

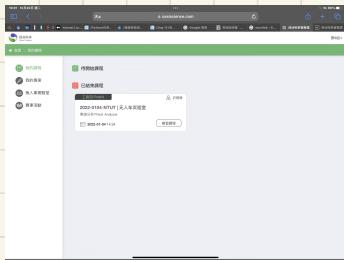
軍事、運動比賽
repo 分享 (發現 → 改進 → 驗證)

going 3D print model
basis license MIT Tech
oasisscience.com

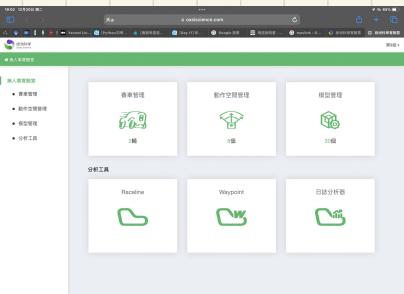
user
pass

994544
994544

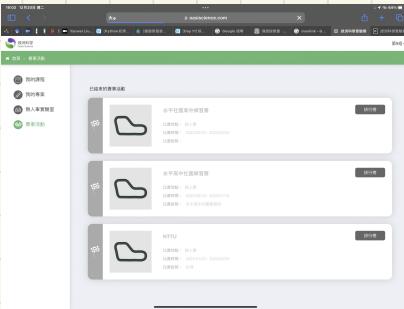
其 15 hr training time



手動操車
由



訓練



課室 - 66 番

Deep Racer

舊型 Training 方式：

on learning base

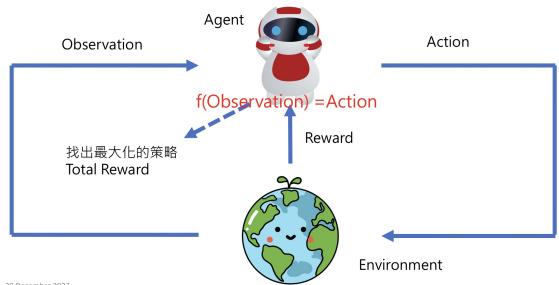
1. 人類先示範給 agent 看 (搜集 phase)
2. 看到什麼樣做什麼動作
相關連起來

行為複製
behavior clone 缺點：

無法窮盡所有可能

	強AI	弱AI
探索能力	○	✗
標籤 Label	✗	○
人類介入	✗	○

強化學習(Reinforcement Learning)



模型管理 Reward
動作空間管理 action


```

1 def reward_function(params):
2     ...
3     # 检查是否在赛道内
4     ...
5     # 检查所有轮子是否在赛道上
6     all_wheels_on_track = params['all_wheels_on_track']
7
8     # 给予一个很轻的奖励以鼓励赛道行驶
9     reward = 1.0
10
11     # 如果所有轮子都在赛道上，给予奖励
12     if all_wheels_on_track:
13         reward = 1.0
14
15     return float(reward)

```

→ get params

all_wheels_on_track

所有輪胎在道路上

上一步

下一步

创建模型 - 奖励函数

```

1 def reward_function(params):
2     ...
3     # 检查是否在赛道内
4     ...
5     # 检查所有轮子是否在赛道上
6     all_wheels_on_track = params['all_wheels_on_track']
7
8     # 给予一个很轻的奖励以鼓励赛道行驶
9     reward = 1.0
10
11     # 检查是否在赛道中心附近
12     if is_left_of_center:
13         reward = -1.0
14
15     return float(reward)

```

→ get params

is-left-of-center

is-right-of-center

中心左边

上一步

下一步

创建模型 - 奖励函数

```

1 def reward_function(params):
2     ...
3     # 检查是否在赛道内
4     ...
5     # 检查所有轮子是否在赛道上
6     all_wheels_on_track = params['all_wheels_on_track']
7
8     # 给予一个很轻的奖励以鼓励赛道行驶
9     reward = 1.0
10
11     # 检查是否在赛道中心附近
12     if is_left_of_center:
13         reward = -1.0
14
15     if is_right_of_center:
16         reward = -1.0
17
18     return float(reward)

```

中心偏左

最近的道路点

上一步

下一步

三个并排的电脑屏幕显示了同一个名为“奖励模型 - 基础函数”的编辑器窗口，展示了Python代码。代码实现了一个名为`reward_function`的函数，该函数根据agent与赛道中心线的距离返回一个浮点奖励值。

左侧屏幕上的代码注释说明了如何根据距离惩罚agent，右侧屏幕上的注释则说明了如何奖励agent靠近中心线的行为。右侧屏幕还包含手写批注：“track_width 道路宽度”和“distance_from_center 离中心距离”，以及“一阶惩罚系数 2.0”。

中间屏幕显示了相同的代码，但注释部分被修改为：“distance_from_center * parameter[1] * distance_center”。

右侧屏幕上的手写批注：“一阶惩罚系数 2.0”旁边有箭头指向注释部分，下方批注：“转向超过 Threshold”指向代码中的`steering_angle_threshold`部分。

```
1 def reward_function(params):
2     # Read input parameters
3     # Example of reading the agent's position
4     # distance_center = params[0]
5     # distance_left = params[1]
6     # distance_right = params[2]
7     # distance_to_center = abs(distance_center)
8     # distance_left_center = abs(distance_left)
9     # distance_right_center = abs(distance_right)
10    # distance_center * parameter[1] * distance_center
11    # distance_left_center * parameter[1] * distance_left_center
12    # distance_right_center * parameter[1] * distance_right_center
13
14    # Calculate 3 markers that are at varying distances away from the center line
15    marker_1 = 0.2 * track_width
16    marker_2 = 0.5 * track_width
17    marker_3 = 0.8 * track_width
18
19    # Give higher reward if the car is closer to center line and vice versa
20    if distance_center <= marker_1:
21        reward = 1.0
22    elif distance_center >= marker_2:
23        reward = 0.5
24    else:
25        reward = 0.1
26
27    # Give a high reward if no wheels go off the track and slightly lower reward if at least one wheel is off the track
28    if all_wheels_on_track and (track_width - distance_from_center) >= 0.5:
29        reward += 0.1
30
31    # Reward for not crashing
32    if steering != 0.0:
33        reward -= 0.5
34
35    # Reward for not crashing close to off track
36
37    return float(reward)
```

创建模型 - 提炼函数

```
1 def reward_function(params):
2     # Example of rewarding an agent to stay inside two borders
3     # and penalizing getting too close to the objects in front
4     #
5     #
6     # All Lanes & on-track & param[1] = wheel_in_track
7     distance_from_center = params['distance_from_center']
8     track_width = params['track_width']
9     objects_distaned = params['closest_objects']
10    objects_lanes = params['closest_lane']
11    next_object_index = params['closest_lane_index']
12    objects_left_of_center = params['left_of_center']
13    objects_right_of_center = params['right_of_center']
14    is_left_of_center = params['is_left_of_center']
15    #
16    # Initialize reward with a small number but not zero
17    # to decide zero means off-track or crashed
18    reward = 0.0
19    #
20    # Reward if the agent stays inside the two borders of the track
21    if (is_left_of_center and (0.5 + track_width - distance_from_center)) >= 0.05:
22        reward += 1.0
23    else:
24        reward -= 1.0
25    #
26    # Reward if the agent is too close to the next object
27    reward -= 1.0
28    #
29    # Distance to the next object
30    distance_to_next_object = objects_distaned[next_object_index]
31    #
32    # If the next object is in the same lane as the agent
33    if objects_lanes[next_object_index] == objects_lanes[0]:
34        #
35        # If it's same lane
36        if (0.5 + distance_to_next_object) < 0.5:
37            reward -= 0.5
38        elif 0.5 + distance_to_next_object < 0.55:
39            reward -= 0.15
40        elif 0.5 + distance_to_next_object < 0.6:
41            reward -= 0.05
42        else:
43            reward -= 0.01
44    else:
45        reward -= 0.01
46    #
47    # Get reward
48    return reward
```

```
    # 檢查距離是否過近，並根據距離調整獎勵
    def calculate_reward(objects, left_of_center):
        reward = 0.0

        # 計算與最近對象的距離
        distance_to_nearest_object = min([obj['distance'] for obj in objects])
        if distance_to_nearest_object < 0.8:
            reward += -0.5 * (1 / distance_to_nearest_object)

        # 計算與左側對象的距離
        distance_to_left_objects = [obj['distance'] for obj in objects if obj['left_of_center']]
        if len(distance_to_left_objects) > 0:
            reward += -0.5 * (1 / min(distance_to_left_objects))

        # 計算與右側對象的距離
        distance_to_right_objects = [obj['distance'] for obj in objects if not obj['left_of_center']]
        if len(distance_to_right_objects) > 0:
            reward += -0.5 * (1 / min(distance_to_right_objects))

        # 根據距離調整獎勵
        reward += 0.5 * (1 / distance_to_nearest_object)
        reward += 0.5 * (1 / min(distance_to_left_objects))
        reward += 0.5 * (1 / min(distance_to_right_objects))

        return reward
```

10:59 12月29日 周一

https://www.zhihu.com/question/28411333

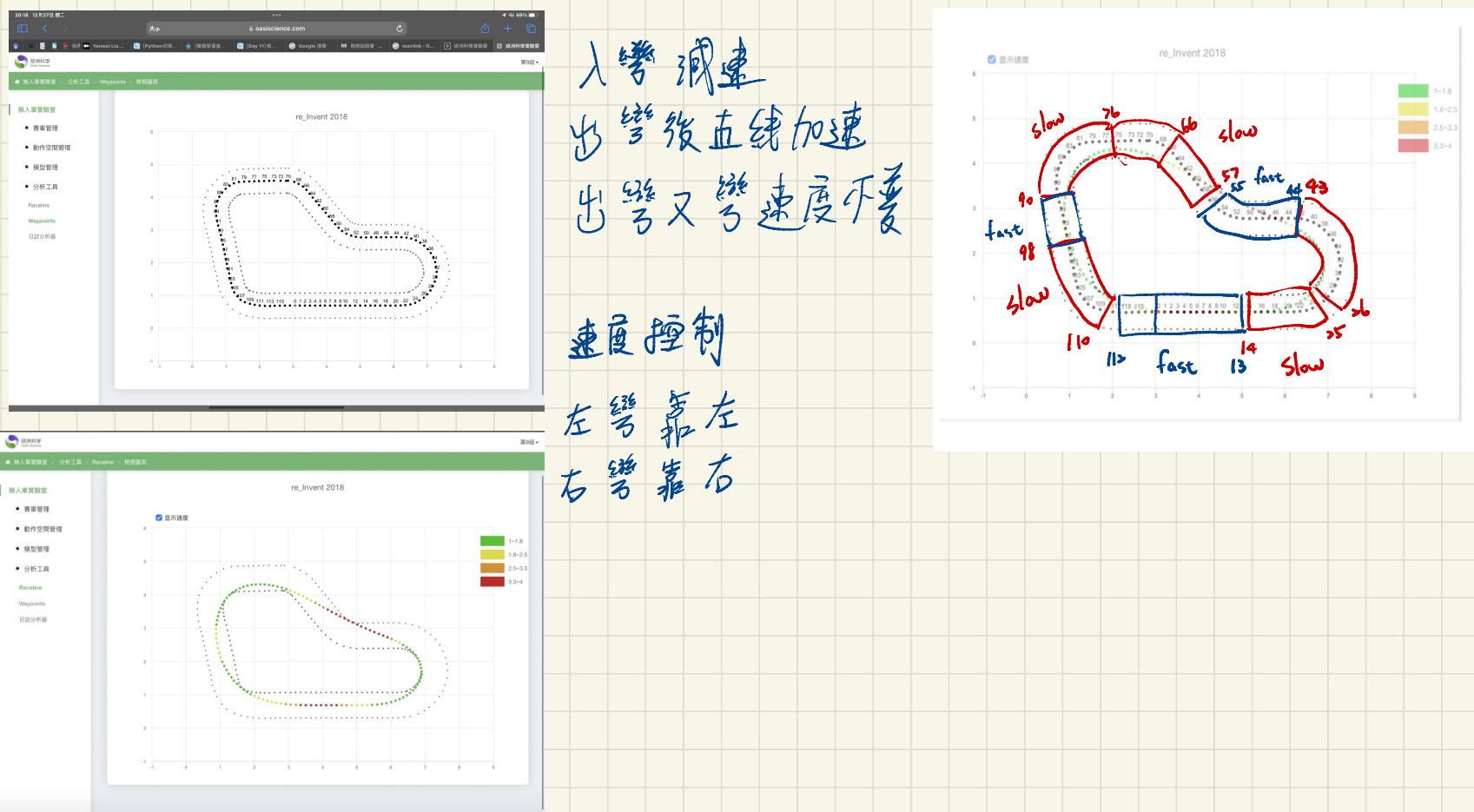
在 assistance.com

...         

割點模型 - 微調函數

代码:

```
12 global NF
13
14 # The 2D track, RANG2 = 1.0M, should be adjust for different track
15
16 waypoints = parse("waypoints1")
17 x = waypoints[0]
18 y = parse("y")
19 x = parse("x")
20 y = parse("y")
21 waypoints = waypoints[0:-1]
22
23 # Use optimization receive algorithm
24 # x_start = 0.0, y_start = 0.0, z_start = 0.0
25 # x_after = 0.0, y_after = 0.0, z_after = 0.0
26 # x_start = 0.0, y_start = 0.0, z_start = 0.0
27 # x_after = 0.0, y_after = 0.0, z_after = 0.0
28 # x_start = 0.0, y_start = 0.0, z_start = 0.0
29 # x_after = 0.0, y_after = 0.0, z_after = 0.0
30
31 # Distance to the raceline
32 distance = dist_2d_point_to_line(x_start, y_start, x_end, y_end)
33
34 reward = 0.0
35
36 if distance <= 0.05:
37     reward = 1.0
38 else:
39     if distance > 0.05:
40         reward = -0.1
41     elif distance > 0.1:
42         reward = -0.15
43     elif distance > 0.15:
44         reward = -0.2
45
46 return reward
```



Learning Rate = 越大步越快，結果二不精細

百分比 = 調整範圍

wheels = \bar{v} 調整



初期 Learning Rate

先快速

後期 慢慢降低

action 選多

探索空間 選大

[x, y, speed, time]

Training time 越久

觀察數據 → 分析 → 改進

應用實務

distribution

布局, 分配

ppo 算法

想解決樣本使用效率低的問題

off-policy

learning rate 如果選擇的太小，收斂速度會很慢，如果太大，loss function 就會在極小值處不停地震盪甚至偏離。

(有一種措施是先設定大一點的學習率，當兩次迭代之間的變化低於某個閥值後，就減小learning rate)。如果學習率太小，會導致網絡loss下降非常慢，或陷入局部最小值。如果學習率太大，那麼參數更新的幅度就非常大，產生不穩定的學習曲線，或者loss直接開始增加。

learning rate	越大	越小	多少誤差是要被學習的
學習	快	慢	
收斂	快	慢	

1 为什么说学习率和batchsize

目前深度学习模型多采用批量随机梯度下降算法进行优化，随机梯度下降算法的原理如下：

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \mathcal{B}} \nabla l(x, w_t).$$

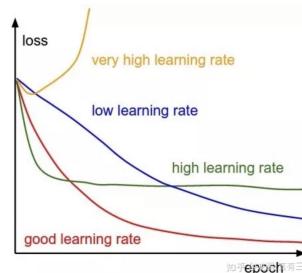
知乎 @龙鹏-言有三

n是批量大小(batchsize)，η是学习率(learning rate)。可知道除了梯度本身，这两个因子直接决定了模型的权重更新，从优化本身来看它们是影响模型性能收敛最重要的参数。

学习率直接影响模型的收敛状态，batchsize则影响模型的泛化性能，两者又是分子分母的直接关系，相互也会影响，因此这一次来详述它们对模型性能的影响。

2.1 初始学习率大小对模型性能的影响

初始的学习率肯定是一个最优值的，过大则导致模型不收敛，过小则导致模型收敛特别慢或者无法学习，下图展示了不同大小的学习率下模型收敛情况的可能性，图来自cs231n。



4 学习率和batchsize的关系

通常当我们增加batchsize为原来的N倍时，要保证经过同样的样本后更新的权重相等，按照线性缩放规则，学习率应该增加为原来的N倍[5]。但是如果要保证权重的方差不变，则学习率应该增加为原来的sqrt(N)倍[7]。目前这两种策略都被研究过，使用前者的明显居多。

从两种常见的调整策略来看，学习率和batchsize都是同时增加的。学习率是一个非常敏感的因素，不能太大，否则模型会不收敛。同样batchsize也会影响模型性能。那实际使用中都如何调整这两个参数呢？

研究[8]表明，衰减学习率可以通过增加batchsize来实现类似的效果，这实际上从SGD的权重更新公式就可以看出来两者确实是等价的，文中通过充分的实验证明了这一点。

研究[9]表明，对于一个固定的学习率，存在一个最优的batchsize能够最大化测试精度，这个batchsize和学习率以及训练集的大小正相关。

对此实际上是有两个建议：

- 如果增加了学习率，那么batchsize最好也跟着增加，这样收敛要稳定。
- 尽量使用大的学习率，因为很多研究都表明更大的学习率有利于提高泛化能力。如果真的要减小，可以尝试其他办法，比如增加batch size。学习率对模型的收敛影响的很大，慎重调整。

Batch Size - 指一次訓練的樣本數，影響 model 最佳化程度和速度
會 affect 梯度下降的方向

Epoch : 使用訓練集全部資料進行一次性的完整訓練，也稱一代訓練

Batch : 訓練集中的一小部分樣本對 model weight 進行一次反向傳播的權重更新
△ - 條件量

Iteration 迭代：使用一個 Batch 資料 model 進行一次權重更新的過程，稱一次訓練

batch_size

即批大小，如果把全部數據放入內存後再加载到磁盤中，空間是不夠的；如果一個一個數據加載訓練並更新模型參數，效率會低，所以考慮一批一本地加載數據，每次送進去的數量就是 batch_size。這樣可以加快速度。

用 minibatch 方法則會定義 batch_size，即把整個數據集分几份後，每份的大小就是 batch-size。假设把10000個樣本，分成500批次送過去，則每次送進20個样本，batch_size=20。

iteration

即迭代，表示每個循環的一遍，一次参数更新。用 minibatch 時就意味著訓練完一個 batch。

一個 epoch 的步驟 $\text{batch size} * \text{iteration}$ 。假设把10000個樣本，分成500批次送過去，則每次送進20個樣本，則 iteration=500，每經過一個 iteration，參數更新一次。

epoch

one forward pass and one backward pass of all the training examples,
in the neural network terminology。(在神經網絡語彙中，所有訓練示例的一次前向傳播和一次反向傳播)

一個 epoch 就是第一遍完整的訓練數據。假设把10000個樣本，這10000個樣本都跑完就算一個 epoch。實驗中一般需要訓練很多 epoch，取均值作為最後的結果，從而減少偶然性，避免直到局部極值。

即所有訓練數據跑一遍，如果有300條，把數據分成了3個 batch，batch_size 是 $300 / 3 = 100$ ，3個 batch 都跑完，即跑了三個 iteration，就是一個 epoch。

episode

episode 常用于強化學習，以遊戲舉例，例如 模型訓練⁽¹⁾ 中達成迭代結束後，玩一局遊戲（例如玩一局飛機大戰）看看本局遊戲贏得多少獎勵。无论谁来还是失败，都是一个 episode。如果玩兩局，从头下到尾的一局就是一个 episode。

一個 epoch 包含多個 episode

$$\text{Dataset} = 10000$$

$$\text{batch size} = 10$$

一個 epoch 包含 $(10000 / 10) = 1000$ 個 step 或 iteration

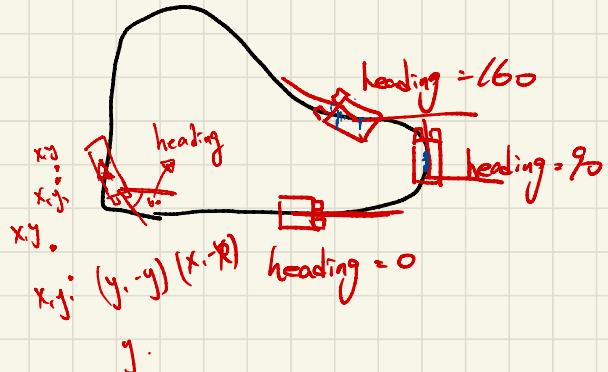
$$\text{設 episode} = 100$$

一個 episode 包含 100 個 step。

一個 epoch 包含 $(1000 / 100) = 10$ 個 episode

heading

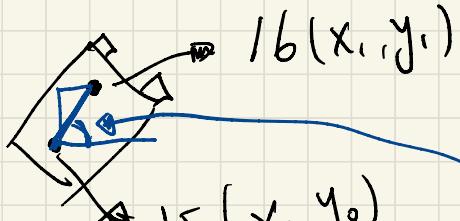
agent車頭 对 X 軸角度



$$|(direction - heading)| =$$

$$180 - 90 = 90$$

ex:



$$(y_1 - y_0), (x_1 - x_0) \\ (y, x) = \text{角度}$$

direction

無法收斂：Training time 太短
速度 > 位置

若曲線趨勢整體有上升的趨勢，就在給 Training time