高级人工智能作业 2

演化神经网络

姓名：董广念 学号：11849058

本报告主要如何演化神经网络解决 5-parity 问题，算法主要思想参考自[1]，相对于[1]做出改进的有 2 点：(1)神经网络的训练采用了 adam 方法[4]进行反向传播的训练，能以更快的速度，更高的概率帮助神经网络收敛到全局最优解；(2)采用小随机种群扩大全局搜索的幅度帮助算法寻找全局最优解；报告分为(1)问题描述，(2)算法设计，(3)结果分析，(4)研究心得，(5)参考文献，5 部分组成。

1. 问题描述

设计演化算法演化人工神经网络解决 5-Parity 问题输入：N 个二进制数字

输出：0 或 1，取决于公式(1)

$$\forall x \in \{0,1\}^N, f(x) = \begin{cases} 1 if \sum_{i=1}^{N} x^{(i)} \ (mod\ \ 2) == 0 \\ 0 \quad otherwise \end{cases} \tag{1}$$

$x^{(i)}$为第输入中 i 个数字，在本次作业中 N=5

2. 算法设计

本算法主要参考自[1]算法主要分为三个部分，结构如下:

1) 神经网络部分 a) 前馈 b) 反向传播 2) 演化计算部分 a) 模拟退火
b) 节点，边的删除 c)
节点，边的增加
3) 随机种群

算法的主要流程如下所述(注，所有参数的确定请参见第四节参数设置):

1) 随机生成 M 个 ANN，并对其中的每一个 ANN 使用训练集进行计算其误差得分，正确预测的数量

2) 对(1)中生成的每个 ANN 做训练到收敛或训练 K0 次
3) 对训练后的数据根据其得分函数从小到大进行排序
4) 如果种群中是否存在满足隐藏层节点数小于等于 2，或程序运行时间超过 4 小时 10 分钟 50 秒，至步骤(10)
5) 按照排序顺序为概率[1]选择种群中的某个个体作为母代，进行模拟退火算法得到子代，如果子代得分大于等于亲代，到步骤(6)，否则训练子代到收敛替换亲代，到步骤(3)
6) 随机删除 1 个母代中隐藏层的节点得到子代，对子代训练到收敛或训练 K0 次，如果子代比当前种群中最差的好，则使用子代替换种群中最差的个体，到步骤(3)，否则到步骤(7)
7) 以母代中任意边权重的绝对值与所有边权的绝对值和的比值为概率，随机选择一条边删除，对子代训练到收敛或训练 K0 次，如果子代比当前种群中最差的好，则使用子代替换种群中最差的个体，到步骤(3)，否则到步骤(8)
8) 按照图存储顺序（即邻接表行列的存储顺序）从向亲代中增加边，后训练到收敛或 K0 次，检查是否比当前种群中最差的好。如果否，再执行增加边的过程，直至有一个比种群中最差的个体好转到步骤(3)，或图变成全连接转到步骤(9)
9) 在亲代图中随机选择一个节点分裂，裂变后的新节点继承原节点所有的连接关系，对子代训练 K0 次，转到步骤(3)
10) 输出种群中排序第一的权值矩阵

为了更好的进行全局搜索，在算法中新增了 birds 种群，birds 种群对神经网络任意一个可能的节点数生成一个全连接的神经网络（总结点数为 N）。即

$$\forall i = 1,2,\dots,N \ \text{connection}[i][j] = True\ if\ node_j\ is\ not\ input\ node$$

在本次作业中，birds 种群由节点数为 9，10，11，12 的全连接神经网络构成。

birds 种群每轮对网络中的参数随机赋值，训练 K0 次，并入迭代停止判断过程。

因为 birds 种群每轮执行的操作均为邻接矩阵的随机赋值和训练，与神经网络的初始化操作类似，故不单独给出伪代码。

算法的参数设置如下：

表 1 算法参数设置

参数名	**意义**	**数值**
M	种群大小	20
K0	训练次数	2000
NODES_RANGE	节点总数取值范围	9~12
VALUR_RANGE	神经网络初始化取值范围	-1~1
CD	初始化边连接密度	0.9

ALPHA	学习率	0.1
β_1	Adam 方法参数	0.5
β_2	Adam 方法参数	0.999
ε	Adam 方法参数	1e-8
T	模拟退火算法初始化温度	100
cool	模拟退火算法冷却率	0.98
NIGHBOUR_STEP	模拟退火算法找邻居的步长	-1~1
stop_iter	模拟退火算法迭代停止条件	1e-8

2.1 神经网络部分

网络存储结构分为以下几个部分:

输入层有一个输入恒为-1 的节点与隐层，输出层所有节点相连，边权代表每个节点的 bias;

connection 矩阵为边连接矩阵，行数为总结点数量+1（+1 代表着输入恒为-1 的输入层节点)，列数为隐藏层和输出层节点数量，边的方向为行标号指向列标号。无连接的边值为 False，有连接边值为 True;

weight 为边权重矩阵。行数为总结点数量+1（+1 代表着输入恒为-1 的输入层节点)，列数为隐藏层和输出层节点数量，边的方向为行标号指向列标号。无连接的边数值为零，有连接值非零

神经网络的评分函数为

$$\text{Score} = \sum_{t=1}^{T} (\hat{y}_t - y_t)^2 \tag{2}$$

隐藏层节点输出的计算方式为:

$$y = f(\sum_{i=1}^{n} w_i x_i - \theta)$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

输出层节点输出的计算方式为:

$$y = \begin{cases} 1 & if \ \ f(\sum_{i=1}^{n} w_i x_i - \theta) > 0.5 \\ 0 & otherwise \end{cases}$$

算法主要包含(1)神经网络的初始化，(2)feedforward 算法，(3)backpropagation 算法三部分组成。，下列为代码中‘·‘代表矩阵乘法，’×‘代表矩阵中对应元素相乘。

以下为详细介绍。

1) 神经网络的初始化

初始化首先随机生成网络节点数量，再生成(节点总数+1)×(隐藏层节点数+输出层节点数)大小的 connection 和 weight 矩阵。再遍历两个数组每个值，以连接密度为概率生成边。

伪代码如下:

算法 1	ANN 初始化
输入	NODES_RANGE-初始化节点取值范围 LR-学习率 CD-边连接密度 INPUT_NODES-输入节点数量
输出	WEIGHT-边权矩阵 CONNECTION-连接情况矩阵
1	初始化 learning_rate = LR, input_nodes_num = INPUT_NODES, nodes_num = 随机选取一个 NODES_RANGE 里的数值，hidden_out_nodes_num = nodes_num - input_nodes_num
2	初始化 connection = nodes_num 行, hidden_out_nodes_num 列，值为 False 的矩阵
3	初始化 weight = nodes_num 行, hidden_out_nodes_num 列，值为 0 的矩阵
4	for i=0,i<nodes_num,i++:
5	for j=0,j<hidden_out_nodes_num,j++:
6	if [0,1)之间的随机数<CD:
7	connection[i][j] = True
8	weight[i][j] = [-1,1]之间的随机数
9	end if
10	end for
11	end for
12	return connection, weight

2) feedforward 算法

Feedforward 算法主要用于计算神经网络的输出。基本思想为：对网络中从前到后的每一个节点计算每一个节点，计算方法与参考文献[1]中相同

算法 2	feedforward 算法
输入	TEST_DATA-训练数据，32×6 的矩阵。前 5 列为输入，后一列为期望输出
输出	output-32×nodes_num 的矩阵。在 32 个训练数据下每个 nodes 的输出
1	初始化 output = 32×nodes_num 的矩阵
2	output[输入层节点] = 训练数据的前 5 列
3	for i=0,i<hidden_out_nodes_num,i++

4	wx = output×weight[第 i 列]
5	output[相应节点] = sigmod(wx)
6	end for
7	return connection, weight

3) 反向传播算法

反向传播算法为经过 adam 方法优化的反向传播算法，算法分为两部分(1)adam 方法[3](2)反向传播算法[2]，伪代码如下：

算法 3	反向传播算法
输入	TEST_DATA-训练数据，32×6 的矩阵。前 5 列为输入，后一列为期望输出 weight-神经网络权重矩阵 connection-神经网络连接矩阵 epochs-最大训练轮次
输出	weight-训练好的神经网络权重矩阵 connection-训练好的神经网络连接矩阵
1	初始化m_0, v_0 为与 weight 结构相同的全零矩阵
2	初始化 $\alpha = 0.1, \beta_1 = 0.5, \beta_2 = 0.999, \varepsilon = 1e-8, \mathrm{t} = 0$
3	While 迭代 epoches 次:
4	old_score = now_score
5	gradient = backprop()
6	t = t+1
7	$m_t = \beta_1 \times m_{t-1} + (1-\beta_1) \times gradient$
8	$v_t = \beta_2 \times v_{t-1} + (1-\beta_2) \times gradient^2$
9	$\widehat{m_t} = m_t/(1-{\beta_1}^t)$
10	$\widehat{v_t} = v_t/(1-{\beta_2}^t)$
	$\mathrm{weight} = \alpha \times \widehat{m_t}/(\varepsilon + \sqrt{\widehat{v_t}})$
11	evaluate()
12	
13	If 正确数量满足要求 or now_score=old_score
14	break
15	end if
16	end while
17	return weight, connection

反向传播算法的数学原理请参见参考文献[2][4]。以下主要为根据[4]中公式，倒序求每个节点的 delta，进而求得输出对边权的偏导。

算法 4	backprop 算法

输入	output-32×node_num 规模的矩阵。为网络中每个节点的输出 TEST_DATA-训练数据，32×6 的矩阵。前 5 列为输入，后一列为期望输出 weight-神经网络边权矩阵 connection-神经网络连接矩阵
输出	deriv_E_w-误差函数对边权的导数矩阵，与 weight 同形
1	初始化 delta = 32×nodes_num 的矩阵，o_net = output×(1-output)
2	delta[输出节点] = 2×(output[输出节点]-TEST_DATA[期望结果])×o_net[输出节点]
3	for i=1,i<hidden_out_nodes_num,i++
4	delta[导数第-1-i 列]=weight[第-1-i 列]· $\mathbf{delta}^T$×o_net[第-1-i 列]
5	end for
6	deriv_E_w=$\mathbf{output}^T \cdot delta \times \mathbf{connection}/32$
7	return deriv_E_w

2.2 演化计算部分

主要包含(1)模拟退火，(2)删除边，(3)删除节点，(4)增加边，(5)增加节点四部分组成。

算法首先按照种群排序为概率依据随机选择一个个体作为亲代，方法与[1]中相同，第(M-j)个个体被选中的概率为

$$p(M-j) = \frac{j}{\sum_{k=1}^{M} k} \quad (3)$$

而后按照模拟退火，删除节点，删除边，增加边，增加节点的顺序得到下一代。经过模拟退火生成的子代与亲代进行比较，如果更优则替换亲代，进行下一轮演化；否则顺序执行后面的方法。之后每经过一个方法得到子代后都与种群中最差的个体比较，如果更优则跳出循环进行下一轮演化；否则顺序执行后面的方法。

以下为详细介绍:

1) 模拟退火

模拟退火方法为全剧搜索算法，可一定程度上优化 BP 算法收敛于局部最优解的情况。是一种最小代价改变网络的方法。经过模拟退火方法的种群个体只能替换本身。

算法 5	模拟退火算法
输入	ann-一个人工神经网络
输出	ann-经过模拟退火运算后的神经网路
1	初始化 offspring={ann}, T=100, cool=0.98
2	while T>1e-8

3	T = T×cool
4	neighbour= offspring[最后一个个体]的权重矩阵加-1~1 的随机数
5	delta = neighbor 的得分 - 亲代个体的得分
6	if delta<0 or [0,1)之间的随机数<exp(-delta/T):
7	offspring=offspring∪neighbour
8	end while
9	对 offspring 集合中的 ANN 按得分从小到大排序
10	return offspring[首个 ann]

2) 删除节点

首先执行节点删除操作是一种启发，优先追求最精简结构的神经网络个体。算法随机在隐藏层节点中寻找一个节点删除，而后对子代网络进行训练，返回。

算法 7	cut_node 算法
输入	ann-一个人工神经网络
输出	ann-经过删除节点运算后的神经网路
1	初始化 offspring = ann, idx=0~隐藏层节点数之间的随机数
2	offspring 的节点数量-1
3	offspring 的权重矩阵和连接矩阵删除编号 idx 对应非输入层节点的行和列
4	对 offspring 进行训练
5	return offspring

3) 删除边

算法以以边权的绝对值占边权绝对值和的比例为概率，随机选择网络中的边进行删除操作，而后对子代网络进行训练，返回。

算法 6	cut_edge 算法
输入	ann-一个人工神经网络
输出	ann-经过边删除边运算后的神经网路
1	初始化 offspring = ann, weight_sum=ann 边权之和, selected_prob=ann 边权绝对值/weight_sum
2	prob = 生成 0~1 之间的随机数矩阵×ann.connection-selected_prob
3	x, y = prob 中数值最大数的行列号
4	设置 offspring 连接矩阵和权重矩阵由 x 指向 y 的边为 False 和 0
5	对 offspring 进行训练
6	return offspring

4) 增加边

该部分与参考文献[1]不同，假设 ann 为当前亲代网络，ann'为与 ann 节点数量相同的全连接神经网络，在增加边操作中，我们顺序遍历亲代神经网络的邻接矩阵。发现

∀edge∈ ann'，edge∉ann，则将该边加入 ann，直到 ann 与 ann'结构相同，下列为代码给出了添加一条边的方法。

算法 8	add_edge 算法
输入	ann-一个人工神经网络
输出	ann-经过增加边运算后的神经网路
1	初始化 offspring = ann
2	for 按存储顺序遍历 ann 的连接矩阵
3	if ∀edge∈ann', edge∉ann:
4	连接节点 x, y，x->y 的边权=-1~1 的随机数
5	break
6	end if
7	end for
8	训练 offspring
9	return offspring

5) 增加节点

增加节点采用节点分裂的方式，即任意选择一个隐藏层节点分裂，分裂后的节点与原节点连接关系完全相同。

算法 9	add_node 算法
输入	ann-一个人工神经网络
输出	ann-经过增加节点运算后的神经网路
1	初始化 offspring = ann, idx=0~隐藏层节点数之间的随机数
2	offspring 的节点数量+1
3	offspring 的权重矩阵和连接矩阵在编号 idx 对应非输入层节点的行和列处插入与编号 idx 对应非输入层节点的行和列相同的向量
4	对 offspring 进行训练
5	return offspring

3. 算法分析

本部分给出算法的运行结果并进行对比分析。而后分析算法的优劣，并给出改进想法。

3.1 实验结果分析

运行结果由三部分组成：(1) adam-BP 算法+随机种群演化，(2) adam-BP 算法，(3)实验结果图示以及对比分析

1) 算法实际运行效果 adam 方法改良的 BP 算法运行效果

表 2 adam-BP 算法+随机种群演化			
运行算法 100 次	**平均**	**最小**	**最大**
节点数量	9	9	9
连接数量	20.99	20	21
演化代数	3.2	0	20
误差函数	1.1092	0.8418	4.3102
运行时间 = 760.97 秒			

2) 无 birds 种群运行效果

由于时间关系，本表数据限定每轮演化运行时间不超过 3 分钟。

表 3 dam-BP 算法			
运行算法 100 次	**平均**	**最小**	**最大**
节点数量	10.31	9	12
连接数量	31.35	20	49
演化代数	76.92	0	663
误差函数	0.8398	0.0032	4.2765
运行时间 = 3213.71			

3) 实验结果图示以及对比分析

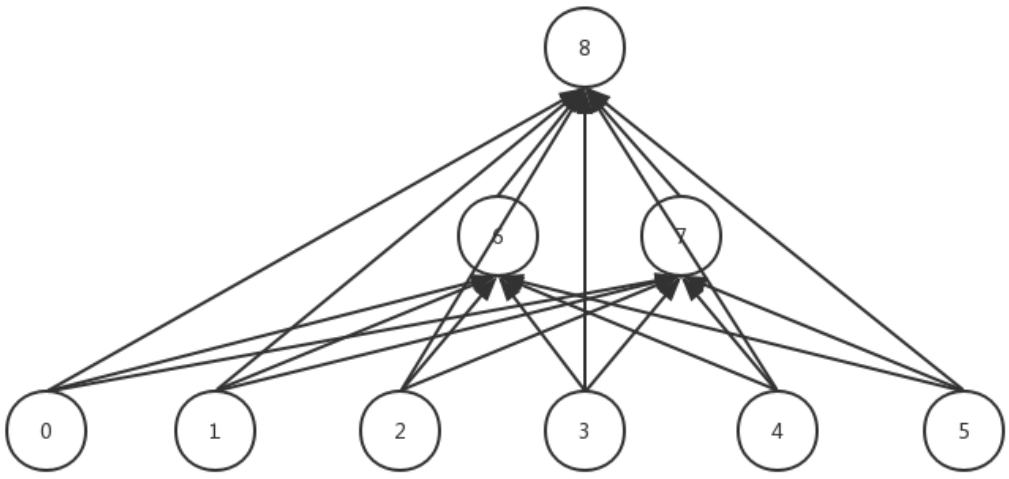

图 1 实验结果图示

表 4 图 1 神经网络节点连接权和 bias								
	Bias	1	2	3	4	5	6	7
6	3.34813837495 91735	-7.6191021037101265	-7.6136741840050055	7.374923858753266	7.375746308129844	7.375805428980396	0.0	0.0
7	-0.9076922692967347	-7.1267693334743811	-7.1304606195507105	7.1211600228874765	7.130284817305559	7.131010601334375	-8.9173047782021734	0.0
8	2.26520956069 19636	1.19557609788 67338	1.20490623744 5644	-1.5800683699293396	-1.5977508246119592	-1.5991381551022676	-3.7800104384185581	10.821016236928099

参考文章[4]以及大量应用已经证明 adam 的有效性，因时间限制本文不在重复证明。

通过表二表三数据对比可知，完全随机种群可以在完全保证输出最优结果的基础上有效的降低程序的运行时间，降低了 3 倍的运行时间。

但是通过数据也可以看出完全随机种群输出的结果误差函数较无完全随机种群的算法大，虽然能够正常输出正确结果，但收敛的不是特别好。有些值仍在 0.5 左右波动恰巧输出正确结果。

3.2 算法分析

- 算法相对于参考文献[1]来说做改动的有三处：

1) 神经网络训练算法由 MBP 改成了 ABP。
2) 添加了随机种群。
3) 演化部分增添边和节点做了修改。

下面给出修改原因，以及继续修改意见：

1) 这点为本算法的优势之一。单纯的根据是否显著降低错误得分函数只能使算法更快速精准的取到局部最优解，但是不能让算法跳出局部最优解限制。并且在梯度下降的多维空间里，每个维度距离最优解的距离是带本维度特性的数值，乘以同一个学习率是非常简单粗暴的方式，因此需要做调整。
2) 这点为本算法的优势之一。对于原解法而言，对于固定结构的神经网络做全局搜索的方式为模拟退火算法。纵使模拟退火能够接受差解，通过一步一步的跳动有全局搜索效果。但是算法受步长限制，仍旧是依赖于上一个点的位置，在步长的空间范围内做搜

索。但是完全随机种群便能突破这一限制，每轮在全局随机搜索。实验结果也说明随机种群有着巨大的贡献。

3) 这一点为本算法的劣势，因为没看懂参考文献[1]中增加节点增加边的具体实现方式，所以做了自己的修改。

- 算法的其他优势：

1) 神经网络中所有的前馈后馈均为矩阵运算，与 for 循环嵌套单个数值计算相比效率提升了 300 倍以上（估计值）。1 次运行在 7 秒左右，能够快速的演化出结果。

- 算法的其他劣势

1) 算法运行中的参数没有经过科学的验证，一部分是从参考文献里拿过来直接用，一部分是凭感觉直接修改。

- 算法的改进：

1) 需要阅读理解文献[4]理解 adam 优化的原理，从而更好地设置梯度下降的参数
2) 需要经过更多的对比试验来设置其他的参数
3) 对于演化部分，删除边和节点的时候当修改成一个亲代繁衍出删除其不同隐藏层节点的子代，再对这些子代进对比行评分，替换种群中比较差的解。而不是随机删除，删除无效就做别的操作。

4. 未来展望

本算法的应用范围较广，给定参数范围可以有效地生成训练演化出用户需要的有效神经网路。但是局限性在于最佳的参数比较难以确定。

5. 参考文献

[1] Xin Yao and Yong Liu. A new evolutionary system for evolving artificial neural networks. IEEE transactions on neural networks, 8(3):694{713, 1997.

[2] 周志华. 机器学习. Qing hua da xue chu ban she, 2016.

[3] https://en.wikipedia.org/wiki/Backpropagation

[4] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.