# 动态噪声的竞争深度 Q 网络 *

张鑫，张席

(深圳大学 计算机与软件学院，深圳 518061)

**摘 要**：深度强化学习通常是通过在动作空间注入噪声来进行探索，但难以根据学习状况进行调整。另一种方法是在参数空间加入噪声，使得探索更加丰富，但会大幅减缓训练过程，使得收敛速度变慢。针对这个问题，提出了结合竞争网络结构和动态噪声的方法，即在动作空间与参数空间均加入噪声，动态地结合了各自优点。在训练前期以动作空间的噪声为主，减少神经网络的参数学习压力，从而更快的适应环境。利用测试平台 OpenAI Gym 进行对比实验，结果表明提出的方法取得了更好的学习性能，训练稳定性有明显提升。

**关键词**：深度 Q 网络；强化学习；探索与利用

**中图分类号**：TPxxx〔查询请参考 http://ztflh.xhma.com〕

# Dueling Deep Q Network with Dynamic Noise

Zhang Xin, Zhang Xi

(College of Computer Science and Software Engineering of Shenzhen University, Shenzhen 518061, China)

**Abstract**: Deep reinforcement learning generally engage in exploratory behavior through noise injection in the action space, but it is difficult to adjust according to the training situation. An alternative is to add noise directly to the parameter space, which can lead to a richer set of behaviors, but also slow down the training process and the convergence speed greatly. To address this problem, a method combining dueling network and dynamic noise is proposed, that is, adding noise both to the action space and parameter space. In the early stage of training, the noise in the action space is mainly used to reduce the parameter learning pressure of the neural network, so as to adapt to the environment more quickly. The OpenAI Gym was used for comparison experiment. The results show that the proposed method achieved better learning performance and significantly improved training stability.

**Key words**: DQN; Reinforcement Learning; exploration and exploitation

## 0 引言

强化学习的基本学习方法是通过智能体在一定时间步中，不断的采取动作与所在环境进行交互，观测到环境对当前智能体状态、动作的反馈信号后调整下一步动作，从而提升自身的表现性能。在结合深度学习之前，传统的强化学习算法如 Q-Learning[1]利用 Q 表来存储状态动作映射的 Q 值，但在高维度状态的环境中其状态空间巨大，保存每个状态的 Q 值会有存储空间浪费、查表效率低下等问题，故局限于低维度状态的环境。深度强化学习将深度学习的高维度感知处理能力与强化学习的决策能力相结合，取得了实质性的突破[2]。

Mnih V 等人[3]首次提出将深度学习与强化学习算法 Q-Learning 结合的方法，即 DQN (Deep Q-network) 深度强化学习，直接将预处理后的原始图片像素作为深度神经网络的输入，预测该输入对应的 Q 值，输出要选择的动作，在部分 Atari 2600 游戏[4]中取得了超过人类玩家的分数[5]。双 DQN[6] (Double DQN) 方法采用不同网络参数来对 Q 值进行估计，解决 DQN 的过估计问题。竞争网络[7] (Dueling Network)改进原本的网络结构，将原始 DQN 的输出解耦成两个分支，分别预测作为标量的状态值函数，与作为矢量的优势函数，后者中的每个值对应一个动作，两个函数相加输出每个动作的 Q 值，提升函数近似效果。为了平衡强化学习中探索和利用之间的关系，通常是在动作空间加入噪声，如 $\varepsilon$-greedy 策略[1]，但参数 $\varepsilon$ 是单调递减的线性函数，对于动态的学习过程，并不能完全适应。文献[8]和文献[9]分别提出两种在参数空间加入噪声的方法，引起网络输出变化，从而影响选择的动作。本文采用动作空间噪声与参数空间噪声动态结合的竞争深度 Q 网络针对控制问题进行研究，在 OpenAI Gym 提供的控制问题环境中取得了更高的分数。

**作者简介**：张鑫（1994-），男，广东揭阳人，硕士研究生，主要研究方向为深度强化学习（296232375@qq.com）；张席（19XX-），男/女（民族）（通信作者），籍贯，职称，职务，学位，主要研究方向为……（abcdefg@abcd.com〔通信作者一定要写邮箱地址，若没有通信作者，邮箱则标注第一作者的〕）；作者姓名 3（19XX-），男/女（民族），籍贯，职称，职务，学位，主要研究方向为…….

## 1 强化学习

### 1.1 强化学习算法

强化学习与传统的搜索策略不同，与环境进行交互的智能体能够通过学习环境中采集的样本，获得从状态到动作的映射，从而更快的得到最优的动作。强化学习的应用场景假设遵循马尔科夫性[10]，只需要当前步的状态信息，即可预测出一个合理的动作，执行动作到环境中得到反馈回报奖励，针对状态 $s_t$ 回报奖励为：

$$R_{t+1}=\sum_{s_{t+1},r_{t+1}} p(s_{t+1},r_{t+1}\mid s_t,a_t)r_{t+1} \tag{1}$$

其中，智能体在在状态 $s_t$ 下执行的动作 $a_t$ 达到下一个状态 $s_{t+1}$，并获得 $r_{t+1}$ 奖励的概率为 $p(s_{t+1},r_{t+1}\mid s_t,a_t)$。状态动作值函数 $Q(s_t,a_t)$ 表示状态 $s$ 在 $t$ 时间步下执行动作 $a$ 获得的期望回报，折扣因子 $\gamma$ 表示未来反馈回报对当前期望回报的影响程度，根据公式(1)递推，可得到递归公式[11]：

$$Q(s_t,a_t)=\sum_{s_{t+1},r_{t+1}} p(s_{t+1},r_{t+1}\mid s_t,a_t)\left[r_{t+1}+\gamma Q(s_{t+1},a_{t+1})\right] \tag{2}$$

给定一个确定性策略 $\pi$，为获得更优策略 $\pi'(\pi'>\pi)$，不断更新使其接近最优策略 $\pi_*$，记动作 $a=\pi(s)$，则：

$$Q_*(s,a)=Q_\pi(s,\pi_*(s))=\max_\pi Q_\pi(s,a) \tag{3}$$

结合公式(2)(3)推导，可得 $Q$-Learning 算法[12]公式如下：

$$Q(s_t,a_t)=Q(s_t,a_t)+\alpha\left[R_{t+1}+\gamma\max_a Q(S_{t+1},a_{t+1})-Q(s_t,a_t)\right] \tag{4}$$

状态动作函数值通过映射表的方式进行存储，但在高维度问题下，巨大的映射表占用大量的存储空间，并且有查表效率低下等问题。

### 1.2 深度 Q 网络

传统 $Q$-Learning 算法在大部分现实问题中，面临“维度灾难”问题。深度 Q 网络以神经网络为非线性函数，利用函数近似解决获取 Q 值的问题，即：

$$\hat{Q}(s,a,\boldsymbol{\theta})\approx Q_\pi(s,a) \tag{5}$$

定义损失函数来更新网络参数 $\boldsymbol{\theta}$，即：

$$L(\boldsymbol{\theta})=E_\pi\left[\left(Q_\pi(s,a)-\hat{Q}(s,a,\boldsymbol{\theta})\right)^2\right] \tag{6}$$

$Q$-Learning 算法属于无模型的 TD (Temporal Difference) 算法[1]，即不用等到训练回合结束，每步更新 Q 值，并通过自举来预测真实样本值 $Q_\pi(s,a)$。深度 Q 网络结合 $Q$-Learning 算法来近似 TD 目标。设当前步为 $t$ 步，TD 目标为 $Y_t$ ：

$$Y_t=R_{t+1}+\gamma\max\hat{Q}(s_{t+1},a_{t+1},\boldsymbol{\theta}) \tag{7}$$

由上可见，TD 目标与网络近似输出的 Q 值用的是同一参数 $\boldsymbol{\theta}$，在训练过程中一起发生改变，导致影响收敛的问题，使用目标网络[5]替换 TD 目标中当前网络的部分可解决该问题，记为 $\hat{Q}(s_{t+1},a_{t+1},\boldsymbol{\theta}^-)$，同时以一定的更新频率更新目标网络。

根据公式(6)(7)，深度 Q 网络以网络近似输出的 Q 值与 TD 目标之间的误差定义的损失函数为：

$$L(\boldsymbol{\theta})=E_\pi\left[\left(Y_t-\hat{Q}(s_t,a_t,\boldsymbol{\theta})\right)^2\right] \tag{8}$$

其中，$Y_t=R+\gamma\max\hat{Q}(s_{t+1},a_{t+1},\boldsymbol{\theta}^-)$，利用目标网络输出 TD 目标。

通过反向传播更新参数 $\boldsymbol{\theta}$，用于更新的梯度公式根据公式(8)对参数 $\boldsymbol{\theta}$ 求偏导可得，表示为：

$$\nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta})=E\left[\left(Y_t-Q(s_t,a_t,\boldsymbol{\theta})\right)\nabla_{\boldsymbol{\theta}}Q(s_t,a_t,\boldsymbol{\theta})\right] \tag{9}$$

为了打断状态之间的相关性，利用经验回放机制，智能体在每个 $t$ 步的信息 $e_t=(s_t,a_t,r_{t+1},s_{t+1},d)$ 存入存储容量为 N 的经验池 $B=\{e_1,e_2,\cdots,e_N\}$ 中，在开始更新网络模型后，每次从经验池中随机采样数量 $n$ 的样本，利用梯度下降更新参数。

### 1.3 改进的竞争网络

竞争结构将状态动作值函数 $Q^\pi(s,a)$ 解耦成状态值函数与动作优势函数，定义为：

$$Q^\pi(s,a)=A^\pi(s,a)+V^\pi(s) \tag{10}$$

其中，状态值函数 $V^\pi(s)$ 表示在状态 $s$ 下由特定策略 $\pi$ 产生的期望回报值，动作优势函数 $A^\pi(s,a)$ 表示在状态 $s$ 下由特定策略 $\pi$ 选择动作 $a$ 带来的优势值。

将竞争结构应用到神经网络中，即把深度 Q 网络的原始输出分为两个全连接层流，分别对应状态值函数、动作优势函数，记为：

$$Q(s,a;\boldsymbol{\theta},\alpha,\beta)=V(s;\boldsymbol{\theta},\beta)+A(s,a;\boldsymbol{\theta},\alpha) \tag{11}$$

其中，$\boldsymbol{\theta}$ 表示用于进行特征处理的卷积层的参数，$\alpha,\beta$ 分别为动作优势层、状态值层的参数，$V(s;\boldsymbol{\theta},\beta)$ 对应状态值函数，输出为标量，$A(s,a;\boldsymbol{\theta},\alpha)$ 对应动作优势函数，输出为矢量，矢量大小对应动作空间的大小。

由于网络最后的输出是状态动作值函数 Q 值，即在公式(10)中，状态值函数 $V$ 与动作优势函数 $A$ 无法被复原，表明 $V$ 不能反映出状态值，$A$ 不能反映出优势值。

因此将动作优势函数的输出矢量和设置为 0，即：

$$A(s,a;\boldsymbol{\theta},\alpha)=A(s,a;\boldsymbol{\theta},\alpha)-\max_{a'}\left(s,a';\boldsymbol{\theta},\alpha\right) \tag{12}$$

为了提高实际应用的稳定性，更好地反映优势变化以获得最优的行动，将公式(11)更改为：

$$A(s,a;\boldsymbol{\theta},\alpha)=A(s,a;\boldsymbol{\theta},\alpha)-\sum_{a'}A\left(s,a';\boldsymbol{\theta},\alpha\right)/|A| \tag{13}$$

因此，竞争网络结构能在无监督的情况下，自动地分别对状态价值函数 $V$ 与动作优势函数 $A$ 产生近似预估。更具体地说，竞争结构不需要了解每个状态对应每个动作的价值，也能决定哪些状态是有价值的。

## 2 动态噪声模型

### 2.1 $\varepsilon$-greedy 策略

在 $Q$-Learning 算法中，智能体获得最优动作的策略是选取最大 Q 值对应的动作，但在训练初期的阶段，智能体对环境了解的信息较少，获得的 Q 值并不能准确的反映最优的动作，影

响训练表现。

因此，需要在探索过程中加入一定随机性，即 $\varepsilon$-greedy 策略，在动作空间中引入参数 $\varepsilon(\varepsilon\in[0,1])$，在 $\varepsilon$ 概率下，智能体会随机选择动作，能在面对未知环境时，收集到更多信息。而随着训练时间进行，智能体获取到较多的环境信息，概率 $\varepsilon$ 单调衰减到一个固定值，从而减少智能体随机选取动作的可能性。

### 2.2 参数空间噪声

考虑在神经网络中加入高斯噪声作为参数，与 $\varepsilon$-greedy 策略不同的是，建立噪声分布，从分布中取样后注入到其他可训练的参数中，在神经网络中引起扰动，同样利用梯度下降进行参数学习。而 $\varepsilon$-greedy 策略在动作空间加入噪声，并不会影响网络输出，且参数 $\varepsilon$ 若下降过快，则未必能保证智能体学习到最优策略。

对有 $p$ 个输入和 $q$ 个输出的线性单元可表示为：

$$\boldsymbol{y}=\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b} \tag{14}$$

其中，$\boldsymbol{x}\in\boldsymbol{R}^{p\times1}$ 对应输入向量，$\boldsymbol{y}\in\boldsymbol{R}^{q\times1}$ 对应输出向量，$\boldsymbol{w}\in\boldsymbol{R}^{p\times q}$ 对应隐藏层的参数，$\boldsymbol{b}\in\boldsymbol{R}^{q\times1}$ 对应偏置值。

将 $\boldsymbol{w}$ 表示为一个分布，记为 $\boldsymbol{w}\sim N\left(\boldsymbol{\mu}^{w},\boldsymbol{\sigma}^{w}\right)$，建立噪声分布 $\boldsymbol{\epsilon}^{w}\sim N(0,1)$，从噪声分布中采样，则：

$$\boldsymbol{w}=\boldsymbol{\mu}^{w}+\boldsymbol{\sigma}^{w}\odot\boldsymbol{\epsilon}^{w} \tag{15}$$

其中，$\boldsymbol{\mu}^{w}\in\boldsymbol{R}^{p\times q}$ 对应均值，$\boldsymbol{\sigma}^{w}\in\boldsymbol{R}^{p\times q}$ 对应方差，同理对偏置值做同样改动，则线性单元可表示为：

$$\boldsymbol{y}=\left(\boldsymbol{\mu}^{w}+\boldsymbol{\sigma}^{w}\odot\boldsymbol{\epsilon}^{w}\right)\boldsymbol{x}+\boldsymbol{\mu}^{b}+\boldsymbol{\sigma}^{b}\odot\boldsymbol{\epsilon}^{b} \tag{16}$$

对于高斯噪声的采样，若对每个元素作为独立的高斯分布进行采样，那么需要从 $p\times q+q\times1$ 个分布中采样。另一种方法是将高斯噪声分解[14]，即采样 $p$ 个高斯分布，记为 $\boldsymbol{\epsilon}_i$；采样 $q$ 个高斯分布，记为 $\boldsymbol{\epsilon}_j$，则：

$$\begin{aligned}\boldsymbol{\epsilon}^{w}&=f\left(\boldsymbol{\epsilon}_i\right)\cdot f\left(\boldsymbol{\epsilon}_j^{T}\right)\\ \boldsymbol{\epsilon}^{b}&=f\left(\boldsymbol{\epsilon}_i\right)\end{aligned} \tag{17}$$

其中，$f(x)=\mathrm{sgn}(x)\sqrt{|x|}$ 作为噪声的缩放函数。

可分解的高斯噪声需采样 $p+q$ 个高斯分布，降低了采样所需的开销，减少了计算代价。

### 2.3 动态噪声

仅在动作空间加入噪声，虽能在短时间内寻找到一个较优策略，训练速度快，但在训练后期 $\varepsilon$ 下降到很小值，难以保证探索度，且 $\varepsilon$ 是单独线性衰减，并没有考虑到训练环境以及训练进度；仅在参数空间加入噪声，在训练前期智能体对环境了解较少，噪声方差参数难以进行学习，且由于需要学习的参数变多，训练速度慢，需要更多的训练回合才能达到良好效果，但在训练后期能保证足够的探索度，增加样本的多样性。

本文提出结合以上两种噪声类型的优点，提出同时在动作空间与参数空间中注入噪声的方法，该方法在训练前期，利用动作空间噪声，使智能体快速进行探索，学习到一个较优策略，随着训练的进行，逐渐增大参数空间噪声权重，以获得相对平滑的训练曲线，设权重系数为 $\beta(\beta\in[0,1])$，参数空间噪声可表示为 $\boldsymbol{\epsilon}^{w}=\boldsymbol{\epsilon}^{w}\cdot\beta$，$\boldsymbol{\epsilon}^{b}=\boldsymbol{\epsilon}^{b}\cdot\beta$，权重系数取值对比将在 **3.2 节**中阐述。

但在动作空间中，采用的是 $\varepsilon$-greedy 策略，$\varepsilon$ 的下降速度并没有考虑实际训练进度，而是单独地自行衰减。为了缓解这个问题，提出一种根据训练进度动态调整 $\varepsilon$ 下降速度的方法，判断训练进度基于当前步与上一步的TD目标差值，表示为 $Y_D$：

$$Y_D=Y_t-Y_{t-1} \tag{18}$$

其中，$Y_t=R_{t+1}+\gamma\max\hat{Q}(s_{t+1},a_{t+1},\boldsymbol{\theta})$。若 $Y_D>0$，则证明当前步比上一步更好；若 $Y_D<0$ 说明智能体采取了较差的行动；若 $Y_D=0$ 说明收敛或者没有进展。

根据 $Y_D$ 动态使用两种参数 $\varepsilon$ 衰减模型，由于在动作空间加入噪声是为了尽快达到较优策略，当 $Y_D>0$ 时，说明智能体训练有所进展，因此使用 $e$ 指数衰减模型 $\varepsilon_E$；当 $Y_D\le0$ 时，说明智能体还需要更多探索，因此使用线性衰减模型 $\varepsilon_L$，两种衰减模型对应公式为：

$$\begin{aligned}\varepsilon_E&=\varepsilon_{\min}+(\varepsilon_{\max}-\varepsilon_{\min})e^{\frac{-n}{N}}\\ \varepsilon_L&=\varepsilon_L-(\varepsilon_{\max}-\varepsilon_{\min})\cdot\frac{-n}{N}\end{aligned} \tag{19}$$

其中，$\varepsilon_{\min}$ 为 $\varepsilon$ 的最小值，$\varepsilon_{\max}$ 为 $\varepsilon$ 的最大值，$n$ 为当前训练次数，$N$ 为总的训练次数。

基于上述描述，伪代码算法流程描述如下：

算法 1　动态噪声的竞争深度 Q 网络算法

输入：环境参数 env，经验回放缓冲池 B，缓冲池大小 $N_B$，训练批量大小 $N_T$，更新目标网络频率 $F_T$，初始网络参数 θ，初始目标网络参数 $\theta^-$，准备更新步数 T，初始 TD 目标差值 $Y_D$，参数噪声布尔值 NOISY。

输出：状态动作值 Q(s,a,θ)。

a) 循环训练回合数 e∈{1,…,M} 有：

b) 初始化环境，得到状态 $S_0\sim$ env，$s=S_0$。

c) 循环每回合步数 t∈{1,…} 有：

d) 采样一个参数噪声 ω 并初始化。

e) 如果不 NOISY，那么：//使用动作空间噪声

如果 $Y_D>0$，那么使用 $\varepsilon_E$ 衰减；

否则使用 $\varepsilon_L$ 衰减；

否则 ε=0。　　//使用参数空间噪声

f) 以 ε 概率选择随机动作 $a_t$；

否则 $a_t=\mathrm{argmax}Q(s,a,\theta,\omega)$

g) 执行动作得到环境反馈，并把信息$(s_t,a_t,r_{t+1},s_{t+1},d)$存入经验池 B 中。

h) 如果 $N_B>N_T$ 且 $N_B>T$：　//准备计算损失函数、更新模型

从经验池 B 中采样 $N_T$ 个样本；

根据梯度下降，利用损失$(Y_t-Q(s,a,\theta,\omega))^2$反向传播，更新网络。

i) 如果 1-ε>λ：NOISY 为真；//作为噪声参数开始学习的阈值

否则 NOISY 为假。

j) 每 $F_T$ 步，更新目标网络。

结束循环。

结束循环。

## 3 实验结果与分析

### 3.1 实验设计

为验证所提方法的效果，本文选取 OpenAI Gym[15]测试平台上选取 CartPole-v0, LunarLander-v2, Acrobot-v1 三个经典控制测试环境进行测试对比。如图 1 所示，图中左上角的为 Cart Pole 控制环境，一根杆子连接到一辆小车，小车沿着无摩擦的轨道移动，杆子初始是直立状态，但一旦小车移动，杆子会有倒下来的可能，控制目标是不让杆子倒下来，当杆子离开垂线 15 度或小车离画面中心超过 2.4 个单位，环境自动结束；图 1 中右上角的为 Lunar Lander 控制环境，控制目标是飞行器底部正确着落在两支旗子之间的着陆台内，飞行器有四个离散的动作，向左、右、下喷射，以及无动作，如果飞行器着陆在着陆台以外的地方，或者顶部触碰地面，环境自动结束；图 1 下方的是 Acrobot 控制环境，模型包括两个关节、两个连杆，其中两个连杆之间的关节是可以被驱动的，初始时，杆子向下悬挂静止，控制目标是将较低连杆的末端摆动到给定高度，超过一定控制步数，环境自动结束。

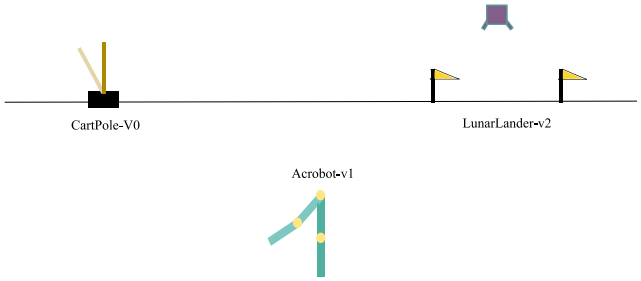



图 1 经典控制环境

Fig.1 Classic control enviorments

深度神经网络的结构分为卷积层与全连接线性层，其中卷积层的具体细节如表 1 所示，线性层采用竞争网络结构，分为动作优势线性层与状态值层，神经元数量为 512，动作优势层的输出为动作空间大小的矢量矩阵，状态值层输出为单个值。

表 1 卷积网络前向传播

Table 1 Convolutional network forward propagation

| 网络层 | 卷积核 | 步幅 | 卷积核数 | 激活函数 |
|---|---|---|---|---|
| Conv1 | 5×5 | 2×2 | 16 | ReLU |
| Conv2 | 5×5 | 2×2 | 32 | ReLU |
| Conv3 | 5×5 | 2×2 | 32 | ReLU |

OpenAI Gym 中环境的图像参数为 400×600×3，为了减少训练的计算量以及便于提取特征，对其进行预处理。先将原始的 RGB 图像转换为 400×600×1 的灰度图像，图像的通道由 3 减少为 1，由于控制环境不需要用到颜色参数，所以转换为灰度图像不会影响训练。环境图像的边角区域也属于无用信息，对其进行裁剪，使图像高度变为原来的 0.4 倍，宽度不进行裁剪，但取图像宽度的 0.6 倍作为可视宽度传入神经网络。此时，图像参数为 160×360×1，对其进行缩放为原来的 0.25 倍，最终，图像参数变为 40×90×1，进行归一化处理后作为卷积网络层的输入。梯度下降算法使用 Adam 算法[16]，训练超参数设置，如表 2 所示。

表 2 超参数设置

Table 2 Hyper-parameters setting

| 超参数 | 大小 | 说明 |
|---|---|---|
| Replay size | 10000 | 经验池存储容量 |
| Replay warmup | 128 | 开始训练之前的步数 |
| Target update freq | 100 | 更新目标网络频率 |
| Epsilon max | 1.0 | 参数 ε 最大值 |
| Epsilon min | 0.1 | 参数 ε 最小值 |
| Learning rate | 0.001 | Adam 算法学习率 |
| Gamma | 0.9 | 折扣因子系数 |
| Batch size | 32 | 批量采样数 |

对于参数噪声权重 $\beta$ 的设置，如下：

(1) 使动作空间与参数空间的噪声同时作用，随着训练进行，慢慢增大参数空间的权重 $\beta$，而动作空间参数 ε 逐渐衰减。

(2) 在训练前期，初始化参数空间噪声后，设置权重 $\beta$ 为 0，只使用动作空间的噪声作为训练前期的策略，在参数 ε 下降到一定值 λ 后，将参数 ε 设置为 0，权重 $\beta$ 设置为 1，开始使用参数空间噪声作为策略。

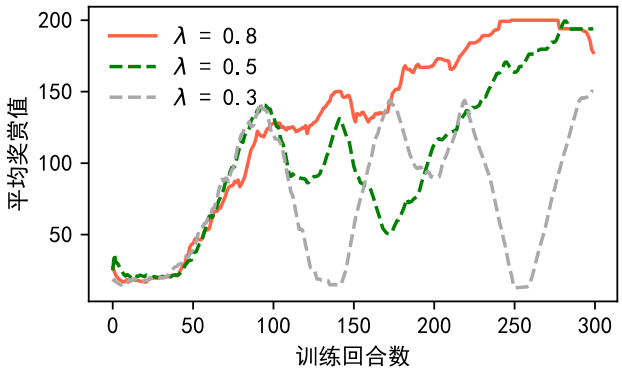
对方法(1)进行测试后，发现同时在动作空间与参数空间产生噪声，导致预测 Q 值偏差过大，难以训练到良好效果，因此，本文采用方法(2)进行实验。

### 3.2 实验结果分析

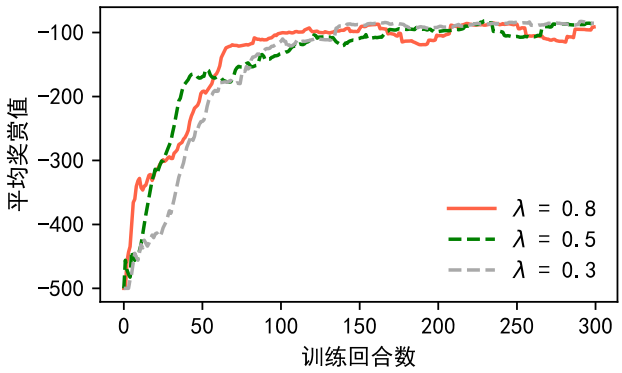
首先对上节提到的方法(2)进行实验，确定 λ 参数的值，以确定什么时候参数空间开始产生噪声。从表 2 可以看到，参数 ε 的取值范围为[0.1, 1]，则均分区间，对 λ 参数取值为 0.3、0.5、0.8 进行对比实验，测试算法为在 Dueling DQN 中加入动态噪声模型。

图 2 给出 λ 参数在三种取值下对训练的影响。λ 的值越大说明使用动作空间产生噪声时间越长。其中，图 2(a)反映了在当 λ=0.3，说明在训练阶段较前期就开始使用参数空间来产生噪声，由于 Cart Pole 环境中，小车一旦移动，杆子很容易偏离超过 15 度，导致回合结束，意味着基本没有受益于 $\varepsilon$-greedy 策略在训练前期的优势，在回合数 100 后才开始有训练效果；当 λ=0.5 与 λ=0.8，在回合数 100 时虽有一定训练效果，但经验池中大部分还是训练初期时获得的样本，此时取样进行参数空间噪声的训练，导致训练表现骤降，并且训练过程不稳定，而在 λ=0.8 的设置中，训练表现总体趋于收敛。在图 2(b)(c)中三种参

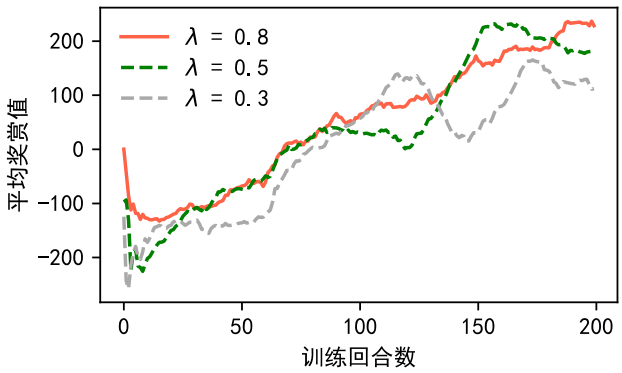
数设置中，各自的训练表现差异不大，但还是能看出 λ=0.8 的设置效果稍优于其他两种，训练曲线更加平稳，且更早能获得较优的策略。故 λ 参数值取值 0.8。




(a) CartPole-v0 场景




(b) Acrobot-v1 场景



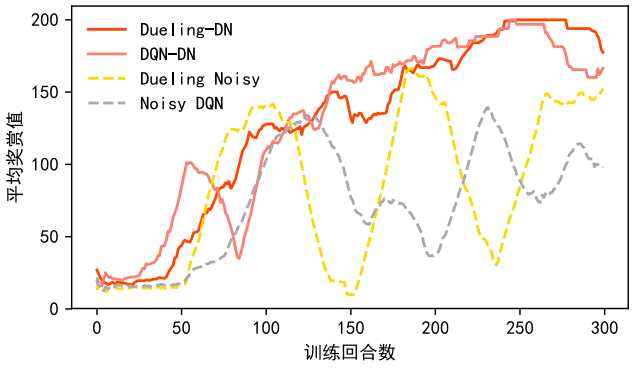

(c) LunarLander-v2 场景

图 2 三种 λ 参数的比较

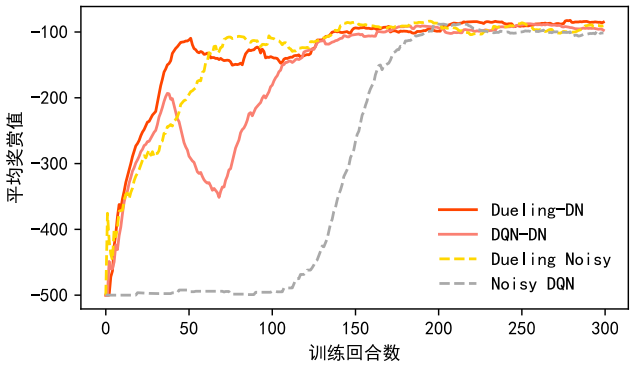
Fig.2 A comparison of the λ parameters

以 Dueling Noisy DQN 为基准算法，即仅在参数空间产生噪声，与本文所提方法 Dueling DQN-DN(Dynamic Noise)在上述三种控制问题环境下进行实验对比，为更好测试模型效果，进行模块测试，即同时以 Noisy DQN 为基准算法，与 DQN-DN(Dynamic Noise)进行实验对比。

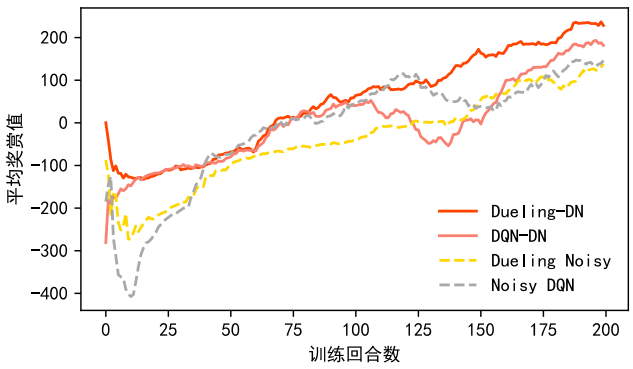
在图 3(a)中，首先对比 Dueling DQN-DN 与基准算法 Dueling Noisy DQN，可以发现，基准算法在 Cart Pole 环境下的训练过程很不稳定，首先 Cart Pole 环境容错率低，其次是由于噪声参数与其他可训练参数，一并在学习，而此时经验池中有价值的样本不多，导致了训练曲线波动，最终在回合数 250 之后才趋于稳定，对比 Dueling DQN-DN，训练过程稳定上升。




(a) CartPole-v0 场景




(b) Acrobot-v1 场景




(c) LunarLander-v2 场景

图 3 在三种环境的算法比较

Fig.3 Algorithms comparison in three environments

对比 DQN-DN 与其对应的基准算法，可以发现类似的结果，且可以看出，基准算法明显需要更多的训练回合数，才能收敛。图 3(b)中得益于竞争网络结构的优势，基准算法 Dueling Noisy DQN 在 Acrobot 环境中选取了有价值的动作，取得了一定训练结果，但对比 Dueling DQN-DN 的训练曲线，在训练前期获取信息还是稍慢。在动态噪声模型的作用对比方面，可以从图 3(b)(c)中看到，在训练前期，基准算法难以取得一个较优的训练表现，训练速度慢，往往需要更多的回合数，才能取得收敛，

而动态噪声模型在训练前期很快取得较优的训练结果，经验池中有价值的样本数量占优，随后利用参数空间可学习的噪声得到进一步的样本补充与泛化能力。

表 3 记录了在三种控制环境中，四种算法在一定训练回合后的平均奖赏值，表明了动态噪声模型在一定训练回合数，能取得比基准算法更高的分数，加快训练进程。Dueling DQN-DN 相比基准算法在 Cart Pole、Acrobot、Lunar Lander 环境中，平均奖赏值分别提高了 77.0%、7.6%、73.1%；DQN-DN 相比其对应的基准算法在三个环境中的平均奖赏值分别提高了 121.4%、11.2%、32.6%，检验结果进一步表明动态噪声模型取得更优的训练表现。

表 3 平均奖赏

Table 3 Mean rewards

| 环境 | Dueling-DN | Dueling Noisy | DQN-DN | Noisy DQN |
|---|---|---|---|---|
| CartPole | **199.5** | 112.7 | **195.5** | 88.3 |
| Acrobot | **-88.0** | -94.7 | **-89.9** | -100.0 |
| LunarLander | **232.9** | 134.5 | **185.8** | 140.1 |

## 4 结束语

本文提出了一种结合动作空间噪声与参数空间噪声的动态模型，能够同时利用两种噪声类型的优势，既能加快训练速度，也能在训练后期为样本提供丰富度，减少过拟合，在测试平台 OpenAI Gym 上的三个经典控制问题环境中进行了实验测试，相比于仅在参数空间添加噪声的算法，训练速度、稳定性和训练效果均有显著提升。

## 参考文献


[1] Sutton R.S, Barto A.G. Reinforcement learning: An introduction [J]. IEEE Transactions on Neural Networks, 1998, 9(5):1054-1054.

[2] 刘全, 翟建伟, 章宗长, 等. 深度强化学习综述 [J].计算机学报,2018, 41(1): 1-27. (Liu Quan,Zhai Jianwei,Zhang Zongchang, *et al*. A survey on deep reinforcement learning [J]. Chinese Journal of Computers, 2018, 41 (1): 1-27. )

[3] Mnih V, Kavukcuoglu K, Silver D, *et al*. Playing atari with deep reinforcement learning. arXiv preprint arXiv: 1312. 5602, 2013.

[4] Bellemare M G, Naddaf Y, Veness J, *et al*. The arcade learning environment：An evaluation platform for general agents[J]. Journal of Artificial Intelligence Research, 2012, 47(1): 253-279.

[5] Mnih V, Kavukcuoglu K, Silver D, *et al*. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518：529-533.

[6] Van H, Guez A, Silver D. Deep reinforcement learning with double q-learning [C]// Thirtieth AAAI conference on artificial intelligence, 2016

[7] Wang Z , Schaul T , Hessel M , *et al*. Dueling Network Architectures for Deep Reinforcement Learning [C]// Proc of the 33rd International Conference on Machine Learning, 2016.

[8] Plappert M, Houthooft R, Dhariwal P, *et al*. Parameter space noise for exploration. arXiv preprint arXiv:1706.01905, 2017.

[9] Fortunato M, Azar MG, Piot B, *et al.* Noisy networks for exploration. arXiv preprint arXiv:1706.10295, 2017.

[10] 邱祎, 董彦彦. 基于马尔可夫过程的线性规划方法探讨 [J]. 统计与决 策, 2017, 10: 88-90. (Qiu Yi, Dong Yanyan. Linear programming method based on Markov process [J]. Statistics & Decision, 2017, 10: 88-90. )

[11] Dolcetta IC, Ishii H. Approximate solutions of the Bellman equation of deterministic control theory[J]. Applied Mathematics and Optimization. 1984, 11(1):161-81.

[12] Watkins C J C H，Dayan P. Q – learning [J]. Machine Learning，1992，8：279-292.

[13] Osband I, Russo D, Wen Z, et al. Deep exploration via randomized value functions [J], Journal of Machine Learning Research, 2017.

[14] 刘全, 闫岩, 朱斐, 等. 一种带探索噪音的深度循环 Q 网络 [J]. 计算机学报, 2019, 42(7):1588-604. (Liu Quan, Yan Yan, Zhu Fei, *et al*. A deep recurrent Q network with exploratory noise[J]. Chinese Journal of Computers, 2019, 42(7):1588-604.)

[15] Zamora I, Lopez N G, Vilches V M, *et al*. Extending the OpenAI Gym for robotics:A toolkit for reinforcement learning using ROS and Gazebo[J]. Robotics, 2016, 211：645-789.

[16] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.