 DOI: https://doi.org/10.26549/bdai.v4i6.15248

Applied Research on Developing Machine Learning Based on Unity Games

Yongxiang Hu

Jiangsu Vocational and Technical College of Finance and Economics, Huai'an, Jiangsu, 223001, China

Abstract

This paper studies the implementation of machine learning technology in Unity environment development games, uses the open source machine learning plug-in ML-Agents, builds the Pytorch platform, and studies the various problems of developing games in machine learning in Unity. Research has shown that the Unity machine learning plug-in ML-Agent can help solve problems related to AI development, multi-agent collaboration, reinforcement learning, visualization tools, and custom training environments, allowing developers to build and train intelligent agents more efficiently to get unexpected results. This paper studies the implementation of machine learning technology in the Unity environment development game, combined with the actual needs at the present stage, to make it more intelligent, rich and perfect and better game experience.

Keywords

Unity; ML-Agents; Pytorch; machine learning

基于 Unity 游戏开发机器学习的应用研究

胡永祥

江苏财经职业技术学院，中国·江苏 淮安 223001

摘　要

论文研究了在Unity环境开发游戏中实施机器学习技术，运用开源的机器学习插件ML-Agents，搭建使用Pytorch平台，研究在Unity中开发游戏实现机器学习的各种问题。研究表明，Unity机器学习插件ML-Agent可以帮助解决与人工智能开发、多智能体协同、强化学习、可视化工具和自定义训练环境相关的问题，使得开发者可以更高效地构建和训练智能代理，得到意想不到的结果。论文通过研究在Unity环境开发游戏中实施机器学习技术，结合现阶段的实际需求，使其更加智能化，丰富完善更好的游戏体验。

关键词

Unity；ML-Agents；Pytorch；机器学习

1 引言

随着游戏产业的不断发展，人工智能（AI）在游戏开发中的地位也日益重要。Unity 作为一款广泛使用的游戏引擎，其强大的功能和灵活性使得它成为了许多游戏开发者的首选。为了实现更复杂的 AI 逻辑，开发者需要借助一些机器学习插件，ML-Agents 是一款基于 Unity 引擎的机器学习插件，它为开发者提供了一个易于使用的框架，用于创建、训练和部署人工智能代理。通过 ML-Agents，开发者可以轻松地实现复杂的 AI 逻辑，并让游戏中的角色展现出高度智能的行为。

2 开发平台的搭建

Unity ML-Agents 包包含 Unity C# 软件开发工具包（SDK），集成到 Unity 场景中。扩展包是一些尚未在基本包中实现的实验性组件。

2.1 ML–Agents 包的安装

①打开或创建一个需要 ML 的 Unity 项目；

②进入 Unity 包管理器（窗口—包管理器）；

③找到 ML-Agents 包并单击 install。

2.2 Pytorch 包的安装

PyTorch 是一个开源的机器学习库，新版本的 ML-Agents 可以使用 Pytorch。要安装 Python 包，最好先创建一个虚拟环境，确保每个项目独立工作。

①创建虚拟环境。

在 Unity 项目的基论文件夹中打开命令行；

创建一个文件夹名为 vr_env 的虚拟环境：

D:\Unity\Example>python -m venv vr_env

②安装 PyTorch 包。

从虚拟环境文件夹导航到 Scripts 文件夹，并启动

【作者简介】胡永祥（1969–），男，中国江苏淮安人，硕士，教授，从事计算机应用技术（虚拟现实、AI等方向）研究。

activate.bat 文件。如果在这些“<>”符号之间可以看到文件夹名称，则表示虚拟环境正在运行。

D:\Unity\Example>cd vr_env/Scripts

D:\Unity\Example\vr_env\Scripts>activate.bat

< vr_env > D:\Unity\Example\vr_env\Scripts>

更新 pip 安装程序（可选）

< vr_env > D:\Unity\Example\vr_env\Scripts>python -m pip install –upgrade pip

根据 GitHub 页面安装最新推荐的 PyTorch 包。

< vr_env > D:\Unity\Example\vr_env\Scripts>pip install torch~-1.7.1 -f https://download.pytorch.org/whl/torch_stable.html

③ Python ML-Agents 包安装。

先确保虚拟环境已定位并激活，再安装 ML-Agents 包：

< vr_env > D:\Unity\Example\vr_env\Scripts>pip install mlagents

最后通过运行命令“mlagents-learn --help”检查安装是否成功。

3 Unity 中机器学习的实现

Unity 3D 由 Unity Technologies 开发的专业游戏引擎，是一个实时 3D 互动内容创作和运营平台，Unity 平台提供一整套完善的软件解决方案，可用于创作、运营和变现任何实时互动的 2D 和 3D 内容，支持平台包括手机、平板电脑、PC、游戏主机、增强现实和虚拟现实设备。论文将最新的机器学习技术与 Unity 相结合，提高研发效率，Unity 机器学习代理工具应运而生。

3.1 Unity 机器学习代理 ML–Agents

ML-Agents 是一款基于 Unity 引擎的机器学习插件，它使得我们可以在游戏环境和模拟环境中训练智能 Agent。这款插件包含了三个高级组件：学习环境、Python API 和外部通讯器。学习环境包含了 Unity 场景和所有游戏角色，Python API 中包含用于训练的所有机器学习算法，将外部学习环境与 Python API 连接起来。通过简单易用的 Python API，运用增强学习、进化策略以及其他机器学习方法。机器学习代理对于高度逼真环境中的复杂机器学习问题具有显著的优势。

ML-Agents 具有多种功能和特点。首先，它支持多种机器学习算法，包括强化学习、模仿学习、神经进化等，开发者可以根据实际需求选择最适合的算法。其次，它提供了易于使用的 Python API，使得开发者可以快速构建和训练智能代理。此外，ML-Agents 还集成了可视化工具和自定义训练环境的功能，帮助开发者更好地理解和监控智能代理的行为。

ML-Agents 对于游戏开发和 AI 研究人员来说都非常有用。它提供了一个集中的平台，使得我们可以在 Unity 的丰富环境中测试 AI 的最新进展，并使结果为更多的研究者和游戏开发者所用。通过使用 ML-Agents，开发者可以创造出更加智能、更加有趣的游戏体验。

3.2 在 Unity 项目中使用 ML–Agents 基本步骤

①创建一个容纳 Agent 的环境：该环境可以从包含少量对象的简单物理模拟环境到整个游戏或生态系统，环境的样式可以多种多样。

②实现 Agent 子类：Agent 子类定义了必要的代码以供 Agent 用于观测自身环境、执行指定动作以及计算用于强化训练的奖励。你同样可以实现可选方法，从而在 Agent 完成任务或任务失败时重置 Agent。

③将实现 Agent 子类的脚本加到适当的 GameObject 上，当该对象在场景中，即代表对应 Agent 在模拟环境中了。

3.3 创建 Unity 项目打开场景

先新建一个 Unity 项目，并且将 ML-Agents 包导入里面，在 Unity 菜单“Edit”->“Project Settings”，在弹出的窗口中，找到“Player”，将“Api Compathbility Level”改为“.NET 4.x”，在 Unity 中需要将 ML-Agents 插件导入 Unity 中。创建一个简单的 ML-Agent 环境。该环境的“physical”组件包括一个 Plane（充当 Agent 移动的地板）、一个 Cube（充当 Agent 寻找的目标）和一个 Sphere（表示 Agent 本身），添加 Agent 球体，场景如图 1 所示。

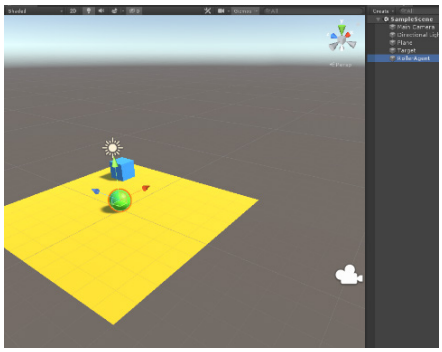

图 1 Unity 项目打开场景

4 机器学习训练过程实现

4.1 实现 Agent

创建 Agent：

①选择 RollerAgent 游戏对象以便在 Inspector 窗口中查看该对象。

②单击 Add Component。

③在组件列表中单击 New Script（位于底部）。

④将该脚本命名为“RollerAgent”。

⑤单击 Create and Add。

4.2 动作（Actions）

Brain 的决策以动作数组的形式传递给 AgentAction() 函数。此数组中的元素主要由 agent 的 Brain 的 Vector Action、Space Type 和 Space Size 来决定的。这里分别代表了向量运动空间、向量运动空间类型以及向量运动空间数，ML-Agents 将动作分为两种：Continusous 向量运动空间是一个可以连续变化的数字向量；Discrete 向量运动空间则将

动作定义为一个表，提供给 Agent 的具体动作是这个表的索引。

4.3 奖励（Rewards）

Reinforcement Learning（强化学习）需要奖励。同样奖励（惩罚）也在 AgentAction() 函数中实现，与上面动作实现的重写函数在一起。学习算法使用在模拟和学习过程的每个步骤中分配给 agent 的奖励来确定是否为 agent 提供了最佳的动作。当 Agent 完成任务时，对它进行奖励。在这个示例中，如果 Agent（小球）到达了目标位置（方块），则给它奖励 1 分。

4.4 AgentAction() 方法

上面的动作和奖励构成了 AgentAction() 方法，主要理解里面每一步的意义是为何，最后 AgentAction() 方法如下：RollerAgent 会计算到达目标所需的距离，当到达目标时，我们可以通过 Agent.SetReward() 方法来将 agent 标记为完成，并给它奖励 1 分，同时使用 Done() 方法来重置环境。

```
public override void AgentAction(float[]      vectorAction)
{
   //Space Type=Continuous  Space Size=2
   Vector3 controlSignal = Vector3.zero;
   controlSignal.x = vectorAction[0];//x 轴方向力
   controlSignal.z = vectorAction[1];//z 轴方向力
    // 当然上面这两句可以互换，因为 Brain 并不知道 action[] 数组中数值具体含义
   rBody.AddForce(controlSignal * speed);

   // 计算自身与目标的距离
    float distanceToTarget = Vector3.Distance(transform.position, Target.position);

   // 不同情况进行奖励
   if (distanceToTarget < 1.42f)
   {// 到达目标附近
      SetReward(1);
      Done();
   }
   if (transform.position.y < 0)
   {// 小球掉落
      Done();
   }
}
```

4.5 训练模型

找到我们之前建好的训练环境，到 ML-Agents 的根目录：

```
< vr_env > D:\Unity\Example\vr_env\Scripts>
```

输入命令：

```
mlagents-learn config/trainer_config.yaml --run-id=RollerBall-1 --train
```

运行 Unity 中的程序，如果 Unity 与 Pytorch 训练环境成功通讯，则会在命令行中发现训练配置，同时，可以看到 Unity 中小球开始自己快速运动，方块也会根据不同状态来随机生成（见图 2）。

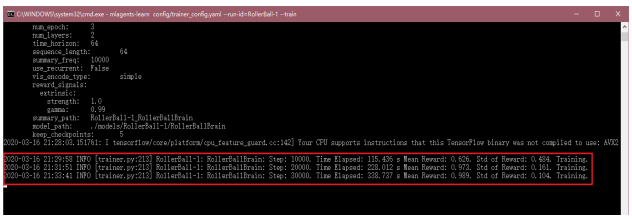

图 2 Unity 中的程序

然后选中场景中的小球，将 Behavior Parameters 组件中 Model 属性中，选择刚才训练好的模型，并将 Behavior Type 选为 Inference Only，然后点击运行，就可以看到小球利用我们训练好的模型开始找方块了（见图 3）。

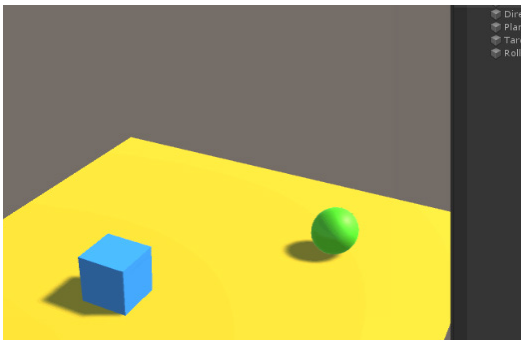

图 3 小球利用模型找方块

5 结语

ML-Agent 支持多种机器学习算法和模型，包括深度学习、强化学习等。这意味着开发者可以根据实际需求选择最适合的机器学习技术，以实现最佳的游戏 AI 效果。在 Unity 中使用机器学习插件 ML-Agent 是一种强大的 AI 解决方案。它不仅提供了多种机器学习技术和强大的多智能体协同功能，还集成了可视化工具和自定义训练环境的功能。通过使用 ML-Agent，开发者可以更高效地构建和训练智能代理，从而创造出更加智能、更加有趣的游戏体验。

参考文献

[1] 余涛,贾如春.基于机器学习算法人工智能技术的发展与应用[J].数学学习与研究,2019(9).

[2] 张量,金益,刘媛霞,等.虚拟现实(VR)技术与发展研究综述[J].信息与电脑(理论版),2019(11).

[3] 万里鹏,兰旭光,张翰博,等.深度强化学习理论及其应用综述[J].模式识别与人工智能,2019(8).

[4] 赵勇,张引琼.Windows下基于TensorFlow平台的Unity3D机器学习[J].电脑知识与技术,2018(6).