(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112292693 A
(43)申请公布日 2021.01.29

(21)申请号 201980033531.1

(22)申请日 2019.05.20

(30)优先权数据
62/673,844 2018.05.18 US

(85)PCT国际申请进入国家阶段日
2020.11.18

(86)PCT国际申请的申请数据
PCT/EP2019/062936 2019.05.20

(87)PCT国际申请的公布数据
WO2019/219965 EN 2019.11.21

(71)申请人 渊慧科技有限公司
地址 英国伦敦

(72)发明人 Z.徐 H.P.范哈塞尔特 D.希尔沃

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 金玉洁

(51)Int.Cl.
G06N 3/00(2006.01)
G06N 3/08(2006.01)

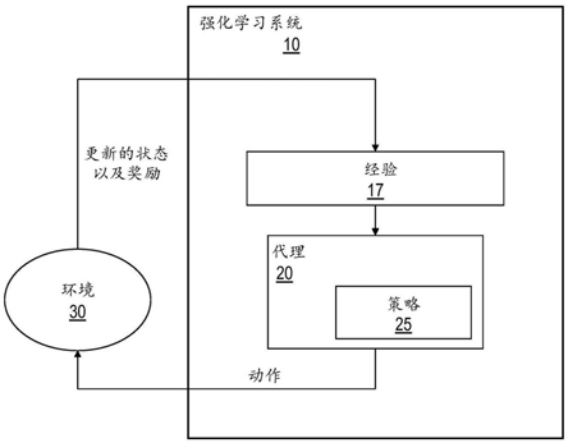
权利要求书3页 说明书19页 附图4页

(54)发明名称

强化学习系统训练返回函数的元梯度更新

(57)摘要

用于强化学习的方法、系统和装置，包括编码在计算机存储介质上的计算机程序。本文所描述的实施例应用元学习(特别是元梯度强化学习)来学习最优返回函数G，从而改善系统的训练。这提供了训练强化学习系统的更有效和高效的手段，因为系统能够通过训练返回函数G更快地收敛到一个或多个策略参数θ的最优集。特别地，使返回函数G取决于一个或多个策略参数θ，并且使用相对于一个或多个返回参数η被微分的元目标函数J'，以改善对返回函数G的训练。



CN 112292693 A

1.一种强化学习系统,包括一个或多个处理器,所述一个或多个处理器被配置为:

从强化学习神经网络检索多个经验,所述强化学习神经网络被配置为控制与环境交互的代理执行任务,以尝试基于所述强化学习神经网络的一个或多个策略参数来实现指定的结果,每个经验包括表征环境状态的观测数据、所述代理响应于所述观测数据而执行的动作以及响应于所述动作而接收到的奖励;

使用基于奖励计算返回的返回函数、基于第一经验集更新所述强化学习神经网络的一个或多个策略参数;以及

基于一个或多个更新的策略参数和第二经验集,更新所述返回函数的一个或多个返回参数,其中所述一个或多个返回参数使用相对于所述一个或多个返回参数被微分的元目标函数经由梯度上升或下降方法来更新,其中所述元目标函数取决于所述一个或多个策略参数。

2.根据权利要求1所述的强化学习系统,其中,更新所述一个或多个返回参数利用所述一个或多个更新的策略参数相对于所述一个或多个返回参数的微分。

3.根据权利要求1所述的强化学习系统,其中,所述一个或多个处理器还被配置为迭代地:

使用所述一个或多个更新的策略参数和所述一个或多个更新的返回参数,检索由所述强化神经网络生成的更新的经验;

使用所述一个或多个更新的返回参数、基于第一更新经验集,进一步更新所述一个或多个策略参数;以及

经由梯度上升或下降方法、基于进一步更新的策略参数和第二更新经验集,进一步更新所述一个或多个返回参数,

直到达到结束条件。

4.根据权利要求1所述的强化学习系统,其中,更新所述一个或多个返回参数包括:应用进一步的返回函数作为所述元目标函数的部分,并且根据当所述进一步的返回函数被应用于所述第二经验集时的返回来评估更新的策略。

5.根据权利要求1所述的强化学习系统,其中,对所述一个或多个策略参数的更新应用以所述一个或多个返回参数为条件的策略和值函数中的一个或多个。

6.根据权利要求5所述的强化学习系统,其中,以所述一个或多个返回参数为条件是经由对所述一个或多个返回参数的嵌入来进行的。

7.根据权利要求1所述的强化学习系统,其中,所述一个或多个返回参数包括所述返回函数的折扣因子和所述返回函数的自举因子。

8.根据权利要求1所述的强化学习系统,其中,所述一个或多个处理器还被配置为:

基于所述第二经验集更新所述强化学习神经网络的一个或多个策略参数;以及

基于所述一个或多个更新的策略参数和所述第一经验集,更新所述返回函数的一个或多个返回参数,其中所述一个或多个返回参数经由梯度上升或下降方法来更新。

9.根据权利要求1所述的强化学习系统,其中,被微分的元目标函数为:

$$\frac{\partial J'(\tau',\theta',\eta')}{\partial \eta}=\frac{\partial J'(\tau',\theta',\eta')}{\partial \theta'}\frac{d\theta'}{d\eta}$$

其中：

η是所述一个或多个返回参数；以及

J′(τ′,θ′,η′)是以所述第二经验集τ′、所述一个或多个更新的策略参数θ′和形成所述元目标函数的部分的进一步的返回函数的一个或多个进一步的返回参数η′为条件的所述元目标函数。

10.根据权利要求9所述的强化学习系统，其中，系统被配置为基于所述更新的策略参数θ′相对于所述返回参数η的微分dθ′/dη来计算所述被微分的元目标函数，所述微分dθ′/dη是通过将更新函数相对于所述返回参数的微分df(τ,θ,η)/dη加到所述策略参数θ相对于所述返回参数η的微分dθ/dη上来计算的，所述更新函数用于更新策略。

11.根据权利要求10所述的强化学习系统，其中，所述更新函数相对于所述返回参数的微分$\partial f(\tau,\ \theta,\ \eta)/\partial\eta$经由以下公式计算：

$$\frac{\partial f(\tau,\theta,\eta)}{\partial\eta}=\alpha\frac{\partial G_{\eta}(\tau)}{\partial\eta}\left[\frac{\partial\log\pi_{\theta}(A|S)}{\partial\theta}+c\ \frac{\partial v_{\theta}(S)}{\partial\theta}\right]$$

其中：

α是当从一个或多个先前策略参数更新所述一个或多个策略参数时所应用的学习率；

$G_{\eta}(\tau)$是基于所述一个或多个返回参数η、计算从所述第一经验集τ的返回的返回函数；

π_{θ}是用于所述强化学习神经网络从状态S确定动作A的策略，所述策略π_{θ}根据所述一个或多个策略参数θ操作；

c是系数；并且

$v_{\theta}(S)$是基于所述一个或多个策略参数θ确定状态S的值的值函数。

12.根据权利要求9所述的强化学习系统，其中：

$$\frac{\partial J'(\tau',\theta',\eta')}{\partial\theta'}=\left(G_{\eta'}(\tau')-v_{\theta'}(S')\right)\frac{\partial\log\pi_{\theta'}(A'|S')}{\partial\theta'}$$

其中：

$G_{\eta'}(\tau')$是基于所述一个或多个进一步的返回参数η′、计算从所述第二经验集τ′的返回的进一步的返回函数；

$\pi_{\theta'}$是用于所述强化学习神经网络从取自所述第二经验集τ′的状态S′确定动作A′的策略，所述策略$\pi_{\theta'}$根据所述一个或多个更新的策略参数θ′操作；并且

$v_{\theta'}(S')$是基于所述一个或多个更新的策略参数θ′确定状态S′的值的值函数。

13.根据权利要求9所述的强化学习系统，其中，所述一个或多个进一步的返回参数保持固定。

14.根据权利要求9所述的强化学习系统，其中，更新所述一个或多个返回参数包括计算：

$$\eta'=\eta-\beta\frac{\partial J'(\tau';\theta',\eta')}{\partial\eta}$$

其中

η′是所述一个或多个更新的返回参数；并且

β是用于更新所述一个或多个返回参数的学习因子。

15.一种用于强化学习的计算机实现的方法,所述方法包括:

从强化学习神经网络检索多个经验,所述强化学习神经网络被配置为控制与环境交互的代理执行任务,以尝试基于所述强化学习神经网络的一个或多个策略参数来实现指定的结果,每个经验包括表征环境状态的观测数据、所述代理响应于所述观测数据而执行的动作以及响应于所述动作而接收到的奖励;

使用基于奖励计算返回的返回函数、基于第一经验集更新所述强化学习神经网络的一个或多个策略参数;以及

基于一个或多个更新的策略参数和第二经验集,更新所述返回函数的一个或多个返回参数,其中所述一个或多个返回参数使用相对于所述一个或多个返回参数被微分的元目标函数经由经由梯度上升或下降方法来更新,其中所述元目标函数取决于所述一个或多个策略参数。

16.一个或多个计算机存储介质,其存储指令,当由一个或多个计算机运行指令时,所述指令使得所述一个或多个计算机执行权利要求15所述的方法的操作。

强化学习系统训练返回函数的元梯度更新

技术领域

[0001] 本说明书涉及强化学习(reinforcement learning)。

背景技术

[0002] 在强化学习系统中,代理(agent)通过执行由强化学习系统响应于接收到表征环境当前状态的观测数据(observation)而选择的动作来与环境交互。

[0003] 一些强化学习系统响应于接收到给定的观测数据,根据神经网络的输出选择要由代理执行的动作。

[0004] 神经网络是机器学习模型,其采用非线性单元的一个或多个层来预测对接收到的输入的输出。一些神经网络是深度神经网络,其除了输出层之外,还包括一个或多个隐藏层。每个隐藏层的输出被用作网络中下一个层(即下一个隐藏层或输出层)的输入。网络的每个层根据相应参数集的当前值从接收到的输入生成输出。

发明内容

[0005] 本说明书总体上描述了一种强化学习系统,该系统选择要由与环境交互的强化学习代理执行的动作。为了使代理与环境交互,系统接收表征环境当前状态的数据,并响应于接收到的数据选择要由代理执行的动作。表征环境状态的数据在本说明书中被称为观测数据。

[0006] 在一些实现方式中,环境是模拟(simulated)环境,并且代理被实现为与模拟环境交互的一个或多个计算机程序。例如,模拟环境可以是视频游戏,并且代理可以是玩视频游戏的模拟用户。作为另一示例,模拟环境可以是运动模拟环境,例如驾驶模拟或飞行模拟,并且代理是通过运动模拟航行的模拟交通工具(vehicle)。在这些实现方式中,动作可以是控制模拟用户或模拟交通工具的控制输入。

[0007] 在一些其他实现方式中,环境是真实世界环境,并且代理是与真实世界环境交互的机械代理。例如,代理可以是与环境交互以完成特定任务的机器人。作为另一示例,代理可以是在环境中航行的自动或半自动交通工具。在这些实现方式中,动作可以是控制机器人或自动交通工具的控制输入。

[0008] 在一个创新方面,本说明书中所描述的主题可以在强化学习系统中实现,该系统包括被配置为从强化学习神经网络中检索多个经验(experience)的一个或多个处理器,该强化学习神经网络被配置为控制与环境交互的代理执行任务,以尝试基于强化学习神经网络的一个或多个策略参数来实现指定的结果,每个经验包括表征环境状态的观测数据、代理响应于观测数据而执行的动作以及响应于动作而接收到的奖励(reward)。一个或多个处理器还被配置为使用基于奖励计算返回(return)的返回函数、基于第一经验集来更新强化学习神经网络的一个或多个策略参数;以及基于一个或多个更新的策略参数和第二经验集来更新返回函数的一个或多个返回参数。一个或多个返回参数使用相对于该一个或多个返回参数被微分的元目标(meta-objective)函数经由梯度上升或下降方法来更新,其中元目

标函数取决于一个或多个策略参数。

[0009] 本文所描述的实施例应用元学习(特别是元梯度强化学习)来学习最优返回函数G,以改善对系统的训练。这提供了训练强化学习系统的更有效和高效的手段,因为系统能够通过逐步训练返回函数G更快地收敛到一个或多个策略参数θ的最优集。特别地,使返回函数G取决于一个或多个策略参数θ,并且使用相对于一个或多个返回参数η被微分的元目标函数J',以改善对返回函数G的训练。

[0010] 元学习可以被认为是训练系统更有效地学习的行为。元目标J'因此可以被认为是改善强化学习神经网络的学习功能的目标。具体地,元目标函数J'可以是用于优化强化学习神经网络的返回参数η的函数。元目标函数J'可以服务于识别使代理中的整体性能最大化的返回函数的目的。这可以通过一个专注于优化返回的元目标(换句话说,策略梯度目标)来直接测量。例如,元目标函数可以计算返回函数和代理用来确定动作的值函数之间的误差(例如均方误差),并且系统可以被配置为更新返回参数以减少(例如最小化)误差。

[0011] 检索经验τ可以包括系统生成经验(即,强化学习神经网络可以形成系统的部分)或者例如,从存储装置或外部系统访问经验。也就是说,经验可以由强化学习系统本身在线生成,或者可以从外部强化学习神经网络获得。因此,更新策略参数θ可以包括将更新的策略参数θ'发送到外部强化神经网络或者更新形成整个系统的部分的强化神经网络。

[0012] 基于各种不同类型的强化学习,可以利用各种不同类型的返回函数G。例如,返回函数G可以是随机梯度上升方法中的返回函数,或者可以作为Q学习方法中的目标。

[0013] 第一经验集和第二经验集可以是相同的。可替代地,第一经验集和第二经验集可以包括不同的经验。通过避免过拟合,将用于更新一个或多个返回参数的不同经验用于那些更新一个或多个策略参数的经验(保留训练数据以用于训练返回函数)改善了训练。

[0014] 更新一个或多个返回参数可以利用一个或多个更新的策略参数相对于一个或多个返回参数的微分。元目标函数J'相对于一个或多个返回参数η的微分可以利用偏导数,将该微分分为两个分量,第一分量是元目标函数J'相对于一个或多个更新的策略参数θ'的微分,并且第二分量是一个或多个更新的策略参数θ'相对于一个或多个返回参数η的微分。因此,这允许系统在更新一个或多个返回参数η时利用更新的(改善的)策略参数θ',从而提高训练的有效性。

[0015] 一个或多个策略参数可以是定义强化学习神经网络的功能的一个或多个参数(定义神经网络采取的动作的一个或多个参数)。一个或多个返回参数可以是定义如何基于奖励来确定返回的参数。

[0016] 一个或多个处理器还可以被配置为迭代地:使用一个或多个更新的策略参数和一个或多个更新的返回参数来检索由强化神经网络生成的更新的经验;使用一个或多个更新的返回参数、基于更新的第一经验集,进一步更新一个或多个策略参数;以及通过梯度上升或下降方法、基于进一步更新的策略参数和更新的第二经验集,进一步更新一个或多个返回参数,直到达到结束条件。

[0017] 因此,系统可以迭代地更新一个或多个策略参数和一个或多个返回参数,以收敛到最优策略。通过在一个或多个策略参数的训练期间更新一个或多个返回参数,系统能够改善计算的奖励,并因此可以更准确地训练策略并(更有效地)使用更少的训练片段(episode)。

[0018] 每次更新策略参数时，可以更新一个或多个返回参数。这为训练系统提供了一种计算更简单且更有效的机制。

[0019] 可替代地，一个或多个返回参数可以在一个或多个策略参数的多次更新中保持固定。在这种情况下，一个或多个返回参数然后可以随时间通过反向传播来更新。

[0020] 更新一个或多个返回参数可以包括应用进一步的(further)返回函数作为元目标函数的部分，并且当被应用于第二经验集时，根据来自进一步的返回函数的返回来评估更新的策略。这可以包括相对于一个或多个返回参数最大化来自进一步的返回函数的总返回的目的。进一步的返回函数G'可以是基于一个或多个进一步的返回参数η'从奖励计算返回的函数。进一步的返回函数可以被认为是具有元返回参数(或者更一般地，元参数)的元返回函数。进一步的返回函数可以作为训练强化学习神经网络以改善返回函数G的手段。进一步的返回函数可以不同于返回函数。进一步的返回参数可以在训练期间保持固定。

[0021] 对一个或多个策略参数的更新可以应用以一个或多个返回参数为条件的策略函数和值函数中的一个或多个。基于一个或多个返回参数来限定(condition)策略和/或值函数使得训练方法更加稳定。由于返回函数在训练期间改变，策略或值函数可能会失效。这可能会导致值评估策略的崩溃(collapse)。通过在一个或多个返回参数上限定策略和/或值函数，代理被强制学习各种返回参数集的通用策略和/或值函数。这允许该方法自由地变动(shift)元参数，而不需要等待逼近器"跟上"。

[0022] 限定可以经由对一个或多个返回参数的嵌入来进行。嵌入可以通过将一个或多个参数输入到嵌入函数中来形成一个或多个嵌入的返回参数。一个或多个嵌入的返回参数可以使用隐藏(潜在)变量来表示一个或多个参数。具体地，策略函数和值函数可以被限定为：

[0023] $$V_\theta^\eta(S) = V_\theta\left(\left[S\ ;\boldsymbol{e}_\eta\right]\right)\pi_\theta^\eta(S) = \pi_\theta\left(\left[S\ ;\boldsymbol{e}_\eta\right]\right)\boldsymbol{e}_\eta = \boldsymbol{w}_\eta^\top\eta$$

[0024] 其中：

[0025] $V_\theta()$是值函数，并且$V_\theta^\eta()$是带条件的值函数；

[0026] $\pi_\theta()$是策略，并且$\pi_\theta^\eta()$是带条件的策略；

[0027] e_η是对一个或多个返回参数η的嵌入；

[0028] $[S;e_\eta]$表示向量S(相应的状态)和e_η的级联(concatenation)；以及

[0029] w_η是以η为输入的嵌入函数，其中w_η可以经由代理训练中的反向传播来学习。

[0030] 一个或多个参数可以包括返回函数的折扣因子(discount factor)和返回函数的自举因子(bootstrapping factor)中的一个或多个。返回函数可以应用折扣因子γ来提供折扣返回。折扣返回可以包括奖励的加权和，其中折扣因子定义加权和的衰减。已经发现优化折扣因子是提高强化学习的效率和准确性的特别有效的方法。同样地，返回函数可以将自举因子λ应用于返回的几何加权组合(自举参数返回函数或λ返回)。返回函数可以计算返回的加权组合，每个返回是在多个步骤(例如，奖励的衰减加权和)上估计的。优化自举因子(可能与折扣因子组合)会导致更有效和更准确的强化学习。

[0031] 一个或多个处理器还可以被配置为：基于第二经验集更新强化学习神经网络的一个或多个策略参数；以及基于一个或多个更新的策略参数和第一经验集来更新返回函数的一个或多个返回参数，其中一个或多个返回参数经由梯度上升或下降方法来更新。这通过

重复更新步骤但交换(swap)用于每次更新的经验集来提高训练的效率和有效性。如上所述,使用不同的经验集避免过拟合。然而,这减少了可以用于每次更新的训练数据量。为了提高数据效率,可以在更新后交换第一经验集和第二经验集,这样第二经验集可以用于训练策略,并且第二经验集可以用于训练返回函数。

[0032] 也就是说,第一经验集可以用于更新一个或多个策略参数,并且可以通过使用第二经验集评估元目标函数来验证该更新的性能。然后,可以交换第一经验集和第二经验集的角色,使得第二经验集用于更新一个或多个策略参数,并且可以通过在第一经验集上评估元目标函数来验证该更新的性能。这样,除了用于训练一个或多个策略参数以便对一个或多个返回参数进行元学习更新的数据之外,所提出的方法不需要任何额外的数据。

[0033] 微分的元目标函数可以是:

[0034] $$\frac{\partial J'(\tau', \theta', \eta')}{\partial \eta} = \frac{\partial J'(\tau', \theta', \eta')}{\partial \theta'} \frac{d\theta'}{d\eta}$$

[0035] 其中:

[0036] η是一个或多个返回参数;以及

[0037] J′(τ′,θ′,η′)是元目标函数,其以第二经验集τ′、一个或多个更新的策略参数θ′和形成元目标函数的部分的进一步的返回函数的一个或多个返回参数η′为条件。

[0038] 上述公式是本说明书后面所描述的公式(2)的等同。

[0039] 该系统可以被配置为基于更新的策略参数θ′相对于返回参数η′的微分dθ′/dη来计算微分的元目标函数,微分dθ′/dη是通过将更新函数相对于返回参数的微分df(τ,θ,η)/dη加到策略参数θ相对于返回参数η的微分dθ/dη上来计算的,该更新函数用于更新策略。

[0040] 更新的策略参数θ′相对于返回参数η的微分dθ′/dη可以使用累积迹线(accumulative trace)被计算为≈dθ/dη,使得:

[0041] $$z' = \mu z + \frac{\partial f(\tau, \theta, \eta)}{\partial \eta}$$

[0042] 其中:

[0043] μ是衰减(decay)参数;以及

[0044] f(τ,θ,η)是用于更新策略的更新函数。

[0045] 更新函数相对于返回参数的微分$\partial f(\tau, \theta, \eta)/\partial \eta$可以经由以下方式计算:

[0046] $$\frac{\partial f(\tau,\theta,\eta)}{\partial \eta} = \alpha \frac{\partial G_\eta(\tau)}{\partial \eta} \left[\frac{\partial \log \pi_\theta(A|S)}{\partial \theta} + c \frac{\partial v_\theta(S)}{\partial \theta} \right]$$

[0047] 其中:

[0048] α是当从一个或多个先前策略参数更新一个或多个策略参数时所应用的学习率;

[0049] $G_\eta(\tau)$是基于一个或多个返回参数η_t、从第一经验集τ计算返回的返回函数;

[0050] π_θ是用于强化学习神经网络从状态S确定动作Aa的策略,策略π_θ根据一个或多个策略参数θ操作;

[0051] c是系数;以及

[0052] $v_\theta(S)$是基于一个或多个策略参数θ确定状态S的值的值函数。

[0053] 上述公式是本说明书后面所描述的公式(13)的等同。

[0054] 换句话，更新的策略参数相对于返回参数的微分可以经由以下方式计算：

[0055] $$\frac{\partial\theta'}{\partial\eta}=\frac{\partial\theta}{\partial\eta}+\alpha\frac{\partial G_{\eta}(\tau)}{\partial\eta}\left[\frac{\partial\log\pi_{\theta}(A|S)}{\partial\theta}+c\frac{\partial v_{\theta}(S)}{\partial\theta}\right]$$

[0056] 对一个或多个返回参数的更新可以利用：

[0057] $$\frac{\partial J'(\tau',\theta',\eta')}{\partial\theta'}=\left(G_{\eta'}(\tau')-v_{\theta'}(S')\right)\frac{\partial\log\pi_{\theta'}(A'|S')}{\partial\theta'}$$

[0058] 其中：

[0059] $G_{\eta'}(\tau')$是基于一个或多个进一步的返回参数η'、从第二经验集τ'计算返回的进一步的返回函数；

[0060] $\pi_{\theta'}$是用于强化学习神经网络从取自第二经验集τ'的状态S'确定动作A'的策略，策略$\pi_{\theta'}$根据一个或多个更新的策略参数θ'操作；

[0061] $v_{\theta'}(S')$是基于一个或多个更新的策略参数θ'确定状态S'的值的值函数。

[0062] 上述公式是本说明书后面所描述的公式(14)的等同。

[0063] 这将根据当在第二经验集τ'上测量的在η'的情况下的总返回，评估更新的策略。

[0064] 一个或多个另外的返回参数可以保持固定。这允许获得相对于一个或多个返回参数的梯度。

[0065] 更新一个或多个返回参数可以包括计算：

[0066] $$\eta'=\eta+\beta\frac{\partial J'(\tau'\ ;\theta',\eta')}{\partial\eta}$$

[0067] 其中

[0068] η'是一个或多个更新的返回参数；以及

[0069] β是用于更新一个或多个返回参数的学习因子。

[0070] 上述公式是本说明书后面所描述的公式(2)至(5)的等同。

[0071] 因此，对一个或多个返回参数的更新可以基于元目标函数相对于一个或多个返回参数的梯度来应用梯度上升方法。

[0072] 在一个创新方面，本说明书中所描述的主题可以具体体现在用于强化学习的计算机实现的方法中。该方法包括：从强化学习神经网络中检索多个经验，强化学习神经网络被配置为控制与环境交互的代理执行任务，以基于强化学习神经网络的一个或多个策略参数来尝试实现指定的结果，每个经验包括表征环境状态的观测数据、代理响应于观测数据而执行的动作以及响应于动作而接收到的奖励；使用基于奖励计算返回的返回函数、基于第一经验集更新强化学习神经网络的一个或多个策略参数；以及基于一个或多个更新的策略参数和第二经验集来更新返回函数的一个或多个返回参数，其中一个或多个返回参数使用相对于一个或多个返回参数被微分的元目标函数经由梯度上升或下降方法来更新，其中元目标函数取决于一个或多个策略参数。

[0073] 本发明的另一方面可以具体体现在存储指令的一个或多个计算机存储介质中，当指令被一个或多个计算机执行时，使得一个或多个计算机执行本文所描述的任何方法的操作。

[0074] 本说明书的主题的某些新颖方面在所附的权利要求中阐述。

[0075] 本说明书中所描述的主题可以在特定实施例中实现，以便实现一个或多个以下优点。

[0076] 该系统的实现方式通过应用元学习来训练强化学习系统的奖励函数，促进了强化学习系统的改善的训练。本文所描述的方法利用了取决于被训练的系统的一个或多个策略参数的元目标函数。元目标函数被微分以获得元梯度，元梯度可用于在与环境互动并从环境中学习的同时、在线适配返回的性质。

[0077] 更具体地，对返回函数的更新是在一个或多个策略参数被更新之后被应用的。策略参数相对于一个或多个返回参数的梯度指示元参数如何影响(多个)策略参数。因此，该方法能够在第二经验样本上测量(多个)更新的策略参数的性能，以改善返回函数的(多个)参数。

[0078] 这在训练时提高了系统的性能，并允许使用更少量的训练数据更有效地训练系统。例如，通过在系统训练时学习改善的返回函数，系统能够使用更少的更新更快地收敛到最优策略参数集。由于在训练过程中使用了改善的奖励，最终训练后的系统也显示了更准确和有效的学习行为。

[0079] 系统的实现可以在数据流上在线地，或者使用存储的数据离线地，或者以这两种方式来训练。系统可以自动地适配其训练以适应特定的训练任务，学习更好地执行这些任务。这有助于自动化训练过程，并使系统的实现能够用于更广泛的不同任务，而不需要针对特定任务进行调节或适配。所提出的方法适用于几乎所有当前的强化学习系统，因为返回总是用于代理更新，并且几乎所有当前的强化学习更新都包括返回的可微函数。例如，这包括基于值的方法(如Q(λ))、策略梯度方法、或动作-评价(actor-critic)算法，如Mnih等人于2016年提出的A3C(Advantage Actor Critic，高级动作-评价)或Espholt等人在arXiv:1802.01561中提出的IMPALA(Importance-Weighted Actor-Learner，重要性加权动作-学习者)。

[0080] 本说明书的主题的一个或多个实施例的细节在附图和以下描述中阐述。根据说明书、附图和权利要求，本主题的其他特征、方面和优点将变得清楚。

附图说明

[0081] 图1示出了用于训练代理与环境交互的示例强化学习系统。

[0082] 图2示出了根据本说明书的应用元学习的强化学习系统。

[0083] 图3示出了图2的系统基于检索到的多个经验更新策略参数和元参数所遵循的过程。

[0084] 图4示出了在更新策略参数之后更新元参数的过程。

[0085] 不同附图中相同的附图标记和名称表示相同的元件。

具体实施方式

[0086] 本文所描述的实现方式涉及强化学习系统。

[0087] 广义而言，强化学习系统是选择由与环境交互的强化学习代理执行的动作的系统。为了使代理与环境交互，系统接收表征环境当前状态的数据，并响应于接收到的数据选

择要由代理执行的动作。表征环境状态的数据在本说明书中被称为观测数据。可选地，在时间步长处的观测数据可以包括来自先前时间步长的数据，例如，在先前时间步长执行的动作、在先前时间步长接收到的奖励等。

[0088] 在一些实现方式中，环境是真实世界环境，并且代理是与真实世界环境交互的机电代理。例如，代理可以是与环境交互以完成特定任务(例如，在环境中定位感兴趣的对象或将感兴趣的对象移动到环境中的指定位置或者导航到环境中的指定目的地)的机器人或其他静止或运动机器；或者代理可以是在环境中航行的自动或半自动的陆地或空中或海上交通工具。

[0089] 在这些实现方式中，观测数据可以包括例如图像、对象位置数据和传感器数据中的一个或多个，以在代理与环境交互时捕获观测数据，例如来自图像、距离或位置传感器或者来自致动器的传感器数据。在机器人或其他机械代理或者交通工具的情况下，观测数据可以类似地包括位置、线速度或角速度、力、扭矩或加速度以及代理的一个或多个部分的整体或相对姿势中的一个或多个。观测数据可以被定义为一维、二维或三维，并且可以是绝对和/或相对观测数据。例如，在机器人的情况下，观测数据可以包括表征机器人当前状态的数据，例如，以下中的一个或多个：关节位置、关节速度、关节力、扭矩或加速度，以及机器人的部分(诸如手臂、和/或由机器人保持物品的)的整体或相对姿势。观测数据还可以包括例如所感测到的电子信号，诸如，马达电流或温度信号；和/或例如来自相机或LIDAR传感器的图像或视频数据，例如来自代理的传感器的数据或来自与环境中的代理分开放置的传感器的数据。

[0090] 在这些实现方式中，动作可以是控制机器人的控制输入，例如机器人关节的扭矩或更高级别的控制命令；或者是控制自动或半自动的陆地或空中或海上交通工具的控制输入，例如对交通工具的控制表面或其他控制元件的扭矩或更高级别的控制命令；或者例如马达控制数据。换句话说，例如，动作可以包括机器人的一个或多个关节或另一机械代理的部分的位置、速度或力/扭矩/加速度数据。动作数据可以包括用于这些动作的数据和/或电子控制数据，诸如马达控制数据，或者更一般地，用于控制环境中的一个或多个电子设备的数据，对电子设备的控制对环境的观测状态有影响。例如，在自动或半自动陆地或空中或海上交通工具的情况下，动作可以包括控制交通工具的导航例如转向以及运动例如制动和/或加速的动作。

[0091] 在一些实现方式中，环境是模拟环境，并且代理被实现为与模拟环境交互的一个或多个计算机。

[0092] 例如，模拟环境可以是对机器人或交通工具的模拟，并且强化学习系统可以在模拟上进行训练。例如，模拟环境可以是运动模拟环境，例如驾驶模拟或飞行模拟，并且代理是在运动模拟中航行的模拟交通工具。在这些实现方式中，动作可以是控制模拟用户或模拟交通工具的控制输入。在真实世界中使用强化学习系统之前，模拟环境对于训练该系统非常有用。在另一示例中，模拟环境可以是视频游戏，并且代理可以是玩视频游戏的模拟用户。一般，在模拟环境的情况下，观测数据可以包括先前描述的观测数据或观测数据类型中的一个或多个的模拟版本，并且动作可以包括先前描述的动作或动作类型中的一个或多个的模拟版本。

[0093] 在另一示例中，环境可以是蛋白质折叠环境，使得每个状态是蛋白质链的相应状

态,并且代理是用于确定如何折叠蛋白质链的计算机系统。在这个示例中,动作是用于折叠蛋白质链的可能的折叠动作,并且要实现的结果可以包括,例如,折叠蛋白质以使蛋白质稳定并从而实现特定的生物功能。作为另一示例,代理可以是执行或控制由系统自动而不需要人类交互地选择的蛋白质折叠动作的机械代理。观测数据可以包括对蛋白质状态的直接或间接观测数据和/或可以从模拟中导出。

[0094] 以类似的方式,环境可以是药物设计环境,使得每个状态是潜在的制药化学药物(pharma chemical drug)的相应状态,并且代理是用于确定制药化学药物的元素和/或制药化学药物的合成途径的计算机系统。例如在模拟中,药物/合成可以基于从药物的目标导出的奖励来设计。作为另一示例,代理可以是执行或控制药物合成的机械代理。

[0095] 在电子代理的情况下,观测数据可以包括来自监控工厂或服务设施的部分的一个或多个传感器的数据,诸如电流、电压、功率、温度,和来自其他传感器和/或表示的电子和/或机械设备部件的功能的电子信号。在一些应用中,代理可以控制包括设备部件的真实世界环境中的动作,例如在诸如数据中心、服务器体、电网干线电力或水配给系统的设施中,或者在制造厂或服务设施中。观测数据可以与工厂或设施的操作有关。例如,除了先前所描述的那些之外或可替代地,它们可以包括对设备的电或水使用的观测数据、或对发电或配电控制的观测数据、或者对资源使用或废物产生的观测数据。代理可以控制环境中的动作以提高效率,例如通过减少资源使用,和/或减少环境中操作对环境的影响,例如通过减少浪费。例如,代理可以控制设施中的电力或其他功耗或水的使用和/或设施和/或设施内项目的温度。动作可以包括控制或强加工厂/设施的设备项目的操作条件的动作,和/或导致工厂/设施的操作中的设置改变的动作,例如调节或打开/关闭工厂/设施的组件。

[0096] 在一些另外的应用中,环境是真实世界环境,并且代理管理跨计算资源(例如在移动设备和/或数据中心中)的任务分布。在这些实现方式中,动作可以包括将任务分配给特定的计算资源。作为另外的示例,动作可以包括呈现广告,观测数据可以包括广告印象或点进(click-through)计数或点击率,并且奖励可以表征一个或多个用户对项目或内容的先前选择。

[0097] 强化学习系统可以被实现为在一个或多个位置的一个或多个计算机上的一个或多个计算机程序,其中实现了本文所描述的系统、组件和技术。

[0098] 图1示出了强化学习系统的示例。强化学习系统10包括基于策略25确定动作的代理20。每次动作被确定时,动作被输出到由代理20控制的环境30。该动作更新环境30的状态。更新的状态连同与该动作相关联的奖励一起被返回到强化学习系统10。这些被强化学习系统10用来确定下一个动作。一般,奖励是数值。奖励可以基于环境30的任何事件或方面。例如,奖励可以指示代理20是否已经完成任务(例如,导航到环境30中的目标位置)或者代理20朝向完成任务的进展。

[0099] 代理20与环境30在一个或多个时间步长上的交互可以由经验元组(tuple)的“轨迹”(即,序列)来表示,其中每个经验元组对应于相应时间步长。与时间步长相对应的经验元组可以包括:(i)表征在时间步长处的环境状态的观测数据,(ii)被选择由代理在时间步长处执行的动作,(iii)表征在代理执行所选动作之后的环境后续状态的后续观测数据,(iv)在代理执行所选动作之后接收到的奖励,以及(v)被选择要在后续时间步长处执行的后续动作。

[0100] 策略25基于环境状态定义系统如何执行动作。当系统10基于经验集17被训练时，代理20遵循的策略25通过根据近似值函数或返回函数评估动作的值来更新，以提高来自策略25所采取的动作的预期返回。这通常通过结合预测和控制来评估由代理20执行的动作的成功来实现，有时被称为“返回”。返回是基于给定动作之后接收到的奖励来计算的。例如，返回可以是多个时间步长上多个奖励值的累积。

[0101] 用于定义系统如何学习的参数中的一些是折扣因子γ和自举参数λ。下面参考等式(6)和(7)更详细地讨论这些参数。

[0102] 折扣因子γ确定返回的时间尺度。接近γ＝1的折扣因子提供了将奖励累积到未来的长远(long-sighted)目标，而接近γ＝0的折扣因子提供了优先考虑短期奖励的短视(short-sighted)目标。即使在期望长远的问题中，经常观测到γ<1的折扣因子值得到更好的结果，尤其是在早期学习期间。众所周知，多种算法在较低的折扣因子值下收敛得更快，但是太低的折扣因子值会导致次优策略。因此，在实践中，最好首先针对短视范围(short-sighted horizon)进行优化，例如，首先令γ＝0，然后在后面的阶段反复地增加折扣因子值。

[0103] 返回也可以被自举到不同的时间范围。n步返回在n个时间步长上累积奖励，然后在第n个时间步长处添加值函数。λ返回是n步返回的几何加权组合。在任一情况下，参数n或λ对于算法的性能(权衡偏差和方差)可能是重要的，并且因此对这些参数的有效选择是期望的。

[0104] 图2示出了根据本说明书的应用元学习的强化学习系统。这里，除了策略之外，还通过将返回函数视为具有可调元返回参数或元参数η的参数函数来学习返回函数本身。例如，这种元参数η可以包括折扣因子γ或自举参数λ。为避免疑问，迄今为止描述的“元返回参数”和“返回参数”等同于下文所描述的“元参数”η。

[0105] 元参数η在代理与状态或环境的交互过程中被调整，允许所述返回既适应特定的问题，也随着时间动态地适应不断变化的学习情境。因此，本文描述了一种实用的基于梯度的元学习方法，并且示出了其可以提高大规模深度强化学习应用的性能。

[0106] 回到图2，本强化学习系统200类似于图1的系统，因为它包括基于策略210和经验250确定动作的代理205。策略210由可以存储在本地存储器中的策略参数215定义。代理205可以使用神经网络来实现。

[0107] 强化学习系统200还包括策略训练模块230和返回函数训练模块240。策略训练模块230被配置为更新策略参数215以训练策略210。策略参数215基于返回来更新，该返回使用由返回参数225所定义的返回函数220、基于经验250来计算。返回参数225可以存储在本地存储器中。

[0108] 强化学习系统200还包括返回函数训练模块240。返回函数训练模块240被配置为使用元目标函数260、基于经验250来更新返回参数225。通过更新(训练)返回函数，系统能够学习更好的返回函数，从而改善对策略的训练。这允许更快且更有效地制定(reach)更准确的策略。

[0109] 在一个实施例中，返回函数训练模块240被配置为更新返回参数以减少返回参数中的误差。这可以相对于代理205用来确定来自动作的预期返回(并且从而确定提供最高预期返回的动作)的值函数。例如，元目标函数可以计算一个或多个经验的返回与值之间的均

方误差，并且返回函数训练模块240可以被配置为更新返回参数以减少(最小化)该均方误差。

[0110] 如上所述，强化学习系统200包括策略210、返回函数220，以及值函数，策略210包括一个或多个策略参数215，返回函数220包括一个或多个返回参数225。系统200从强化神经网络(其中强化神经网络可以形成或可以不形成系统200的部分)检索多个经验250，该强化神经网络被配置为控制与环境交互的代理执行任务，以尝试基于强化学习神经网络的一个或多个策略参数215来实现指定的结果。

[0111] 多个经验250中的每一个都包括表征环境状态的观测数据、代理响应于观测数据而执行的动作以及响应于动作而接收到的奖励。

[0112] 在一些实现方式中，系统200可以生成经验250(即，强化学习神经网络可以形成系统200的部分)。可替代地，系统200可以例如从存储装置或从外部系统访问多个经验250。在后一种实现方式中，随着策略参数215被更新，这些参数可以与神经网络共享以训练神经网络。

[0113] 在本实施例中，系统200被配置为使得多个经验250被分为第一经验集和第二经验集，其中第一经验集和第二经验集中的每一个可以结合返回参数225(也被称为元返回参数或元参数)使用来更新策略210的策略参数215。例如，这可以使用策略训练模块230来实现。元目标函数260然后被用来调整返回参数225。例如，这可以使用返回函数训练模块240来实现。这个过程可以重复多次，以迭代地更新策略参数和返回参数。

[0114] 当被实现时，处理器可以存储接收经验250的代理205。处理器的策略训练模块230可以基于经验250更新策略参数215，并且这进而更新了由代理205所执行的策略210。处理器的返回函数训练模块240然后可以调整存储在存储器中的返回参数225。策略训练模块230可以访问更新的返回参数。因此，这迭代地更新策略210和返回函数220。

[0115] 图3示出了系统200在优化策略210和返回函数220时采取的过程300。在步骤302中，系统200检索多个经验250。在一些实现方式中，系统200可以生成经验250(即，强化学习神经网络205可以形成系统200的部分)。可替代地，系统200可以从包括策略参数的外部系统检索多个经验。在后一种情况下，每次策略参数215被更新时，它们被发送到外部系统，用于生成下一个经验集。例如，多个经验250可以由强化学习系统200本身在线生成，或者可以从外部强化学习神经网络205获得。

[0116] 在步骤304中，然后使用更新函数240并基于多个经验250的第一经验集250a来更新策略参数215，以形成更新的策略参数。

[0117] 在步骤306中，基于多个经验250的第二经验集、元目标函数260和返回函数220来交叉验证更新的策略参数。

[0118] 在步骤308中，基于对更新的策略参数的交叉验证和元目标函数260来更新返回参数，以形成一个或多个元参数225，然后元参数225更新更新函数和返回函数220。在随后的参数更新中，基于最近的交叉验证来更新来自先前更新的现有元参数225。

[0119] 然后，在步骤310中，系统确定是否已经达到结束标准。结束标准可能是最大迭代次数或预定性能水平。性能水平可以是策略的性能(例如，预定义的累积奖励或返回)或返回函数的性能(例如，返回函数相对于真实(ground truth)返回的误差)。如果没有达到结束标准，系统200返回到步骤302，以进行新迭代以进行新的参数更新。如果满足结束标准，

则系统200完成训练，并在步骤312中输出结果。

[0120] 输出可能是优化后的策略(策略参数集)或优化后的返回函数(返回参数集)，如在最近的迭代中所确定的。策略可以用于实现经训练的代理。返回函数可以用来帮助更有效地训练更多强化学习代理。

[0121] 如前所述，在使用第二经验集交叉验证更新的策略参数之前，使用第一经验集更新策略参数的优点在于，避免过拟合并且减少用于每次迭代更新的训练数据量。

[0122] 更详细地，值函数$v_\theta(S)$和策略$\pi_\theta(A|S)$210由具有参数θ215的神经网络来近似。为了更好地逼近值函数和策略210，该方法包括更新函数，

[0123] $\theta' = \theta + f(\tau, \theta, \mu)$ #(1)

[0124] 该更新函数从一系列经验$\tau_t = \{S_t, A_t, R_{t+1}, \cdots\}$调整参数，其中S表示状态，A表示动作，R表示奖励，并且t表示已执行的参数更新次数。更新函数f的性质由返回参数或元参数η225确定。

[0125] 本说明书中所描述的元梯度强化学习方法是使用连续的经验样本、基于在线交叉验证的原则的。基础强化学习方法被应用于第一经验集τ，并且其性能使用第二经验集τ′来测量。具体地，该方法从策略参数θ215开始，并将更新函数应用于第一经验集τ，产生新的参数θ′。这些更新的梯度$d\theta/d\eta$然后指示元参数η225如何影响这些新的策略参数θ′。然后，该方法利用可微元目标$J'(\tau', \theta', \eta')$，测量后续独立的第二经验集τ′上新的策略参数θ′的性能。当在第二经验集τ′上验证性能时，J′中的固定元参数η′被用作参考值。这样，形成了元参数η的可微函数，并且通过取元目标J′相对于η的导数并应用链式法则，可以获得η的梯度：

[0126] $$\frac{\partial J'(\tau', \theta', \eta')}{\partial \eta} = \frac{\partial J'(\tau', \theta', \eta')}{\partial \theta'} \frac{d\theta'}{d\eta} \quad \#(2)$$

[0127] 参数形成加法序列，因此策略参数更新的梯度$d\theta'/d\eta$可以表示为

[0128] $$\frac{d\theta'}{d\eta} = \frac{d\theta}{d\eta} + \frac{\partial f(\tau, \theta, \eta)}{\partial \eta} + \frac{\partial f(\tau, \theta, \eta)}{\partial \theta} \frac{d\theta}{d\eta} = \left(I + \frac{\partial f(\tau, \theta, \eta)}{\partial \theta}\right) \frac{d\theta}{d\eta} + \frac{\partial f(\tau, \theta, \eta)}{\partial \eta} \quad \#(3)$$

[0129] 其中I是单位矩阵。

[0130] 梯度$\partial f(\tau, \theta, \eta)/\partial \eta$较大，并且可以使用累积迹线(trace)被近似为$z \approx d\theta/d\eta$，这样

[0131] $$z' = \mu z + \frac{\partial f(\tau, \theta, \eta)}{\partial \eta} \quad \#(4)$$

[0132] 也就是说，更新的策略参数θ′相对于返回参数η的梯度($d\theta'/d\eta$近似为z')可以通过将(使用策略参数θ和返回参数η在第一经验集τ上所评估的)更新函数相对于返回参数η的微分($\partial f(\tau, \theta, \eta)/\partial \eta$)加到先前梯度z(策略参数θ相对于返回参数η的梯度)上来迭代地计算。

[0133] 公式(3)的梯度可以使用固定元参数η来定义，或者被在线适配。为了进行这种适配，参数$\mu \in [0,1]$衰减该迹线，且只关注最近进行的更新。例如，选择$\mu = 0$会产生仅考虑元参数η对策略参数θ的即时影响的迹线。

[0134] 然后，元参数η225可以被更新为更新的元参数，以优化元目标函数J′260。

[0135] $$\Delta\eta = -\beta \frac{\partial J'(\tau',\theta',\eta')}{\partial \theta'} z' \qquad \#(5)$$

[0136] 这里,β是用于更新元参数η225的学习率。

[0137] 例如,这种更新可以通过应用随机梯度下降来在元梯度的方向上更新元参数η来完成。可替代地,元目标函数J′可以通过任何其他已知的梯度上升或下降方法来优化。

[0138] 然后在检索下一组多个经验250时,更新的元参数可以用作下一次迭代的元参数η225。

[0139] 下面的潜在实现方式考虑元参数η225用于使用时间差更新进行预测的情况,以及元参数225用于控制的情况,其中这是使用规范(canonical)动作-评价更新函数和策略梯度元目标来实现的。本领域技术人员将会理解,使用这种元梯度方法进行强化学习的多种其他替代实现方式也是可能的。

[0140] 图4示出了更新一个或多个元参数η并将其应用于返回函数G的步骤400。在策略参数θ已经被更新为更新的策略参数θ′并被交叉验证之后,该过程形成图3中先前示出的步骤308的一种实现方式。

[0141] 在步骤402中,确定返回函数G220相对于一个或多个元参数η225的梯度。

[0142] 在步骤404中,确定更新函数f相对于一个或多个元参数η225的梯度。

[0143] 在步骤406中,确定元目标函数J′260相对于一个或多个元参数η225的梯度。

[0144] 除了图4所示的步骤之外,且取决于为系统200所选择的元目标函数J(τ,θ,η)的选择,还可以计算值函数相对于策略参数215的梯度(即,$\partial v_\theta(S)/\partial\theta$),以确定更新函数f相对于一个或多个元参数η225的梯度。

[0145] 因此,可以为一个或多个元参数η225来评估元梯度,并且可以在步骤408中相应地调整一个或多个元参数η225,以确保最优返回函数G,从而形成更新的元参数。然后,这些更新的元参数可以在随后的更新迭代中用作元参数η225(其中得出结论,尚未达到最优返回函数)。

[0146] 更详细地,返回函数$G_\eta(\tau_t)$220被定义为经验的片段或截断n步序列的函数,即$\tau_t = \{S_t, A_t, R_{t+1}, \cdots, S_{t+n}\}$。如前所述,返回函数$G_\eta$的性质由一个或多个元参数η确定。

[0147] n步返回函数G_η220在序列上累积奖励,然后从值函数自举,从而

[0148] $$G_\eta(\tau_t) = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n v_\theta(S_{t+n}) \qquad \#(6)$$ 其中$\eta = \{\gamma, n\}$。

[0149] 自举参数λ返回函数或λ返回是n步的几何混合,因此返回函数G_η可以被重新定义为

[0150] $$G_\eta(\tau_t) = R_{t+1} + \gamma(1-\lambda) v_\theta(S_{t+1}) + \gamma\lambda G_\eta(\tau_{t+1}) \qquad \#(7)$$

[0151] 其中$\eta = \{\gamma, \lambda\}$。λ返回具有相对于元参数γ和λ225完全可微的优点。

[0152] 在这种情况下,选择的元参数η,γ和λ,可以被认为是导致返回终止($\gamma = 0$)、自举($\lambda = 0$)或继续到下一步长($\gamma = 1$或$\lambda = 1$)的门(gate)或条件。

[0153] 传统上,典型的强化学习算法将手动选择元参数η,诸如折扣因子γ和自举参数λ,并且这些将在整个训练中保持固定。然而,在本文所描述的强化学习系统中,返回函数G由元参数η参数化的,然后元参数η可以被微分,以便理解返回函数G对η的依赖性。这进而允许确定更新函数f相对于元参数η的梯度$\partial f/\partial\eta$,因此也可以确定元目标函数J′的元梯度

$\partial J'(\tau', \theta', \eta')/\partial \eta$。这允许系统评估哪个返回函数G 220导致最优性能，并根据公式(2)至(5)相应地调整元参数η225。

[0154] 在系统200的特定实现方式中，用于预测的规范TD(λ)算法可以用于基于所选元参数η225来做出关于最优返回函数的预测。TD(λ)算法的目标是最小化值函数逼近器$v_\theta(S)$和λ返回$G_\eta(\tau)$之间的平方误差，

[0155] $$J(\tau,\theta,\eta)=\left(G_\eta(\tau)-v_\theta(S)\right)^2\frac{\partial J(\tau,\theta,\eta)}{\partial\theta}=-2\left(G_\eta(\tau)-v_\theta(S)\right)\frac{\partial v_\theta(S)}{\partial\theta} \quad \#(8)$$

[0156] 这里，τ是从初始状态S开始的第一经验集，并且$\partial J(\tau,\theta,\eta)/\partial\theta$是半梯度。例如，λ返回被视为常数。

[0157] TD(λ)更新函数f(τ,θ,η)应用随机梯度下降来更新代理的策略参数θ215，以降低目标相对于策略参数θ215的梯度，从而

[0158] $$f(\tau,\theta,\eta)=-\frac{\alpha}{2}\frac{\partial J(\tau,\theta,\eta)}{\partial\theta}=\alpha\left(G_\eta(\tau)-v_\theta(S)\right)\frac{\partial v_\theta(S)}{\partial\theta} \quad \#(9)$$

[0159] 这里，α是用于更新策略参数θ的学习率。这里的更新函数f相对于元参数η225是可微的，因此

[0160] $$\frac{\partial f(\tau,\theta,\eta)}{\partial\eta}=-\frac{\alpha}{2}\frac{\partial^2 J(\tau,\theta,\eta)}{\partial\theta\,\partial\eta}=\alpha\frac{\partial G_\eta(\tau)}{\partial\eta}\frac{\partial v_\theta(S)}{\partial\theta} \quad \#(10)$$

[0161] 在该实现方式中，元梯度预测的目的是在实现最佳预测准确度的方向上调整元参数η225。这是在前面所描述的步骤中测量的，并在图3的步骤306中所示出，其中，使用均方误差(MSE)元目标函数J′并取其半梯度、基于从状态S′开始的第二经验集τ′，对更新的策略参数θ′进行交叉验证，其形式为

[0162] $J'(\tau',\theta',\eta')=(G_\eta(\tau')-v_{\theta'}(S'))^2$

[0163] $$\frac{\partial J'(\tau',\theta',\eta')}{\partial\theta'}=-2\left(G_{\eta'}(\tau')-v_{\theta'}(S')\right)\frac{\partial v_{\theta'}(S')}{\partial\theta'} \quad \#(11)$$

[0164] 因此，元目标函数的元梯度$\partial J'(\tau',\theta',\eta')/\partial\theta'$结合公式(2)至(5)来确定和实现，以到达元参数η的必要更新。

[0165] 在该实现方式中，元目标函数J′可以使用无偏(unbiased)且长远的返回函数G220，例如使用η′＝{γ′,λ′}，其中γ′＝1且λ′＝1。

[0166] 在系统200的另外的实现方式中，元梯度可以被应用于控制，诸如A2C动作-评价算法。在此实现方式中，动作-评价更新函数将预测和控制两者结合到对策略的单次更新中。

[0167] A2C元目标函数的半梯度定义为：

[0168] $$-\frac{\partial J(\tau,\theta,\eta)}{\partial\theta}=\left(G_\eta(\tau)-v_\theta(S)\right)\frac{\partial\log\pi_\theta(A|S)}{\partial\theta}+c\left(G_\eta(\tau)-v_\theta(S)\right)\frac{\partial v_\theta(S)}{\partial\theta}+d\frac{\partial H\left(\pi_\theta(.|S)\right)}{\partial\theta} \quad \#(12)$$

[0169] 在该公式中，第一项表示配置策略π_θ210以最大化返回函数220的测量奖励的控制目标。第二项表示配置值函数逼近器v_θ以准确地估计返回函数$G_\eta(\tau)$的返回的预测目标。第

三项是用于正则化(regularize)策略210的熵H的项,并且c和d是对元目标函数中的不同项进行适当加权的系数。

[0170] A2C更新函数f(τ,θ,η)应用随机梯度下降来更新策略参数θ215。该更新函数相对于元参数η225是可微分的,因此

[0171] $$f(\tau,\theta,\eta) = -\alpha\frac{\partial J(\tau,\theta,\eta)}{\partial\theta}$$

[0172] $$\frac{\partial f(\tau,\theta,\eta)}{\partial\eta} = \alpha\frac{\partial G_\eta(\tau)}{\partial\eta}\left[\frac{\partial\log\pi_\theta(A|S)}{\partial\theta} + c\,\frac{\partial v_\theta(S)}{\partial\theta}\right] \qquad \#(13)$$

[0173] 这里,α是当从一个或多个先前策略参数更新一个或多个策略参数时所应用的学习率。

[0174] 在该实现方式中,对元目标函数260的选择是用于确保返回函数最大化代理的性能的选择。这可以通过使用如下形式的策略梯度目标来实现:

[0175] $$\frac{\partial J'(\tau',\theta',\eta')}{\partial\theta'} = \left(G_{\eta'}(\tau') - v_{\theta'}(S')\right)\frac{\partial\log\pi_{\theta'}(A'|S')}{\partial\theta'} \qquad \#(14)$$

[0176] 这里,$G_{\eta'}$是进一步的返回函数,当被应用于第二经验集τ′时,其根据从该进一步的返回函数的返回来评估更新的策略,$v_{\theta'}(S')$是状态S′的与更新的策略相关联的值函数,并且$\pi_{\theta'}(A'|S')$是动作A′响应于状态S′的更新的策略。

[0177] 因此,该公式使用第二经验集τ′、考虑在η′的情况下计算的返回,评估更新的策略参数θ′的成功。

[0178] 在该实现方式中,当更新的策略参数θ′通过使用元目标函数的交叉验证来评估时,可以使用表示代理的真实目标的良好近似的固定元参数η′。这可以包括基于在实践中表现良好的值来选择η′的合理值。

[0179] 在这种实现方式下的元梯度学习算法可以以下面的方式实现。首先,使用公式(12)中所示的A2C更新函数、基于第一经验集τ更新策略参数θ215。这和公式(13)中所示的更新函数的梯度$\partial f(\tau,\theta,\eta)/\partial\eta$被累加到如前面公式(4)中所示的迹线z中。然后,使用公式(14)中所示的策略梯度元目标$\partial J'(\tau',\theta',\eta')/\partial\theta'$、在第二经验集τ′上交叉验证性能。最后,元参数η225然后可以根据公式(2)至(5)、根据元目标函数260的梯度来更新。

[0180] 上述实现方式的一个方面是返回函数$G_\eta(\tau)$220是非平稳的,因为整个训练过程中与元参数η225一起更新。这可能导致值函数v_θ变得不准确,因为它可能接近过期(out of date)返回。

[0181] 例如,值函数v_θ最初可以在γ=0的训练过程的开始处形成良好的近似,但是在γ已经被适配到γ=1之后,在之后的训练过程中形成较差的近似。

[0182] 这同样适用于策略π220,其也可能基于过期返回来形成。

[0183] 因此,为了解决值函数v_θ和策略π220的这种非平稳方面,可以实现类似于通用值函数近似(universal value function approximation,UVFA)的方法。这里,元参数η作为附加输入被提供以限定值函数v_θ和策略π220,其形式为

[0184] $$v_\theta^\eta(S) = v_\theta([S;\boldsymbol{e}_\eta]) \qquad \pi_\theta^\eta(S) = \pi_\theta([S;\boldsymbol{e}_\eta])$$ $e_\eta = W_\eta\eta$

[0185] 其中e_η是η的嵌入，[s;e_η]是向量s和e_η的级联，并且W_η是在训练期间通过反向传播更新的嵌入矩阵(或行向量，对于标量η)。

[0186] 在该实现方式中，代理然后明确地学习对于元参数η225的任何给定值、最合适的值函数v_θ和策略π220。这具有允许元参数η被调整而无需等待逼近器“跟上”的优点。

[0187] 到目前为止所描述的系统200的方法和各种实现方式可以按比例放大。例如，为了提高效率，A2C目标和元目标函数可以在n步经验集内的所有时间步长上累积。A2C目标函数可以通过RMSProp无需动量(momentum)，进行优化。这导致元参数η225的可微函数，并且因此可以类似于随机梯度下降(见公式13)被替换。就像在IMPALA中一样，可以使用基于V形迹线返回的非策略(off-policy)校正。为了更有效地进一步实现，可以并行计算小型经验集，或者经验集可以被重复使用两次，用于更新函数和交叉验证。例如，为了减少元学习所需的数据，经验可以用于代理训练和元学习两者。例如，经验τ可以用于将θ更新为θ′，并且可以经由在经验τ′上评估J′来验证此更新的性能。反之亦然，τ和τ′的角色可以交换，以便经验τ′可以用于将θ更新为θ′，并且可以经由在经验τ上评估J′来验证此更新的性能。这样，除了用于训练代理参数θ以对η进行元学习更新的数据之外，所提出的方法不需要额外的数据。

[0188] 对于由一个或多个计算机组成的系统来说，被配置为执行特定的操作或动作意味着该系统已经在其上安装了软件、固件、硬件或它们的组合，这些在操作中导致该系统执行操作或动作。对于被配置为执行特定操作或动作的一个或多个计算机程序来说，意味着该一个或多个程序包括指令，当该指令被数据处理设备装置时，使得该装置执行操作或动作。

[0189] 在本说明书中所描述的主题和功能操作的实施例可以在数字电子电路中、在有形具体化的计算机软件或固件中、在计算机硬件(包括在本说明书中所公开的结构和它们的结构等同物，或者在它们中的一个或多个的组合中)中实现。本说明书中所描述的主题的实施例可以被实现为一个或多个计算机程序，即被编码在有形的非暂时性程序承载上的计算机程序指令的一个或多个模块，用于由数据处理装置执行或控制数据处理装置的操作。可替代地或附加地，程序指令可以被编码在人工生成的传播信号(例如，机器生成的电、光或电磁信号)上，该信号被生成以编码信息，用于到合适的接收器装置的传输，以由数据处理装置执行。计算机存储介质可以是机器可读存储设备、机器可读存储基底、随机或串行访问存储器设备，或者它们中的一个或多个的组合。然而，计算机存储介质不是传播信号。

[0190] 术语“数据处理装置”包含用于处理数据的所有种类的装置、设备和机器，包括例如可编程处理器、计算机或多个处理器或计算机。该装置可以包括专用逻辑电路，例如，现场可编程门阵列(FPGA)或专用集成电路(ASIC)。除了硬件之外，该装置还可以包括为所讨论的计算机程序创建执行环境的代码，例如，形成处理器固件、协议栈、数据库管理系统、操作系统或它们中的一个或多个的组合的代码。

[0191] 计算机程序(也可以被称为或描述为程序、软件、软件应用、模块、软件模块、脚本或代码)可以用任何形式的编程语言(包括编译或解释语言、声明或过程语言)编写，并且计算机程序可以以任何形式部署，包括作为独立程序或作为模块、组件、子例程或适合在计算环境中使用的其他单元。计算机程序可以，但不是必须，对应于文件系统中的文件。程序可以被存储在保存其他程序或数据的文件的部分(例如存储在标记语言文档中、存储在专用于所讨论的程序的单个文件中、或者存储在多个协同文件(例如存储一个或多个模块、子程序或代码部分的文件)中的一个或多个脚本)中。计算机程序可以被部署为在一个计算机上

或者位于一个站点或分布在多个站点并通过通信网络互连的多个计算机上执行。

[0192] 在本说明书中，“引擎”或“软件引擎”是指提供不同于输入的输出的软件实现的输入/输出系统。引擎可以是编码的功能块，诸如库、平台、软件开发工具包(SDK)或对象。每个引擎可以在任何适当类型的计算设备(例如服务器、移动电话、平板计算机、笔记本计算机、音乐播放器、电子书阅读器、膝上型或台式计算机、PDA、智能电话或其他固定或便携式设备)上实现，计算设备包括一个或多个处理器和计算机可读介质。此外，引擎中的两个或更多个可以在相同的计算设备上实现，或者在不同的计算设备上实现。

[0193] 本说明书中所描述的过程和逻辑流程可以由一个或多个可编程计算机执行，该可编程计算机执行一个或多个计算机程序，以通过对输入数据进行操作并生成输出来执行功能。过程和逻辑流程也可以由专用逻辑电路来执行，并且装置也可以被实现为专用逻辑电路(例如，现场可编程门阵列(FPGA)或专用集成电路(ASIC))。例如，过程和逻辑流程可以由图形处理单元(GPU)来执行，并且装置也可以被实现为图形处理单元(GPU)。

[0194] 适于运行计算机程序的计算机包括例如通用或专用微处理器或两者，或者任何其他类型的中央处理单元。一般，中央处理单元将从只读存储器或随机访问存储器或者两者接收指令和数据。计算机的基本元件是用于执行(perform)或运行(execute)指令的中央处理单元以及一个或多个用于存储指令和数据的存储器设备。一般，计算机还将包括或可操作地耦合到用于存储数据的一个或多个大容量存储设备(例如磁盘、磁光盘或光盘)，以从其接收数据或向其传送数据，或两者兼有。然而，计算机不需要具有这样的设备。此外，计算机可以嵌入到另一个设备(例如移动电话、个人数字助理(PDA)、移动音频或视频播放器、游戏控制台、全球定位系统(GPS)接收器或便携式存储设备(例如通用串行总线(USB)闪存驱动器)，仅举几个示例)中。

[0195] 适用于存储计算机程序指令和数据的计算机可读介质包括所有形式的非易失性存储器、介质和存储器设备，包括例如半导体存储器设备(例如EPROM、EEPROM和闪存设备)；磁盘(例如内部硬盘或可移动磁盘)；磁光盘；以及CD ROM和DVD-ROM。处理器和存储器可以由专用逻辑电路来补充或并入专用逻辑电路。

[0196] 为了提供与用户的交互，本说明书中所描述的主题的实施例可以在计算机上实现，该计算机具有用于向用户显示信息的显示设备(例如阴极射线管(CRT)或液晶显示器(LCD)监视器)，以及键盘和指示设备(例如鼠标或轨迹球)，用户可以通过键盘和指示设备向计算机提供输入。也可以使用其他类型的设备来提供与用户的交互；例如，提供给用户的反馈可以是任何形式的感官反馈，例如视觉反馈、听觉反馈或触觉反馈；并且可以以任何形式接收来自用户的输入，包括声音、语音或触觉输入。此外，计算机可以通过向用户使用的设备发送文档和从用户使用的设备接收文档来与用户交互；例如，通过响应于从网络浏览器接收到的请求，将网页发送到用户客户端设备上的网络浏览器。

[0197] 本说明书中所描述的主题的实施例可以在计算系统中实现，该计算系统包括后端组件(例如作为数据服务器)，或者包括中间件组件(例如应用服务器)，或包括前端组件(例如具有图形用户界面或网络浏览器的客户端计算机，用户可以通过该图形用户界面或网络浏览器与本说明书中所描述的主题的实现进行交互)，或者一个或多个这样的后端、中间件或前端组件的任意组合。系统的组件可以通过任何形式或介质的数字数据通信(例如通信网络)来互连。通信网络的示例包括局域网(LAN)和广域网(WAN)，例如互联网。

[0198] 计算系统可以包括客户端和服务器。客户端和服务器一般彼此远离，并且通常通过通信网络进行交互。客户机和服务器的关系是通过运行在相应计算机上并且彼此具有客户机-服务器关系的计算机程序产生的。

[0199] 尽管本说明书包含多个具体的实现方式细节，但是这些不应被解释为对任何发明或所要求保护的范围的限制，而是对特定发明的特定实施例的特定特征的描述。本说明书中在单独实施例的上下文中所描述的某些特征也可以在单个实施例中组合实现。相反，在单个实施例的上下文中所描述的各种特征也可以在多个实施例中单独实现或者在任何合适的子组合中实现。此外，尽管特征可以在上文中被描述为在某些组合中起作用，并且甚至最初被如此要求保护的，但是在一些情况下，来自所要求保护的组合的一个或多个特征可以从该组合中被删除，并且所要求保护的组合可以针对子组合或子组合的变型。

[0200] 类似地，尽管在附图中以特定顺序描述了操作，但是这不应该理解为要求以所示的特定顺序或顺序地执行这些操作，或者要求执行所有示出的操作，以获得期望的结果。在某些情况下，多任务和并行处理可能是有利的。此外，上述实施例中的各种系统模块和组件的分离不应该被理解为在所有实施例中都需要这种分离，并且应该理解，所描述的程序组件和系统一般可以被一起集成在单个软件产品中或者封装到多个软件产品中。

[0201] 已经描述了主题的特定实施例。其他实施例在所附权利要求的范围内。例如，权利要求中所列举的动作可以以不同的顺序执行，并且仍然获得期望的结果。作为一个示例，附图中所描述的过程不一定需要所示的特定顺序或顺序地来实现期望的结果。在某些实现方式中，多任务和并行处理可能是有利的。

[0202] 可以根据以下条款提供某些实现方式：

[0203] 1.一种强化学习系统，包括一个或多个处理器，一个或多个处理器被配置为：

[0204] 从强化学习神经网络检索多个经验，该强化学习神经网络被配置为控制与环境交互的代理执行任务，以尝试基于强化学习神经网络的一个或多个策略参数来实现指定的结果，每个经验包括表征环境状态的观测数据、代理响应于观测数据而执行的动作以及响应于动作而接收到的奖励；

[0205] 使用基于奖励计算返回的返回函数、基于第一经验集更新强化学习神经网络的一个或多个策略参数；以及

[0206] 基于一个或多个更新的策略参数和第二经验集，更新返回函数的一个或多个返回参数，其中一个或多个返回参数经由使用相对于一个或多个返回参数被微分的元目标函数的梯度上升或下降方法来更新，其中元目标函数取决于一个或多个策略参数。

[0207] 2.根据条款1的强化学习系统，其中，更新一个或多个返回参数利用一个或多个更新的策略参数相对于一个或多个返回参数的微分。

[0208] 3.根据条款1的强化学习系统，其中，一个或多个处理器还被配置为迭代地：

[0209] 使用一个或多个更新的策略参数和一个或多个更新的返回参数，检索由强化神经网络生成的更新的经验；

[0210] 使用一个或多个更新的返回参数、基于第一更新经验集，进一步更新一个或多个策略参数；以及

[0211] 经由梯度上升或下降方法、基于进一步更新的策略参数和第二更新经验集，进一步更新一个或多个返回参数，

[0212] 直到达到结束条件。

[0213] 4.根据条款1的强化学习系统，其中，更新一个或多个返回参数包括：应用进一步的返回函数作为元目标函数的部分，并且当被应用于第二经验集时，根据来自该进一步的返回函数的返回来评估更新的策略。

[0214] 5.根据条款1的强化学习系统，其中，对一个或多个策略参数的更新应用以一个或多个返回参数为条件的策略和值函数中的一个或多个。

[0215] 6.根据条款5的强化学习系统，其中，以一个或多个返回参数为条件是经由对一个或多个返回参数的嵌入来进行的。

[0216] 7.根据条款1的强化学习系统，其中，一个或多个参数包括返回函数的折扣因子。

[0217] 8.根据条款1的强化学习系统，其中，一个或多个处理器还被配置为：

[0218] 基于第二经验集更新强化学习神经网络的一个或多个策略参数；以及

[0219] 基于一个或多个更新的策略参数和第一经验集，更新返回函数的一个或多个返回参数，其中一个或多个返回参数经由梯度上升或下降方法来更新。

[0220] 9.根据条款1的强化学习系统，其中，被微分的元目标函数为：

[0221] $$\frac{\partial J'(\tau';\boldsymbol{\theta}_{t+1},\eta')}{\partial \eta_t}=\left(\frac{\partial J'(\tau';\boldsymbol{\theta}_{t+1},\eta')}{\partial \boldsymbol{\theta}_{t+1}}\right)^{\top}\frac{\partial \boldsymbol{\theta}_{t+1}}{\partial \eta_t}$$

[0222] 其中：

[0223] η_t是一个或多个返回参数；以及

[0224] $J'(\tau',\theta_{t+1},\eta')$是以第二经验集$\tau'$、一个或多个更新的策略参数$\theta_{t+1}$和形成元目标函数的部分的进一步的返回函数的一个或多个进一步的返回参数η'为条件的元目标函数。

[0225] 10.根据条款9的强化学习系统，其中：

[0226] $$\frac{\partial \boldsymbol{\theta}_{t+1}}{\partial \eta_t}=\frac{\partial}{\partial \eta_t}\boldsymbol{\theta}_t+\alpha\nabla_{\eta_t}G_{\eta_t}(\mathbf{x})\left[\nabla_{\theta_t}\log\pi_{\theta_t}(a|s)+c\nabla_{\theta_t}V_{\theta_t}(s)\right]$$

[0227] 其中：

[0228] α是当从一个或多个先前策略参数更新一个或多个策略参数时所应用的学习率；

[0229] $G_{\eta(x)}$是基于一个或多个返回参数η_t、计算第一经验集x(其中x对应于τ)的返回的返回函数；

[0230] π_{θ_t}是用于强化学习神经网络从状态s确定动作a的策略，该策略π_{θ_t}根据一个或多个策略参数θ_t操作；

[0231] c是系数；并且

[0232] $V_{\theta_t}(s)$是基于一个或多个策略参数θ_t确定状态s的值的值函数。

[0233] 11.根据条款9的强化学习系统，其中：

[0234] $$\nabla_{\theta_{t+1}}J'(\tau';\boldsymbol{\theta}_{t+1},\eta')=(G_{\eta'}(\tau')-V_{\theta_{t+1}}(s'))\nabla_{\theta_{t+1}}\log\pi_{\theta_{t+1}}(a'|s')$$

[0235] 其中：

[0236] $G_{\eta'}(\tau')$是基于一个或多个进一步的返回参数η'、根据第二经验集τ'计算返回的进一步返回函数；

[0237] $\pi_{\theta_{t+1}}$是用于强化学习神经网络从取自第二经验集τ'的状态s'确定动作a'的策

略，该策略$\pi_{\theta_{t+1}}$根据一个或多个更新的策略参数θ_{t+1}操作；并且

[0238] $V_{\theta_t+1}(s')$是基于一个或多个更新的策略参数θ_{t+1}确定状态s′的值的值函数。

[0239] 12.根据条款9的强化学习系统，其中，一个或多个进一步的返回参数保持固定。

[0240] 13.根据条款9的强化学习系统，其中，更新一个或多个返回参数包括计算：

[0241] $$\eta_{t+1} = \eta_t + \alpha_\eta \frac{\partial J'(\tau'; \boldsymbol{\theta}_{t+1}, \eta')}{\partial \eta_t}$$

[0242] 其中

[0243] η_{t+1}是一个或多个更新的返回参数；并且

[0244] α_η是用于更新一个或多个返回参数的学习因子。

[0245] 14.一种用于强化学习的计算机实现的方法，该方法包括：

[0246] 从强化学习神经网络检索多个经验，该强化学习神经网络被配置为控制与环境交互的代理执行任务，以尝试基于强化学习神经网络的一个或多个策略参数来实现指定的结果，每个经验包括表征环境状态的观测数据、代理响应于观测数据而执行的动作以及响应于动作而接收到的奖励；

[0247] 使用基于奖励计算返回的返回函数、基于第一经验集更新强化学习神经网络的一个或多个策略参数；以及

[0248] 基于一个或多个更新的策略参数和第二经验集，更新返回函数的一个或多个返回参数，其中一个或多个返回参数经由使用相对于一个或多个返回参数被微分的元目标函数的梯度上升或下降方法来更新，其中元目标函数取决于一个或多个策略参数。

[0249] 15.一个或多个计算机存储介质，其存储指令，当由一个或多个计算机运行指令时，指令使得一个或多个计算机执行条款14的方法的操作。

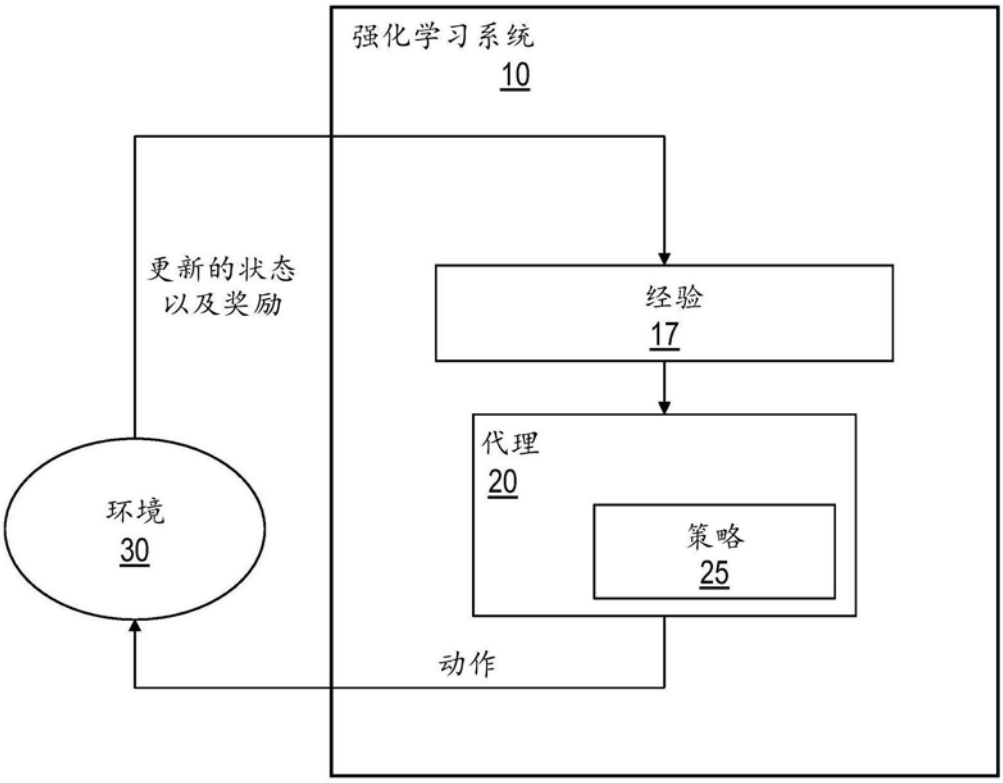


图1

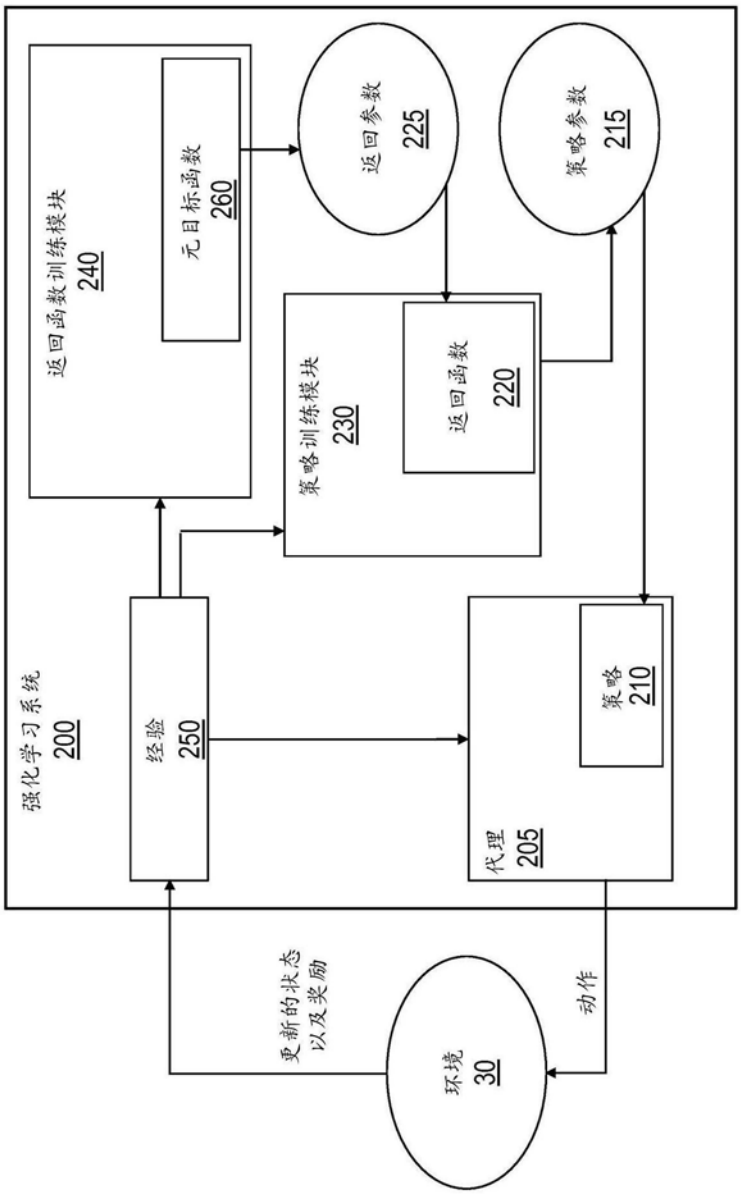

强化学习系统
200
返回函数训练模块
240
元目标函数
260
返回参数
225
策略参数
215
策略训练模块
230
返回函数
220
经验
250
代理
205
策略
210
更新的状态
以及奖励
环境
30
动作

图2

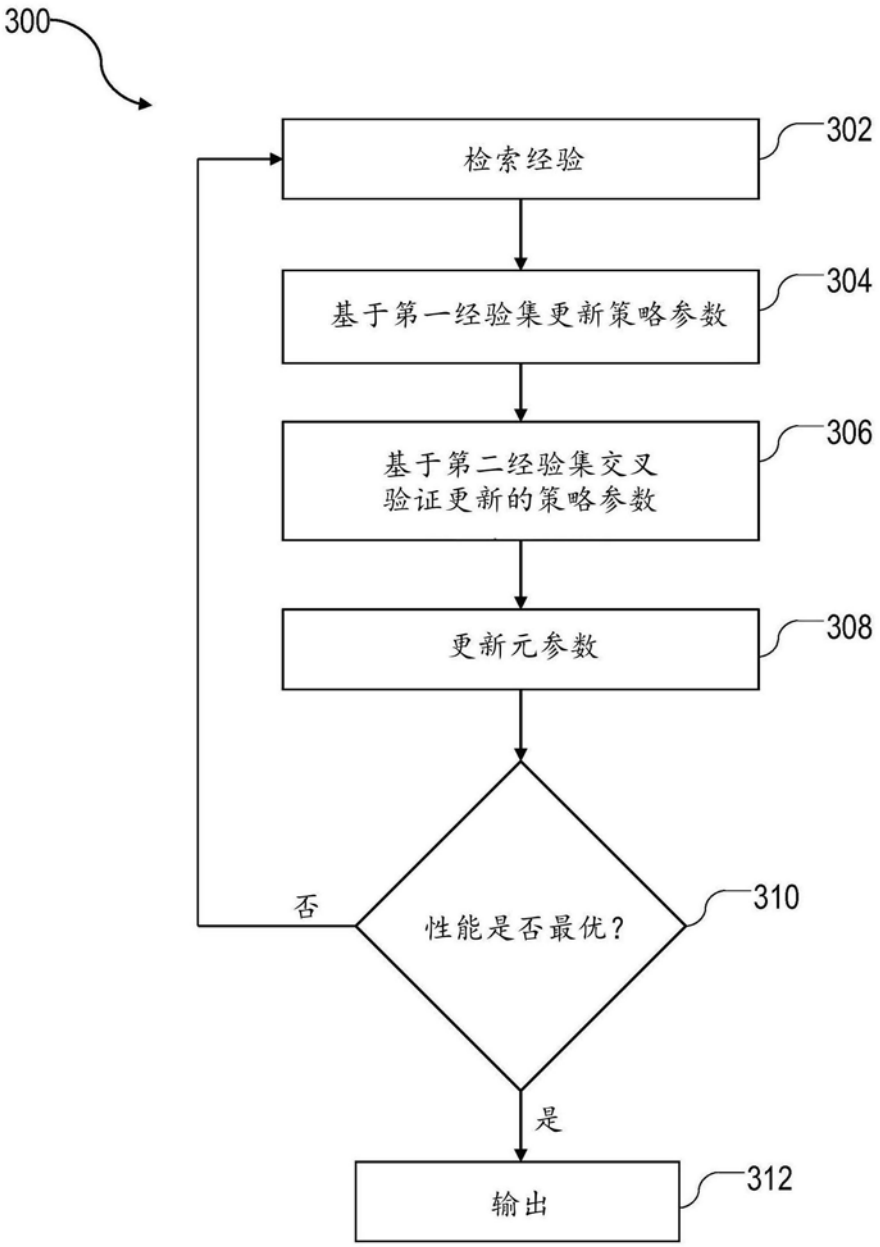

300
检索经验
302
基于第一经验集更新策略参数
304
基于第二经验集交叉
验证更新的策略参数
306
更新元参数
308
性能是否最优？
310
否
是
输出
312

图3

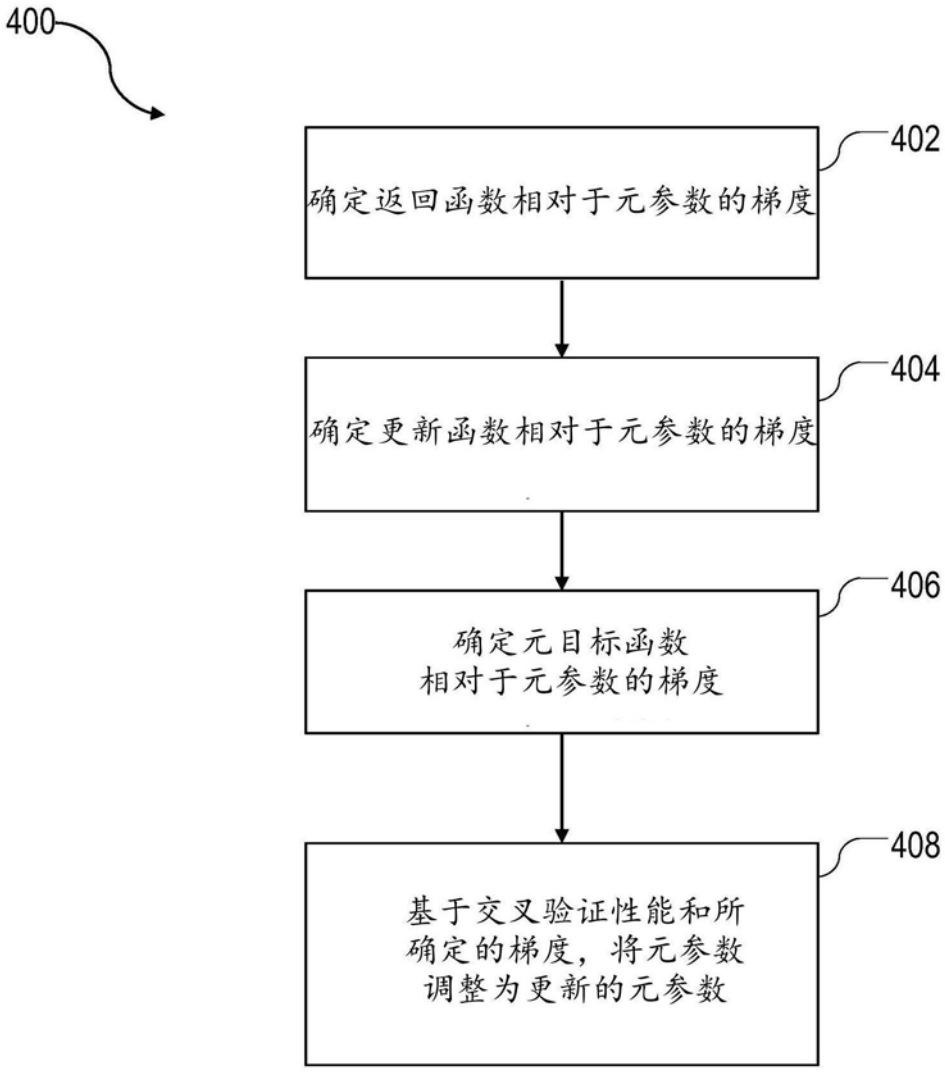

400
确定返回函数相对于元参数的梯度
402
确定更新函数相对于元参数的梯度
404
确定元目标函数
相对于元参数的梯度
406
基于交叉验证性能和所
确定的梯度，将元参数
调整为更新的元参数
408

图4