Artificial Intelligence A-Z™: Learn How To Build An AI

SuperDataScience Team

Abstract

This course will help give you the core concepts using Reinforcement Learning in AI along with some practical models and annexes for neural networks. We also use PyTorch as one of the libraries which is becoming one of the most prominent for AI/Deep Learning research and it's built with Python. I'd recommend pulling up documentation as you progress along with the recommended readings that can be found here: https://www.superdatascience.com/artificial-intelligence/.

First, we would like to thank all of the students who have contributed to the discussions and Q&A sections. It's always a great idea and method of enhancing the materials to be able to discuss key concepts. In the following PDF you will see some of the main questions that we see frequently asked. If you don't see the answer that you are looking for please post in the Q&A in the course. In addition, the following section has some of the main setup and general questions for the entire course:

- **I'm having some audio and video issues, is there any recommended troubleshooting steps?** Yes, you can use the following steps provided by Udemy at:

  `https://support.udemy.com/hc/en-us/articles/229231227-Video-Audio-Issues-Troubleshooting`

  and also at:

  `https://support.udemy.com/hc/en-us/articles/360003054894-Troubleshooting-Issues-With-Udemy-s-App`

- **What IDE is the instructor using?** We are using Spyder inside Anaconda.
- **Where can I access the course materials and recommended readings?** You can visit the following link to obtain the materials

  `https://www.superdatascience.com/artificial-intelligence/`

  .
- **Which version of PyTorch will work with our algorithms and is it supported by Windows?** PyTorch does support Windows but we need to use versions ¡ 0.4.0 of PyTorch for our algorithms. You can find the installation steps, whl files and more at the following for Mac and Linux: `https://pytorch.org/previous-versions/` and for Windows at: `https://github.com/peterjc123/pytorch-scripts`.
- **Which Python version does the course use?** The course was originally built with Python 2.7 but Python 3 can be used for the modules. Since we are working with Anaconda you can set up multiple environments with different Python versions.
- **I'm running into an error stating: TypeError: multinomial() missing 1 required positional arguments: "num_samples", what can I do?** To resolve this you need to install a PyTorch version ¡ 0.4.0. You can check the current version by running pip list

or conda list in the terminal or prompt and checking the output. Please see the steps mentioned above to install PyTorch.

- **Where can I obtain the Artificial Intelligence Handbook?** You can obtain the handbook by visiting `https://www.superdatascience.com/artificial-intelligence/` and scroll down until you see the section for the Artificial Intelligence A-Z Handbook
- **For an updated document on the common debugging tips (recently added Breakout and Doom setup for Windows and YML files for specific environments) please visit lecture 29** `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/8824208?start=0`

Table of Contents

Summary

1 Section: 1 - Welcome To The Course **8**
- 1.1 Lecture 1 - Why AI? . . . 8
- 1.2 Lecture 2 - Introduction . . . 8
- 1.3 Lecture 3 - Where to get the Materials . . . 8
- 1.4 Lecture 4 - Some Additional Resources!! . . . 8

2 Section: 2 - Fundamentals of Reinforcement Learning **9**
- 2.1 Lecture 5 - Welcome to Part 0 - Fundamentals of Reinforcement Learning . . . 9

3 Section: 3 Q Learning Intuition **9**
- 3.1 Lecture 6 - Plan of Attack . . . 9
- 3.2 Lecture 7 - What is reinforcement learning? . . . 9
- 3.3 Lecture 8 - The Bellman Equation . . . 9
- 3.4 Lecture 9 - The "Plan" . . . 10
- 3.5 Lecture 10 - Markov Decision Process . . . 10
- 3.6 Lecture 11 - Policy vs Plan . . . 10
- 3.7 Lecture 12 - Adding a "Living Penalty" . . . 11
- 3.8 Lecture 13 - Q-Learning Intuition . . . 12
- 3.9 Lecture 14 - Temporal Difference . . . 12
- 3.10 Lecture 15 - Q-Learning Visualization . . . 12

4 Section 4 - Part 1 - Self-Driving Car (Deep Q-Learning) **13**
- 4.1 Lecture 16 - Welcome to Part 1 - Self-Driving Car (Deep Q-Learning) . . . 13

5 Section 5 - Deep Q-Learning Intuition **13**
- 5.1 Lecture 17 - Plan of Attack . . . 13
- 5.2 Lecture 18 - Deep Q-Learning Intuition - Learning . . . 13
- 5.3 Lecture 19 - Deep Q-Learning Intuition - Acting . . . 14
- 5.4 Lecture 20 - Experience Replay . . . 14
- 5.5 Lecture 21 - Action Selection Policies . . . 14

6 Section 6 - Installation for Part 1 **15**
- 6.1 Lecture 22 - Plan of Attack (Practical Tutorials) . . . 15
- 6.2 Lecture 23 - Where to get the Materials . . . 15
- 6.3 Lecture - 24. Windows Option 1: End-to-End installation steps . . . 15
- 6.4 Lecture 25 - Windows Option 2 - Part A: Installing Ubuntu on Windows . . . 16
- 6.5 Lecture 26 - Windows Option 2 - Part B: Installing PyTorch and Kivy on your Ubuntu VM . . . 16
- 6.6 Lecture 27 - Mac or Linux: Installing Anaconda . . . 16

6.7 Lecture 28 - Mac or Linux: Installing PyTorch and Kivy 16
6.8 Lecture 29 - Common Debug Tips 17
6.9 Lecture 30 - Getting Started 17

7 Section 7 - Creating the environment 17
7.1 Lecture 31 - Self Driving Car - Step 1 17
7.2 Lecture 32 - Self Driving Car - Step 2 18

8 Section 8 - Building an AI 19
8.1 Lecture 33 - Self Driving Car - Step 3 19
8.2 Lecture 34 - Self Driving Car - Step 4 19
8.3 Lecture 35 - Self Driving Car - Step 5 19
8.4 Lecture 36 - Self Driving Car - Step 6 20
8.5 Lecture 37 - Self Driving Car - Step 7 20
8.6 Lecture 38 - Self Driving Car - Step 8 20
8.7 Lecture 39 - Self Driving Car - Step 9 20
8.8 Lecture 40 - Self Driving Car - Step 10 21
8.9 Lecture 41 - Self Driving Car - Step 11 21
8.10 Lecture 42 - Self Driving Car - Step 12 21
8.11 Lecture 43 - Self Driving Car - Step 13 22
8.12 Leture 45 - Self Driving Car - Step 15 22
8.13 Lecture 46 - Self Driving Car - Step 16 23

9 Section 9 - Playing with the AI 23
9.1 Lecture 47 - Self Driving Car - Level 1 23
9.2 Lecture 48 - Self Driving Car - Level 2 23
9.3 Lecture 49 - Self Driving Car - Level 3 24
9.4 Lecture 50 - Self Driving Car - Level 4 24
9.5 Lecture 51 - Challenge Solutions 24

10 Section 10 - Part 2 - Doom (Deep Convolutional Q-Learning) 24
10.1 Lecture 52 - 52. Welcome to Part 2 - Doom (Deep Convolutional Q-Learning) 24

11 Section 11 - Deep Convolutional Q-Learning Intuition 25
11.1 Lecture 53 - Plan of Attack 25
11.2 Lecture 54 - Deep Convolutional Q-Learning Intuition 25
11.3 Lecture 55 - Eligibility Trace 25

12 Section 12 - Installation for Part 2 25
12.1 Lecture 56 - Where to get the Materials 25
12.2 Lecture 57 - Installing Open AI Gym and ppaquette 26
12.3 Lecture 58 - Installing Open AI Gym Walk Through (Mac Version) 26
12.4 Lecture 59 - Installing Open AI Gym Walk Through (Ubuntu Version) 26
12.5 Lecture 60 - Common Debug Tips 26

13 Section 13 - Building an AI **26**
13.1 Lecture 61 - Doom - Step 1 26
13.2 Lecture 62 - Doom - Step 2 26
13.3 Lecture 63 - Doom - Step 3 26
13.4 Lecture 64 - Doom - Step 4 26
13.5 Lecture 65 - Doom - Step 5 26
13.6 Lecture 66 - Doom - Step 6 27
13.7 Lecture 67 - Doom - Step 7 27
13.8 Lecture 68 - Doom - Step 8 27
13.9 Lecture 69 - Doom - Step 9 27
13.10Lecture 70 - Doom - Step 10 27
13.11Lecture 71 - Doom - Step 11 27
13.12Lecture 72 - Doom - Step 12 27
13.13Lecture 73 - Doom - Step 13 27
13.14Lecture 74 - Doom - Step 14 27
13.15Lecture 75 - Doom - Step 15 27
13.16Lecture 76 - Doom - Step 16 28
13.17Lecture 77 - Doom - Step 17 28

14 Section 14 - Playing with the AI **28**
14.1 Lecture 78 - Watching our AI play Doom 28

15 Section 15 - ————- Part 3 - Breakout (A3C) ————- **28**
15.1 Lecture 79 - Welcome to Part 3 - Breakout (A3C) 28

16 Section 16 - A3C Intuition **28**
16.1 Lecture 80 - Plan of Attack 28
16.2 Lecture 81 - The three A's in A3C 28
16.3 Lecture 82 - Actor-Critic 28
16.4 Lecture 83 - Asynchronous 29
16.5 Lecture 84 - Advantage 29
16.6 Lecture 85 - LSTM Layer 29

17 Section 17 - Installation for Part 3 **29**
17.1 Lecture 86 - Installing OpenCV 29

18 Section 18 - Building an AI **29**
18.1 Lecture 87 - Breakout - Step 1 29
18.2 Lecture 88 - Breakout - Step 2 29
18.3 Lecture 89 - Breakout - Step 3 29
18.4 Lecture 90 - Breakout - Step 4 30
18.5 Lecture 91 - Breakout - Step 5 30
18.6 Lecture 92 - Breakout - Step 6 30
18.7 Lecture 93 - Breakout - Step 7 30
18.8 Lecture 94 - Breakout - Step 8 30
18.9 Lecture 95 - Breakout - Step 9 30

18.10Lecture 96 - Breakout - Step 10 30
18.11Lecture 97 - Breakout - Step 11 30
18.12Lecture 98 - Breakout - Step 12 30
18.13Lecture 99 - Breakout - Step 13 30
18.14Lecture 100 - Breakout - Step 14 30
18.15Lecture 101 - Breakout - Step 15 31

19 Section 19 - Annex 1: Artificial Neural Networks 31
19.1 Lecture 102 - What is Deep Learning? 31
19.2 Lecture 103 - Plan of Attack 31
19.3 Lecture 104 - The Neuron 31
19.4 Lecture 105 - The Activation Function 31
19.5 Lecture 106 - How do Neural Networks work? 31
19.6 Lecture 107 - How do Neural Networks learn? 31
19.7 Lecture 108 - Gradient Descent 32
19.8 Lecture 109 - Stochastic Gradient Descent 32
19.9 Lecture 110 - Backpropagation 32

20 Section 20 - Annex 2: Convolutional Neural Networks 33
20.1 Lecture 111 - Plan of Attack 33
20.2 Lecture 112 - What are convolutional neural networks? 33
20.3 Lecture 113 - Step 1 - Convolution Operation 33
20.4 Lecture 114 - Step 1(b) - ReLU Layer 33
20.5 Lecture 115 - Step 2 - Pooling 33
20.6 Lecture 116 - Step 3 - Flattening 33
20.7 Lecture 117 - Step 4 - Full Connection 33
20.8 Lecture 118 - Summary 33
20.9 Lecture 119 - Softmax & Cross-Entropy 33

21 Conclusion 33

1 Section: 1 - Welcome To The Course

1.1 Lecture 1 - Why AI?

- **I don't have a background in technology, can I still learn about AI with this course?** Absolutely, this course will help give you the core concepts using Reinforcement Learning in AI along with some practical models and annexes for working with neural networks. We also use PyTorch as one of the libraries which is becoming one of the most prominent for AI/Deep Learning research and it's built with Python. I'd recommend pulling up documentation as you progress along with the recommended readings that can be found here: `https://www.superdatascience.com/artificial-intelligence/`.

1.2 Lecture 2 - Introduction

- **My high school math isn't that strong, will I be able to understand the concepts?** Yes, you can understand the main AI concepts that we discuss since we always try to make the complex simple. In addition it's always a great idea to always reinforce the math knowledge as you go along with the lectures. You can discuss concepts in the QA section as you progress and view outside documentation for math concepts. The additional reading materials are also accessible clarify concepts further at: `https://www.superdatascience.com/artificial-intelligence/`. If you are progressing with the lectures it might help to pull up for example on the bellman equation further resources if the math isn't clear but you can see the main ideas of reinforcement learning and the practical approach with the modules.

1.3 Lecture 3 - Where to get the Materials

- **Where can I obtain the course materials?** They can be obtained at: `https://www.superdatascience.com/artificial-intelligence/`

1.4 Lecture 4 - Some Additional Resources!!

- Altough this isn't a question we want to make this resource available to the students. In order to ease in to this amazing field, we've selected a great episode you can listen to on your commute, at breakfast or wherever to get you fired up to continue in this fantastic journey.Click here to get started: https://www.superdatascience.com/podcast-unstoppable-data-science-reckless-commitment-artificial-intelligence/

2 Section: 2 - Fundamentals of Reinforcement Learning

2.1 Lecture 5 - Welcome to Part 0 - Fundamentals of Reinforcement Learning

- **Do I need prior knowledge of Python?** You can understand AI concepts without specific Python knowledge as it walks you through implementation such as applying it to the self driving car, doom etc. As you progress you will gain a better understanding of both. After completion of the course you will be able to apply it to real world problems especially for example knowing the theoretical and practical knowledge behind q learning and deep q learning. Also as an extra I have seen multiple students recently taking what they learned in the course and have applied it to their own projects.

3 Section: 3 Q Learning Intuition

3.1 Lecture 6 - Plan of Attack

- **I'm looking for some papers on zero shot and unseen event related to reinforcement learning. Do you know of any that might help?** Take a look at the following as they might help get you started regarding zero shot with reinforcement learning:`https://arxiv.org/pdf/1706.05064.pdf` `https://arxiv.org/abs/1707.08475`

3.2 Lecture 7 - What is reinforcement learning?

- **I'm not able to download the paper, what can I do?** If you visit the link and click on the PDF icon it should open the paper. If you still are having trouble accessing the resource please post a thread in the QA.

3.3 Lecture 8 - The Bellman Equation

- **Does the discount factor always start with the value of 0.9?** Overall the 0.9 in this lecture is just used as an example so we wouldn't know it would start with a 0.9. If you go through the next few lectures it should make more sense and be clearer but for this lecture think of it's use to convey the information and main points for using gamma. If you would like some additional information have a look at the following:

`https://stats.stackexchange.com/questions/221402/understanding-the-role-of-the-discount-factor-in-reinforcement-learning`.

- **Why do we have the value of 1 right next to the goal state?** If we have v=1 it is stating that it's the best possible move to lead us to get

the reward and reach the goal which is what we need the agent to do to complete it's task

3.4 Lecture 9 - The "Plan"

- **Can there be a tie path, or what information should we take into consideration when looking at a tie in the environment?** You can take into consideration that the agent looks for the shortest optimal path since it has a living penalty, but what you might see in the current lecture is that there are three states with the same value - 73. In the next few videos you will see how those values change, calculate the discounting factor (gamma) and more!

3.5 Lecture 10 - Markov Decision Process

- **Why is it Non-Deterministic Search?**With the non-deterministic search we are modeling a scenario that is basically out of control of the agent (the probabilities) that can't be predicted.

- **How did we arrive at calculating the probability of each state?** As an overview for the values in the states is that we are using the MDP to calculate the value of each state but in this scenario we know the optimal route or can calculate a path for the agent to take working back from the final reward of 1. Normally when the agent would go through this scenario it would run multiple iterations and learn while establishing the values (reinforcement learning).

3.6 Lecture 11 - Policy vs Plan

- **At around 9:24 in the lecture I was a little unclear on why the agent acted like that, could you elaborate?** The confusion might come from the agent accruing a living penalty as it looks to minimize it's moves and complete the maze in the smallest amount of iterations possible and our explanation which is focusing on the explanation of the movement through the maze. But with that being said, the agent can move backwards. It has the element and possibility of the 10% of randomness, which can have it move (but think about the living penalty again).

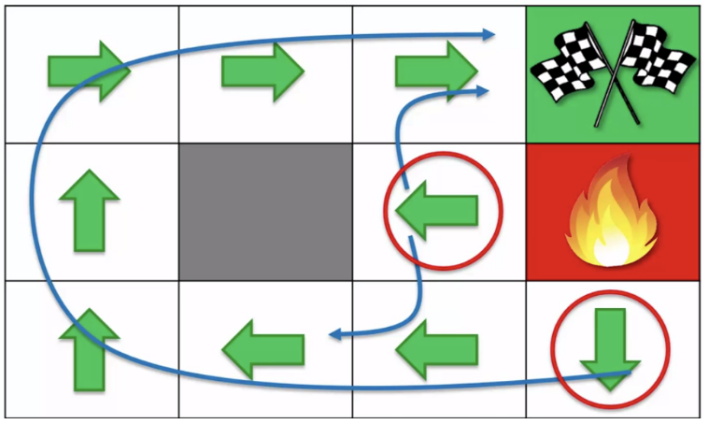
  You can see the following discussions for additional information:

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/3260850` and `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/2939478`

- **Can the agent iterate backwards?** The agent can go backwards especially if you take in the probability of randomness of the action (10%), but for the purpose of establishing the policy and for the tutorial the values are explained to show iterations through the map. In addition, the agent has

the living penalty so that it looks to establish the smallest of moves possible to minimize the penalty which may eliminate the movement backwards in most cases.

- **How did you determine the directions of the arrows?**



Overall, the MDP is a framework essentially that is used by the agent to decide what to do so when the agent is run (factoring in the bellman equation) but take a look around 9:50 since I think it will help give you the information that your looking for. Depending on the values that you get from factoring in the MDP will determine the direction of arrows. If you get the chance it would be highly recommended to go through section 3 and re-visiting the lectures to understand the arrows further. For example, visit lecture 9 on the markov decision process (especially around the 7:50 mark) because the information presented there I think will help clarify your questions. `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7228806?start=0`

3.7 Lecture 12 - Adding a "Living Penalty"

- **Is there a best R(s) value or choice, for example with the self driving car?** It depends on the environment. For the stated example with the self driving car using -0.5 would be way too much. It's a better option to use -0.1 to punish less the car when it is getting further away from the goal. If you punish it with -0.5 it will not want to take any road that does some detour before reaching the goal.

- **Is the process of calculating the state value the main process of training in reinforcement learning?** Yes it's part of reinforcement learning but overall we can think of the following breakdown: "the agent can visit a finite number of states and in visiting a state, a numerical reward will be collected, where negative numbers may represent punishments. Each state has a changeable value attached to it. From

every state there are subsequent states that can be reached by means of i. The value of a given state is defined by the averaged future reward which can be accumulated by selecting actions from this particular state. Actions are selected according to a policy which can also change. The goal of an RL algorithm is to select actions that maximize the expected cumulative reward (the return) of the agent." `http://www.scholarpedia.org/article/Reinforcement_learning`.

That being said yes it's part of the process of training in reinforcement learning.

3.8 Lecture 13 - Q-Learning Intuition

- **What is the relationship between value and reward? Why does the reward not take into account the probability of going into a different state like the expected value does?** The agent can see what the value of the action is and that depends on the rewards and the discounted factor along with all of the possible actions but it has to look at the other options as well, so on a simplistic level it's more to see that it has to look at the reward (so it learns) with the value and also the possible actions to decide which action to take overall. Reward is also more specific to that state to establish the choice or to help the agent learn.

3.9 Lecture 14 - Temporal Difference

- **What is Temporal Differnece learning?** "Temporal Difference is an approach to learning how to predict a quantity that depends on future values of a given signal. It can be used to learn both the V-function and the Q-function, whereas Q-learning is a specific TD algorithm used to learn the Q-function." `http://www.scholarpedia.org/article/Temporal_difference_learning`

3.10 Lecture 15 - Q-Learning Visualization

- **I'm running into a syntax error involving: fixedState = 3, 2147483648L, 507801126L, 683453281L, 310439348L, 2597246090L**. This is caused by running Python 3 but an easy fix, take a look towards the middle and bottom of the discussion at `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/259882` and also `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/2542374`. Using Python 3 a short fix would be to drop the L since all integers in Python 3 are considered Long.

- **I'm receiving the following error Method not implemented: getQ-Value at line 54, what can I do?** There were some modifications made to the file and this error can be resolved by using the code provided at: `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/`

2655974 If the issue persists or you need any assistance please post in the Q&A

4 Section 4 - Part 1 - Self-Driving Car (Deep Q-Learning)

4.1 Lecture 16 - Welcome to Part 1 - Self-Driving Car (Deep Q-Learning)

- **Are their any ideas or recommendations for extending the algorithms?** Sure! First, you can build a more advanced self driving car, with for examples multi destinations, multi options and new goals. Then, you can build new AIs for a lot of other games on Open AI Gym. There are plenty of challenging ones. Also you could make your own AIs for other purposes. The models are the same, you just have to link them to the environment of the problem. That is exactly learning by doing.

5 Section 5 - Deep Q-Learning Intuition

5.1 Lecture 17 - Plan of Attack

- **Where is Annex 1 located?** You can find it at: https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7165988?start=0 or starting at lecture 103.

5.2 Lecture 18 - Deep Q-Learning Intuition - Learning

- **Why is it referred to as Loss and not Cost Function? Is there a difference?** They can be basically seen as synonymous in RL or general representation referring to the same error margin, but for some further context the following discussion is useful

  https://stats.stackexchange.com/questions/179026/objective-function-cost-function-loss-function-are-they-the-same-thing.

- **Do we count the value of Q-target first using Q-learning formula and then use it as the Q-target for ANN training? Also after we used ANN, do we still use the Q-learning formula to update the Q-values?**

  Correct, you can feed the information from the states into the NN (depending on the hidden layers and setup) but we do have the q values and state of the maze fed into the neural network. Also to clarify if (using the maze) say we calculate the maze and q first to then feed it through the

ANN for further training? In deep q learning we have the NN predicting the values and then compare the calculation to the previous set and calculation (stored for the future).

5.3 Lecture 19 - Deep Q-Learning Intuition - Acting

- **I'm confused abput X1 and X2, how can it product Q-value result that has to do with the states environment?** We have our input state that enters the environment as a vector x1 and x2 that is used to run through the network and uses our q target values to update the weights (how our agent learns). For a better understanding of the map and how it's defined and as an example take a look at the commented version of the map.py file, I think it will help clarify your question for you. In addition on how it can produce q value results, take a look at lecture 17 around 11:00.

- **Where can I find Annex 2?** If you go to the course content area and scroll down to section 20 you should see Annex 2, starting at lecture 111. Also this is a link to the first video in Annex 2: `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7165990?start=0`.

5.4 Lecture 20 - Experience Replay

- **What will be the input for X1 and X2?** It can depend on the states, but we can see with experience replay that the states that its in don't get put into the network right away. They are saved into memory until it reaches a specific threshold then we learn (we have the batch of experiences) that it then randomly selects - takes the sample (state, action, state it moved into, reward) for the experience and then goes through the network to learn.

- **What does the batch include and can you elaborate on random pick?** For batch, each is containing a sample and your correct including the s, a, s', and r (see some more info here for general batch information:

  `https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network`

  As for a uniform pick it is more defined (instead of random) as you sample it from a set where drawing each element is equally probable.

5.5 Lecture 21 - Action Selection Policies

- **Why is it not recommended to choose policies completely randomly when training and then only pick the optimal one when applying?** Using policies completely random when training tends to not work well for complex problems especially where the search space is large.

In addition when using the MDP or the benefit of this method is that "When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment. Additionally, Q-learning can handle problems with stochastic transitions and rewards, without requiring any adaptations. It has been proven that for any finite MDP, Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable. "https://en.wikipedia.org/wiki/Q-learning"

6 Section 6 - Installation for Part 1

For the installation of Part 1 and Module 2 &3, please use any version of PyTorch <0.4.0. In addition, the development version of Kivy is highly recommended for installation. Please check the notes and upcoming lectures for further information and if you have any questions regarding the setup please post in the QA.

Currently, YML files are available for the first module on both Mac and Ubuntu, and also for Doom on Ubuntu. Please see the following lecture for setup information: `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/8824208?start=0`

6.1 Lecture 22 - Plan of Attack (Practical Tutorials)

- **For PyTorch documentation on Mac/Linux please see:** `https://pytorch.org/previous-versions/`
- **For PyTorch documentation and version information for Windows please see the following commands to install:** `https://github.com/peterjc123/pytorch-scripts`

6.2 Lecture 23 - Where to get the Materials

- **Please download the course materials here:** `https://www.superdatascience.com/artificial-intelligence/`

6.3 Lecture - 24. Windows Option 1: End-to-End installation steps

- **I'm running into the following error: TypeError: multinomial () missing 1 required positional arguments : "num_samples", what can I do?** To resolve this error please install any PyTorch version ¡ 0.4.0.

- **I'm on Windows and receiving a DLL load error - how can this be resolved?** The DLL load can be caused by an issue with installing PyTorch. Please check if you are using GPU enabled PyTorch that you have Nvidia drivers. To resolve it you can install the CPU version with:

  #for CPU only packages
  conda install -c peterjc123 pytorch-cpu

6.4 Lecture 25 - Windows Option 2 - Part A: Installing Ubuntu on Windows

- **I installed the VM but can't find the 64 bit option.** Please see the following for instructions on setting up the 64 bit option: `https://askubuntu.com/questions/675251/virtualbox-has-no-64-bit-options`

6.5 Lecture 26 - Windows Option 2 - Part B: Installing PyTorch and Kivy on your Ubuntu VM

- **I received the following error: multinomial() missing 1 required positional arguments: "num_samples".** Please uninstall and re-install PyTorch with any version <0.4.0.

6.6 Lecture 27 - Mac or Linux: Installing Anaconda

- **Can I work on the projects in PyCharm?** You can use PyCharm - it's an awesome IDE and whatever other development tool you are comfortable with, you can run the files from PyCharm but it might take a slight configuration if you have any issue for path settings. You can also always edit them in PyCharm and run from terminal as another option. If you have any questions please ask in the Q&A.

6.7 Lecture 28 - Mac or Linux: Installing PyTorch and Kivy

- **I'm having an issue on Mac installing Kivy, is there another method to install?** You can use Homebrew to install Kivy with the development version using the steps here: `https://kivy.org/doc/stable/installation/installation-osx.html#using-homebrew-with-pip`

- **I'm showing an error stating that the whl file is not a supported wheel on this platform.** Please check to make sure that the VM is 64 bit and also that the whl file is for using the Python version that you have installed. You can check by reading the whl file for example `http://download.pytorch.org/whl/cpu/torch-0.3.1-cp27-cp27m-linux_x86_64.whl` would be for Python 2.7

- **I ran into a permission error when running a command on Linux, is there a way to resolve it?** Normally permission errors are resolved by using sudo if you have administrator access to the machine, for example sudo pip install Cython==0.26.1.

6.8 Lecture 29 - Common Debug Tips

- **In this section we try to keep an updated downloadable PDF of common debug tips for the section. You can check the PDF for solutions or post in the Qfor help from an instructor.**

6.9 Lecture 30 - Getting Started

- **What is Kivy?** To quote the kivy documentation for a quick insight:

  "Kivy is focused. You can write a simple application with a few lines of code. Kivy programs are created using the Python programming language, which is incredibly versatile and powerful, yet easy to use. In addition, we created our own description language, the Kivy Language, for creating sophisticated user interfaces. This language allows you to set up, connect and arrange your application elements quickly. We feel that allowing you to focus on the essence of your application is more important than forcing you to fiddle with compiler settings. We took that burden off your shoulders." `https://kivy.org/docs/philosophy.html`

7 Section 7 - Creating the environment

7.1 Lecture 31 - Self Driving Car - Step 1

- **What is last_x and last_y?** last_x and last_y are used to keep the last point in memory when we draw the sand on the map

  last_x = 0
  last_y = 0
  n_points = 0 the total number of points in the last drawing
  length = 0 the length of the last drawing

- **How does action2rotation = [0, 20, -20] cause a rotation?** The actions are encoded in 3 numbers (0,1,2) so 0, 20, -20 are represented by the vectors and actions 0 1 and 2. If the action is selected let's say 0 then the value would be 0 so the car would go straight. Action 1 would make the car rotate 20 degrees to the right and index 2 would cause it to rotate to the left.

- **I see np used often, whats np?** np refers to Numpy and it's probably one of the most common and important packages you will see used for scientific computing: `http://www.numpy.org/` and it's a common statement to use when developing so that when you are writing your code you can simply refer to calling the library by using np. For example you may also see at times pd for Pandas or matplotlib as plt etc. When you import the library into the code you can import it strictly as numpy, or using the option to reference numpy each time with np by using:

  import numpy as np

7.2 Lecture 32 - Self Driving Car - Step 2

- **Can you elaborate on the density of sand calculation?** For the density of sand, we are taking big squares around each sensor (200x200) which we then divide each of the squares we divide the number of ones in the square by the total number of cells in the square 20 x 20 = 400 and that will give us our density which is done for each sensor that gives us our signals. What I recommend to help clarify further is to take a look at the map.py commented file and look at the notes around building the sensors (and the environment in general) and use it to compare to the lecture.

- **What is the graph that we see in the end?** It's displaying our learning and performance with time for our self driving car. The following discussions might help and are insightful to see how some students have experimented with the performance:

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/3144388`

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/2604982`

- **Is there a method to increase the window size for the self driving car?** You can edit the map.py file and use config to customize the window, for example:

  Config.set('graphics', 'width', '1200')
  Config.set('graphics', 'height', '700')

- **I'm running into the following error: Unable to get a Window, abort. SystemExit: 1. What can I do?** As a first step please make sure pygame is installed by running: pip install pygame. You can also try updating conda with: conda update –all

If this doesn't resolve the error please check the debugging PDF's or ask for assistance in the Q&A.

8 Section 8 - Building an AI

8.1 Lecture 33 - Self Driving Car - Step 3

- **I'm receiving the DLL error, how can I resolve it?** The DLL load can be caused by an issue with installing PyTorch. Please check if you are using GPU enabled PyTorch that you have Nvidia drivers. To resolve it you can install the CPU version with:

  #for CPU only packages
  conda install -c peterjc123 pytorch-cpu

- **What are Tensors in PyTorch?** Please see the following documentation from PyTorch for further context and useful examples:

  `https://pytorch.org/docs/stable/tensors.html`

8.2 Lecture 34 - Self Driving Car - Step 4

- **What is nn.module?** It's a reference from the neural network module from PyTorch: `http://pytorch.org/docs/master/nn.html` and we use it to inherit the tools from the module as we use class Network(nn.module).

- **I received the following error: AttributeError: 'module' object has no attribute 'module', how can I fix it?** Please check the syntax in the file if you are writing it as you progress with the lectures and compare it to the downloaded file.

8.3 Lecture 35 - Self Driving Car - Step 5

- **Could you elaborate on how you chose the number of neurons, layers etc?** Mainly it came down to Hadelin experimenting and pruning which will happen quite often with deep learning and AI (see `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7138864?start=0`) at 11:00+, to get the best results. You can also experiment by replacing it with other values as well.

- **Are the q_values being returned weighted activated neurons as a result of a few epochs of backpropagation?**The q_values are being returned as the output neurons of the neural network (q_values = self.fc2(x))

  You can also see that we are using them with the forward function that activates the neuron:

```
def forward(self, state):
x = F.relu(self.fc1(state))
q_values = self.fc2(x)
return q_values
```

8.4 Lecture 36 - Self Driving Car - Step 6

- **In this lecture, what is (object) when creating the ReplayMemory class?** When using ReplayMemoery class object is extending (such as a base class that you will see in many languages and has the property of object). There is a difference between class distinction comparing python 2.x and 3.x but with Python 3 it's deemed "new-style".

8.5 Lecture 37 - Self Driving Car - Step 7

- **Are there any recommended readings to help or provide some further information?** We have a list of recommended outside readings and resources for the course at: `https://www.superdatascience.com/artificial-intelligence/`

8.6 Lecture 38 - Self Driving Car - Step 8

- **Why do we need to use torch.cat?** We are converting x (the samples) once lambda is applied, but because each batch thats in the sample (a1, a2, a3, etc) we have to concatenate it to the first dimension corresponding to the state so that it is aligned. Yes, we are applying it to x and the 0 is the dimension that we want to apply with the concatenation. If you would like to see some additional examples take a look at the following: `http://pytorch.org/docs/0.3.1/torch.html?highlight=torch%20cat#torch.cat`

8.7 Lecture 39 - Self Driving Car - Step 9

- **What exactly is stochastic optimization?** Stochastic optimization mainly refers to a collection of methods for minimizing or maximizing an objective function when randomness is present

- **How did you calculate 100000 transitions?** 10000 transitions (events) was chosen since it will help optimize the training (setting it higher would have the training become much slower) and it's used to have memory to sample the memory to get the random transitions that the model will learn from. But take a look at the following lecture: `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7138944?start=0` around 3:00+

8.8 Lecture 40 - Self Driving Car - Step 10

- **Can you elaborate on the temperature parameter?** For the temperature parameter we are using it to basically control the "certainty" of the decisions by the AI. For example the closer to 0 the less certain and the higher you increase it the more certain that the AI's decisions will be. When we set to 0 it's actions as seen in the video are totally random (since you have deactivated the AI) with setting the temperature to 0. When we set it to 7 you can also see the actions are not as random and it is learning and reaching the goals with the positive rewards.

- **Can you elaborate on action.data[0,0]?** The actions are either 0, 1, or 2 so using action2rotation are converted (indexes) into rotations, 0,20,-20. If the action is 0 it corresponds to the first index 0. Examining the lecture around 14:00+ here: `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7138972?start=0` and also 9:00 here for the calculations: `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7138978?start=0`

8.9 Lecture 41 - Self Driving Car - Step 11

- **Why are we using unsqueeze1?** The batch state has a fake dimension and batch_action doesn't have it. We have to unsqueeze the actions so that the batch action has the same dimension as the batch state. We use 0 for the fake dimension of the state and 1 for the actions.

- **What does self.model(batch_state) do** self.model(batch_state) takes all the input states in your batch, and predicts the Q-values for all of them. The network object is calling the forward function to return the predictions.

n

8.10 Lecture 42 - Self Driving Car - Step 12

- **Why do we use batches?** Batches help with the NNs for efficiently and optimization as well, for example:

  `https://datascience.stackexchange.com/questions/16807/why-mini-batch-size-is-better-than-one-single-batch-with-all-training-data`.

  In addition, batches help so that parallel operations in a GPU can take place.

8.11 Lecture 43 - Self Driving Car - Step 13

- **How can I see the connections between the files?** One of the main or first methods to check on connections between files are the import statements. With the ai.py we built the brain for the self driving car and with the map.py we import:

  # Importing the Dqn object from our AI in ia.py

  from ai import Dqn

- **Why do we need the score method?** It's a great question. We are using score to compute the score of the sliding window of the rewards and we have:

  def score(self): return sum(self.reward$_window$)/($len(self.reward\_window)$+ 1.)

to compute the mean of the rewards. Taking it a part we have the following that will sum the elements in reward_window: (self.reward_window) and remember we have in the DQN class in the __init__:

self.reward_window = []

8.12 Leture 45 - Self Driving Car - Step 15

- **Why do we add state_dict() after self.model?** When we use self.model.state_dict() to be able to save the parameters of the model to the first key. State_dict in general with Pytorch is easy to use and comes in handy loading models(for some general information). With the documentation from PyTorch we can see the example:

  state_dict
  Returns a dictionary containing a whole state of the module.

  Both parameters and persistent buffers (e.g. running averages) are included. Keys are corresponding parameter and buffer names. When keep_vars is true, it returns a Variable for each parameter (rather than a Tensor).

  Example

```
module.state\_dict().keys()
['bias', 'weight']
```

  http://pytorch.org/docs/master/nn.html?highlight=state_dict#torch.nn.Module.state_dict

8.13 Lecture 46 - Self Driving Car - Step 16

- **Why do we save the optimizer?**Since we are using the keys and identifiers in the dictionary we have to give the value and one is for the optimizer because we want to save the parameters for it (the state_dict) as we are using it with the learning rate and for the weights to update correctly.

9 Section 9 - Playing with the AI

9.1 Lecture 47 - Self Driving Car - Level 1

- **What is the graph for?** The graph displayed is basically monitoring and displaying our learning and performance with time for our self driving car, but take a look at the following discussion for some additional details:

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/3434902`

9.2 Lecture 48 - Self Driving Car - Level 2

- **How can I make the car window bigger?** You can edit the file the kivy file. Inside you can replace or change:

```
<Car>:
    size: 20, 10
    canvas:
        PushMatrix
        Rotate:
            angle: self.angle
            origin: self.center
        Rectangle:
            pos: self.pos
            size: self.size
        PopMatrix
```

by

```
<Car>:
    size: 40, 20
    canvas:
        PushMatrix
        Rotate:
```

```
            angle: self.angle
            origin: self.center
        Rectangle:
            pos: self.pos
            size: self.size
        PopMatrix
which doubles the size of your car.
```

9.3 Lecture 49 - Self Driving Car - Level 3

- **Has there been any discussions or extensions to the code?** If you are looking for some methods on how to improve or build upon the self driving car see the following:

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/3897818`

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/3100850`

  `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/2773628`

9.4 Lecture 50 - Self Driving Car - Level 4

- **Challenge**

9.5 Lecture 51 - Challenge Solutions

- **Challenge Solutions**

10 Section 10 - Part 2 - Doom (Deep Convolutional Q-Learning)

10.1 Lecture 52 - 52. Welcome to Part 2 - Doom (Deep Convolutional Q-Learning)

- **Welcome to Part 2**

11 Section 11 - Deep Convolutional Q-Learning Intuition

11.1 Lecture 53 - Plan of Attack

11.2 Lecture 54 - Deep Convolutional Q-Learning Intuition

- **How many inputs can go into a neural network?**It would really depend on the type of neural network and architecture that is being used. But as a reference for choosing the number of nodes this discussion has some useful information regarding a feedforward neural network as an example:

  `https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw`

11.3 Lecture 55 - Eligibility Trace

- **Is there an additional reading available for this section?** Yes, `https://mitpress.mit.edu/books/reinforcement-learning`, if you scroll down on the left of the page you will find the option:

  This is an open access title. Available format: HTML `http://incompleteideas.net/book/the-book-1st.html` That will give you access to the book.

12 Section 12 - Installation for Part 2

12.1 Lecture 56 - Where to get the Materials

- **Please download the course materials here:** `https://www.superdatascience.com/artificial-intelligence/`

12.2 Lecture 57 - Installing Open AI Gym and ppaquette

12.3 Lecture 58 - Installing Open AI Gym Walk Through (Mac Version)

12.4 Lecture 59 - Installing Open AI Gym Walk Through (Ubuntu Version)

12.5 Lecture 60 - Common Debug Tips

13 Section 13 - Building an AI

13.1 Lecture 61 - Doom - Step 1

13.2 Lecture 62 - Doom - Step 2

13.3 Lecture 63 - Doom - Step 3

- **I received this error: ImportError: No module named registration, how can I resolve it?** Please check the debugging tips, YML files or setup instructions to check the versions that are installed. If you run pip list or conda list you can check the version for Gym and Doom.

13.4 Lecture 64 - Doom - Step 4

- **Is there any difference between Matlab and PyTorch?** Matlab is more intended for scientific computing and testing (more of a legacy platform) along with being a language itself whereas PyTorch is library geared toward more state of the art options for building AI related algorithms. See the following for some further information:

`https://www.quora.com/What-is-the-comparison-between-Matlab-and-Tensorflow-in-terms-of-custom-deep-CNN-imple`

13.5 Lecture 65 - Doom - Step 5

- **Why do we input 1 as the fake dimension?**Inside the rand function since we are adding the image to the nn and it only accepts batches we have to create and use the fake dimension, and we use 1 that corresponds to the batch.

- **Why pool the convolutions to count the number of neurons?**We need to pool to count the number of neurons in a large vector after our convolutions are applied. Moreover, max pooling is used especially regarding a cnn to first filter the image for the different widows than takes the most responsive (higher activation) unit as a transition to the next layer from that interest window of the image. In addition it's commonly used when applying Relu layers to apply the pooling layer "also referred

to as a downsampling layer. In this category, there are also several layer options, with maxpooling being the most popular

13.6 Lecture 66 - Doom - Step 6

13.7 Lecture 67 - Doom - Step 7

13.8 Lecture 68 - Doom - Step 8

- **Where can I find more information on multinomial in PyTorch?** You can see further information at the following link: `http://pytorch.org/docs/master/torch.html?highlight=multinomial#torch.multinomial`

13.9 Lecture 69 - Doom - Step 9

13.10 Lecture 70 - Doom - Step 10

13.11 Lecture 71 - Doom - Step 11

13.12 Lecture 72 - Doom - Step 12

13.13 Lecture 73 - Doom - Step 13

- **Could we use eligibility trace for the self driving car module to obtain better results?** Please see the following, a few students explored this concept with some great information, examples and discussion: `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/3868240`

13.14 Lecture 74 - Doom - Step 14

- **Is the size of each series in the batch=10 transitions and are we getting the input of 1st and last transition of each batch at each iteration?** Correct, we will be computing the reward on 10 steps (our series in the batch (all the series in the input batch of 10 transitions), getting the max q values for current state and current action. At each step we get the max q values during the iterations and when reaching the last state its = 0 so it won't be updated further.

13.15 Lecture 75 - Doom - Step 15

- **Could you explain a little more about the state and target meaning, is the state mean its current position or like going to left or right?** We are using a series to represent the steps/states, but yes we are adding the first state (the input state) and get the input state of the first step with state = series[0].state and then the target associated with the first transition with target (q value of the first step).

For some further info please see the following: `https://www.udemy.com/artificial-intelligence-az/learn/v4/questions/4530742`

13.16 Lecture 76 - Doom - Step 16

- **Why are we taking average of 100 steps for training?** For our 100 steps we are computing the moving average for the 100 steps in order to keep track of the average during training. With our 10 steps we will sum the rewards of the 10 steps to get the cumulative rewards over 10 steps which is eligibility trace. If you get the chance have a look at this lecture here since I think it will help clarify it further: `https://www.udemy.com/artificial-intelligence-az/learn/v4/t/lecture/7265104?start=0`

13.17 Lecture 77 - Doom - Step 17

- **How does the optimizer.step() get to know which loss error should be considered** By using the predictions and targets (we are running this in our batch) we get the loss error with loss_error = loss(predictions, targets) and then it allows us to proceed with the backprop using optimizer.zero_grad() and we apply the backward method to it. Once it's backpropogated we then update the weights using SGD using .step(). In addition for the CNN we make the connection by using the brain (cnn) with optim.Adam(cnn.parameters().

14 Section 14 - Playing with the AI

14.1 Lecture 78 - Watching our AI play Doom

15 Section 15 - ————– Part 3 - Breakout (A3C) ————-

15.1 Lecture 79 - Welcome to Part 3 - Breakout (A3C)

16 Section 16 - A3C Intuition

16.1 Lecture 80 - Plan of Attack

16.2 Lecture 81 - The three A's in A3C

16.3 Lecture 82 - Actor-Critic

- **What's the difference in values between Policy and Critic?** We use the neural net to predict the value of the state through the input and this part is called the critic. There are two outputs now (actor and critic) so the q values and the value of state which make up actor - critic.

16.4 Lecture 83 - Asynchronous

- **How is A3C beneficial compared to other methods?** What A3C introduces is that you can have 3 agents or 1+ agents attacking the same environment that are initialized (starting) from different points so that they go through it in different ways and explore so you then get triple (using 3 for example) the amount of experience. Really on a simple level you can visualize or think that you are introducing more agents to get more information, reducing the probability of agents getting stuck and the experiences are shared between each other.

16.5 Lecture 84 - Advantage

- **Why should the advantage have to be maximized?** Since we have the action to choose in the state and since the critic is shared it can tell us the value in the state (contributing to V), overall the critic knows the V so it can compare the Q to the known V value. We select the Q and then can compare it to the known value of the state (the advantage).

16.6 Lecture 85 - LSTM Layer

17 Section 17 - Installation for Part 3

17.1 Lecture 86 - Installing OpenCV

- **Is there any other option to install? I'm running into an error with the current command.** Yes, please try the following: pip install opencv-python

18 Section 18 - Building an AI

18.1 Lecture 87 - Breakout - Step 1

18.2 Lecture 88 - Breakout - Step 2

- **What is the variance of a tensor of weights?** Variance is basically adding specific (greater) weights to our tensors (with degrees of variance) which we need since we will be building two fully connected layers (one for the actor and one for the critic) and we will need to set the standard deviation for each, small for the actor and big for the critic.

18.3 Lecture 89 - Breakout - Step 3

- **What are fan_in and fan_out?** Fan in and fan out are working with establishing the dimensions to establish the weights, for example we can see:

```
fan_in = np.prod(weight_shape[1:4]) # dim1 * dim2 * dim3


fan_out = np.prod(weight_shape[2:4]) * weight_shape[0] # dim0 * dim2 * dim3
```

18.4 Lecture 90 - Breakout - Step 4

18.5 Lecture 91 - Breakout - Step 5

- **What is self.train used for?** Self.train is a method inherited from the nn module that allows us to activate dropout/batch normalization to put the module in train mode.

18.6 Lecture 92 - Breakout - Step 6

18.7 Lecture 93 - Breakout - Step 7

18.8 Lecture 94 - Breakout - Step 8

- **What does the seed stand for?** We are desynchronizing each training agent (we use rank) to shift the seeds, as seeds are fixed numbers (each seed determines a specific environment), but take a look at the following for some additional examples of the use of seed in a3c:

  `https://arxiv.org/pdf/1602.01783.pdf`

  `https://arxiv.org/pdf/1704.04651.pdf`

18.9 Lecture 95 - Breakout - Step 9

18.10 Lecture 96 - Breakout - Step 10

- **Why do we need to clamp the reward between -1 and 1?** Clamp is used as a precaution so that the reward remains between -1 and 1 to take the max/min values returned.

18.11 Lecture 97 - Breakout - Step 11

18.12 Lecture 98 - Breakout - Step 12

18.13 Lecture 99 - Breakout - Step 13

18.14 Lecture 100 - Breakout - Step 14

- **What does the term 'threads' refer to?** Threading refers to the CPU (or a single core in a multi-core processor) to execute multiple processes or "threads" concurrently, supported by the operating system

18.15 Lecture 101 - Breakout - Step 15

- **If you are having any issues running the algorithm please visit the FAQ or post in the QA.**

19 Section 19 - Annex 1: Artificial Neural Networks

19.1 Lecture 102 - What is Deep Learning?

- **Could you briefly state the difference between Deep Learning and Artificial Intelligence?** "Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned. ... Deep learning is a subfield of machine learning. While both fall under the broad category of artificial intelligence, deep learning is what powers the most human-like artificial intelligence" `https://www.zendesk.com/blog/machine-learning-and-deep-learning/`

19.2 Lecture 103 - Plan of Attack

19.3 Lecture 104 - The Neuron

- **Would you be able to provide some further details on what Linear Regression is?** Linear regression is an approach to modeling the relationship between a dependent variable and independent variable (a very basic definition for it). If you visit the following link for Scikit-learn it's a great library for prototyping and experimenting with some models such as LR. If you get the chance take a look: `http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html`

19.4 Lecture 105 - The Activation Function

19.5 Lecture 106 - How do Neural Networks work?

19.6 Lecture 107 - How do Neural Networks learn?

- **Why are the weights are same if every y has different value?** You have our weights being fed through the network (the feed forward neural net / perceptron) that are then run through the algorithm to have our output y value. I think the confusion that you might have is that once we have compared we feed the information back into the neural net to the weights and then the weights get updated. Take a look at about 4:50+ if you get the chance as it will reference this and might help clarify it further.

19.7 Lecture 108 - Gradient Descent

- **What is meant by 1000 combinations?** The reference was related to using the 'brute force option' of trying 1000 different inputs for weights. This won't work because when you have more weights and increase the synapses it will lead to the curse of dimensionality.

- **How was the curve in the gradient descent explanation obtained?** It's based off the cross function since we are differentiating the slope at the specific point (finding out if it's positive or negative) to the right is downhill rolling the ball down.

19.8 Lecture 109 - Stochastic Gradient Descent

- **How does batch gradient descent get stuck in a local minimum. Is it because if it first starts out at a certain spot, it is more likely to go to the minimum that is next to it rather than the global minimum?** The following discussion from

  `https://stats.stackexchange.com/questions/90874/how-can-stochastic-gradient-descent-avoid-the-problem-of-a-local-minimum`

  does a great job of examining this, for example: ""In stochastic gradient descent the parameters are estimated for every observation, as opposed the whole sample in regular gradient descent (batch gradient descent). This is what gives it a lot of randomness. The path of stochastic gradient descent wanders over more places, and thus is more likely to "jump out" of a local minimum, and find a global minimum (Note*). However, stochastic gradient descent can still get stuck in local minimum."

19.9 Lecture 110 - Backpropagation

- **In Step 5, should the method only be used when we perform reinforcement learning and not the others Learning method like unsupervised learning and supervised learning?** It's the method or operation that we use in reinforcement learning as update the weights after each observation. You can see that it also is discussing updating the weights after a batch of observations for batch learning but the updating of the weights after each batch is to be used for reinforcement learning. If you were using other models you would be changing the method for the update.

20 Section 20 - Annex 2: Convolutional Neural Networks

20.1 Lecture 111 - Plan of Attack

20.2 Lecture 112 - What are convolutional neural networks?

20.3 Lecture 113 - Step 1 - Convolution Operation

20.4 Lecture 114 - Step 1(b) - ReLU Layer

20.5 Lecture 115 - Step 2 - Pooling

20.6 Lecture 116 - Step 3 - Flattening

20.7 Lecture 117 - Step 4 - Full Connection

- **The same values are passed to both the cat and the dog. Is the determination of the output is based upon the weights as well?** Correct, we do have the two outputs for the dog and cat based on the prediction that is from the artificial neural network and then with back propagation it is optimized and trains on the data. If you have any additional questions please let me know!

20.8 Lecture 118 - Summary

20.9 Lecture 119 - Softmax & Cross-Entropy

21 Conclusion

Please check back often as we will continuously update the FAQ document, and if you have any questions please feel free to post in the course Q&A.