Chapter 1: Introducing Machine Learning and ML-Agents

<table>
<thead>
<tr>
<th>shot</th>
<th>velocity</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>2.28</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>9.12</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>36.49</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>57.02</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>111.75</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
<td>145.96</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>228.06</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>328.41</td>
</tr>
<tr>
<td>9</td>
<td>65</td>
<td>385.42</td>
</tr>
<tr>
<td>10</td>
<td>75</td>
<td>513.14</td>
</tr>
</tbody>
</table>

$y = 5.3527x$

$R^2 = 0.8595$

$d = 5.3527v$
\[
v = \frac{d}{5.3527}
\]

\[
v = \frac{300}{5.3527} = 56.05
\]
Chapter 2: The Bandit and Reinforcement Learning

\[ V(a) = V(a) + \alpha \times (r - V(a)) \]

\[ V(a) = \]

Configuration

Completed

Multi-armed Bandit (stateless)

Contextual Bandit

Full RL
\[ Q(s, a) = Q(s, a) + \alpha \times (r - Q(s, a)) \]
\[ Q(s, a) = r + \gamma \max_{a'} Q(s', a') \]

\[ r = \text{reward} \]

\[ \gamma = \text{gamma (reward discount factor 0 \text{--} 1.0)} \]

\[ \max_{a'} = \text{maximum of all actions for state} \]
\[ Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha (r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_{t+1}, a)) \]

\[ \alpha = \text{learning rate} \]
Chapter 3:
Deep Reinforcement Learning with Python

```
Advanced Options

- Add Anaconda to my PATH environment variable
  Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

- Register Anaconda as my default Python 3.6
  This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.6 on the system.

Anaconda, Inc.

[Install] [Back] [Cancel]
```
```python
model = Sequential()
model.add(Flatten(input_shape=(1,) + env.observation_space.shape))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(nb_actions))
model.add(Activation('linear'))
policy = EpsGreedyQPolicy()
memory = SequentialMemory(limit=50000, window_length=1)
dqn = DQNAgent(model=model, nb_actions=nb_actions, memory=memory, policy=policy, target_model_update=1e-2, policy=policy)
dqn.compile(Adam(lr=1e-3), metrics=[['mae']])
```
from rl.
from rl.
from rl.

ENV_NAME = "your_env_name"

# Get the environment
env = gym.make(ENV_NAME)
np.random.seed(0)
env.seed(0)
nb_actions = env.action_space.n

model = Sequential()
model.add(Dense(64, input_shape=(100,)))
model.add(Dense(32, activation='relu'))
model.add(Dense(nb_actions, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

print(model.summary())
policy = EpsGreedyQPolicy()
Chapter 4: Going Deeper with Deep Learning
\[ \text{Advantage : } A = Q(s, a) - V(s) \]

\[ \text{Advantage : } A = R - V(s) \]

\[ \text{ValueLoss : } L = \sum (R - V(s))^2 \text{(SumSquaredError)} \]

\[ \text{PolicyLoss : } L = -\log(\pi(a|s)) \cdot A(s) \]

\[ H(\pi) = -\sum P(x) \log(P(x)) \]

\[ \text{PolicyLoss : } L = -\log(\pi(a|s)) \cdot A(s) - \beta \cdot H(\pi) \]

\[ L = 0.5 \cdot \sum (R - V(s))^2 - \log(\pi(a|s)) \cdot A(s) - \beta \cdot H(\pi) \]
Chapter 5: Playing the Game
### Inspector
**Academy**
- Static: False
- Tag: Untagged

**Transform**
- **Wall Jump Academy (Script)**
  - Script: WallJumpAcademy
  - Max Steps: 10000

**Reset Parameters**
- big_wall_min_height: 8
- small_wall_height: 4
- no_wall_height: 0
- big_wall_max_height: 8

**Specific to WallJump**
- Agent Run Speed: 1.5
- Agent Jump Height: 2.75
- Goal Scored Material: SuccessGround
- Fail Material: FailGround
INFO:unityagents:
Lesson changed. Now in Lesson 8: big_wall_min_height -> 4.0, big_wall_max_height -> 8.0
INFO:unityagents: SmallWallBrain: Step: 90000. Mean Reward: 0.827. Std of Reward: 0.151.
INFO:unityagents: BigWallBrain: Step: 90000. Mean Reward: 0.345. Std of Reward: 0.761.
INFO:unityagents: Saved Model
INFO:unityagents: SmallWallBrain: Step: 100000. Mean Reward: 0.449. Std of Reward: 0.744.
INFO:unityagents: BigWallBrain: Step: 100000. Mean Reward: 0.540. Std of Reward: 0.753.
INFO:unityagents: SmallWallBrain: Step: 102000. Mean Reward: 0.903. Std of Reward: 0.027.
INFO:unityagents: BigWallBrain: Step: 102000. Mean Reward: -0.221. Std of Reward: 0.995.
INFO:unityagents:
Lesson changed. Now In Lesson 9: big_wall_min_height -> 4.5, big_wall_max_height -> 8.5
INFO:unityagents: SmallWallBrain: Step: 104000. Mean Reward: 0.655. Std of Reward: 0.596.
INFO:unityagents: BigWallBrain: Step: 104000. Mean Reward: 0.432. Std of Reward: 0.749.
INFO:unityagents: BigWallBrain: Step: 106000. Mean Reward: 0.414. Std of Reward: 0.694.
Chapter 6: Terrarium Revisited – A Multi-Agent Ecosystem