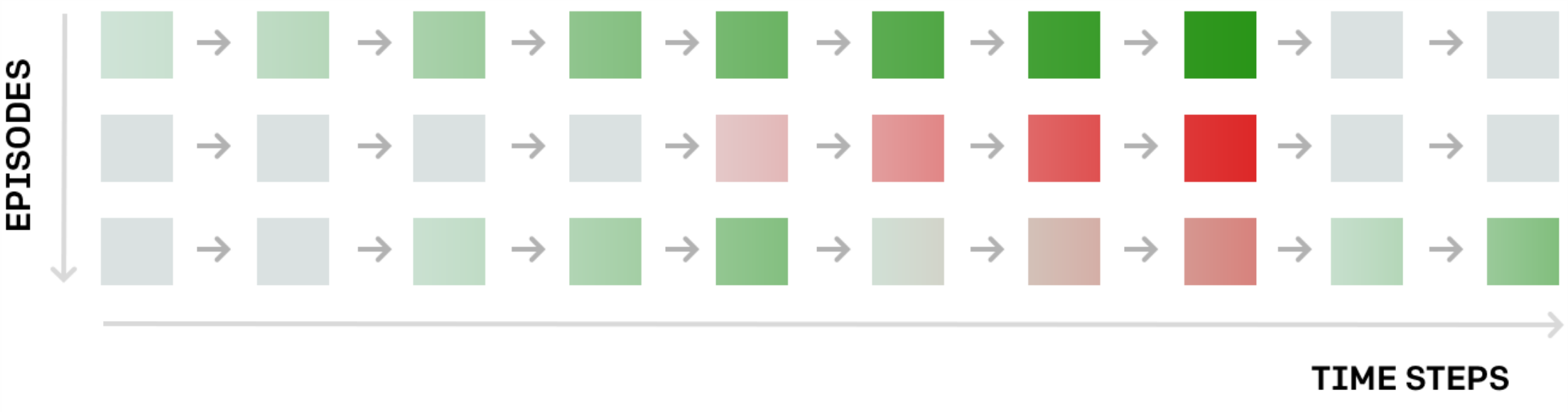
蔡鑫

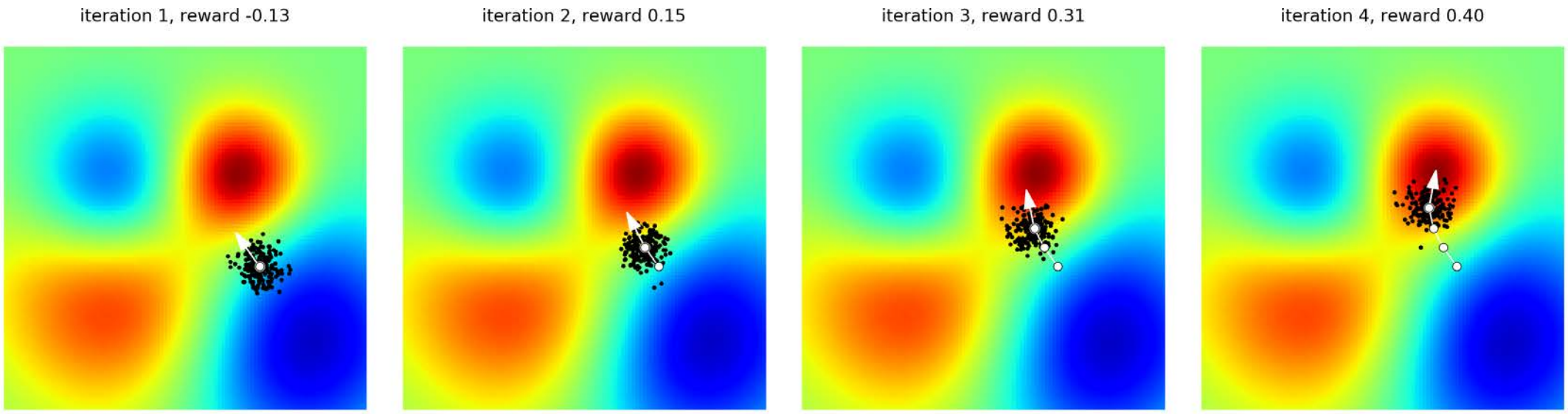
Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, Ilya Sutskever

强化学习在行为的空间搜索



进化策略在产生行为的模型参数空间搜索



Evolution Strategies

Algorithm 1 Evolution Strategies

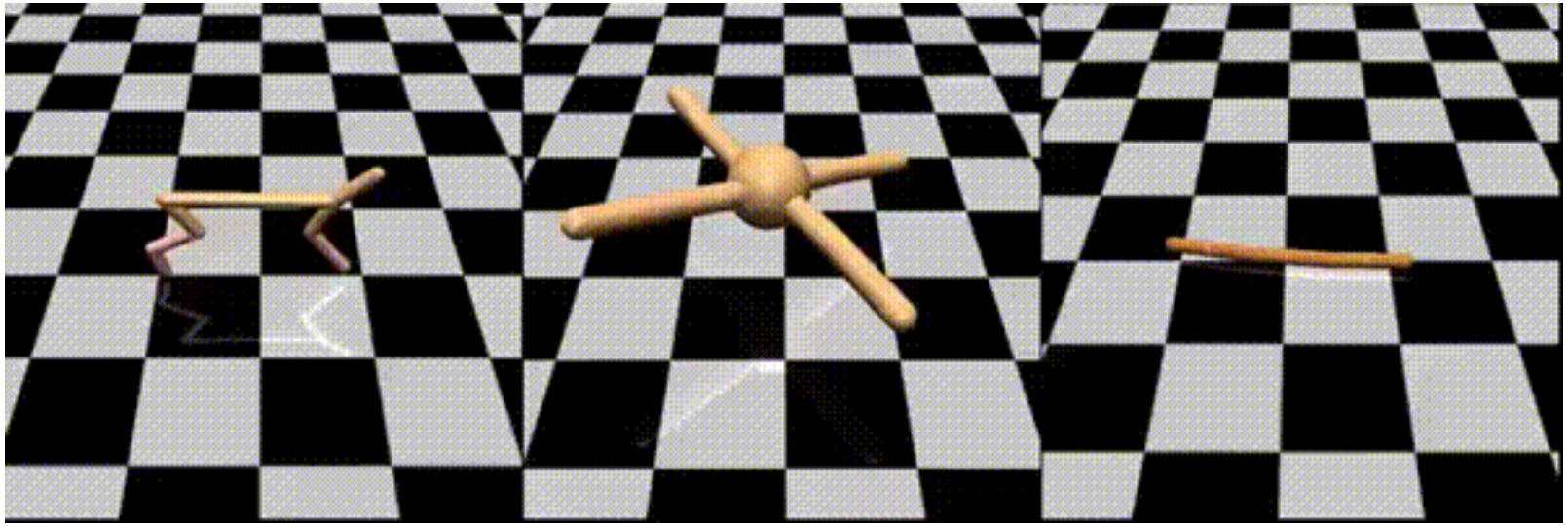
Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
    Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
    Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
    Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
end for

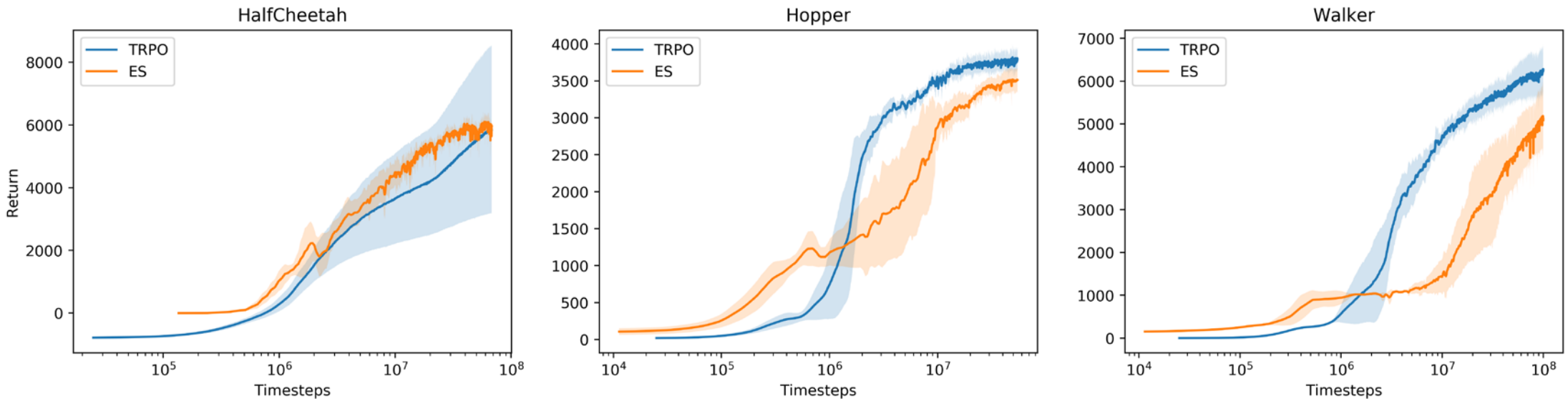
并行化进化策略

Algorithm 2 Parallelized Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
Initialize: n workers with known random seeds, and initial parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  for each worker $i = 1, \ldots, n$ **do**
    Sample $\epsilon_i \sim \mathcal{N}(0, I)$
    Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$
  end for
  Send all scalar returns F_i from each worker to every other worker
  for each worker $i = 1, \ldots, n$ **do**
    Reconstruct all perturbations ϵ_j for $j = 1, \ldots, n$ using known random seeds
    Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$
  end for
end for

MuJoCo 控制任务




HalfCheetah
Hopper
Walker
TRPO
ES
Return
Timesteps
8000
6000
4000
2000
0
4000
3500
3000
2500
2000
1500
1000
500
7000
5000
1000
10^4
10^5
10^6
10^7
10^8

优点

- 不需要反向传播。
- 高度可并行。
- 高度稳健。
- 架构探索。

优点

- 80 台机器和 1440 个 CPU 内核的计算集群上，10 分钟内就训练出一个 3D MuJoCo 人形步行者
- 在 32 核上，A3C 需要大约 10 小时。
- 使用 720 核，Atari 上实现可与 A3C 媲美的表现，同时还能将训练时间从 1 天降低至 1 小时。

缺点

- 在任务有明确的优化函数下，进化策略的收敛速度和效果比Bp要差

储欣

Genetic CNN

ICCV 2017

Introduction

- 本文探索自动学习深度神经网络结构的可能性。

- 假设神经网络具有有限数量的层数，每层定义一组预定义的模块，例如卷积和池化层。即使在这些限制下，可能的网络结构总数随着层数呈指数增长，因此枚举所有候选者并找到最佳候选者是不切实际的。

- 我们将此问题表述为在大型搜索空间中的优化，并应用遗传算法有效地探索空间。

Binary Network Representation

- 考虑那些可以分几个阶段组织的网络结构。在每个阶段，数据大小（宽度，高度和通道数）不变，相邻阶段通过池化操作连接。一个阶段内的所有卷积运算都具有相同数量的filter，即数据通道。
- 网络可以编码成固定长度的二进制字符串。
- 一个网络有S个阶段，在第s个阶段有K_s个节点，用v_{s,k_s}表示，$k_s = 1, 2, \ldots, K_s$。每个阶段中的节点都是有序的，只允许从编号较低的节点连接到编号较高的节点。
- 每个节点对应一个卷积操作，该操作在将所有输入节点(连接到它的编号较低的节点)相加后执行。卷积操作后，相继执行batch normalization和ReLU激活函数
- 每个阶段的二进制字符串长度为：$1 + 2 + \ldots + (K_s - 1) = \frac{1}{2} K_s (K_s - 1)$
- 整个网络的二进制字符串长度：$L = \frac{1}{2} \sum_s K_s (K_s - 1)$

Examples

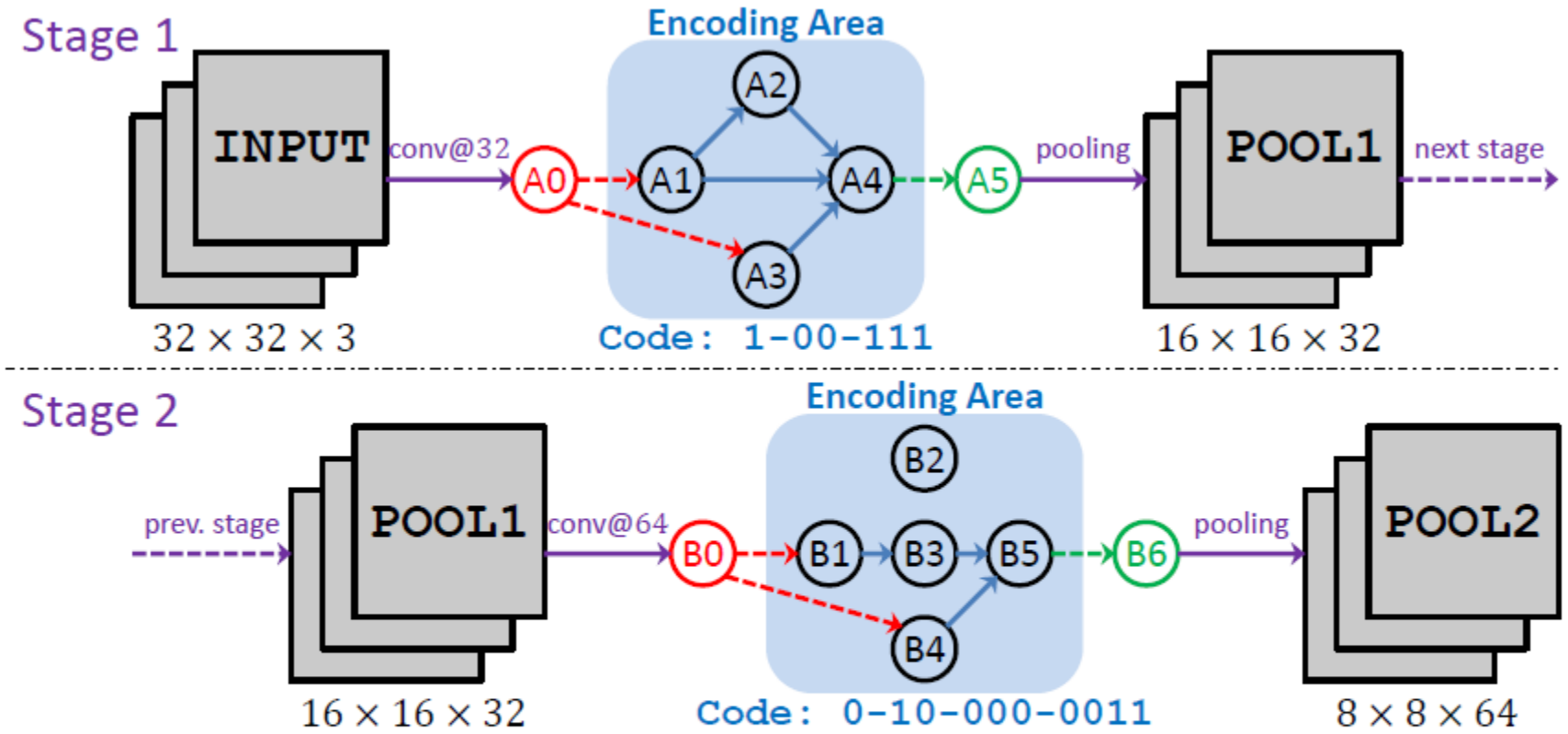


Figure 1. A two-stage network ($S = 2$, $(K_1, K_2) = (4, 5)$) and the encoded binary string (best viewed in color). The default input and output nodes (see Section 3.1.1) and the connections related to these nodes are marked in red and green, respectively. We only encode the connections between the ordinary codes (regions with light blue background). Within each stage, the number of convolutional filters is a constant (32 in Stage 1, 64 in Stage 2), and the spatial resolution remains unchanged (32×32 in Stage 1, 16×16 in Stage 2). Each pooling layer down-samples the data by a factor of 2. ReLU and batch normalization are added after each convolution.

Examples

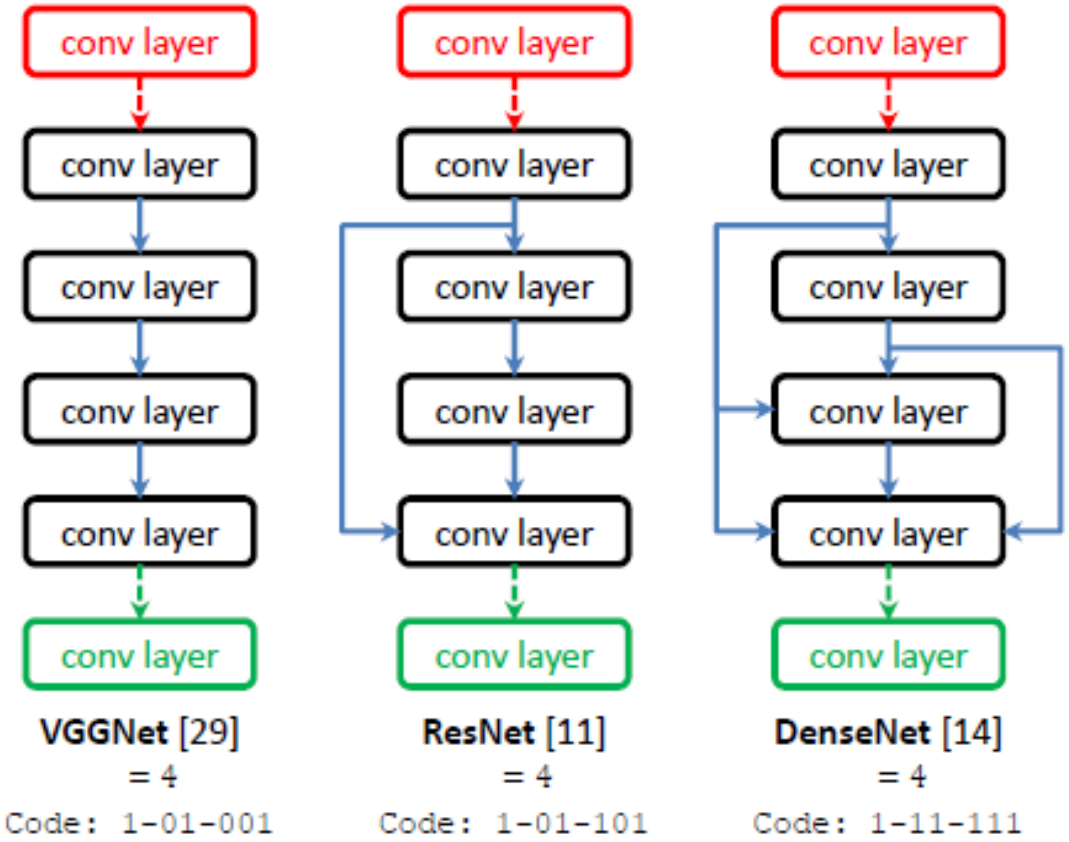


Figure 2. The basic building blocks of **VGGNet** [32], **ResNet** [13] and a variant of **DenseNet** [15] can be encoded as binary strings defined in Section 3.1.

Genetic Operations

- 初始化
  - 初始化N个随机个体
  - 初始化一组随机模型 $\{\mathbb{M}_{0,n}\}_{n=1}^{N}$ 。每个模型都是一个L位的二进制字符串，即，$\mathbb{M}_{0,n} : \mathrm{b}_{0,n} \in \{0,1\}^{L}$。每个个体中的每一个比特都是从伯努利分布中独立采样的。之后，我们评估每个个体以获得其适应度函数值。
  - 不同的初始化策略对遗传性能的影响不大。

Genetic Operations

- 选择
  - 选择过程在每一代的开始执行。
  - 在第t代之前，给每个个体指定一个适应度函数，该适应度值直接影响每个个体在选择过程中存活的概率。
    - 该适应度函数定义为前代或初始化得到的识别准确率$r_{t-1,n}$。
  - 下一代的个体是通过在当前集合上的非均匀采样确定的。每个个体的采样概率和对应的适应度函数值成正比
  - 这意味着最好的个体被选中的概率最大，而最差的个体总是被淘汰。由于个体数量N保持不变，上一代人中的每一个个体都可以被多次选择。

Genetic Operations

- 突变和交叉
  - 个体$M_{t,n}$的突变是指以概率qM独立地翻转每个二进制位。
  - 在实践中，qM通常很小，例如，0.05，所以突变不太可能改变一个个体太多。这是为了保存一个幸存个体的良好特性，同时提供一个尝试新可能性的机会。

  - 交叉过程同时改变两个个体。交叉中的基本单元是一个基因片段，它的动机是需要在每个阶段中保留局部结构。通常交叉概率qC也较小。

Genetic Operations

- 评估
  - 经过上述过程，对每个网络$M_{t,n}$进行评估，得到适应度函数值。
  - 参考数据集D是预先定义的（CIFAR10），我们单独训练每个模型$M_{t,n}$。如果$M_{t,n}$是之前求过值的，我们只需再次求它的值，然后计算所有出现值的平均精度。这种策略至少在一定程度上缓解了训练过程随机性带来的不稳定性。

Algorithm

Algorithm 1 The Genetic Process for Network Design

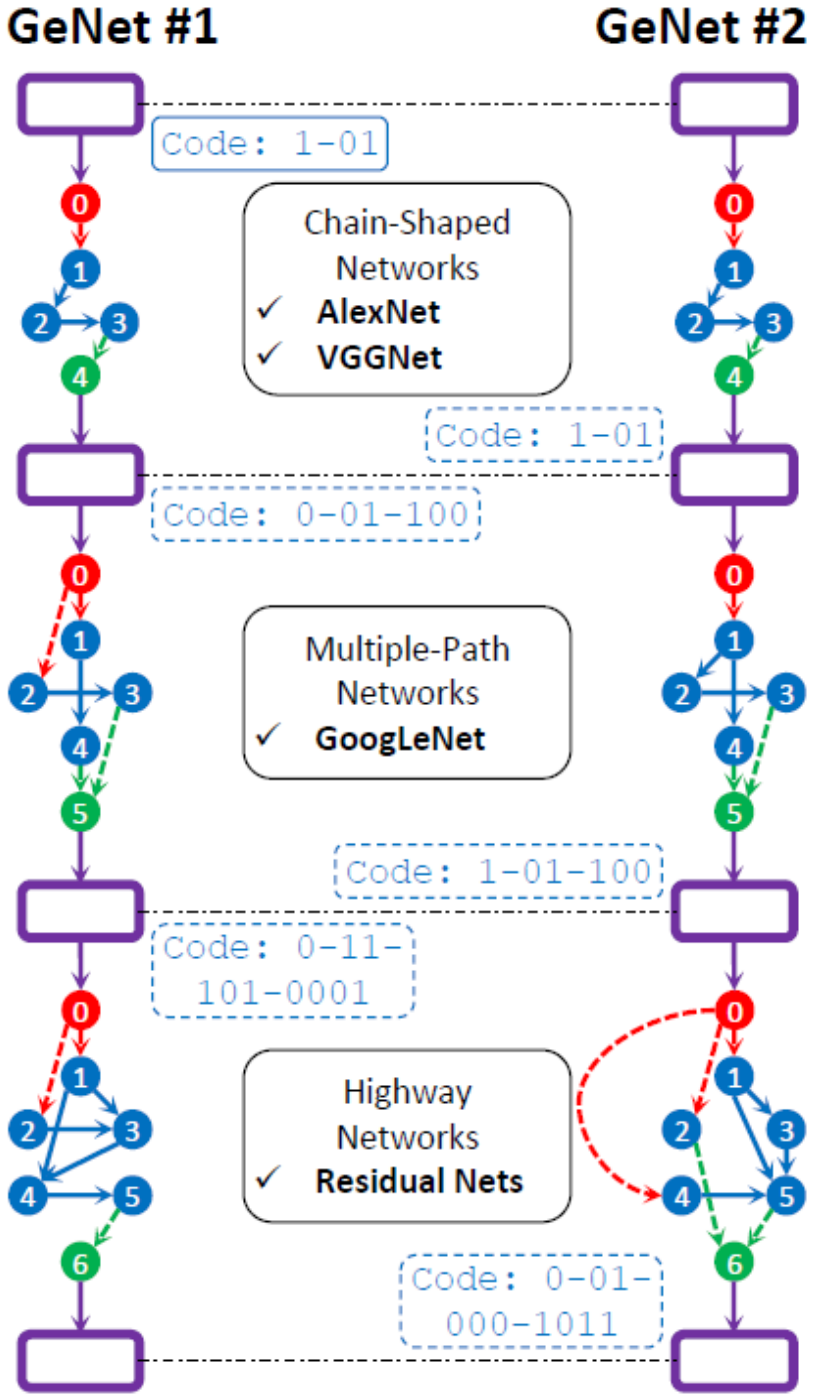
Input: the reference dataset $\mathcal{D}$, the number of generations T, the number of individuals in each generation N, the mutation and crossover probabilities p_{M} and p_{C}, the mutation parameter q_{M}, and the crossover parameter q_{C}.
Initialization: generating a set of randomized individuals $\{\mathbb{M}_{0,n}\}_{n=1}^{N}$, and computing their recognition accuracies;
for $t = 1, 2, \ldots, T$ **do**
    Selection: producing a new generation $\{\mathbb{M}'_{t,n}\}_{n=1}^{N}$ with a Russian roulette process on $\{\mathbb{M}_{t-1,n}\}_{n=1}^{N}$;
    Crossover: for each pair $\{(\mathbb{M}_{t,2n-1}, \mathbb{M}_{t,2n})\}_{n=1}^{\lfloor N/2 \rfloor}$, performing crossover with probability p_{C} and parameter q_{C};
    Mutation: for each non-crossover individual $\{\mathbb{M}_{t,n}\}_{n=1}^{N}$, doing mutation with probability p_{M} and parameter q_{M};
    Evaluation: computing the recognition accuracy for each new individual $\{\mathbb{M}_{t,n}\}_{n=1}^{N}$;
end for
Output: a set of individuals in the final generation $\{\mathbb{M}_{T,n}\}_{n=1}^{N}$ with their recognition accuracies.

Experiments

- 遗传算法需要非常大的计算资源，这使得直接对ILSVRC2012这样的大规模数据集进行评估非常困难。
- 策略是在一个小数据集CIFAR10上探索高效的网络结构，然后将这些结构迁移到大规模数据集中。
- 参数
  - S=3
  - （K1,K2,K3）=（3,4,5）
  - L=19
  - N=20
  - T=50

Experiments

- 右图显示了两个最好的网络学习结果
- 遗传算法能够学习到一些手工设计的结构,如:
  - Chain-Shaped networks
  - Multiple-path networks
  - Highway networks
- 这两个网络由独立的遗传算法获得,但有点相似,这表明遗传过程通常收敛于类似的网络结构。



Experiments

- 小规模数据集迁移实验
  - CIFAR10, CIFAR100和SVHN
- 识别准确率随着遗传过程迭代次数增加而提高
- 虽然准确率低于一些最先进的方法[16][15]，但这些网络要深得多（约40-100层，GeNet#1、GeNet#2只有17层）

	SVHN	CF10	CF100
Zeiler *et.al* [43]	2.80	15.13	42.51
Goodfellow *et.al* [10]	2.47	9.38	38.57
Lin *et.al* [26]	2.35	8.81	35.68
Lee *et.al* [24]	1.92	7.97	34.57
Liang *et.al* [25]	1.77	7.09	31.75
Lee *et.al* [23]	1.69	6.05	32.37
Zagoruyko *et.al* [42]	1.77	5.54	25.52
Xie *et.al* [39]	1.67	5.31	25.01
Huang *et.al* [16]	1.75	5.25	24.98
Huang *et.al* [15]	**1.59**	**3.74**	**19.25**
GeNet after G-00	2.25	8.18	31.46
GeNet after G-05	2.15	7.67	30.17
GeNet after G-20	2.05	7.36	29.63
GeNet #1 (G-50)	1.99	7.19	**29.03**
GeNet #2 (G-50)	**1.97**	**7.10**	29.05
GeNet from WRN [42]	**1.71**	**5.39**	**25.12**

Table 2. Comparison of the recognition error rate (%) with the state-of-the-arts. We apply data augmentation on all these datasets. **GeNet** #1 and **GeNet** #2 are the structures shown in Figure 5.

Experiments

- 大规模数据集迁移实验
  - ILSVRC2012数据集

- 与VGGNet-16和VGGNet-19相比，学习到的模型具有更好的性能

	Top-1	Top-5	# Paras
AlexNet [19]	42.6	19.6	62M
GoogLeNet [36]	34.2	12.9	13M
VGGNet-16 [32]	28.5	9.9	138M
VGGNet-19 [32]	28.7	9.9	144M
GeNet #1	28.12	9.95	156M
GeNet #2	**27.87**	**9.74**	156M

Table 3. Top-1 and top-5 recognition error rates (%) on the **ILSVRC2012** dataset. For all competitors, we report the single-model performance without using any complicated data augmentation in *testing*. These numbers are copied from this page: `http://www.vlfeat.org/matconvnet/pretrained/`. **GeNet** #1 and **GeNet** #2 are the structures shown in Figure 5.

缺点

- 首先,大量的网络结构仍然未被探索,包括一些像Maxout、通道连接这样的新模块，并未将多尺度引入卷积。

- 在目前的工作中,遗传算法只用于探索网络结构,而网络训练过程是分开执行的。可以考虑将遗传算法与网络结构和权重同时结合起来。

董建洲

论文介绍

- Bannister C A, Halcox J P, Currie C J, et al. A genetic programming approach to development of clinical prediction models: A case study in symptomatic cardiovascular disease[J]. PloS one, 2018, 13(9): e0202685.

论文背景

- 遗传编程（GP）是一种能够识别大型数据集中复杂的非线性模式的进化计算方法。尽管GP相对于更典型的，频繁的统计方法方法具有潜在的优势，但其在生存分析中的应用至少是罕见的。本研究的目的是确定GP用于临床预测模型自动开发的效用。

论文方法

- 论文使用来自SMART研究的数据，对心血管风险评分的发展和表现，将GP与常用的Cox回归技术进行了比较，该研究是对症状性心血管疾病患者的前瞻性研究。复合终点是心血管死亡，非致命性中风和心肌梗塞。共有3,873名年龄在19-82岁的患者参加了1996 - 2006年的研究。队列以70:30的比例被分成派生和验证集。推导集用于GP和Cox回归模型的开发。然后使用这些模型预测t = 1,3和5年的离散危险度。两种模型的预测能力根据其风险辨别和使用验证集的校准进行评估。

论文结果

- 这两种模式具有可比性。在时间点t = 1,3和5年时，GP和Cox回归模型的C指数分别为0.59,0.69,0.64和0.66,0.70,0.70。在同一时间点，使用校准曲线评估的两个模型的校准和Hosmer-Lemeshow检验统计量的推广也具有可比性，但Cox模型更好地校准到验证数据。

论文结论

- 使用经验数据，我们证明GP自动开发的预测模型具有与手动调整的Cox回归相当的预测能力。GP模型更复杂，但它是以完全自动化的方式开发的，并且包含较少的协变量。此外，它不需要通常需要的专业知识来推导，从而减轻了知识瓶颈。总体而言，GP作为自动开发用于诊断和预后目的的临床预测模型的方法显示出相当大的潜力。

胡荐苛

主题：神经网络进化(Neuroevolution)

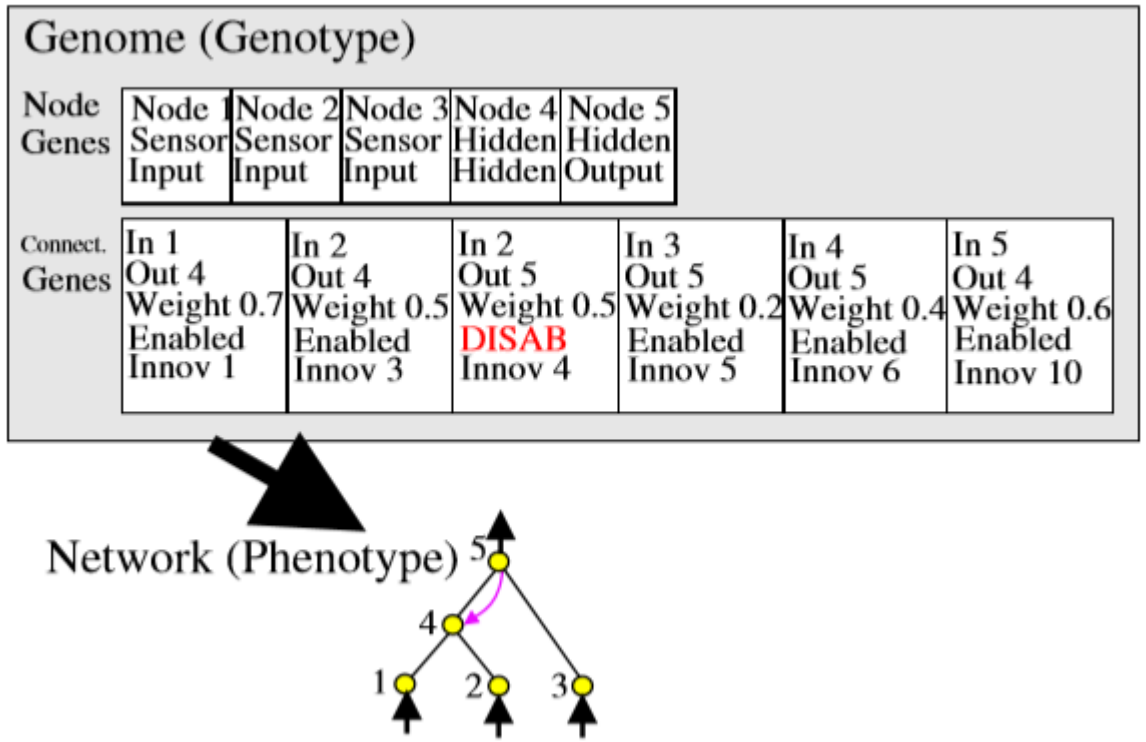
论文：Efficient Evolution of Neural Network Topologies

简介:

神经网络进化可以看做生物进化算法与神经网络应用的结合。而本篇论文中介绍的NEAT算法，通过不断对全连接神经网络尝试变异，改变权重的同时也改变网络拓扑结构，按照“适者生存，不适者淘汰”的定律使网络得到进化。

NEAT算法：

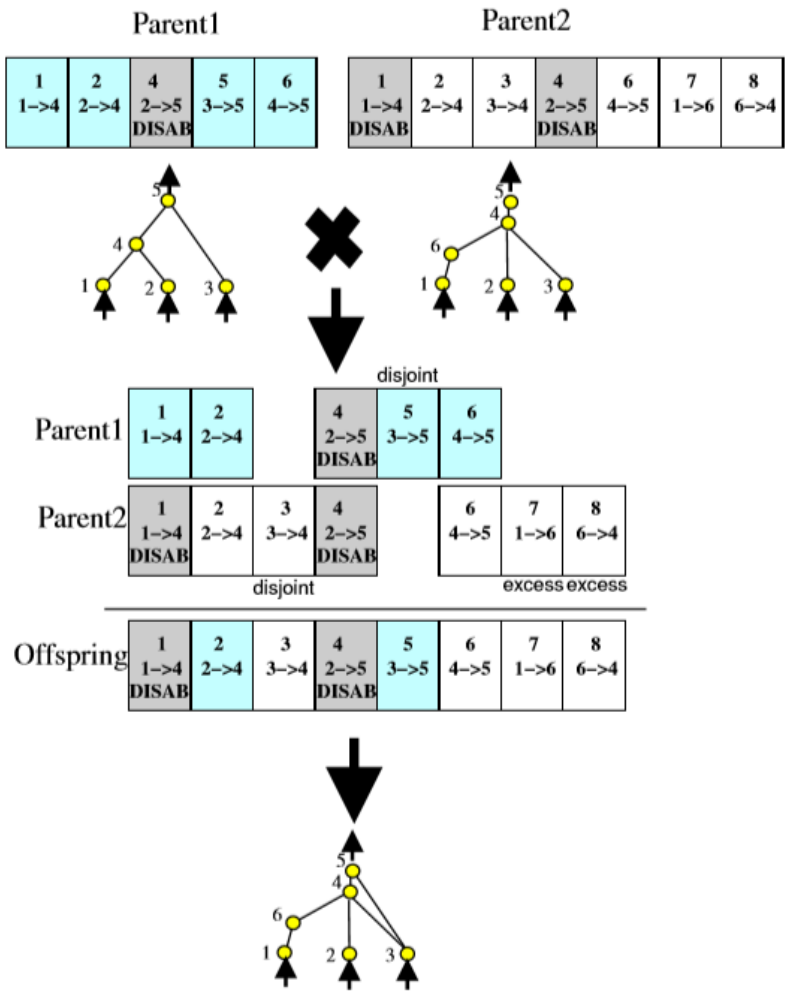
1. 基因编码(coding)



每一个基因配对(Connect. Genes)定义了节点之间的连接形式：从输入节点(In)到输出节点(Out)，链接的权重(Weight)；链接是否被使用(Enabled or DISAB)；唯一的标识号(Innov ID)

NEAT算法：

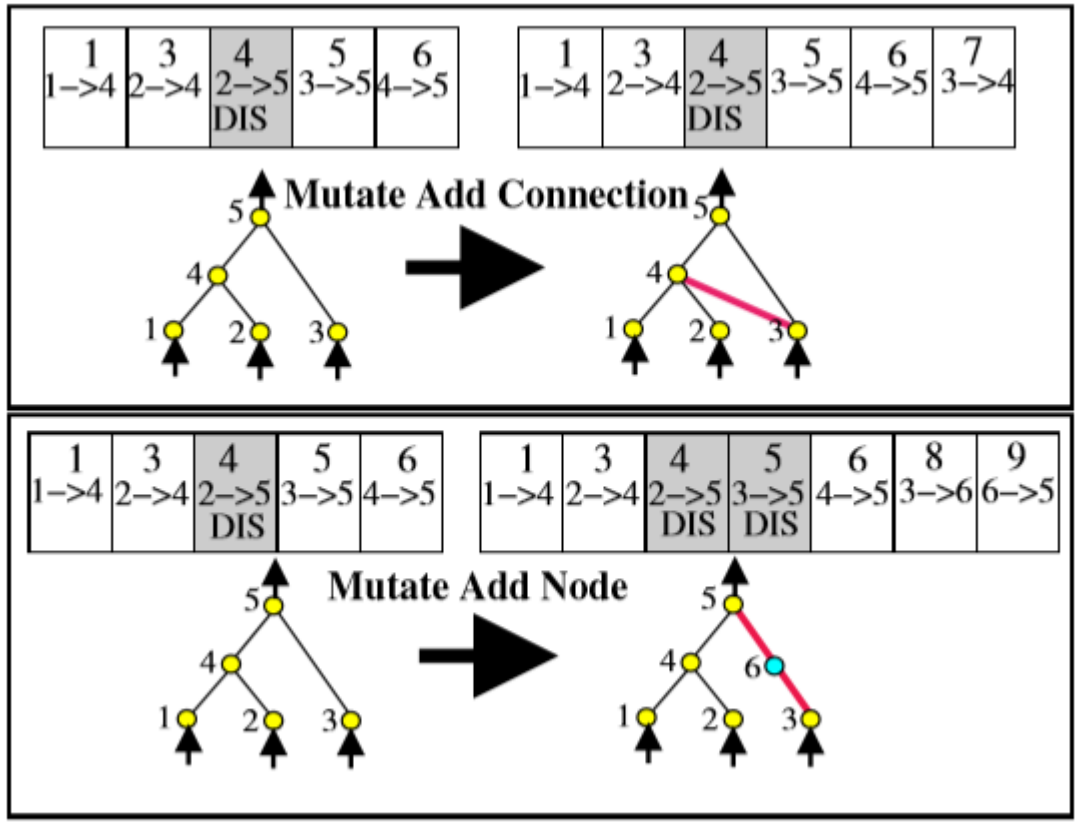
2. 交叉配对(Crossover)



不同的网络拓扑结构配对，保留双方都有的基因对，继承各自独有的基因对。

NEAT算法：

3. 基因突变(Mutation)



NEAT突变不仅改变网络的权重，还会改变网络的结构。结构突变包括新增链接和新增节点两种方式。新增节点是在原有链接上增加新的节点，原有链接将会被disable掉。每一次突变产生的基因Innov ID都会依次递增。

NEAT算法：

4. 物种进化(Innovation)

计算子孙的网络拓扑结构相似度并划分为不同的物种(species)，实现物种的多样性。通过评价函数计算各个物种的适应程度(fitness)，按照适者生存的规则在更适应的物种中继续繁衍实现进化。

NEAT算法贡献：

1. 提出标记的方式实现不同网络拓扑结构的配对
2. 通过划分物种(species)在小范围内繁衍来保留进化成果
3. 从最小网络结构开始进化，只有必要时才会改变结构

合作协同进化算法解决作业调度问题

A Cooperative Coevolutionary Algorithm with Application to Job Shop Scheduling Problem

21821117

贾程皓

Job shop problem （JSP）

- 给定n个作业和m台机器，每项作业由一组有确定进行时间并且要在有限台机器完成的操作（operations）组成。目标是在约束条件下，确定每项operation的起始时间和结束时间，从而优化指标。

- 约束:
  - 一项作业不能两次使用同一台机器
  - 不同作业的operation之间没有偏向性
  - 每台机器同一时间只能处理一个作业

Job shop problem （JSP）

- 数学定义

- c_ik: job i 在machine k上的完成时间
- p_ik: job i 在machine k上的作业时间
- M: 大正数
- a_ihk: 指示系数；如果对于作业i，先在machine h上作业，后再k上作业，则为1，否则为0.
- x_ijk: 指示变量；如果在machine k上，作业i先于作业j，则为1，否则为0.

$$\min \max_{1\le k\le m}\{\max_{1\le i\le n} c_{ik}\} \quad (1)$$

$$s.t. \quad c_{ik} - p_{ik} + M(1-a_{ihk}) \ge c_{ih} \quad (2)$$

$$c_{jk} - c_{ik} + M(1-x_{ijk}) \ge p_{jk} \quad (3)$$

$$c_{ik} \ge 0 \quad (4)$$

$$x_{ijk} = 0 \text{ or } 1 \quad (5)$$

$$a_{ihk} = 0 \text{ or } 1 \quad (6)$$

Algorithm framework

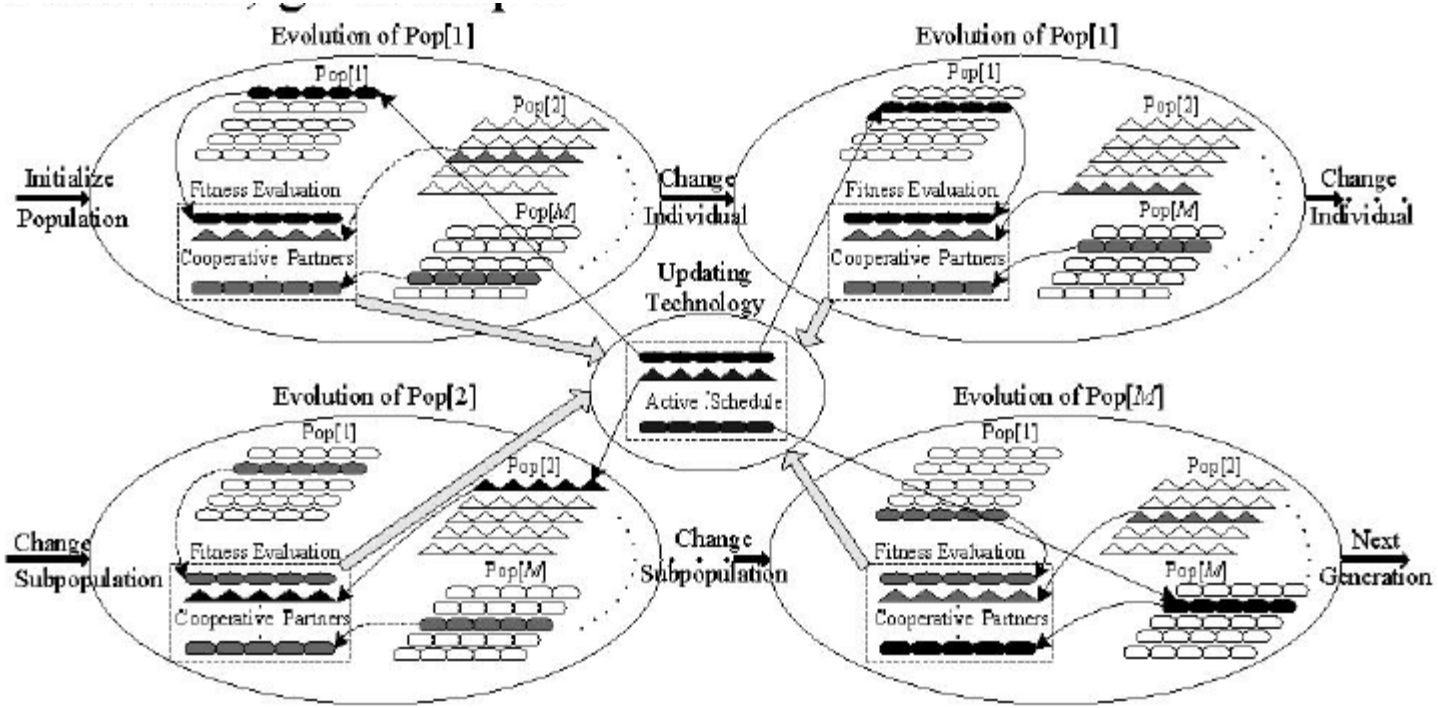


Figure 2. Improved Cooperative Cooevolutionary Algorithm

Algorithm framework

- Step 1：生成M组初始子种群 Pop[m] (m=1,2,…,M)
- Step 2：初始评估每个子种群中每个个体，把最佳适应度复制给f_best。选择随机个体和保持不变的个体作为合作伙伴。
- Step 3：使用更新技术，用对应机器上的最终 operation 序列替换个体。
- Step 4：协同进化， m=1
  - Step 4.1：评估Pop[m]中的第s（s=1,2,…,S）个个体，s=1
  - Step 4.2：基于适应度选择其他父代，应用LOX算子生成子代
  - Step 4.3：应用交叉变异生成子代

Algorithm framework

- Step 4：协同进化
  - Step 4.4：计算子代适应度。选择最佳子代、随机子代和不变子代作为合作伙伴
  - Step 4.5：如果子代适应度高于父代，应用更新技术。否则，保留父代。如果最好子代适应度高于f_best, 更新f_best。
  - Step 4.6：更新s←s+1，如果s<S，转到Step 4.2，否则，转到Step 4.7
  - Step 4.7：更新m←m+1。如果m<M，转到Step 4.1，否则，转到Step 5
- Step 5：若满足终止标准，停止算法，否则，转到Step 4.

改进的双群进化规划算法

- 进化规划 (Evolutionary Programming, EP) 是一种群体全局搜索随机算法。进化规划是一个反复迭代、不断进化的过程。EP的进化操作主要依赖突变[2], 进行突变时需要使用高斯变异算子[3]。而高斯变异算子容易导致早熟收敛现象, 使算法陷入局部极值点。
- 双群进化规划算法 (Bi-group Evolutionary Programming, BEP) , 将种群按一定规则分成2个子群, 其中一个子群使用振荡的高斯变异算子, 以实现子群中的个体能尽可能分散地到达解空间的各个位置, 另一个子群使用递减的高斯变异算子, 以保证子群中的个体在每个局部能找到尽可能好的解。

算法总体思想

- 改进双群进化规划算法将种群划分为2个子群。2个子群使用不同粒度的变异算子。
- 第一个子群对高斯变异算子进行导向性调节，能充分利用函数值信息较快地到达最优解区域。
- 第二个子群采用递减的高斯变异算子，确保子群能够以较高的精度找到局部极值。而种群的重组可实现子群间的信息交流，从而提高解空间的搜索效率。

算法具体步骤

- (1) 参数初始化。设置种群个体数目，随机竞争个体数目，最大进化代数，种群进化代数，进化终止条件。
- (2) 种群初始化。在问题的可行解空间中随机产生u个个体作为初始种群X。
- (3) 终止进化判断。若满足终止条件则停止进化, 输出计算结果, 否则转步骤 (4) 。
- (4) 随机选取u/2个个体组成子群, 其他个体组成种群。
- (5) 变异产生后代。
- (6) 重组组成临时种群X, 计算X中2u个个体的适应度函数值, 使用随机q竞争法则, 选取前u个个体组成新的种群X。
- (7) 转步骤 (3) 。

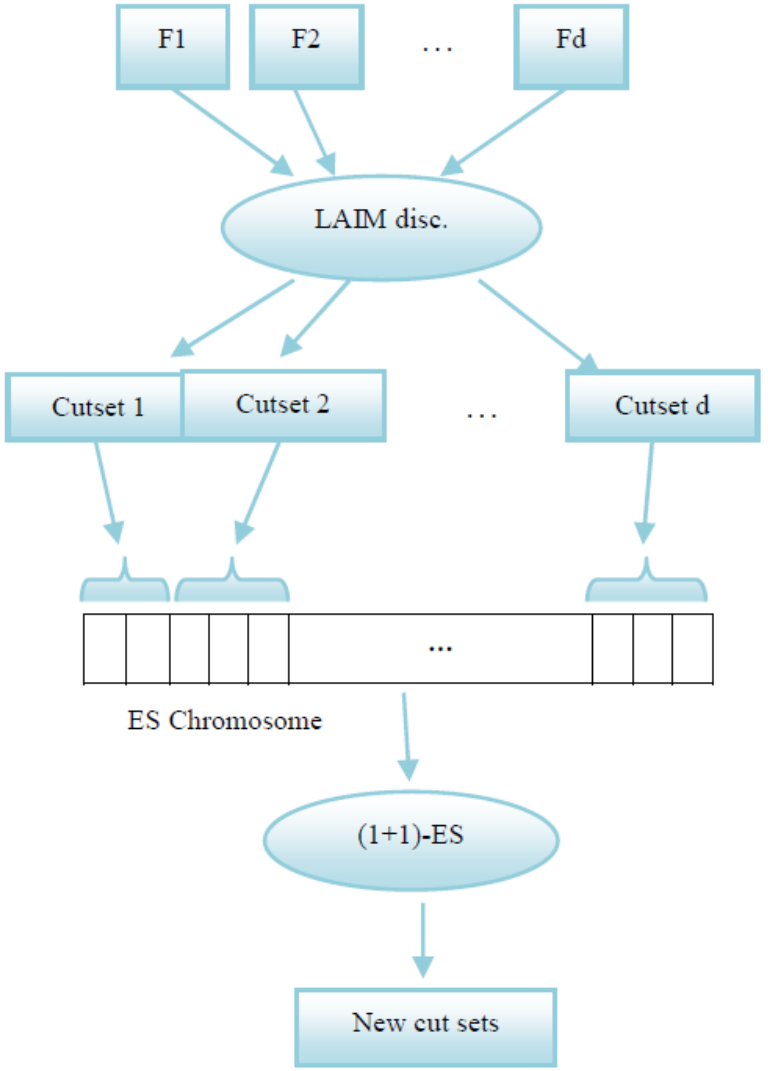
李建新

利用进化策略改进适用于多标签数据的LAIM离散化方法

Improving LAIM discretization method for multilabel data using evolution strategy

CSIEC2018

1. 对多标签数据的每一个特征使用*LAIM discretization*方法得到对应特征的分割点集
2. 将对应的分割点集加入ES染色体，作为进化的父代
3. 运用（1+1）策略，使用高斯变异产生一个新的子代
4. 通过准确率、召回率等比较父代与子代，保存较好的一个



李凯

Optimization of Dynamic Mobile Robot Path Planning based on Evolutionary Methods

概述

- 本研究使用进化算法来优化动态环境下的移动机器人路径规划。这种方法可以避免在选定点的位置以及从开始到目标点的所规划的路径上碰撞到障碍物。 在基于网格的环境中，路径规划在全局上应针对路径的距离，平滑度和安全性进行优化。 在一些路径中，优化的路径规划算法在目标函数和时间上相互比较
- 使用到的进化方法包括遗传算法，模式搜索算法和粒子群优化算法。

问题描述

- 环境表示：使用基于网格的环境来表示具有可行路径和不可行路径的环境
- 目标：在基于网格的环境中找到位置，以便机器人只能在这些位置上改变方向，并最小化起点和目标点之间的目标函数
- 关键点：选择位置不能位于障碍物位置，也不能选择路径穿越障碍物位置
- 环境动态性：障碍物可以在基于网格的环境中移动

问题描述

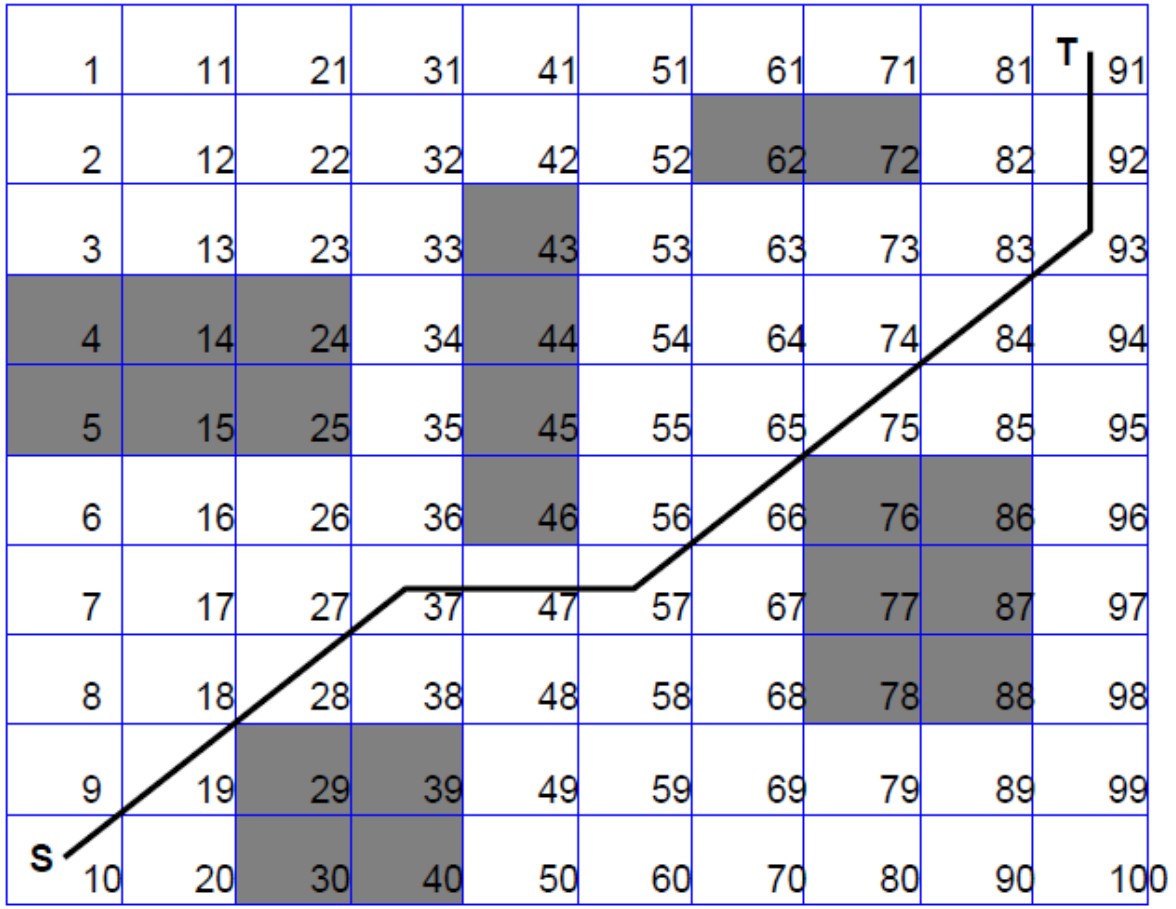


Fig. 1. Grid-based environment

问题描述

- 目标函数：这里我们使用路径距离，平滑度和安全性的组合来优化路径
- 可行路径

$$f_{feas} = \alpha \sum_{i=1}^{n-1} d\left(P_i, P_{i+1}\right) + \beta \sum_{i=1}^{n-1} |\theta_i| + \gamma \ f_{semi} \tag{1}$$

$$f_{semi} = \begin{cases} C, \ if \ \exists \ L_i, |L_i| < k, for \ i = 1, \ldots, n-1 \\ 0, \ otherwise \end{cases} \tag{2}$$

- 不可行路径

$$f_{infeas} = f_{feas} + penalty \tag{3}$$

进化算法

- 模式搜索算法：搜索当前点周围的一组点，以找到使得目标函数值更小的点
- 遗传算法：是一种迭代的随机全局搜索方法
- 粒子群优化算法：用于处理连续和不连续目标函数的优化，是基于群体的搜索方法，每个解决方案在群体中显示为粒子，称为群体。粒子可以在多维搜索空间中改变它们的位置，以便达到它们的平衡和最佳点，直到发生迭代限制。

进化算法

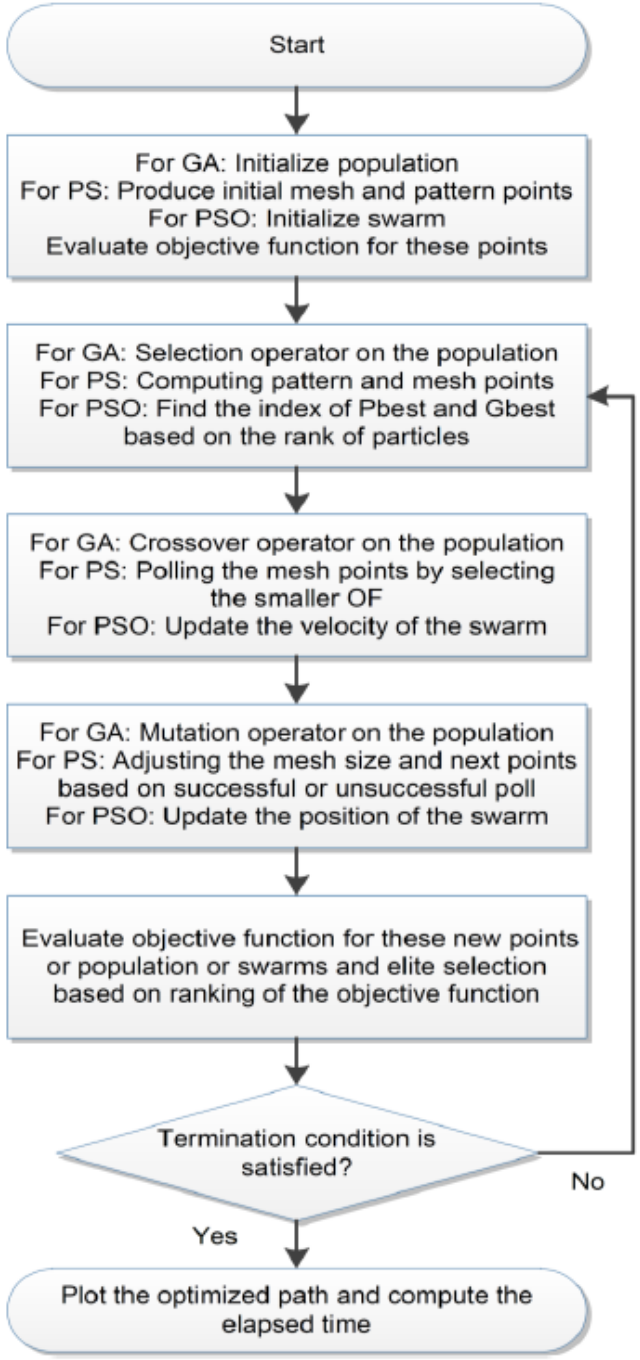
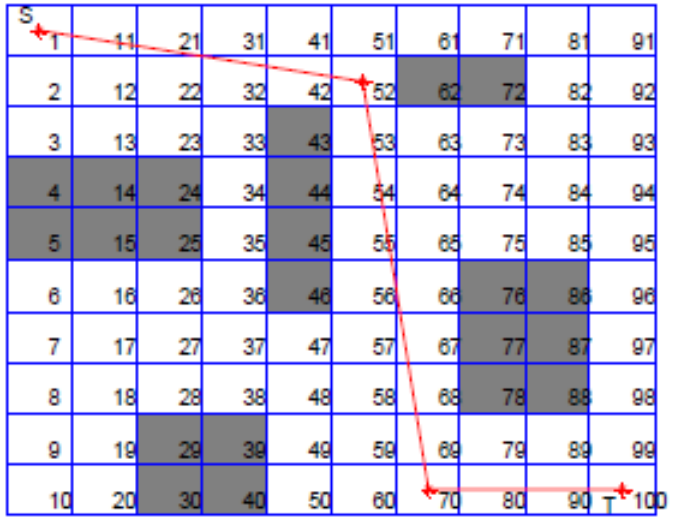


Fig. 3. Flowchart of iterative evolutionary algorithms.

模拟结果与讨论



ig. 4 Path planning using GA in the initial environment

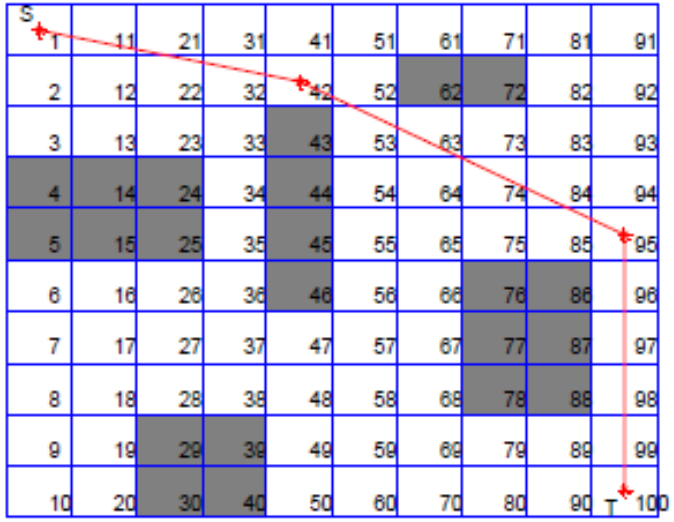
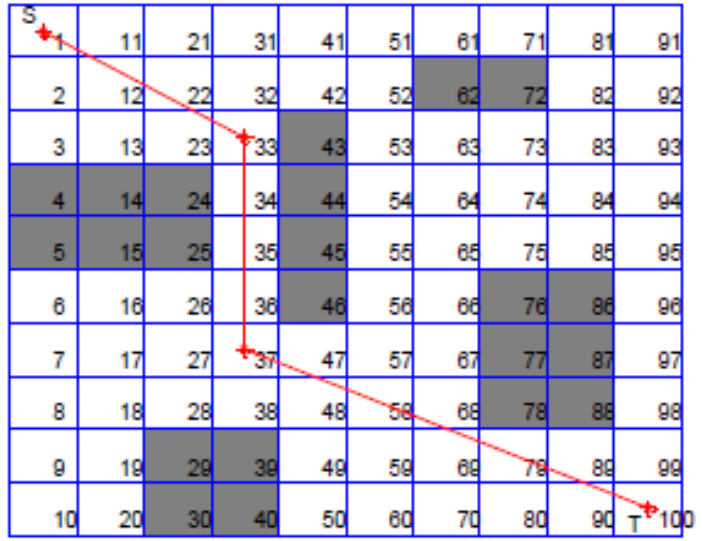


Fig. 5 Path planning using PS in the initial environment



ig. 6 Path planning using PSO in the initial environment

模拟结果与讨论

Table 1. Comparison results in evolutionary methods for mobile robot path planning in the initial and modified environment

Algorithms	Objective function at initial environment	Objective function at modified environment	Time for initial environment (sec)	Time for modified environment (sec)
Genetic Algorithm	17.8051	18.4410	0.9310	1.1180
Pattern Search Algorithm	17.3103	17.7430	0.6800	0.8310
Particle Swarm Optimization	16.9362	17.7001	2.2300	2.5590

林泽培

主题：进化深度网络

论文：EDEN: Evolutionary Deep Networks for Efficient Machine Learning

简介

提出了一种进化深度网络（Evolutionary Deep Network/EDEN），即一种神经进化（neuro-evolutionary）算法。该算法结合了遗传算法和深度神经网络，并可用于探索神经网络架构的搜索空间、与之相关联的超参数和训练迭代所采用的 epoch 数量。

算法目标

用神经进化算法来创造 1D 卷积网络和优化嵌入层以实现情感分析任务

算法大致过程

- 神经网络由 EDEN 染色体编码
- 随机初始化染色体群（population of chromosomes），每一个染色体代表优化问题的候选解
- 基于适应度函数，我们选择多对较优个体（父母）
- 适应度高的个体更易被选中繁殖，即将较优父母的基因传递到下一代。

解决的问题

- 在众多神经网络算法优化问题领域中演化优良的一般性架构和超参数
- 这种演化是否可以在单块 GPU 上完成，不是像以前一样需要在大型计算集群上完成

卢涛

CMA Evolution Strategy

- CMA是一种随机的，不需要计算梯度的数值优化算法。主要用来解决非线性、非凸的优化问题，是最有名，应用最多，性能最好的进化策略算法之一，在中等规模的复杂优化问题上具有很好的效果，是黑箱优化算法中最好的方法之一。

- 黑箱随机优化：考虑一个黑箱搜索情景，想要最小化代价函数，目标是寻找一个或者多个点x,使得函数f(x)尽可能的小。而黑箱搜索，所能提供的信息只有函数f(x),搜索点可以自由的选择。但是同时意味着大的搜索信息量。

黑箱随机优化流程

初始化分布参数θ

迭代代数 g：0,1,2，.....

- 从分布中采样n个独立的点 $P(x/\theta^{(g)}) \rightarrow x_1, \ldots, x_n$
- 利用$f(x)$评估样本 $x_1, \ldots, x_n$
- 更新参数$\theta^{(g+1)} = F_\theta(\theta^{(g)}, (x_1, f(x_1)), \ldots, (x_n, f(x_n)))$
- 中断条件满足，结束

CMA-ES 算法步骤

CMA-ES的核心想法是通过对正态分布 $N(m_t, \sigma_t^2 C_t)$ 中协方差矩阵 C 的调整来处理变量之间的依赖关系和scaling。算法基本可以分成以下三步

- 采样产生新解；
- 计算目标函数值；
- 更新分布参数 m_t, C_t, σ_t

ES算法设计的核心就是如何对这些参数进行调整，尤其是步长参数和协方差矩阵的调整，以达到尽可能好的搜索效果。对这些参数的调整在ES算法的收敛速率方面有非常重要的影响。**CMA-ES调整参数的基本思路是，调整参数使得产生好解的概率逐渐增大（沿好的搜索方向进行搜索的概率增大）。**

CMA-ES 算法步骤

- **采样产生新解**

在CMA-ES中，每一次迭代从 $N(m_t, \sigma_t^2 C_t)$ 中产生 λ 个解(一般 λ 取值为log(n)的量级)

$$x_i = m_t + \sigma_t y_i,\ y_i \sim N(0, C_t)$$

这里的每一个 $y_i \in R^n$ 是一个搜索方向。一般的， y 可以通过协方差矩阵C的特征分解 $C_t = BD^2B^T$ 或者Cholesky 分解由标准正态分布得到，即

$$y_i = BDz_i,\ z_i \sim N(0, I).$$

CMA-ES 算法步骤

- **计算目标函数值**

对新产生的解计算对应的目标函数值 $f(x_i)$ ，并对这些目标函数值排序

$$f(x_{1:\lambda}) \leq f(x_{2:\lambda}) \leq \cdots \leq f(x_{\lambda:\lambda})$$

其中的下标表示在这些样本中排第i位。对于截断选择来说，取前 $\mu = \lfloor \frac{\lambda}{2} \rfloor$ 个解用于更新分布参数。还存在其他的选择方式，这里为方便使用截断选择。值的注意的是，这里的顺序只依赖于目标函数值的比较，而不依赖于目标函数值本身，即comparison based，或者说 objective value-free。

CMA-ES 算法步骤

- **分布参数更新**
  - **均值**

分布的均值即为所选择的 μ 个解的加权的最大似然估计

$$m_{t+1} = \sum_{i=1}^{\mu} w_i x_{i:\lambda} = m_t + \sum_{i=1}^{\mu} w_i (x_{i:\lambda} - m_t)$$

此式表明，均值沿着平均搜索方向移动一步.

CMA-ES 算法步骤

- 分布参数更新
  - 搜索路径

$$p_{t+1} = (1-c)p_t + \sqrt{c(2-c)}\sqrt{\mu_w}\frac{m_{t+1}-m_t}{\sigma_t}$$

它描述了分布均值的移动，并且将每次迭代中移动方向做加权平均，使得这些方向中相反的方向分量相互抵消，相同的分量则进行叠加。这类似于神经网络优化中常用的Momentum，加快收敛速度

CMA-ES 算法步骤

- 分布参数更新
  - 协方差矩阵

协方差矩阵的更新的原理是

$$argmax\ p(p_{t+1}|m,C),\ \ argmax\ \prod_{i=1}^{\mu} p\left(\frac{x_{i:\lambda}-m_t}{\sigma_t}\middle|m,C\right).$$

更新原理是增大沿成功搜索方向的方差，即增大沿这些方向采样的概率

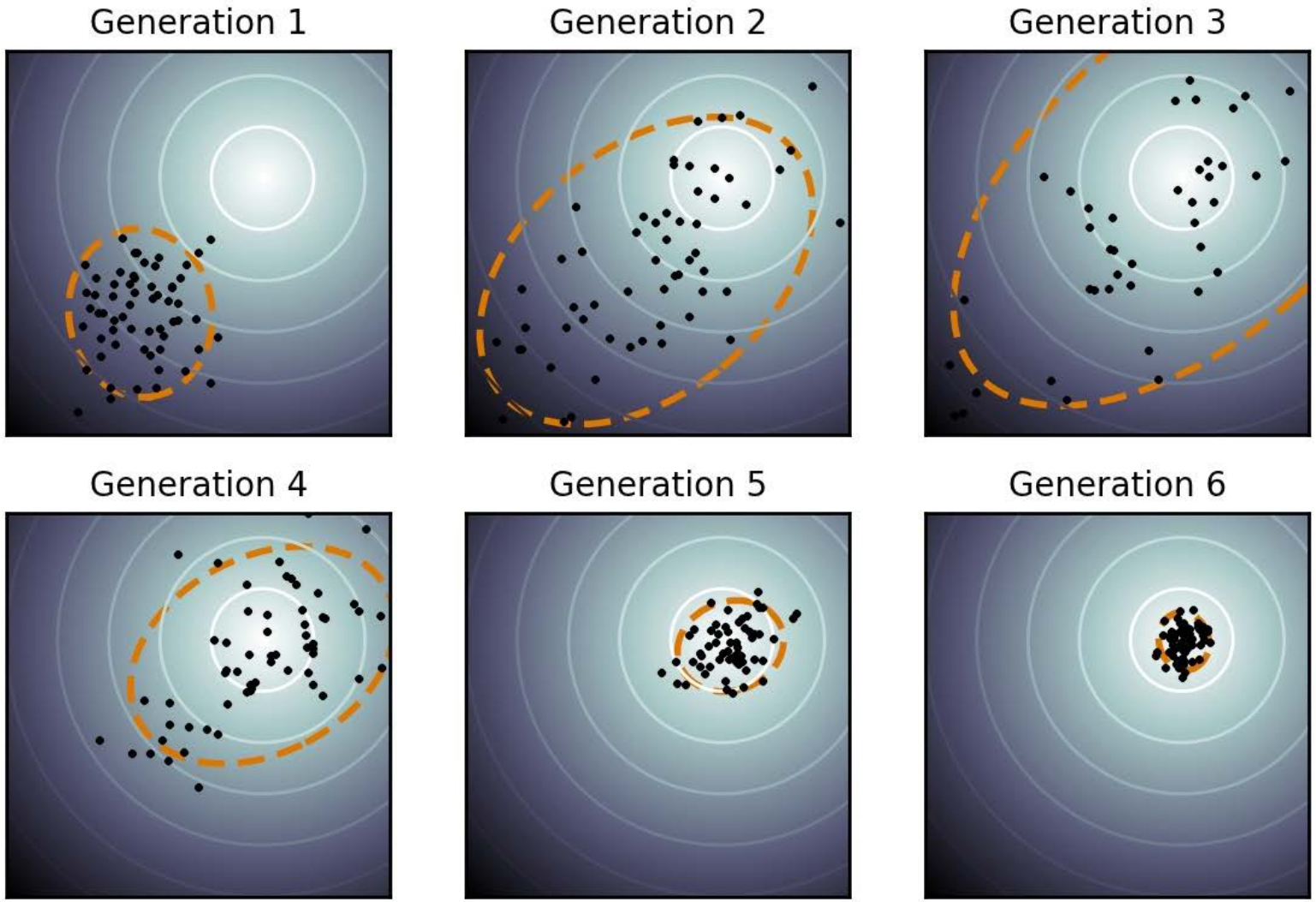
CMA-ES 算法步骤

- **分布参数更新**
  - **步长更新**

CMA-ES默认使用累积式步长调整(Cumulative step size adaptation， CSA) 。CSA是当前最成功、用的最多的步长调整方式。CSA的原理可以从方面来理解：**相继搜索的方向应该是共轭的。**

- 如果相继的搜索方向之间正相关（夹角小于pi/2），那么表明步长太小，应该增大；
- 如果相继搜索方向之间是负相关的，那么表明步长太大，应该减小。

CMA-ES 搜索过程


Generation 1
Generation 2
Generation 3
Generation 4
Generation 5
Generation 6

CMA-ES 算法的基本特点：

- 无梯度优化，不使用梯度信息
- 局部搜索中无梯度算法通常比梯度算法慢
- 在复杂优化问题non-separable, ill-conditioned, or rugged/multi-modal 上表现良好

马宁

- Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity Inspired by Network

- **ieda:**
- Biological neural networks have good properties(e.g. sparsity, scale-freeness)
- Artificial neural networks should not have fully-connected layers
- Propose sparse evolutionary training of artificial neural networks

SET(Sparse Evolutionary Training)

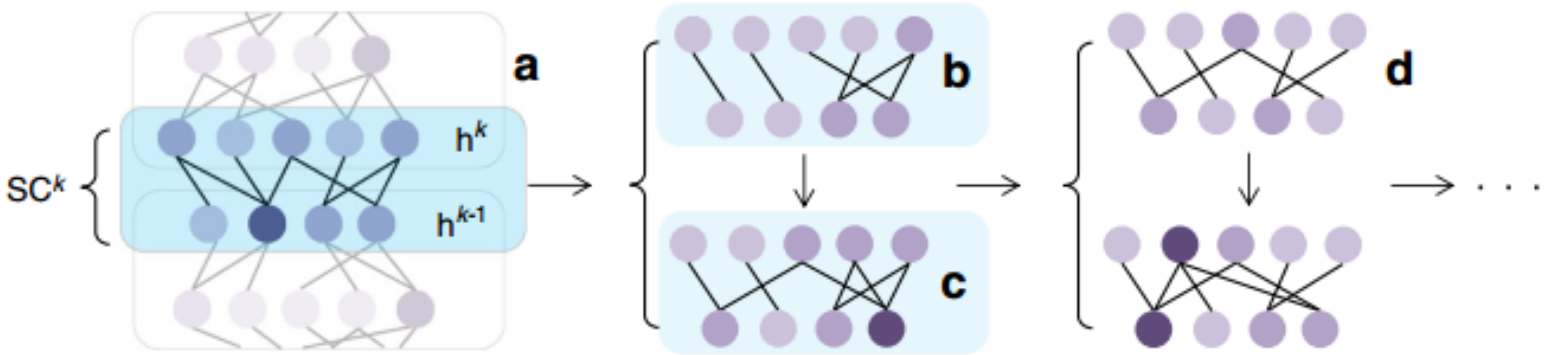
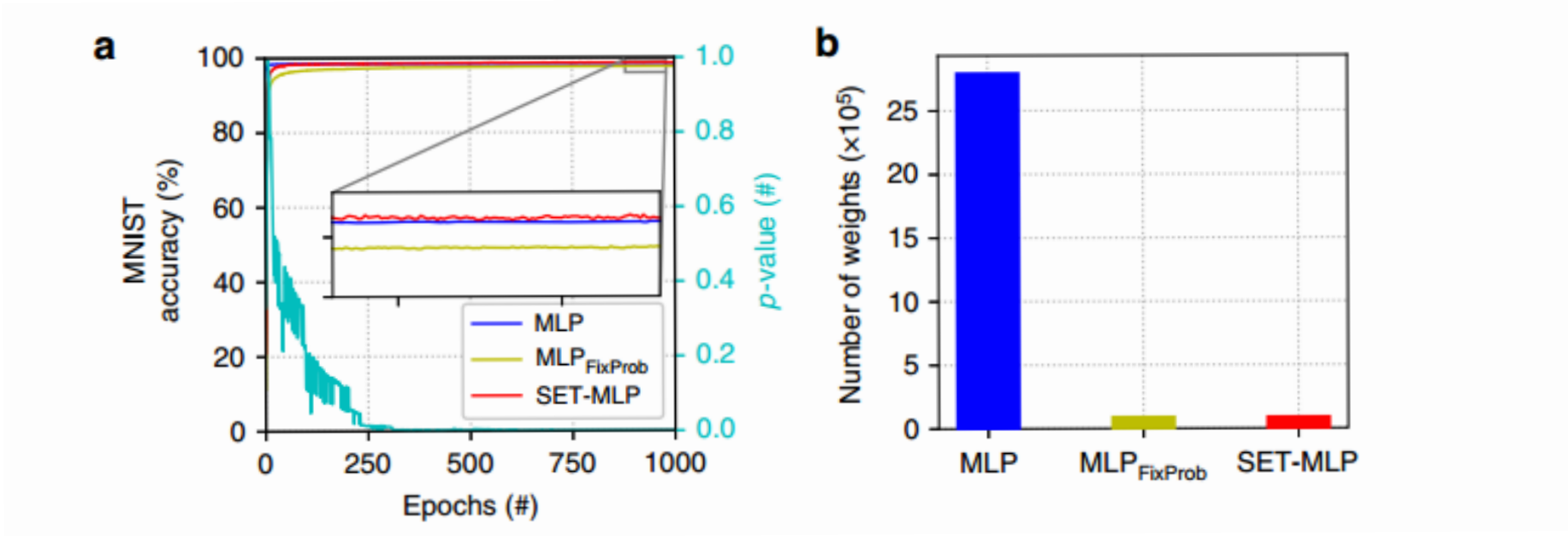
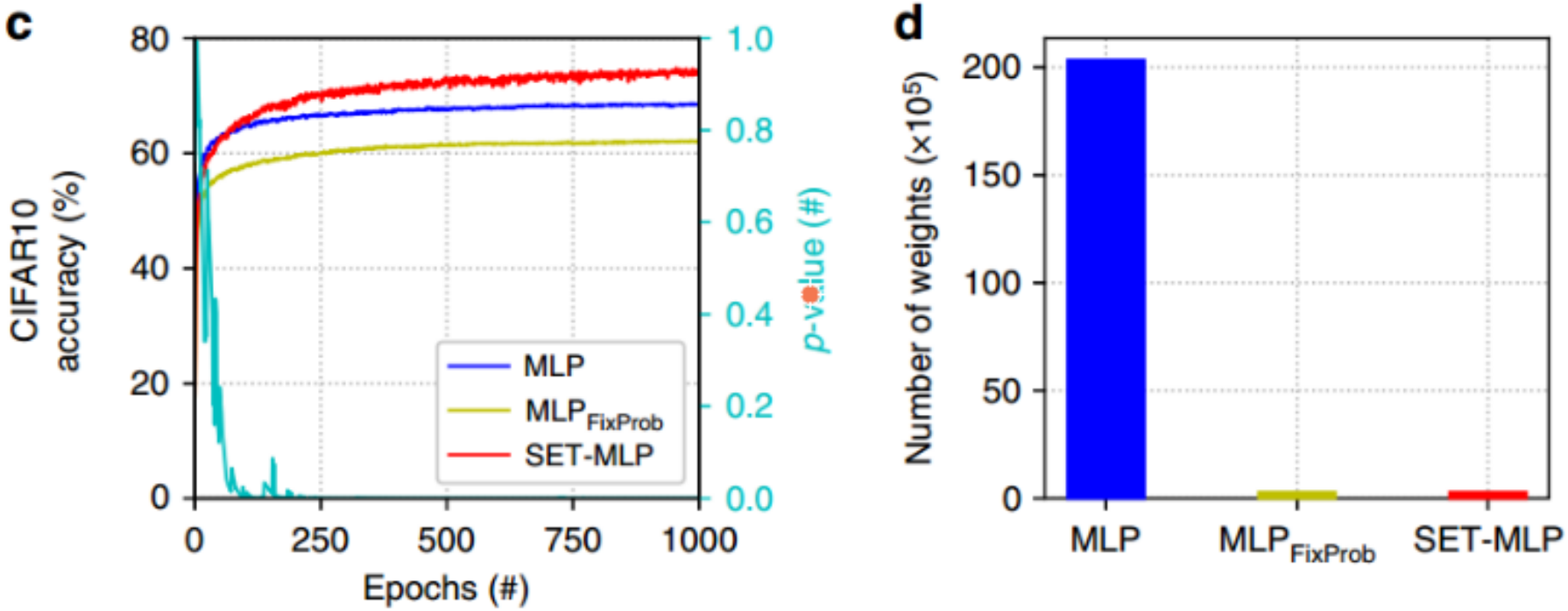


Fig. 1 An illustration of the SET procedure. For each sparse connected layer, SC^k (**a**), of an ANN at the end of a training epoch a fraction of the weights, the ones closest to zero, are removed (**b**). Then, new weighs are added randomly in the same amount as the ones previously removed (**c**). Further on, a new training epoch is performed (**d**), and the procedure to remove and add weights is repeated. The process continues for a finite number of training epochs, as usual in the ANNs training

Performance on MLPs



Performance on MLPs



Performance on CNNs

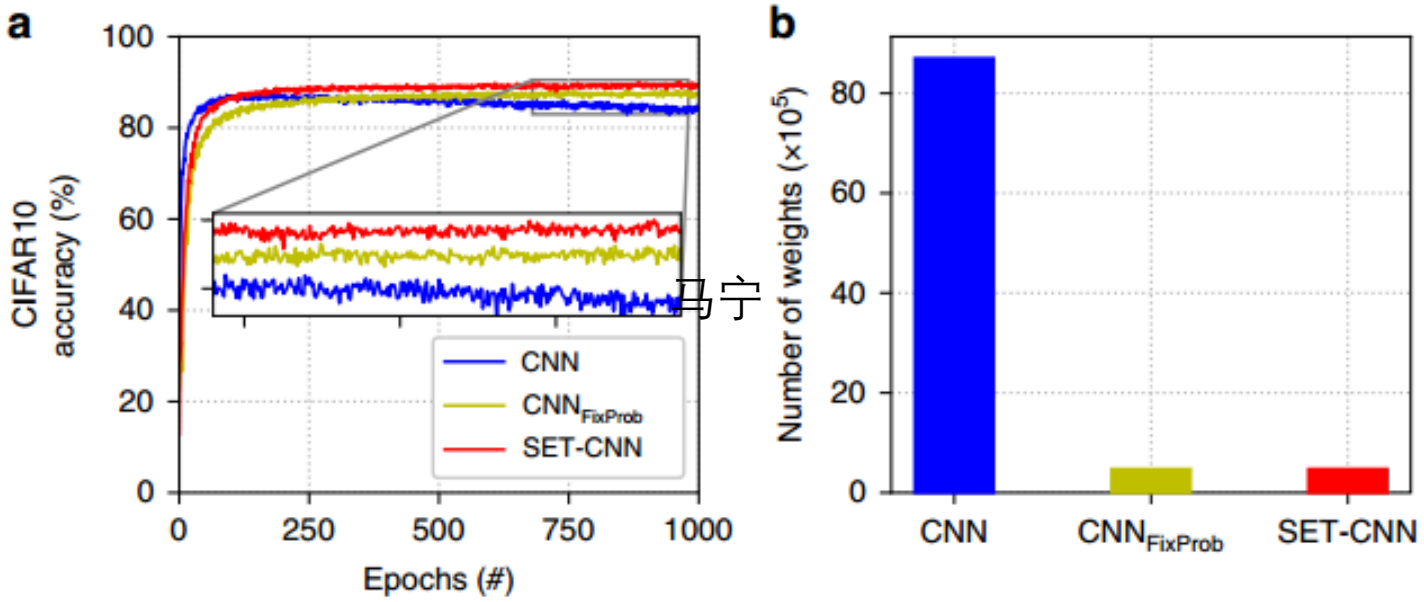


Fig. 7 Experiments with CNN variants on the CIFAR10 dataset. **a** Models performance in terms of classification accuracy (left *y* axes) over training epochs (*x* axes). **b** The number of weights of the three models on each dataset. The convolutional layers of each model have in total 287,008 weights, while the fully connected (or the sparse) layers on top have 8,413,194, 184.842, and 184,842 weights for CNN, $CNN_{FixProb}$, and SET-CNN, respectively

陈亚博

Limited Evaluation Cooperative Co-evolutionary Differential Evolution for Large-scale Neuroevolution

该论文中提出用“有限评估合作协同差分进化”的方法来优化高纬度的人工神经网络，其进化策略主要涉及以下几个算法：

1、（Limited Evaluation/LE）LE/LEEA算法：神经进化的fitness evaluation计算量很大，因此该算法进行计算量的优。

2、（Cooperative Co-evolutionary/CC）合作协同进化算法：用启发式的方法分解高维度的神经网络，分解成一个个较小的子问题。

3、（Differential Evolution/DE）差分进化算法：用于求解多维空间中整体最优解

两个主要问题：

1、结合使用CC算法的DE算法是否比标准的DE算法准确度更高？

2、在不降低神经网络准确性的前提下，结合使用LE算法的DE算法是否减少了运行的时间？

提出的对比实验：对DE、LEDE、CCDE、LECCDE四种算法在三个不同的数据集上的优化结果进行对比。

叶茂鑫

Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science

论文结构

- 1 INTRODUCTION
- 2 RELATED WORK
- 3 MTHOD
- 4 RESULTS
- 5 DISCUSSION
- 6 CONCLUSIONS
- 7 ACKNOWLEDGMENTS
- 8 REFERENCES

INTRODUCTION and RELATED WORK

- 随着数据科学领域的不断发展，对非专家可以访问机器学习的工具的需求将不断增加。在本文中，作者介绍了基于树的管道优化的概念，以自动化机器学习 - 管道设计中最繁琐的部分之一。作者在Python中实现了一个基于树的开源管道优化工具（TPOT），并展示了它在一系列模拟和真实世界基准数据集上的有效性。

Method

- 在这里，列出了当前在基于树的管道优化工具（TPOT）中实现的四种主要类型的管道运算符。所有管道运营商都在scikit-learn中使用现有的实现。有关这些运算符的进一步阅读，请参阅scikit-learn在线文档和。
- 预处理器。实现了一个标准缩放运算符，它使用样本均值和方差来扩展特征（StandardScaler），一个强大的缩放运算符，它使用样本中值和四分位间范围来扩展特征（RobustScaler），以及一个生成交互特征的运算符 通过数值特征的多项式组合（多项式特征）。
- 分解。实现了RandomizedPCA，一种使用随机奇异值分解（SVD）的主成分分析变体。
- .特征选择。我们实现了一个递归特征消除策略（RFE），一个选择前k个特征的策略（SelectKBest），一个选择前n个特征百分位（SelectPercentile）的策略，以及一个删除不满足最小方差的特征的策略 阈值（VarianceThreshold）。
- 楷模。在本文中，我们专注于监督学习模型。我们实现了基于个体和集合树的模型（DecisionTreeClassifier，RandomForestClassifier和GradientBoostingClassifier），非概率和概率线性模型（SVM和LogisticRegression），以及k-最近邻（KNeighborsClassifier）。

RESULTS 、DISCUSSION and CONCLUSIONS

- 对TPOT的分类性能与两个对照的性能进行比较
- 总结发现在许多情况下，自动化机器学习管道设计和优化可以提供对基本机器学习分析的显着改进，同时几乎不需要输入或用户的先验知识但是，重要的是要注意自动化管道设计的目标不是取代数据科学家，也不是机器学习从业者。 相反，我们的目标是基于树的管道优化工具（TPOT）成为“数据科学助手”，它探索数据，发现数据中的新特征，并向用户推荐管道。

应升宇

多级彩色图像分割的差分进化与细胞差分进化研究

Bouteldja, Mohamed Abdou, and Mohamed Batouche. "A study on differential evolution and cellular differential evolution for multilevel color image segmentation." *2017 Intelligent Systems and Computer Vision (ISCV)*. IEEE, 2017.

一、论文介绍

图像分割是将图像分割成多个区域的过程，并且广泛用于分离和搜索给定图像内的特定对象。多级阈值处理是用于分割图像的最重要技术之一。然而，选择最佳阈值仍然是一个具有挑战性的问题。由于进化算法在十年中被应用于文献中以提高多级阈值方法的准确性和计算效率，因此在许多情况下，这种随机算法可能过早地收敛到局部最优并且缺乏准确性和稳定性。结构化进化算法的提出就是来克服这些限制。因此，这篇论文的研究工作的主要目的是提出一种基于细胞差分进化的多级阈值分割方法，用于彩色图像分割。本文中提出了一项比较性能研究，以评估标准DE和Cellular DE的效率和准确性，还展示了这项比较研究的结果，该研究表明DE中的群体结构在准确性，稳健性和速度收敛方面提高了其性能，还通过使用众所周知的基准图像将其与其他三种算法进行比较来评估所提出的算法。

二、进化算法在图像分割中的应用

图像分割包括将图像分成多个部分。这些部分描述了一些更有意义且更易于分析的结构。分割广泛用于各种领域，例如计算机视觉，对象识别，图像压缩等。存在各种方法来执行图像分割：其中最常用的是阈值处理。阈值处理方法使用从图像定义的一些特征来选择一组阈值。为了确定阈值，很多方法将阈值处理作为优化问题，通过最大化或最小化某些目标函数，例如类间方差，熵和交叉熵。根据阈值的数量，有双层阈值和多级阈值。大多数双层阈值处理方法可以很容易地扩展到多级阈值处理。然而，当阈值数量增加时，这些方法非常耗时。这些经典方法的另一种解决方案是进化算法，它们受到许多研究人员的极大关注，以解决多级阈值问题。在这类算法中，我们发现了遗传算法（GA），遗传编程（GP），进化规划（EP），进化策略（ES）和差分进化（DE）。为了在优化该过程，文中对进化算法引入了一些改进，其中包括使用了结构化的进化算法，群体以某种方式分散，在许多情况下，这些分散算法提供了更好的搜索空间采样，相对于标准版本中的等效算法，导致数值行为得到改善。

在这项工作中，这篇论文提出了一种基于 Cellular DE的多级阈值处理方法，用于分割彩色图像。这种DE使用邻域的概念，即个体可能只在繁殖循环中与其附近的邻居交互。 Cellular DE的重叠小邻域有助于探索搜索空间，因为解决方案通过群体的诱导缓慢扩散提供了一种探索，而探索通过遗传操作在每个邻域内发生。

基于作为目标函数的类间方差，将所提出的Cellular DE的性能与DE的标准版本进行比较。实验结果表明，Cellular DE 具有优于标准 DE 的优点。

三、 基于类别方差的多层阈值

在多级阈值处理问题中，其目的是根据诸如类间方差和熵之类的一些区分标准以最佳方式分离图像的灰度级区域。 在类间方差的情况下，目标是选择最大化这种目标函数的阈值。在本文中，考虑了彩色（RGB）图像的多级阈值问题。 在RGB图像中，通过组合三个分量（红色，绿色和蓝色）来计算每个像素的值。 每个组件分别被视为灰度图像。 因此，可以为每个组件写入多个阈值的目标函数，如下所示：