该向右移动. 确定处于任意位置的最大价值的动作是强化学习的目标.

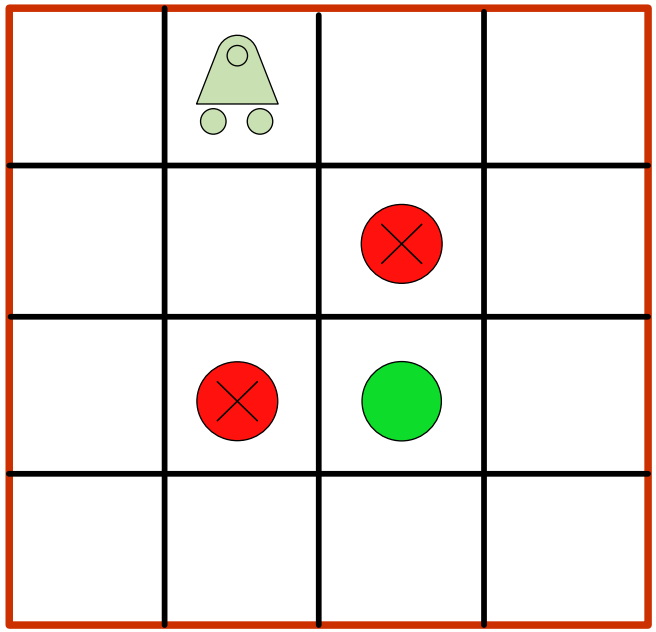
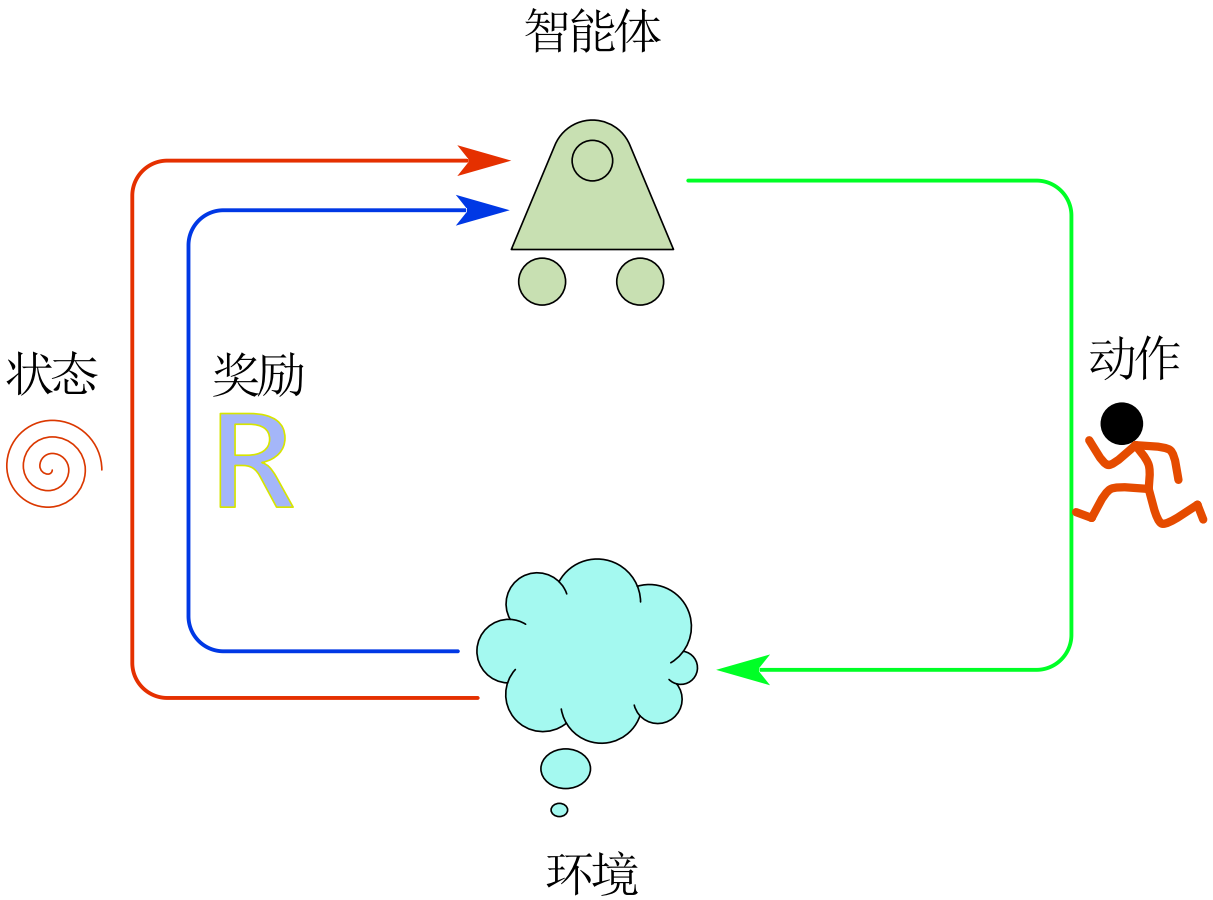


图 6: 处于某一个状态的机器人

强化学习

下图是强化学习的基本示意图. 本文将 BNNs 的训练看作是一个强化学习过程. 从而不直接使用梯度的方法来更新 BNNs 的参数. 图中展示了强化学习的 5 个基本元素.



强化学习

强化学习的目的是找到一个最优的动作策略，价值函数用于评价动作的好坏. 而在训练 BNNs 的过程中，则是通过学习动作找到一个最优的状态使得价值函数最大.

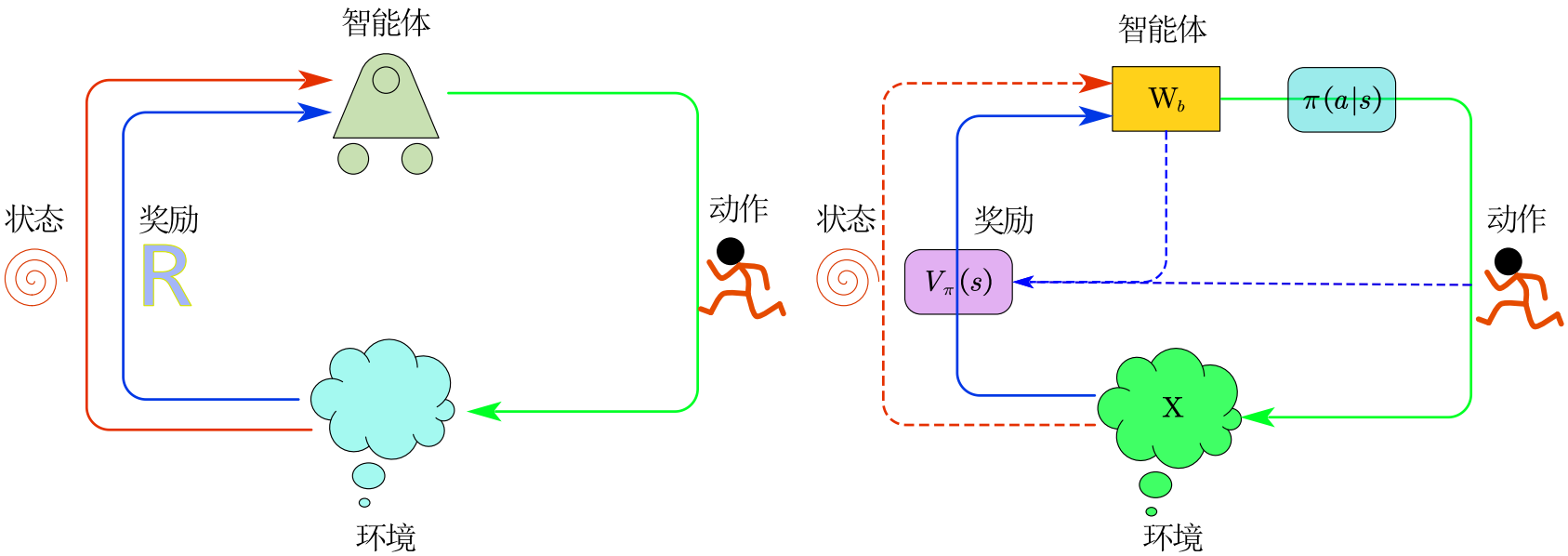


图 8: 强化学习下的 BNNs

调整动作的一个列子

下图展示了如何调整动作来使得直线将两个类别分离出来.

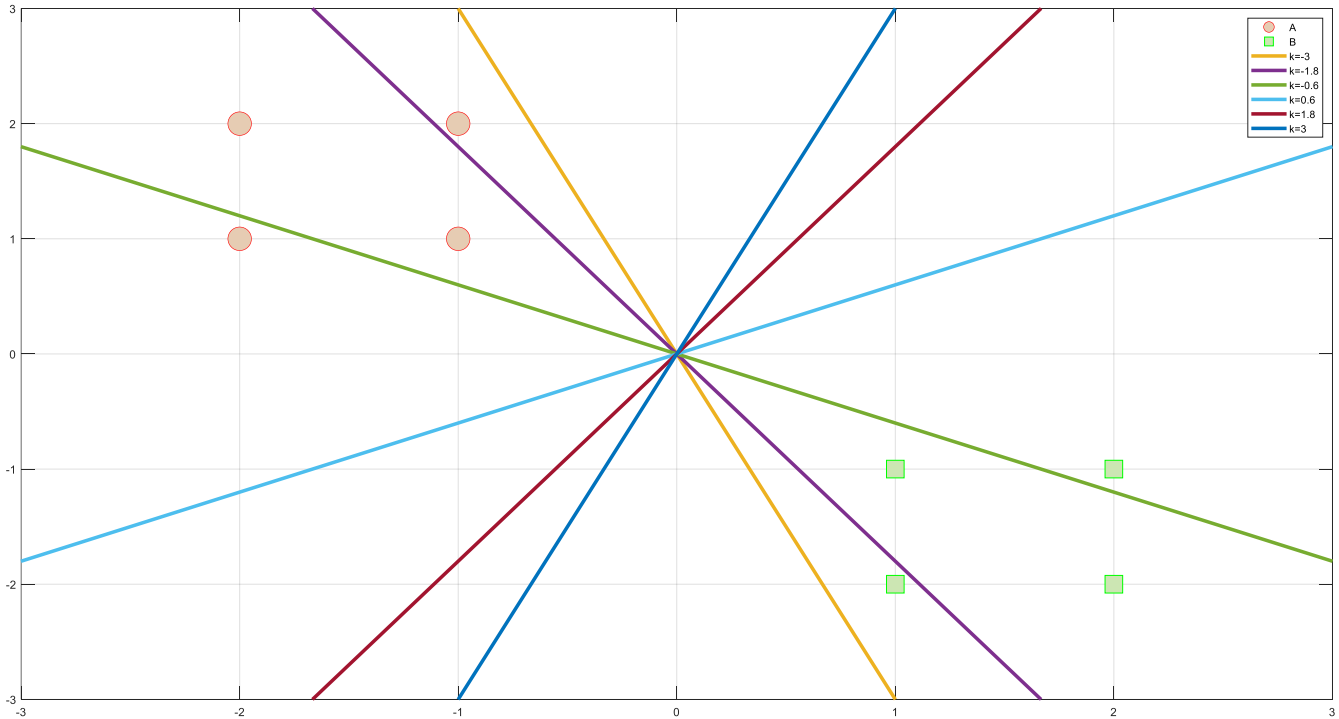


图 9: 调整动作 (这里是斜率 k) 来使得直线将 A 和 B 分为两个类别.

移步到 BNNs 中

在 BNNs 中，由于二值参数的非连续性. 将其看做是动作的更新而不是通过梯度来更新是很自然的. 当然与图5的问题比起来我们的模型将更加复杂. 因为在迷宫问题中，机器人只有一个动作需要更新: 即上下左右中的一个. 而对于一个参数矩阵来说，所有的矩阵元素都将可能是我们需要更新的动作. 我们可以看做是向前或向后或原地不动. 就如同人类的婴儿学习走路一样，我们应该如何让自身的关节相互协调，以达到平稳行走的目的.

假设 $\mathcal{D} = \{\mathrm{X}, \boldsymbol{y}\}$ 是包含了输入和输出的数据集. 一个 BNNs 模型被定义如下:

$$\mathcal{F} = \mathcal{F}\left(\mathrm{X}; \left[\mathrm{W}_b^1, \mathrm{W}_b^2, \mathrm{W}_b^3\right]\right), \tag{4}$$

$$\mathcal{L} = \mathcal{L}\left(\mathcal{F}\left(\mathrm{X}\right), \boldsymbol{y}\right). \tag{5}$$

移步到 BNNs 中

为了方便, 我们定义二值矩阵为 2×2 的矩阵. 那么可能让 BNN 学会平稳行走的策略可以由下图表示:

$$\mathrm{W}_b^1 = \begin{pmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{pmatrix} \quad \mathrm{W}_b^2 = \begin{pmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \end{pmatrix} \quad \mathrm{W}_b^3 = \begin{pmatrix} w_{11}^3 & w_{12}^3 \\ w_{21}^3 & w_{22}^3 \end{pmatrix}$$

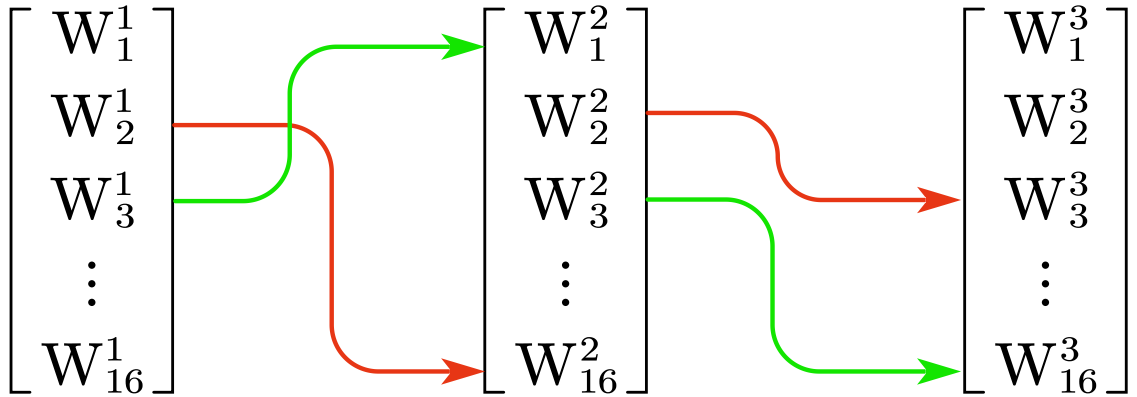


图 10: 不同颜色的线条表示我们的不同策略

可行性分析

理论的支撑

据文献 [10] 记载，早在 1954 年 Minsky [1]首次提出强化一次的概念．并且确立了试错是强化学习的核心机制．早期的强化学习是为了求解最优控制问题，并开发了动态规划和马尔科夫决策过程的策略迭代方法．1988 年 Minh 提出时序差分算法 [11]，并且于 1989 年，Watking 提出了 Q-Learning[12] 算法，并给出了算法的收敛性证明，强化学习理论开始逐渐开始完备．

实践的探索

目前，已有不少研究使用强化学习的方式自动生成 ANN 的结构 ([13, 14, 15] 等)．Aenugu 提出了一种使用强化学习训练脉冲神经网络的方法 [16]．James 等人提出了一种使用学习自动机 [17] 来训练 ANN 的方法，而不需要误差的反向传播．Lazarus[18] 使用 BNNs 替换 DQN 中的 ANN，提出了一种 BQN-L 算法，但是该算法仍然没有放弃用全精度计算梯度的思想．Sourabh 等人提出了一种使用策略梯度 (PG) 训练神经网络的方案 [19]．该方法重心在对神经元激活值 (而不是直接对权重) 的建模，通过另一个 ANN 的学习来导出梯度．

[1] Minsky 是人工智能之父，并获得 1969 年图灵奖

强化学习算法

强化学习大约可以分为有模型和无模型的强化学习算法:

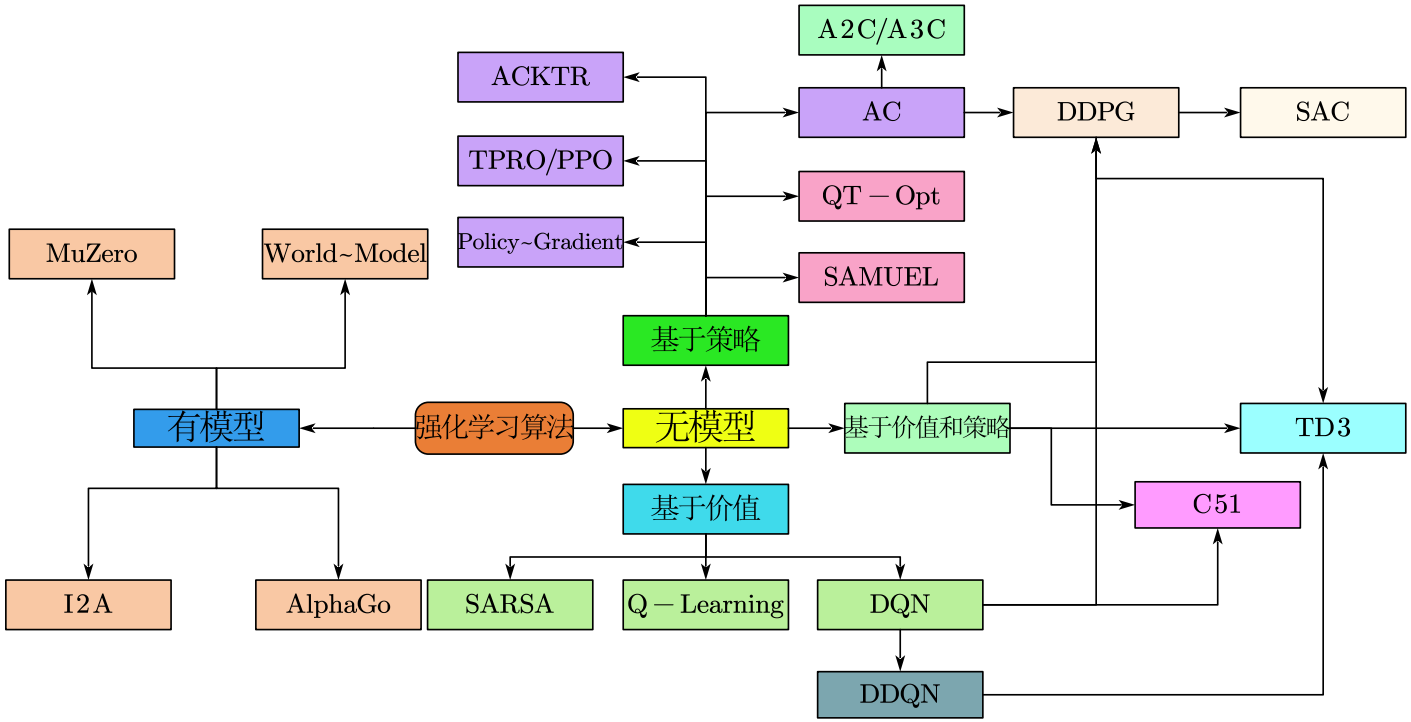


图 11: 常见强化学习算法分类

有模型和无模型的区别

下图展示了有模型和无模型的区别:

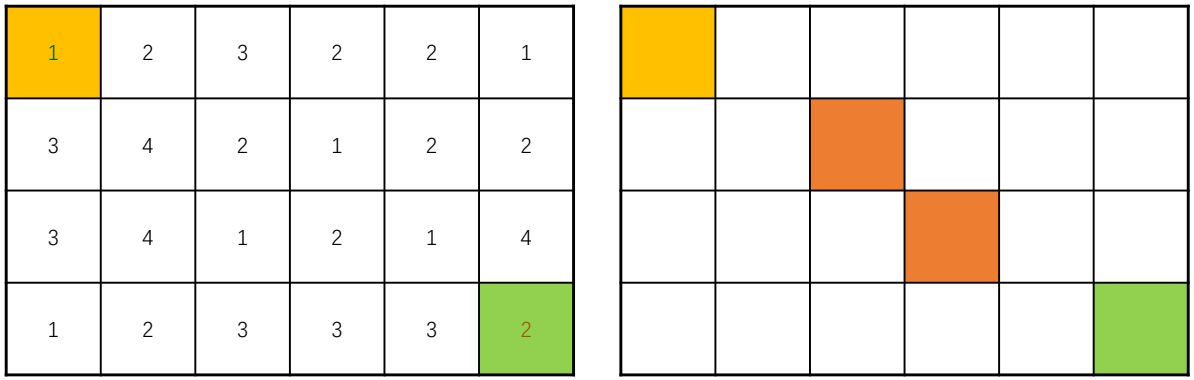


图 12: 有模型 & 无模型

假设网格的值用 M 表示, D_{ij} 表示到达 i 行 j 列时的最大奖励. 那么状态转移方程可以表示如下:

状态转移方程

$$D_{ij} = \max\left(D_{i-1,j}, D_{i,j-1} + M_{ij}\right) \tag{6}$$

策略梯度定理

定义 1 (策略与奖励)

我们定义在状态 s 下动作执行 a 的概率为 $\pi_\theta(a|s)$. 这种策略称为随机策略. 我们用 r_t 表示在 t 时刻的奖励, R_t 表示从 0 时刻到 t 时刻的总奖励和. 并用 γ 表示折扣因子, 这是一个超参数, 是我们对未来的获得奖励的衰减值系数.

定义 2 (价值函数)

状态价值函数

$$V_\pi(s) \doteq \mathbb{E}_\pi\left[R_t \mid s_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right] \tag{7}$$

状态-动作价值函数

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi\left[R_t \mid s_t = s, a_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right] \tag{8}$$

策略梯度定理

定理 1 (策略梯度定理 [10])

强化学习的学习目标是最大化奖励:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\mathrm{T}} R_t\right] = \sum_{t=0}^{\mathrm{T}} \mathbb{E}_{\tau\sim\pi_\theta}\left[R_t\right] \tag{9}$$

那么我们有:

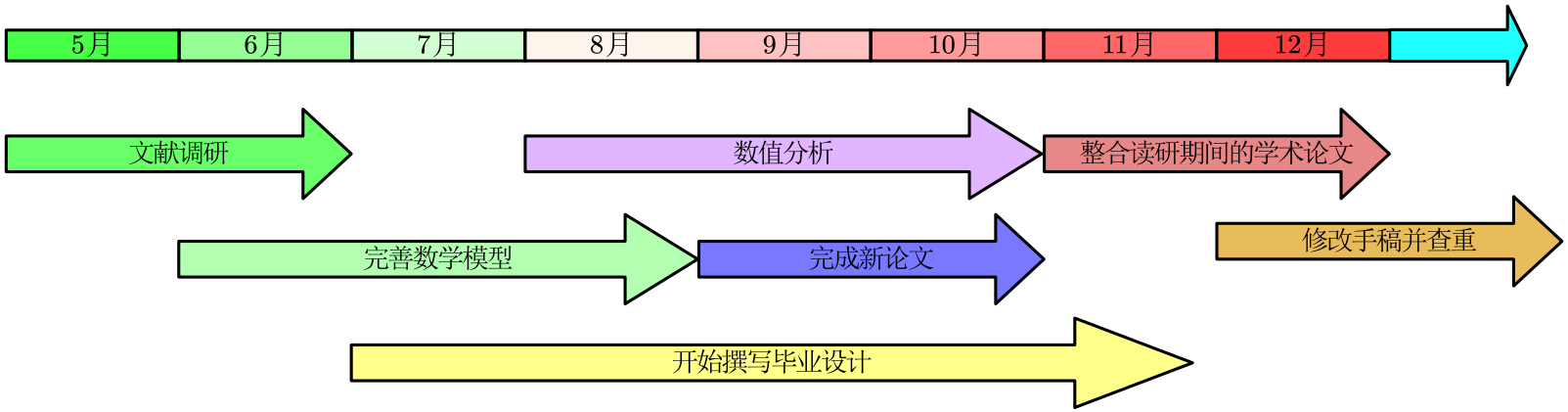
$$\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t'=0}^{T} \nabla_\theta \log \pi_\theta\left(a_{t'} \mid s_{t'}\right) \gamma^{t'} \sum_{t=t'}^{T} \gamma^{t-t'} R_t\right] \tag{10}$$

这是策略梯度的基本形式.

需要克服的问题

1. 与一般的强化学习模型不同, 训练 BNNs 的过程更像是多智能体学习, 而不是简单的只有一个机器人走迷宫问题. 然而关于多智能体学习的研究还相对较少.
2. 没有一套合适的二值计算工具, 能让计算过程全部以逻辑值 (0&1) 进行. 而 (如 MATLAB 等) 直接将矩阵元素看做是 0&1 时, 它的数据类型仍然是一个浮点型或者整型.
3. BNNs 不是一种 ANN 模型, 而是一类 ANN 模型. 新的训练方法需要尽可能多的能够迁移到其它 ANN 框架.

时间表


5月
6月
7月
8月
9月
10月
11月
12月
文献调研
数值分析
整合读研期间的学术论文
完善数学模型
完成新论文
修改手稿并查重
开始撰写毕业设计

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.

[2] Kuan Wang, Yuyu Zhang, Diyi Yang, Le Song, and Tao Qin.
Gnn is a counter? revisiting gnn for question answering.
arXiv preprint arXiv:2110.03192, 2021.

[3] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al.
Distilling the knowledge in a neural network.
arXiv preprint arXiv:1503.02531, 2(7), 2015.

[4] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros.
Dataset distillation.
arXiv preprint arXiv:1811.10959, 2018.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David.
Binaryconnect: Training deep neural networks with binary weights during propagations.
Advances in neural information processing systems, 28, 2015.

[6] Fengfu Li, Bo Zhang, and Bin Liu.
Ternary weight networks.
arXiv preprint arXiv:1605.04711, 2016.

[7] Song Han, Huizi Mao, and William J Dally.
Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.
arXiv preprint arXiv:1510.00149, 2015.

[8] Yoshua Bengio, Nicholas Léonard, and Aaron Courville.
Estimating or propagating gradients through stochastic neurons for conditional computation.
arXiv preprint arXiv:1308.3432, 2013.

[9] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe.
Binary neural networks: A survey.
Pattern Recognition, 105:107281, 2020.

[10] R. S. Sutton and A. G. Barto.
Reinforcement learning.
A Bradford Book, volume 15(7):665–685, 1998.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al.
Human-level control through deep reinforcement learning.
nature, 518(7540):529–533, 2015.

[12] Christopher JCH Watkins and Peter Dayan.
Q-learning.
Machine learning, 8(3):279–292, 1992.

[13] Natasha Jaques, Shixiang Gu, Richard E Turner, and Douglas Eck.
Tuning recurrent neural networks with reinforcement learning.
2017.

[14] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar.
Designing neural network architectures using reinforcement learning.
arXiv preprint arXiv:1611.02167, 2016.

[15] Barret Zoph and Quoc V Le.
Neural architecture search with reinforcement learning.
arXiv preprint arXiv:1611.01578, 2016.

[16] Sneha Aenugu.
Training spiking neural networks using reinforcement learning.
arXiv preprint arXiv:2005.05941, 2020.

[17] James Lindsay and Sidney Gigivi.
A novel way of training a neural network with reinforcement learning and without back propagation.
In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020.

[18] Christopher Lazarus and Mykel J Kochenderfer.
Deep binary reinforcement learning for scalable verification.
arXiv preprint arXiv:2203.05704, 2022.

[19] S. Bose and M. Huber.
Training neural networks with policy gradient.
In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

Thanks!