

引用格式:唐长成,叶佐昌.基于强化学习的参数化电路优化算法[J].微电子学与计算机,2019,36(1):46-50. [TANG C C, YE Z C. Parametric Circuit Optimization with Reinforcement Learning[J]. Microelectronics & Computer, 2019, 36(1): 46-50.]

基于强化学习的参数化电路优化算法

唐长成,叶佐昌

(清华大学 微电子所,北京 100083)

摘 要: 本文主要致力于解决参数化形式的优化问题,即 $\min_{\theta} f(\theta, \mathbf{w})$, 其中 θ 是需要优化的变量, $\mathbf{w}$ 则是对应不同优化问题的参数, 在现实中经常会遇到需要解决一系列不同参数下的优化问题. 在对某种特定结构的问题下, 通过对不同的参数训练一个模型来解决所有参数下的优化问题. 和传统的方法不一样, 并不是通过对不同的参数多次独立抽样来训练我们的模型, 而是利用强化学习的方法加速训练过程. 强化学习算法中分别用策略网络来得到优化结果和利用价值网络来评价策略好坏, 通过迭代地训练两个网络来优化策略. 在后面一些数学例子和电路优化的例子中显示强化学习的方法取得了比较好的效果.

关键词: 优化算法; 强化学习; 参数化优化

中图分类号: TP301.6

文献标识码: A

文章编号: 1000-7180(2019)01-0046-05

Parametric Circuit Optimization with Reinforcement Learning

TANG Chang-cheng, YE Zuo-chang

(Institution of Microelectronic, Tsinghua University, Beijing 100083, China)

Abstract: In this paper we are focusing on solving parametric optimization problems, i. e. $\min_{\theta} f(\theta, \mathbf{w})$, where θ is the variable to be optimized and $\mathbf{w}$ is a vector that parameterize the optimization problem. Such kind of problems are very commonly seen in reality. We propose an efficient method to train a model that connects the solution to the parameters and thus solve all the problems with the same structure and different parameters at the same time. During training process, instead of solving a series of optimization problems with randomly sampled $\mathbf{w}$ independently, we adopt reinforcement learning to accelerate the training process. Two networks are trained alternately. The first network is a value network, and it is trained to fit the target loss function. The second network is a policy network, whose output is connected to the input θ of the value network and it is trained to minimize the output of the value network. Experiments demonstrate the effectiveness of the proposed method.

Key words: optimization algorithm; reinforcement learning; parametric optimization

1 引言

一个有效的、简单且鲁棒的优化算法在很多领域内都是非常重要的. 传统的全局优化算法, 例如进化算法(evolutionary algorithms, EA)^[1], 通常是通过广泛地搜索来得到最小值, 或者是依赖于梯度的算法, 类似随机梯度下降, 动量法等优化算法.

在很多应用中, 人们经常会遇到参数化的优化问题, 即很多不同的问题, 但是他们具有相同的结

构, 仅仅是参数不一样而已. 一个比较典型的应用是在电路设计中, 很多情况下人们都是对于同一个拓扑但是工艺参数不同的电路进行优化, 例如 SRAM, 逻辑门电路和运放电路. 虽然这些电路的特性人们都研究的非常透彻, 但是设计师在优化电路参数上依然需要花大量的时间, 因为不同的工艺节点、电压源和优化约束下都对参数有不同的要求. 如果可以解决这样参数化的优化问题的话, 那么模拟电路设计师的工作便大大减轻.

在文章 Learning to Learn by Descent by Descent^[2]中,作者训练了一个优化器学习梯度优化算法从而取代了人工设计的优化器.训练得到的优化器在每次迭代的过程中都能提供一个比较好的更新量去更新优化变量使得损失函数的收敛曲线收敛得最快.在另外一篇文章^[3]中,作者表明用有监督学习来训练优化器是不可行的,即输入当前的变量等信息,输出为更新量,最后往往会导致损失函数发散.

但强化学习则很好的解决了这个问题.通过提供当前问题的部分信息来得到合适的更新量与强化学习中的任务非常相似.强化学习的主要思想主要是通过定义一个回报函数以奖励好的优化器并惩罚差的优化器.

在本文中,我们提出了通过训练一个模型以优化一系列的参数化问题的优化算法.利用搭建两对神经网络来拟合策略函数和价值函数,并利用 DDPG (Deep Deterministic Policy Gradients) 算法^[4]来训练我们的网络.在之后的几个例子中均取得了不错的效果.

2 参数化优化方法

本文主要关注以下形式的优化问题: $\min_{\theta} f(\theta, \omega)$, 式中, $f(\theta, \omega)$ 是我们需要优化的目标函数; ω 则是代表不同目标函数的参数.

在一些比较简单的问题中,可以通过抽样并利用全局优化算法得到每个样本的最优值来训练一个回归模型来解决这一系列的问题.但由于问题不断变得复杂,优化变量的纬度不断变大,对抽样数量的需求也以指数级上升.而强化学习则很好地解决了这样地问题,通过将优化空间从原来的优化变量 θ 扩展到优化变量和参数 $\{\theta, \omega\}$, 在搜索的过程中能将之前的优化路径记忆下来,从而有效地利用不同参数直接最优解的关联性找到最优的搜索路径.

2.1 强化学习

一般的强化学习任务都能描述成这种形式: 一个智能体 (Agent) 与一个环境 (Environment) 在一个离散时间系统中的交互过程, 如图 1 所示. 环境一般是一个非常复杂的系统, 我们可以认为它是一个黑盒子, 可以通过观测来得到环境目前的状态 s_t 以及其他信息. 智能体基于每次的观测 o_t 来生成一个动作 a_t 以更新环境中的状态, 随后环境由我们定义的回报函数来回馈一个回报 $r_t \in \mathbb{R}$, 来表示智能体作出动作的好坏程度.

首先, 智能体对环境当前的状态进行观测; 然后

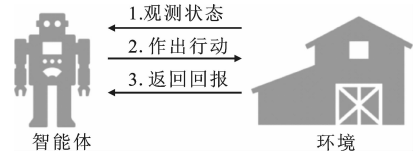


图 1 环境与智能体的关系

基于观测量作出一个动作; 随后环境更新当前状态并返回当前的回报给智能体. 评价我们动作的好坏可以通过一下的价值函数进行评价:

$$Q(s, a) = \max_{\pi} E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s, a_t = a, \pi]$$

式中, γ 是回报的折价因子; π 则是我们的策略函数, 即决定了我们智能体对不同观测量作出何种反应的行为的函数. 由贝尔曼方程可以将这个难以计算的函数转化为迭代形式, 即:

$$Q(s_t, a_t) = E_{r_t, s_{t+1}}[r_t + \gamma E_{a_{t+1} \sim \pi}[Q(s_{t+1}, a_{t+1})]]$$

也就是说我们可以通过迭代来近似地计算价值函数的大小, 通常我们利用神经网络来近似他们.

算法的目标是优化策略函数 $\pi: O \times A \rightarrow \mathcal{R} \geq 0$, 它决定了在某状态下, 智能体作出动作的分布. 也可以简单地表示为 $a_t \sim \pi(a_t | o_t)$ ^[5].

强化学习中有针对离散动作的算法, 例如 Q-learning 算法和 DQN, 和针对连续空间的算法, 例如 DDPG 和 PPO 等^[6]. 在离散动作空间中, 动作的样本空间是有限多个的, DQN 等算法一般是通过枚举所有动作并计算其 Q 值, 选取其中最大的来决定当前的动作. DQN 非常稳定且有效, 但并不能解决连续动作下的问题, 因为连续动作的可能性有无穷多种, 无法一一枚举. 我们选用 DDPG 这样的连续动作空间下的强化学习算法来解决我们的问题.

2.2 DDPG: Deep Deterministic Policy Gradients

DDPG 是一种连续状态 - 动作空间下的算法^[7]. 主要的思想是使用神经网络来拟合状态与动作空间之间的关系. 一般的强化学习算法是通过近似计算 Q 值来确定最优的策略, 称为 Q-learning 方法, 但直接利用神经网络去学习 Q 值已经被证明在很多情况下是非常不稳定的^[4], 因为在训练的过程中, 价值函数 $Q(s_t, a_t | \theta^Q)$ 同时被更新和作为学习目标. 而 DDPG 算法中使用了两套网络, 分别是策略网络 $\eta(s_t | \theta^\pi) \eta(s_t | \theta^\pi)$, 和判定网络 $Q(s_t, a_t | \theta^Q)$, 分别用于决定当前的动作和学习 Q 值, 这两个网络称为学习网络, 除此之外, 两个网络各自有一份拷贝 $\eta'(s_t | \theta^i)$, $Q'(s_t, a_t | \theta^Q)$, 他们的拷贝用于作为训练自身时的目标, 称为目标网络. 这样, 训练网络是

的优化目标则是使得损失函数最小:

$$L(\theta^Q) = E_{s_t, a_t, r_t} [(Q(s_t, a_t | \theta^Q) - y_t)^2]$$

式中, y_t 为目标判定网络得到的 Q 值.

为了保证算法的稳定性, 目标网络并不会在训练过程中更新他们的权重, 而是采用一种软更新策略:

$$\theta_{t+1}^* \leftarrow \tau \theta_t + (1 - \tau) \theta_t^*$$

式中, $\tau \ll 1$ 是一个软更新常数, 实际上是利用学习网络中学习到的权重逐步更新目标网络, 以达到迭代训练的效果, 虽然软更新的过程比较慢, 但是比直接学习 Q 值要更稳定.

当策略-判决网络训练好了之后, 我们可以仅依赖于策略网络来决定当前状态下应该做出什么样的动作.

3 方法

对于上述从实际问题中抽象出来的问题, 我们可以让目标函数 $f: S \rightarrow \mathbb{R}$ 作为强化学习任务中的环境 E , 目标函数中的优化变量 θ_i 作为状态 s_i , 每次智能体从环境中观测得到的观测值除了状态之外, 还包括了目标函数中的参数 w , 而智能体充当了优化过程中的优化器的作用, 每次提供的动作作为当前状态的更新量 $\Delta\theta$. 最后我们定义回报函数为 $r_t = f(\theta_{t+1}) - f(\theta_t)$, 也就是说当优化器使得目标函数减小的程度越多, 得到的回报就越大. 现在我们成功的将一个参数的优化问题转化为了一个强化学习的任务, 如图 2 所示.



图 2 转化成为强化学习任务的优化器框架

4 实验

4.1 实现细节

在我们的实验中, 我们使用了简单的浅层全连接网络来搭建策略网络和判定网络. 在训练过程中, 使用 Adam 优化器来优化网络权重, 学习率被设置为 0.001, 训练步数设置为 30 000 次, 其中一步表示智能体作出一次动作. 回报折价因子设置为 0.99, 软更新常数设置为 0.001. OpenAI Gym 是一个为强化学习算法提供环境接口的工具箱, 里面有许多集成的环境, 包括离散动作空间和连续动作空间等, 里

面每个环境都有公共的接口, 也可以自己实现环境的接口来自定义环境, 只需要实现以下几个接口即可: reset 接口负责将环境内部的所有状态初始化为原始状态, 可以是随机初始化也可以初始化为特定值; sample 接口负责从动作空间 (是 gym 包中的一个 spaces 对象) 中随机抽取一个动作; step 接口接收一个动作作为输入参数, 更新环境内部的状态后, 返回新的观测值 o_t , 回报 r_t , 一个布尔值表示当前回合是否结束 (结束后会将环境初始化然后重新开始) 和一些其他用于 debug 的信息. 其他的接口可以根据实际情况再作出具体的实现, 而 step 接口是环境中最重要接口, 直接决定了环境的回报函数和行为.

4.2 二次型函数

我们首先在一个简单的二次型函数上做了实验, 二次型函数是一个凸函数, 可以表示为以下形式:

$$f(\theta; W, b) = \|W\theta - b\|_2^2$$

在这个实验中, 我们利用 DDPG 算法训练了一个简单的优化器来优化一个 8 维的二次型函数. 其中优化变量为 θ , 即长度为 8 的向量, 参数为 $\{W, b\}$, 分别是 8×8 大小的矩阵和长度为 8 的向量. 在优化过程中, 我们将 W 转化为 64 维的向量, 与 b 拼接起来作为参数. 每次初始化环境时, θ 和 $\{W, b\}$ 随机地由高斯分布生成, 用于训练优化器.

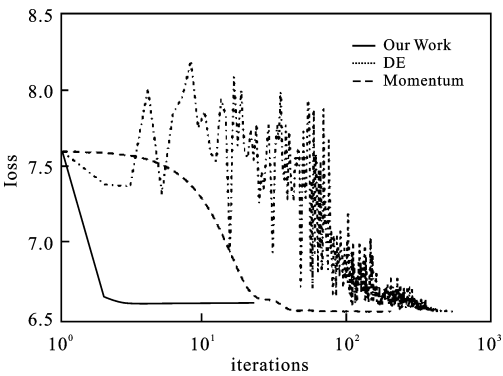


图 3 二次型函数的收敛曲线

使用传统的优化算法 (差分进化)、梯度下降算法和训练好的优化器的收敛曲线如图 3. 为了定量地比较收敛情况, 我们分别从收敛步数和收敛的相对误差来比较各个算法的表现, 相对误差定义为如下形式:

$$\text{relative_loss} = \frac{\text{loss}_{\text{final}}^* - \text{loss}_{\text{final}}^{\text{ddpg}}}{\text{loss}_{\text{init}}}$$

$\text{loss}_{\text{final}}^*$ 为目标函数在各算法下收敛后的大小.

即我们用收敛后的目标函数大小与优化器(DDPG 算法)得到的目标函数大小相比,若其比优化器得到的损失值更小,则相对误差为负,表示其性能比优化器要更优,反之说明表现不如训练好的优化器.在二次函数上各算法的优化结果如表 1 所示,可以看到在保证相对误差非常小的情况下,强化学习算法花费了更少的步数.

表 1 二次函数的优化实验结果

算法	花费步数(min/mean/max)	相对误差
强化学习算法	1/1/1	0/0/0
梯度下降法	10/23/36	-0.07 / -0.02 / -0.003
差分进化法	57/294/527	-0.07 / -0.02 / -0.003

4.3 非凸函数的优

Ackley 函数是一个比较有代表性的非凸函数,它拥有许多局部最优值,图 4 中是二维的 Ackley 函数图像.

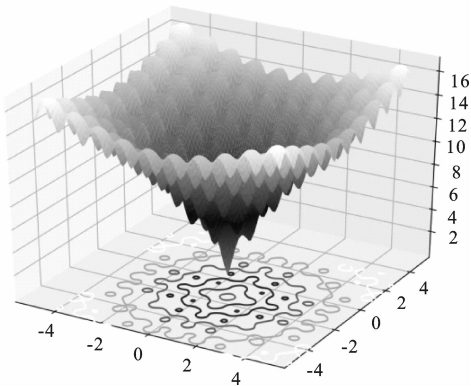


图 4 二维的 Ackley 函数图像

从理论上来说,我们训练好的优化器并不依赖于函数的凸性,而函数本身的复杂度也对我们的算法影响较小.也就是说即使函数拥有很多局部最优值,优化器总能找到全局最优解(因为主要能得到最大的回报).而梯度下降法(本文中我们使用的是动量法的梯度下降)在有很多个局部最优的情况下,很难找到全局最优解,而差分进化算法也会在某些情况下陷入了局部最优.

优化过程的收敛曲线如图 5.横轴是对数坐标,表示算法迭代的次数.可以观察到,动量梯度下降法(Momentum)几乎不能找到全局最优解,而差分进化算法也在几个例子中陷入了局部最优.具体的优化结果如表 2 所示,差分进化的花费步数平均在千步左右,而梯度下降法几乎无法找到最优解.

4.4 SRAM 工艺

在 SRAM 的生产过程中,SRAM 的性能非常

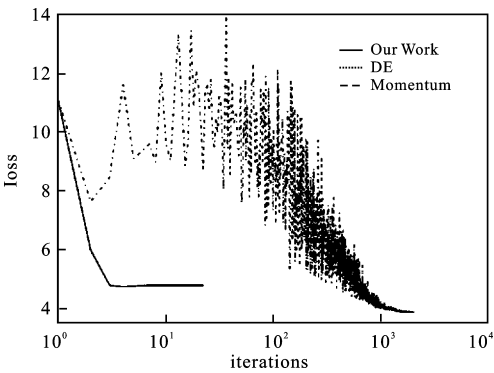


图 5 各算法在 Ackley 函数下的收敛曲线
表 2 Ackley 函数的优化实验结果

算法	花费步数(min/mean/max)	相对误差
强化学习算法	1/1/1	0/0/0
梯度下降法	-/-/-	0.15/0.57/0.85
差分进化法	833/1 044/1 344	-0.07 / -0.03 / 0.13

依赖于工艺参数 w 和工作环境变量 θ .当工作环境变量超出某个限定范围的时候,器件将无法正常工作.在这个问题中,我们需要找到某参数下器件可工作的最大范围,问题可表述为一个约束下的优化问题:

$$\begin{aligned} \min_{\theta} \quad & \|\theta\|_2^2 \\ \text{s. t.} \quad & f(\theta, w) \leq \gamma \end{aligned}$$

式中, $f(\cdot)$ 为 SRAM 的静态噪声容限(Static Noise Margin, SNM).当约束条件不满足的时候,器件无法工作.为了优化这个问题,我们把这个带约束的优化问题转化为带惩罚项的无约束优化问题:

$$\min_{\theta} \|\theta\|_2^2 + c \cdot \max\{0, f(\theta, w) - \gamma\}$$

式中, c 为惩罚因子,我们在优化过程中可以把它设置的比较大,当 $f(\theta, w) \leq \gamma$ 时惩罚为 0,违反约束的时候,施加线性大小的惩罚.

在优化的过程中,不可避免地需要多次调用仿真器来计算 $f(\theta, w)$,由于仿真器对我们的优化算法来说类似一个黑箱,难以计算其中各变量的梯度,只能使用启发式的搜索算法来得到他的最优解.在使用差分进化优化这个问题的时候,一般情况下找到他的最优值需要调用上百甚至上千次的仿真器找到他的最优解.在本次实验中,环境变量即我们的待优化变量 θ 为长度为 18 的向量,工艺参数为长度为 8 的向量 w .由于无法获取模型的梯度,我们只使用差分进化算法和强化学习训练好的优化器相比较,共优化了 20 组数据作为对比.

在使用差分进化算法优化的 20 组数据中,参数 w 和变量 θ 采用随机初始化,每次的样本都调用了

将近 2 000~3 000 次仿真器得到最优解. 而使用强化学习训练时,仅训练了 5 000 次不到时,优化器以及收敛到了一定的范围. 对于更复杂的问题来说,使用传统的优化算法可能需要花费 $O(n)$ 的时间去优化得到结果,在本例中每优化一个样本需要花费约一个小时,而采用训练好的优化器只需要 $O(1)$ 的时间,即不到一分钟就能给出一个令人满意的结果.

各算法的收敛曲线如图 6 所示,具体的优化结果如表 3 所示. 训练好的优化器依然能在 1~2 步内找到最优解,而差分进化算法每次都花费了上千步^[8-9].

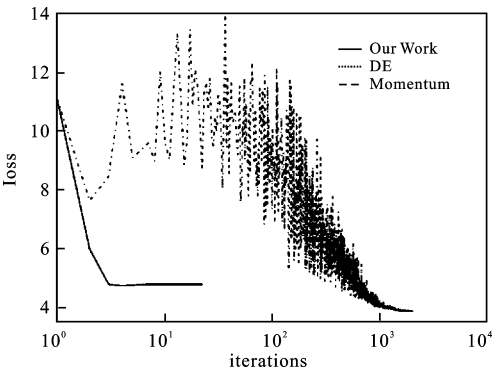


图 6 SRAM 实验中强化学习和差分进化算法的收敛曲线

表 3 SRAM 实验的优化实验结果

算法	花费步数(min/mean/max)	相对误差
强化学习算法	1/1/1	0/0/0
差分进化法	182/343/594	-0.27 / -0.13 / -0.03

5 结束语

对于某一类的优化问题,我们可以将其参数化,然后利用强化学习训练一个优化器去优化这个参数化的问题,并且几乎总能在 $O(1)$ 的时间内得到最优解.

参考文献:

[1] Das S, Suganthan P N. Differential evolution: a survey of the state-of-the-art[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1):4-31.

[2] Andrychowicz M, Denil M, Gomez S, et al. Learning to learn by gradient descent by gradient descent[EB/OL]. <https://arxiv.org/abs/1606.04474>.

[3] Li K, Malik J. Learning to optimize[EB/OL]. <https://arxiv.org/abs/1606.01885>.

[4] Heess N, Wayne G, Silver D, et al. Learning continuous control policies by stochastic value gradients[J]. International Conference on Neural Information Processing Systems, 2015:2944-2952.

[5] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2015, 8(6):A187.

[6] Balduzzi D, Ghifary M. Compatible value gradients for reinforcement learning of continuous deep policies[J]. Computer Science, 2015, 8(6):A187.

[7] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]// International Conference on International Conference on Machine Learning. Beijing, China, 2014:387-395.

[8] Peters J, Schaal S. Reinforcement learning of motor skills with policy gradients [J]. Neural Networks, 2008, 21(4):682-697.

[9] Bakker B. Reinforcement learning with long short-term memory[C]// International Conference on Neural Information Processing Systems: Natural and Synthetic. Cambridge, Massachusetts, MIT Press, 2001: 1475-1482.

作者简介:

唐长成 男, (1994-), 硕士研究生. 研究方向为 CAD 仿真设计、机器学习. E-mail: tangcc15@mails. tsinghua. edu. cn.

叶佐昌 男, (1980-), 博士, 副教授. 研究方向为机器学习、集成电路 CAD 仿真算法.