arXiv:2301.01348v1 [cs.LG] 3 Jan 2023

DADAgger：分歧增强数据集聚合

Akash Haridas akashh@cs.toronto.edu
Karim Hamadeh kar@cs.toronto.edu
Samarendra Chandan Bindu Dash dashsam1@cs.toronto.edu

摘要： DAGGER 是一种模仿算法，它通过在训练过程中对遇到的所有样本查询专家，来聚合其原始数据集。为了减少查询的样本数量，我们提出了对 DAGGER 的一种改进，称为 DADAGGER，它只针对分布外（OOD）的状态-动作对查询专家。通过测量模型集成在每个状态上的动作预测方差来识别 OOD 状态，我们使用 Dropout 来模拟这种方差。在 Car Racing 和 Half Cheetah 环境中进行测试，结果显示其性能与 DAGGER 相当，但专家查询次数更少，且优于随机采样基线。我们还展示了，我们的算法可以用于构建高效、均衡的训练数据集：从没有初始数据的状态开始运行，仅在需要解决不确定性时查询专家。

关键词： 模仿学习、集成

1 问题描述

DAGGER 算法 [1]通过在测试运行期间，对智能体遇到的所有状态再次查询专家，并将这些新样本与现有数据集聚合，从而解决行为克隆中遇到的协变量偏移。

然而，在某些情况下，查询专家的成本可能很高。DAGGER 的一个缺点是，它会对所有已收集的观测查询专家，而不考虑哪些动作对训练出良好的策略最有价值。DRIL 算法 [2] 通过在策略优化中添加正则化项来处理协方差偏移问题，隐式地倾向于选择能够进入某些状态的策略，使这些状态下策略集成的方差最小。其依据是，分布内状态的方差应该较低，因此这应促使学习者策略停留在该分布范围内，前提是这样足以实现良好性能。这种方法的一个缺点是，它可能无法从不完整数据集中学习，也无法进一步探索。

因此，我们设计了一种不会不必要地查询专家、同时又能有选择地查询专家以了解确实尚未遇到的状态的算法。该算法结合了 DAGGER 的灵活性与探索能力，同时利用集成来识别分布外状态的能力，这与 Brantley 等人 [2] 所采用的方法相同。

2 相关工作

该领域以往的工作主要包括估计深度神经网络预测的不确定性，以及提高 DAGGER 算法的样本效率。

Blundell 等人 [3] 学习神经网络权重的概率分布，使网络能够对未见数据做出更合理的预测。Gal 和 Ghahramani [4]从深度神经网络的随机失活层估计模型不确定性。估计模型不确定性的一种常见方法是使用学习器集成：当集成的预测方差较高时，就认为不确定性较高。Wen 等人 [5] 通过在各个网络间共享部分权重，降低集成的计算成本。Brantley 等人 [2] 将集成方差纳入一个额外的损失项，从而隐式地训练模仿策略，使其避开未见过示范的状态。

多项研究尝试提高 DAGGER 模仿学习算法的采样效率。Kim 和 Pineau [6] 提出一种查询度量，用于判断遇到的状态与迄今收集的数据分布有多接近；只有当该度量高于某个阈值时，才查询专家。他们使用最大均值差异作为度量，该方法最初用于判断两组数据是否来自不同的概率分布。Zhang 和 Cho [7]通过采用一个名为安全策略的独立网络，实现类似目标：该网络预测当前正在训练的策略是否可能出错，随后利用此信息确定哪些已访问状态需要纳入下一次 DAGGER迭代。同样，Laskey 等人 [8] 使用支持向量机分类器确定特定状态的风险等级。Menda 等人 [9] 使用集成来估计模型不确定性，从而构建 DAGGER 的概率变体；他们利用模型对其预测动作的不确定性来确定何时查询专家。他们这样做是为了确保模型安全，而非像我们的工作那样，在训练时选择何时扩充数据集。

3 方法

我们提出了一种名为 DADAGGER（分歧增强数据集聚合算法）的新算法。它基于 DAGGER，并借鉴了 [2] 这一原理：分布外状态会导致策略集成中的分歧更大。我们利用这一点对 DAGGER 进行改进，仅在分歧较大的状态（已访问状态中的特定比例）上查询专家，力求通过更少的查询获取最多的信息并消除最多的不确定性。与 DAGGER 相比，该方法减少了专家调用次数，同时又能处理不完整数据集，不像完全离线的 DRIL。重要的是，该方法并不通过查询专家来判断学习器是否遇到了无法处理的状态，而是完全依赖学习器集成的分歧。

在此，若输出之间可以定义距离（与 [2] 中的实现一致），分歧可定义为输出的方差；若输出是离散且非传递的，则可定义为熵。由于训练神经网络集成的成本高得难以承受，为了高效地获得方差，我们改用随机失活层来近似估计器的不确定性。将同一输入 M 次传入一个每层之后都启用了随机失活的网络，理论上即可通过高斯过程采样得到不同的网络 [10]。DADAGGER 的正式定义如下。可调超参数包括根据方差要保存的状态比例 α，以及 M（集成中策略的数量）。由于我们使用随机失活来模拟集成，因此选择 M 不会显著增加训练时间，因为只有一个网络的权重会得到更新。为确定特定动作对应的不确定性，需要调用网络 M 次。较小的 α 会减少查询的样本数量，但由于数据集增广程度降低，收敛也会更困难。请注意，在评估策略时（即测试时），我们只对网络采样一次来预测动作，而不获取集成均值，以确保与 DAGGER 进行公平比较。

图 1 展示了一个简化示意图：不同模型在数轴$\mathcal{R}$ 原点周围的区域（代表分布内状态）达成一致，但在原点之外的区域（即未被显式学习的分布外区域）则意见不一。

算法1DADAgger 集成算法

初始化 $D \leftarrow \emptyset$。
初始化 $\hat{\pi}_{1,1}, \hat{\pi}_{1,2}..\hat{\pi}_{1,M}$
对于 $i = 1$到 $i = n$ **执行**
  使用 $\hat{\pi}_{i,1}$采样 T 步轨迹
  计算所有已访问状态 s 上 $\hat{\pi}_{i,1}(s), \hat{\pi}_{i,2}(s)..\hat{\pi}_{i,M}(s)$ 的方差。
  保留方差最高的 α 百分比。
  获取所有筛选后的状态 s 对应的数据集 $D_i = \{s, \pi^*(s)\}$。
  聚合数据集 $D \leftarrow D_i \cup D$
  在 D 上训练分类器 $\hat{\pi}_{i+1,1}, \hat{\pi}_{i+1,2}..\hat{\pi}_{i+1,M}$
结束循环
返回最佳$\hat{\pi}_{i,1}$ 在验证集上

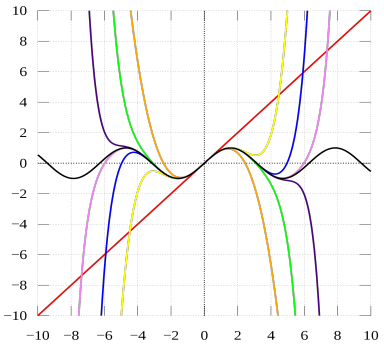


图1：函数在 $\mathcal{R}$ 区域内一致、在区域之外不一致的简化示意图。

算法 2带 Dropout 的 DADAgger 算法

初始化 $D \leftarrow \emptyset$使用随机失活层初始化 $\hat{}\pi_1$ **，** $i = 1$至 $i = n$ **执行**使用 $\hat{}\pi_i$采样 T 步轨迹使用 $\hat{}\pi_i$ 生成 $\hat{}y_{i,\ j} = \hat{}\pi\text{‘}_{i,\ j}(s)$，其中 $\hat{}\pi\text{‘}_{i,\ j}$ 是 $\hat{}\pi_i$ 的一种变体，它通过随机失活层随机关闭部分神经元。计算 $\hat{}y_{i,\ 1}$、$\hat{}y_{i,\ 2}$.....、$\hat{}y_{i,\ M}$ 所有已访问 s 的方差。保留方差最高的 α %。对所有筛选后的 s 获取数据集 $D_i = \{s, \pi^*(s)\}$ 。汇总数据集 $D \leftarrow D_i \cup D$在 D上训练分类器 $\hat{}\pi_{i+1}$ **结束循环，返回**验证集上表现最佳的 $\hat{}\pi_i$

4 实验

4.1 基线与参数调优

我们在 OpenAI Gym 提供的 Car Racing 环境上实现了我们的方法 [11]。对于策略网络，我们使用一个卷积神经网络，并在全连接部分加入随机失活层。该网络接收当前状态的图像，并预测转向动作。

我们的第一个实验通过改变超参数 M 和 α，评估它们对 DADAGGER 收敛的影响。请注意，并不存在“最优参数”。增加 M和 α 理论上应能减少达到收敛所需的迭代次数（有待测试验证），但代价是集成中更多模型的训练时间更长，且需要更多专家查询。我们将算法的性能和收敛特性与 DAGGER（通过将 α 和 M 均设为 1 得到）进行比较；同时也与一种智能体进行比较，该智能体会在观测到的状态中随机选取状态并查询专家（通过在不同 α下将 M 设为 1 得到），以检验采样质量是否会影响收敛。我们使用 5 个不同的随机种子运行算法 10 次迭代，并测量收敛的运行所占比例。我们还测量数据集中累积的动作分布，以定量评估算法探索动作空间的效率。所报告的标准差来自 $p = 0.5$ 二项估计量，该估计量保证给出误差的上界。

4.2 Half Cheetah

在简单环境中实现算法后，我们在 Half Cheetah MuJoCo[12] 环境中对其进行测试。与 Racer 相比，HalfCheetah 的动力学更复杂，动作空间维度更高且取值连续。观测是状态向量，而非图像，因此我们用多层感知机（带有随机失活层）替换卷积神经网络，其余部分保持

算法不变。我们重复前一项实验中使用的 M 和 α 的每个取值，并测量每个策略在每次 DADAGGER 迭代结束时获得的奖励。我们将其性能和收敛特性与基础 DAGGER 进行比较，同时也与一种在观测到的状态中随机选取状态并向专家查询的智能体进行比较。

4.3 数据集构建

最后一项实验利用我们选择专家查询方式的高效性，构建一个精简且具有代表性的数据集，以支持快速的单次训练。与之前的实验使用初始训练数据集不同，我们使用空的初始数据集运行 DADAGGER，并在多个迭代中逐步构建数据集，仅在需要消除不确定性时查询专家。我们的假设是，收敛后的最终数据集规模会相当小，因为其中只包含模型在训练不同阶段意见不一致的样本点。我们使用 $\alpha = 0.1$，并进行 50 次迭代；根据我们的发现，采用此技术需要这么多次迭代才能实现良好收敛。随后，我们测量最终数据集中各类动作的分布，并将其与使用初始数据集时 DAGGER 和 DADAGGER 生成的数据集进行比较。

5 结果与讨论

5.1 基线与参数调优

结果如表1所示。对于 $\alpha = 0.4$，所有训练模型在所有运行中均收敛，包括随机查询。我们推测，这是由于相邻帧之间的差异相对较小，因此平均而言，跳过一帧不会对训练造成太大影响。当 α 为 0.2时，无论选择何种 M，我们的算法都能保持 100% 的收敛率，而随机查询仅有 40% 的成功率。因此，在这一指标上，我们的算法优于随机基线（$p < 0.003$）。最后，在 α的最低设置下，随机采样完全无法收敛，而我们的算法有时能够收敛。收敛通常发生在第 9 或第 10 次迭代，因此，即使在未终止的情况下，也完全有可能随后收敛；不过，为了使所有方法的测试条件一致，我们将测试限制在 10 次迭代。请注意， $M = 25$ 与 $M = 10$之间的差异在统计上并不显著。

为验证我们所获改进背后的机制，我们在图 3 中绘制了不同设置下已保存动作的直方图。可以看到，DAGGER 大量采样直线动作（靠近中间位置）。相较于数据集的大小，我们的算法显著提高了所采样转向动作的比例，同时总体采样量也更少。有趣的是，初始数据集比 DAGGER / DADAGGER 迭代生成的任何数据集都更加均衡，却仍未能收敛，这意味着数据集的质量并非完全取决于其均衡程度。

5.2 Half Cheetah

图 2 显示，DADAGGER 的所有变体在 Half Cheetah 上都收敛到与基础 DAGGER 相近的性能，这表明我们的不确定性度量和采样技术也能处理多维连续动作空间。仅需向专家查询基础 DAGGER 所需次数的 10%（$\alpha = 0.1$），就能达到这一效果；这再次表明，该方法只采样对于学习策略和消除不确定性最重要的数据点。有趣的是，这两种方法（包括基础 DAGGER）都未能达到专家的性能，这可能是由于环境的复杂性增加所致。

5.3 数据集构建

大约在第 50 次迭代时达到收敛（因此我们决定停止）。相比策略的性能，我们更感兴趣的是考察所构建数据集的特征。最终数据集由 746 个样本组成，明显少于我们用于 DAGGER 和 DADAGGER的初始数据集，该数据集包含 1139 个样本。因此，无论选择 α如何，该数据集都远小于此前实验生成的数据集。值得注意的是，使用该数据集训练的策略无需任何增广即可收敛（即仅进行 1 次 DAGGER 迭代，或进行单次训练），这表明样本质量很高。我们再次查看直方图（图 3e），

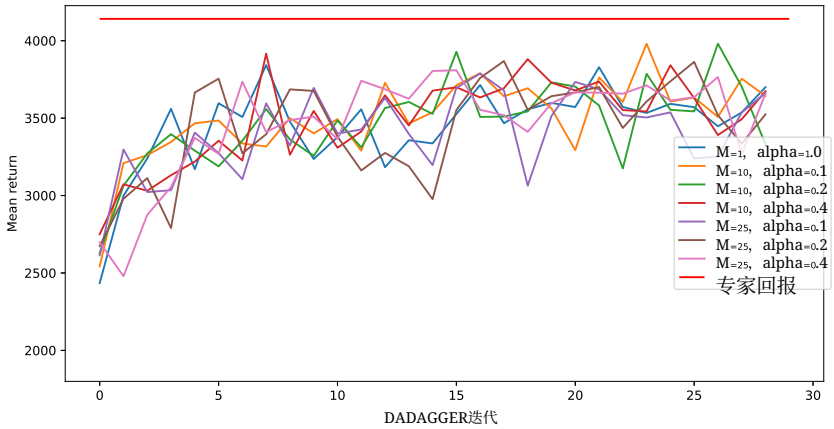


图 2：DADAGGER 的所有变体都与基础 DAGGER（$M = 1$和 $\alpha = 1$）一样，收敛到相近的性能，在 Half Cheetah 环境中向专家查询的次数则显著减少。然而，这些方法的性能都未能达到与专家相当的水平

M / α	0.1	0.2	0.4
10	0 ± 20	100 ± 20	100 ± 20
25	60 ± 20	100 ± 20	100 ± 20
随机	40 ± 20	40 ± 20	100 ± 20

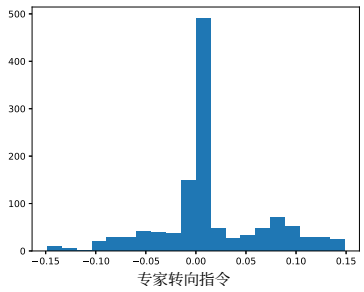
表 1：在 Car Racing环境中成功完成一圈的试验百分比。

并发现它是所有数据集中分布最均衡的，尤其是在急剧转弯方面；即便在 DADAGGER 实验中，这类样本也相对不足。因此，DADAGGER 可用于构建小型且高效的数据集，以进行单次训练。

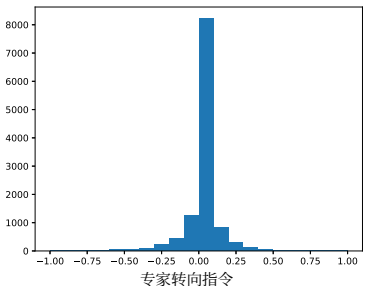
6 局限性与结论

尽管 DADAGGER 能够提高 DAGGER 的采样效率，但仍存在一些不足。首先，智能体仍需像使用 DAGGER 时一样多次与环境交互，这意味着若交互成本较高，所需回合数在收敛前并没有减少。数据集初始化为空时尤其如此，因为需要更多回合才能实现收敛。我们在测试中偶尔观察到的另一个问题是：对于错误结果给出高置信度预测，这可能导致跳过必要的采样并重复同一错误。因此，要使我们的算法正常运行，预测器之间必须具备一定程度的独立性。任何偏差都会在分布外状态中产生一致意见，进而可能导致算法陷入如下循环：不进行采样，因该错误而失败，并且再次不采样所需状态。事实上，在数据集构建实验的 50 次迭代中，有 30 次以相对相似的形式失败，这可能暗示稀疏数据集中存在这一问题。

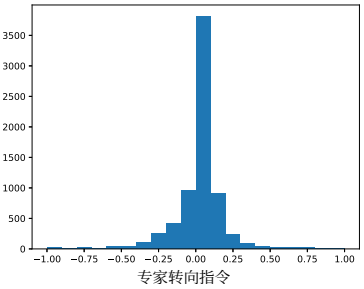
总之，本文介绍了 DADAGGER，这是一种新方法，可在不影响收敛的情况下提高 DAGGER 算法的查询效率。我们的方法在 Car Racing 和 Half Cheetah 环境中的表现均与 DAGGER 相当。此外，由于采用了 Dropout，我们的方法执行效率很高；同时，凭借其选择性，也可用作数据集生成器。未来的探索方向包括研究评估不确定性的其他方法，以便使用绝对指标而非比例 α，并避免针对每个动作进行多次网络评估。另一个尚未解决的问题是，在不考虑动作的情况下处理样本空间，并尝试设计一种度量，衡量特定样本在多大程度上属于分布外，且该度量完全独立于生成动作的预测器。



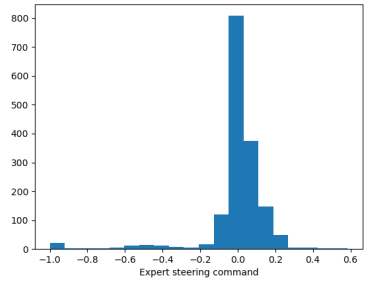
(a) 初始数据集



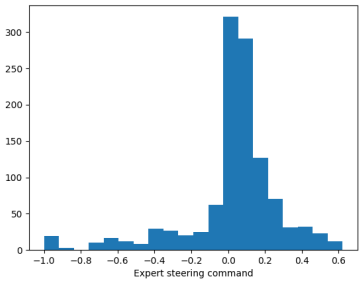
(b) 经过一次 DAGGER 运行后的数据集



(c) **我们的方法**：经过一次 DADAGGER 运行后的数据集



(d) **我们的方法**：使用 $M = 10$, $\alpha = 0.4$ with $M = 10, \alpha = 0.1$ 运行一次 DADAGGER 后的数据集



(e)**我们的方法**：DADAG-GER 从空的初始 数据集开始

。

图3：Car Racing环境中的采样分布比较。初始数据集主要包含直线示例，而 DAGGER 在转弯处至少会采样一些点。我们的算法 DADagger 通过减少在直线路段的冗余采样，总体上采集的样本少得多（请注意 y 轴刻度的差异）。以空初始数据集启动 DADAGGER，得到的数据集最为均衡，尤其是在急剧转弯方面；与其他实验相比，这些位置的采样相对不足。

贡献

KH 负责算法设计和实验3。实验1由所有成员共同完成。AH 和 SCBD 进行了实验2。SCBD 和 AH 共同编写、改编并优化了用于开展所有实验的大部分代码。

参考文献

[1] S. Ross、G. Gordon 和 D. Bagnell。将模仿学习和结构化预测归约为无悔在线学习。载于第十四届国际会议

人工智能与统计学，第 627–635 页。JMLR Workshop and Conference Proceedings，2011 年。[2] K. Brantley、W. Sun 和 M. Henaff。分歧正则化模仿学习。载于国际表征学习会议，2019 年。[3] C. Blundell、J. Cornebise、K. Kavukcuoglu 和 D. Wierstra。神经网络中的权重不确定性。载于国际机器学习会议，第 1613–1622 页。PMLR，2015 年。[4] Y. Gal 和 Z. Ghahramani。Dropout 作为贝叶斯近似：表示深度学习中的模型不确定性。载于国际机器学习会议，第 1050–1059 页。PMLR，2016 年。[5] Y. Wen、D. Tran 和 J. Ba。Batchensemble：一种高效集成与终身学习的替代方法。*arXiv* 预印本 *arXiv:2002.06715*，2020 年。[6] B. Kim 和 J. Pineau。最大均值差异模仿学习。载于机器人：科学与系统，2013 年。[7] J. Zhang 和 K. Cho。面向端到端自动驾驶的查询高效模仿学习。*arXiv* 预印本 *arXiv:1605.06450*，2016 年。[8] M. Laskey、S. Staszak、W. Y.-S. Hsieh、J. Mahler、F. T. Pokorny、A. D. Dragan 和 K. Goldberg。SHIV：在 DAGGER 中使用支持向量降低监督者负担，从高维状态空间中的示范高效学习。载于*2016 IEEE* 国际机器人与自动化会议（*ICRA*），第 462–469 页。IEEE，2016 年。[9] K. Menda、K. Driggs-Campbell 和 M. J. Kochenderfer。DropoutDAgger：一种安全模仿学习的贝叶斯方法。*arXiv* 预印本 *arXiv:1709.06166*，2017 年。[10] N. Srivastava、G. Hinton、A. Krizhevsky、I. Sutskever 和 R. Salakhutdinov。Dropout：一种防止神经网络过拟合的简单方法。机器学习研究杂志，15 (1):1929–1958，2014 年。[11] G. Brockman、V. Cheung、L. Pettersson、J. Schneider、J. Schulman、J. Tang 和 W. Zaremba。OpenAI Gym，2016 年。[12] E. Todorov、T. Erez 和 Y. Tassa。MuJoCo：一种用于基于模型的控制的物理引擎。载于*2012 IEEE/RSJ* 智能机器人与系统国际会议，第 5026–5033 页。IEEE，2012 年。