深度强化学习门诊决策

由于人工智能特别是深度学习最近的进步，已经实施了许多数据驱动的决策支持系统，以方便医生提供个性化护理。我们专注于本文中的深度强化学习（DRL）模型。DRL模型在计算机视觉和游戏玩法的任务中展示了人类级甚至卓越的性能，例如围棋和Atari游戏。然而，在临床决策优化中采用深度强化学习技术的情况仍然很少。我们在此提出第一份调查，总结了使用深度神经网络（DNN）进行临床决策支持的强化学习算法。我们还讨论了一些case studies，其中应用了不同的DRL算法来解决各种临床挑战。我们进一步比较和对比各种DRL算法的优点和局限性，并提供了如何为特定临床应用选择合适的DRL算法的初步方针。

简介

深度学习（深度神经网络）和强化学习技术的有效组合称为深度强化学习（DRL），最初是为游戏而发明的（Mnih等，2013; Silver等，2016），后来出现了作为解决大规模，高维状态和动作空间的复杂控制问题的有效方法（Mnih等，2015）。深度强化学习对这些领域的巨大应用依赖于对基础过程的知识（例如，游戏规则）。在医疗保健领域，临床过程非常动态。制定临床决策的“规则”通常不明确（Marik，2015）。比如，让我们考虑一下这样的门诊决策问题：患者可以从器官移植中受益吗？在什么情况下移植是一个比较好的选择？在移植后的处方中最好的药物应该是什么？其剂量是多少？为了找到这些问题的最佳决策，进行随机临床试验（RCT）通常是比较好的选择。然而，RCT在某些临床条件下可能不实用且不可行。因此，分析观测数据开始成为另一种选择。随着数据采集和DRL技术的进步，我们看到基于DRL的决策支持系统有很大的潜力来优化治疗建议。

技术意义

本调查文件总结并讨论了已应用于临床应用以提供临床决策支持的主要DRL算法类型。此外，我们讨论了这些DRL算法的利弊和其基础假设。本文旨在作为指导，帮助我们的受众为其特定的临床应用选择合适的DRL模型。据我们所知，这是关于DRL治疗推荐或临床决策支持的第一份调查。

临床相关

DRL被证明可以实现人类在学习特定领域（例如视频游戏，棋盘游戏和自主控制）中的复杂序列决策的能力。在医疗保健方面，DRL尚未广泛应用于临床应用。在本文中，我们调查了主要的DRL算法，这些算法在临床领域提供序列决策支持。我们相信，从大量收集的电子健康记录（EHR）数据中学习，DRL能够提取和总结优化新患者治疗所需的知识和经验。DRL还有可能通过自动探索各种治疗方案并估计其可能的结果来扩展我们对当前临床系统的理解。

强化学习的基础知识

MDP相关

强化学习（RL）（Sutton等，1998）是一种面向目标的学习工具，其中代理或决策者学习通过与环境交互来优化长期奖励的策略。在每个步骤中，RL代理获得关于其动作性能的评估反馈，从而允许其改善后续动作的性能（Kiumarsi等，2018）。在数学上，这种顺序决策过程称为马尔可夫决策过程（MDP）（Howard，1960）。

马尔可夫决策过程主要有4个组成部分：

- 状态空间S：在每个时间点t, 环境都在状态$s_t \in S$中。
- 动作空间A：在每个时间点t, agent选择一个动作$a_t \in A$, 该动作会影响下一个状态s_{t+1}
- 转移函数 $P(s_{t+1}|s_t, a_t)$：给定现在的状态和动作时候的下一个状态的概率，代表了与agent交互的环境。
- 奖励函数$r(s_t, a_t) \in R$： 通过给定的状态和动作对(s_t, a_t) 得到观测的反馈。

在临床环境中，我们将agent视为临床医生。我们可以将状态视为患者的健康状况。患者的状态可以取决于他的人口统计学（例如年龄，性别，种族等），身体和生理报告（例如，实验室测试，生命体征，医学图像报告等）和一些其他临床特征。动作是临床医生对患者起作用的治疗（例如，某些药物的处方，外科手术的顺序等）。转换函数$P\left(s_{t+1}|s_t, a_t\right)$ 可以被视为患者自己的生理系统，给定当前的健康状况和治疗方式，患者将进入下一个时间点的状态s_{t+1} 。如果情况得到改善，我们会给agent奖励，如果患者病情恶化或在干预后停滞不前，我们会对agent进行惩罚。

强化学习的agent的目标被定义为最大化累积奖励的期望，总奖励的期望是按照轨迹τ 遵从$p(\tau)$ 的分布得来的，$p(\tau)$ 可能是一个从状态到动作的映射$\pi(a|s)$ 。

$$\pi^* = \arg\max \underbrace{E_{\tau\sim p(\tau)}\left[\sum_t \gamma^t r\left(s_t, a_t\right)\right]}_{J} \quad (1)$$

该式中的J 就是我们想要关于策略$\pi(a|s)$ 的优化的目标，最优策略π^* 就是最大化目标函数J。

为了定义一个状态或者一个动作的好坏程度，我们分别定义了Q-function和Value-function：

$$Q^\pi\left(s_t, a_t\right) = \sum_{t'=t}^{T} E_\pi\left[r\left(s_{t'}, a_{t'}\right)|s_t, a_t\right] \quad (2)$$

$$V^\pi\left(s_t\right) = E_{a_t\sim\pi(a_t, s_t)}\left[Q^\pi\left(s_t, a_t\right)\right] \quad (3)$$

Q-function $Q^\pi(s_t, a_t)$ 是给定当前状态和动作对(s_t, a_t) 下的累积奖励的条件期望，Value-function $V^\pi(s_t)$ 是(s_t, a_t) 时根据策略π 采取动作a_t 得到的期望Q-value。

如果我们知道环境中所有可能状态和动作的实际Q-function或值函数，那么最优策略将很容易从(1)推导出来。一个agent会在每个状态s_t挑选一个能最大化Q-function或Value-function的动作。

深度强化学习

对于具有高维状态的环境，由于大规模存储的要求和高计算量的要求，表格表示法并不十分适用。因此深度强化学习通过采取深度神经网络(DNN)(Ciresan et al.,2012)来替代传统强化学习中用表格表示$Q(s_t, a_t)$, $V(s_t)$和 π 的方法。

不同的DRL算法以不同方式使用DNN。例如，在策略梯度RL（Sutton等，2000）中，DNN可用于优化策略$\pi(\theta)$，其中是θ 是DNN中的参数（权重）。DNN的输入是当前状态，输出是动作。通过DNN的梯度，在每个时间点，DNN中的参数会被更新，从而改进策略。

在Value-based RL(Kaelbling et al., 1996)中，DNN可以用来近似Value-function或者Q-function。对于Value-function而言，DNN利用当前的状态s_t 作为输入，输出是给定策略π下近似的值$\hat{V}^{\pi}\left(s_t\right)$ 。对于Q-function而言，DNN将状态动作对$\left(s_t, a_t\right)$ 作为输入，输出是$\hat{Q}_{\phi}\left(s_t, a_t\right)$ 的值，其中ϕ 是value-based RL中DNN的网络参数。在model-based RL中(Doya et al., 2002)，DNN被用来近似转移函数$p_{\phi}\left(s_{t+1} | s_t, a_t\right)$ 。

关键步骤

大多数RL算法可以分成三步：

- 收集样本
- 评估目标函数
- 提高策略

以策略梯度RL为例，在第一步中，RL算法与环境交互并按照未学习的（可能是随机初始化的）策略生成样本（即状态动作对）。状态动作对一般是按照轨迹τ 的有序集合。在第一步中，RL算法可以生成多条轨迹。第二步，RL算法通过计算目标函数J 来评估生成的样本的好坏。第三步，RL算法将通过增加访问奖励高的轨迹的机会，降低访问奖励低的轨迹的机会来最大化目标函数J ，这一步也叫策略更新。然后agent将循环这三个步骤，这个过程中策略就在"trial and error"中不断更新了。

大多数RL算法遵循这三个步骤，但是这三个步骤的重要性对不同算法而言是不同的。对于具体问题来说，哪种RL算法最适合也不尽相同。现在我们介绍一下各个步骤的成本。首先对于第一步来说，如果是从真实物理环境中生成样本，那么这一步代价十分昂贵，因为需要实时数据。但是如果是从模拟环境中（比如Atari游戏里）获得样本，那么相对来说要容易很多。

对于第二步，如果使用策略梯度来改进策略，可以通过"trail and error"来改进策略，那么目标函数J 将是来自这些trail的奖励之和。然而，对于model-based RL，使用DNN拟合模型的代价比较大，我们需要确保拟合模型收敛并接近实际模型。

在第三步中，策略梯度RL仅需要我们计算梯度并将梯度应用于策略。它仍然相对简单。但是，对于DNN的model-based RL，我们需要对大量时间进行反向传播以改进策略。
当涉及到真实世界的应用时，特别是在临床应用中，数据生成通常是一个大问题，因为它是实时的，并且许多观测数据（来自电子健康记录）是私有的，我们不容易获得。我们也不希望通过"trail and error"来估计数值，因为它通常不太道德且昂贵。因此，当我们选择RL算法时，我们需要了解这些约束。

在本文中，我们将首先集中讨论应用于临床决策支持的RL算法的主要类型，并简要介绍一种在临床应用中很少应用的RL算法。我们还将讨论与医疗保健领域的其他算法相比，特定算法不那么实用的根本原因。

RL算法的类型

Policy Gradient Reinforcement Learning

策略梯度RL遵从上述的三个步骤。第一步中，策略梯度RL首先生成随机$\pi(\theta)$ 并在此策略下生成对应的trajectories。在这个算法中我们并不需要知道转移函数，因为在策略梯度RL中不用学习具体的转移函数。在收集到了一些trajectories之后，我们通过第二步来估计$J(\theta)$， 它可以用刚才生成的一系列trajectories来近似 $J(\theta) \approx \frac{1}{N} \sum_i \sum_t r\left(s_{i, t}, a_{i, t}\right)$ 。在第三步中，我们计算其梯度，并用于提高策略$\pi(\theta)$ ，其新的策略参数$\theta^{\prime}=\theta+\alpha \nabla_{\theta} J(\theta)$

。接下来我们需要在更新之后的策略$\pi(\theta')$上采样新的状态动作对并且重复进行上述操作。我们管这种方式叫做on-policy(Krstic et al., 1995), 因为每当策略改变的时候，我们都需要在新的策略下采样新的数据。不同于on-policy的方法，off-policy的方法是我们不需要从策略中生成新的数据就可以提高策略的方法。因此策略梯度RL是一种on-policy的方法。

在策略梯度RL中，DNN用于在第三步里构造策略，DNN的输入是状态，输出是动作。图一说明的是策略梯度RL的MDP图示。通过$J(\theta)$的梯度更新DNN的权重可以学习到策略。

在临床环境中，与其他RL算法相比，策略梯度并不十分流行。主要是因为策略梯度的方法是on-policy的算法，它需要每次得到新的策略之后相对应的采样新的数据。整个算法是根据"trail and error"学习得到的。许多临床环境无法承担收集实时临床数据的成本。例如，要了解脓毒症患者用药剂量的最佳临床决策，使用"trail and error"是不道德的，并且也是耗时的。然而，策略梯度RL在机器人控制和计算机游戏等其他领域仍然很受欢迎，这些领域的环境中的模拟器可以承受"trail and error"的负担。

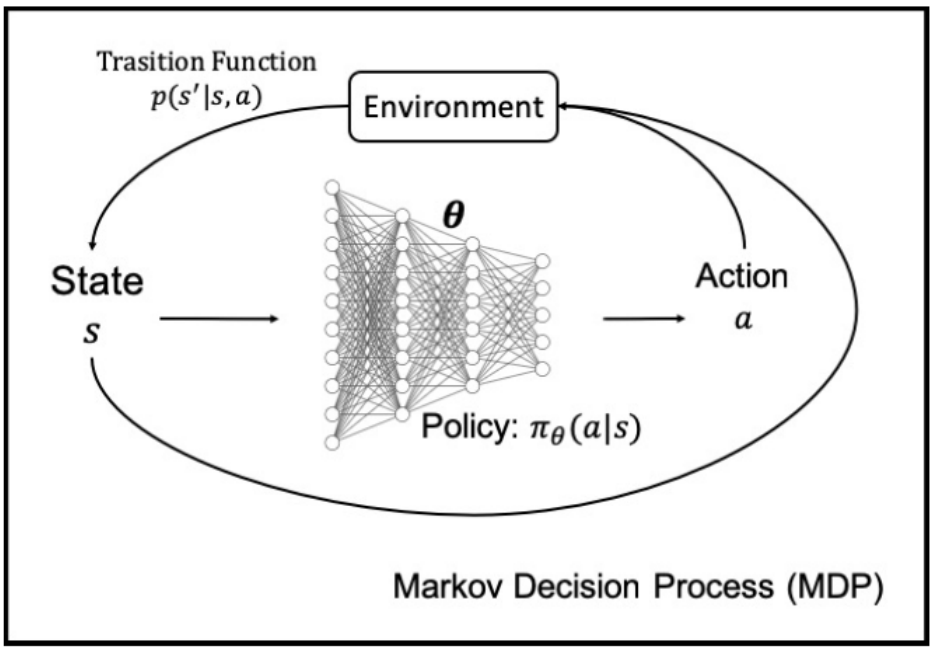

Figure 1: MDP for policy gradient RL algorithm

Value-based Reinforcement Learning

Value-based RL试图通过对具体的状态和动作的Value-function或者Q-function进行估计。Value-based RL 利用DNN从累积奖励中估计Value-function $V^\pi(s_t)$ 或Q-function $Q^\pi(s_t)$。对于Value-function来说，DNN输入的是s_t，输出是$V^\pi(s_t)$；对于Q-function来说，DNN的输入是状态动作对(s_t, a_t)，输出是$Q^\pi(s_t, a_t)$。DNN利用MSE进行优化。MSE定义为$L(\phi) = \frac{1}{2}\left\|\hat{V}_\phi^\pi(s_t) - y_{t+1}\right\|^2$，这里$y_{t+1}$ 是目标值，对于value-function来说是$y_{t+1} = \max_{a_{t+1}} \gamma Q^\pi(s_{t+1}, a_{t+1})$，对于Q-function来说是$y_{t+1} = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q^\pi(s_{t+1}, a_{t+1})$。这个优化过程可以通过很多种梯度下降的方式实现，比如SGD等。在第三步中，仅当从π' 获取的动作导致Q-function最大时，Value-based RL才将策略π 更新为π'.

这两种算法分别被称为**Fitted Value Iteration(FVI)** 和 **Fitted Q Iteration(FQI)** 。FQI中的一个特例是**Q-learning**，在Q-learning中我们只取一组样本(s_t, a_t, s_{t+1}, r_t) 用于估计Q-functiion，然后用一个gradient step进行参数更新，并采样下一组样本迭代优化。FVI，FQI和Q-learning是off-policy中的例子，他们并不像策略梯度RL方法那样方差很大。然而，他们的缺点在于当应用于非线性函数估计（比如DNN）的时候，他们经常会不收敛。此外，从Q-learning中采样的样本具有时序性，因此样本高度相关，这样我们的网络可能只能得到局部最优值。另外，由于目标值y_{t+1} 是只使用一组样本进行估计的，并且进行了多次单步估计的迭代，这可能会让目标值非常不稳定。

Lin(1992) 使用**'replay buffer'** 解决了样本高度相关的问题。带有**relay buffer**的Q-learning采样一个batch的样本(s_t, a_t, s_{t+1}, r_t) ，并且用这个batch来进行单步的梯度更新，然后采样下一个batch的样本放入replay buffer中。这种方式下，样本不再相关，batch中的多个样本也保证了梯度方差比较小。为了解决目标值不稳定的问题，Mnih et al.(2016)提出了**target network** 的方法，目标值y_{t+1} 通过固定的值来估计，并且在学习几轮之后才进行更新。Mnih et al.(2015) 通过将Q-learning与'replay buffer'和'target network'结合，提出了**Deep-Q-Network(DQN)**。然而，DQN并不完美。它与Q-learning同样具有容易高估Q-function的问题，这个问题是因为在估计目标值时的'max'项$y_{t+1} = r\left(s_t, a_t\right) + \gamma \max_{a_{t+1}} Q^{\pi}\left(s_{t+1}, a_{t+1}\right)$， 这里容易受到数据采集时候噪声的影响。其中一个解决方案是**Double Q-Learning** 的方法，即通过实现两个DNN来学习两个Q-function，用其中一个估计Q-value，另一个选择动作。使用两个Q-function是为了去除动作和Q-function中的噪声的相关性。

Value-based RL经常应用于临床环境中。

Actor-Critic Reinforcement Learning

Actor-Critic RL是将策略梯度RL与Value-based RL结合的算法。在第一步中，我们像策略梯度RL一样根据策略采样出状态动作对，不过在这里，状态动作对是根据策略$\pi(\theta)$ 行动的DNN(actor)生成的。在第二步里，我们利用$\hat{V}_{\phi}^{\pi}\left(s_t\right) = \sum_{i=t}^{T} E_{\pi_\theta}\left[r\left(s_i, a_i\right) | s_t\right]$ 通过另一个DNN(critic)采样出奖励之和。这个DNN的输入是s_t， 输出是$\hat{V}^{\pi}\left(s\right)$ 。我们引入一个新的项$\hat{A}^{\pi}\left(s_i, a_i\right)$ ，其定义为$\hat{A}^{\pi}\left(s_i, a_i\right) = r\left(s_i, a_i\right) + \hat{V}_{\phi}^{\pi}\left(s'\right) - \hat{V}_{\phi}^{\pi}(s)$ ，这里$\hat{V}_{\phi}^{\pi}\left(s'\right)$ 是下一个状态s' 的估计值。Advantage function告诉我们根据value-function来看，该动作比这个状态下的平均动作好多少。在第三步中，我们使用轨迹上的累积advantage $\hat{A}^{\pi}\left(s_i, a_i\right)$ 的梯度更新actor DNN上的参数θ来学习更好的策略。critic DNN可以根据监督学习来优化，$L(\phi) = \frac{1}{2}\left\|\hat{V}_{\phi}^{\pi}\left(s_t\right) - y_t\right\|^2$ ，这里的y_t 可以通过该轨迹上的奖励来进行蒙特卡洛估计，或者使用bootstrap的方式进行估计。Actor-Critic RL是off-policy算法，但是另一个版本也可以是on-policy的算法。不同点在于，第一步中，我们不再收集一个batch的trajectories，而是只收集一个trajectory并更新策略，然后从更新的策略中生成新的样本。

Model-based Reinforcement Learning

之前讨论的RL算法都是Model-free的RL算法，在Model-free RL中，我们假设我们不知道准确的转移函数$p(s_{t+1}|s_t, a_t)$。 所以即使给定现在的状态动作对，我们也不知道真正的下一个状态到底是什么。Model-free RL不会试图学习精确的转移函数，而是通过从环境中采样来绕过这个问题。然而，知道真正的转移函数经常会很有用，而且在某些情况下，比如我们自己定义规则的简单棋盘游戏里，可以明确知道转移函数。在临床应用中，大多数时候

我们无法确认精确的转移函数，但是我们会对环境的动态过程有一定的了解。比如临床医生通常都知道在用适当的药物剂量治疗患者后，患者将逐渐从患病恢复到健康状态。即使我们不了解环境的全貌，但是依然可以提出几种模型来估计真正的转移函数并且来进行优化。这就叫做Model-based RL。我们可以使用很多模型来估计转移函数，比如Gaussian Process(GP), DNN, Gaussian Mixture models(GMM)等模型。对于DNN的model-based RL来说，DNN的输入是(s_t, a_t)，输出是s_{t+1}。通常DNN是在第二步中用来设计成转移函数的。与DNN不同，GP的样本利用效率非常高，GP可以用很少的样本来对下一次的状态进行合理预测，由于大多数临床应用都存在数据利用的问题，因此GP在临床上十分有用。然而GP的坏处在于当转移函数不平滑的时候，GP会表现的很差。此外，如果样本数量很大并且在高维中，GP可能会非常慢。因此在临床环境中，当输入的状态是医学图像（高维度）的时候，DNN比GP更适用于model-based RL。

其他RL算法

Hierarchical Reinforcement Learning

Hierarchical RL包含两个层次的结构。低层网络就像一般RL算法一样，给出一个合适的策略。高层的网络有"meta-policy"，它用来训练选择在一个trajectory上使用哪种低层的策略。与随机初始化的策略相比，HRL更容易找到全局最优值，并且它可以从低层策略中学习到之前任务的相关知识。在临床环境中，由于人类交互的复杂行为，状态-动作空间可能十分巨大。因此在临床上应用HRL是非常适合的。然而，HRL的结构更难以训练，并且不合适的迁移学习可能会导致'negative transfer'，导致最终的策略并不一定比低层的策略好。

Recurrent Reinforcement Learning

在现实世界中，马尔科夫属性很难被满足，因此常见的问题经常难以用MDP的方式来解决。在医疗领域里，我们不太可能观测到完整的患者的临床状态。这种情况被称为部分观测的马尔科夫问题(POMDP)。POMDP问题的四元组为(S, A, R, O)，其中O是观测。传统的DQN在这里只在观测可以反映出潜在状态时候才有用。Hausknecht and Stone (2015)提出了一个DQN的扩展，用来解决POMDP的问题，该网络是把DQN中的全连接层换成了LSTM层。这种新的RL算法叫做Deep Recurrent Q-Network(DRQN)。

Inverse Reinforcement Learning

在临床中，大多数标准的RL算法都要手工设计奖励函数，然而我们并不知道真实的奖励是什么，这种奖励函数设计非常容易出错。Inverse RL可以让我们从专家演示里推断出正确的奖励函数，而无需手工设计奖励函数。Inverse RL是直接从专家的行为中学习，一般用模仿学习（imitation learning）来实现。然而模仿学习的缺陷在于专家可能选择不同并且倾向于采用不完美的策略，这将导致Inverse RL学到的是次优解。因此，通常在Inverse RL中，我们给出状态动作和次优策略。Inverse RL的目标是找到正确的奖励函数。然后我们可以使用学习到的奖励函数来获得比次优策略更好的新策略。奖励可以通过DNN学到，DNN的输入是由次优策略$\pi^{\#}$得到的状态动作对，输出是奖励函数$r_\Phi(s, a)$，Φ是我们学到的DNN的参数，在得到了$r_\Phi(s, a)$之后，我们可以用新的奖励函数来学习更好的策略。

临床应用中的DRL