

Bibliography

1. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning. MIT Press (1998)
2. Bellman, R.: A Markovian decision process. *Indiana Univ. Math. J.* **6**(4), 679–684 (1957)
3. Markov Decision Process, wikipedia.org. https://en.wikipedia.org/wiki/Markov_decision_process. Accessed Aug 2018
4. Markov Chain, wikipedia.org. https://en.wikipedia.org/wiki/Markov_chain. Accessed Aug 2018
5. Markov Property, wikipedia.org. https://en.wikipedia.org/wiki/Markov_property. Accessed Aug 2018
6. Solving MDPs with dynamic programming, towardsdatascience.com. <https://towardsdatascience.com/reinforcement-learning-demystified-solving-mdps-with-dynamic-programming-b52c8093c919>. Accessed Aug 2018
7. Dynamic Programming in Python for reinforcement learning, medium.com. <https://medium.com/harder-choices/dynamic-programming-in-python-reinforcement-learning-bb288d95288f>. Accessed Aug 2018
8. Understanding RL—The Bellman Equation, joshgreaves.com. <https://joshgreaves.com/reinforcement-learning/understanding-rl-the-bellman-equations/>. Accessed Aug 2018
9. Reinforcement Learning in Finance, Coursera. <https://coursera.org/lecture/reinforcement-learning-in-finance>. Accessed Aug 2018
10. Deep Reinforcement Learning Demystified, medium.com. <https://medium.com/@m.alzantot/deep-reinforcement-learning-demystified-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa>. Accessed Aug 2018
11. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016). <https://arxiv.org/abs/1606.01540>
12. Docs—Open AI Gym. <https://gym.openai.com/docs/>. Accessed Aug 2018
13. Keras-RL, GitHub Repository Keras-RL. <https://github.com/keras-rl>. Accessed Aug 2018
14. Shani, G.: A Survey of Model-Based and Model-Free Methods for Resolving Perceptual Aliasing. Ben-Gurion University (2004)
15. Temporal Difference Learning, Wikipedia. https://en.wikipedia.org/wiki/Temporal_difference_learning. Accessed Aug 2018
16. Sutton, R.: Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**(1), 9–44 (1988)
17. Eligibility Traces, incompleteideas.net. <http://www.incompleteideas.net/book/ebook/node72.html>. Accessed Aug 2018
18. SARSA, Wikipedia. <https://en.wikipedia.org/wiki/State%E2%80%93action%E2%80%93reward%E2%80%93state%E2%80%93action>. Accessed Aug 2018

19. Diving deep into reinforcement learning, freecodecamp.org. <https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe>. Accessed Aug 2018
20. Tokic, M.: Adaptive epsilon-greedy exploration in reinforcement learning based on value differences. In: KI 2010: Advances in Artificial Intelligence, pp. 203–210. Springer, Berlin (2010)
21. Imaddabburra, Bandit Algorithms, github.io. <https://imaddabburra.github.io/blog/data%20science/2018/03/31/epsilon-Greedy-Algorithm.html>. Accessed Aug 2018
22. Sewak, M., Rezaul Karim, Md., Pujari, P.: Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python. Packt Publishing (2018). ISBN: 1788392302, 9781788392303.
23. Sewak, M., Sahay, S.K., Rathore, H.: An overview of deep learning architecture of deep neural networks and autoencoders. In: International Conference on Intelligent Computing, 25–27 Oct 2018. Amrita University, Proceedings in Journal of Computational and Theoretical Nanoscience (2018)
24. Sewak, M., Sahay, S.K., Rathore, H.: Comparison of deep learning and the classical machine learning algorithm for the malware detection. In: 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPDC (2018)
25. Sewak, M., Sahay, S.K., Rathore, H.: An investigation of a deep learning-based malware detection system. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, pp. 26:1–26:5 (2018)
26. “OpenAI Universe”, GitHub Repository. <https://github.com/openai/universe>
27. “OpenAI Retro”, GitHub Repository. <https://github.com/openai/retro>
28. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016)
29. “DeepMind Lab”. <https://deepmind.com/blog/open-sourcing-deepmind-lab/>
30. “DeepMind Control Suite”, GitHub Repository. https://github.com/deepmind/dm_control
31. Johnson, M., Hofmann, K., Hutton, T., Bignell, D.: The Malmo platform for artificial intelligence experimentation. In: Kambhampati, S. (ed.) Proceedings 25th International Joint Conference on Artificial Intelligence, pp. 42–46. AAAI Press, Palo Alto, California USA (2016). <https://github.com/Microsoft/malmo>
32. Duan, Y., Chen, X., Houthoofd, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33rd International Conference on Machine Learning (ICML) (2016)
33. “DeepMind’s TRFL”, GitHub Repository. <https://github.com/deepmind/trfl>
34. “OpenAI Baselines”, GitHub Repository. <https://github.com/openai/baselines>
35. Plappert, M.: “Keras-RL”, GitHub Repository (2016). <https://keras-rl>. <https://github.com/keras-rl/keras-rl>
36. Caspi, I., Leibovich, G., Novik, G., Endrawis, S.: Reinforcement learning coach (2017). <https://doi.org/10.5281/zenodo.1134899>
37. Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J.E., Jordan, M.I., Stoica, I.: RLlib: abstractions for distributed reinforcement learning. In: International Conference on Machine Learning (ICML) (2018)
38. DQN, deepmind.com. <https://deepmind.com/research/dqn/>. Accessed Aug 2018
39. AlphaGo, deepmind.com. <https://deepmind.com/research/alphago/>. Accessed Aug 2018
40. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing Atari with Deep Reinforcement Learning (2013). <https://arxiv.org/abs/1312.5602>
41. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* **518** (2015). [10.1038/nature14236](https://doi.org/10.1038/nature14236)

42. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized Experience Replay (2015). <https://arxiv.org/abs/1511.05952>
43. Lin, L.-J.: Reinforcement learning for robots using neural networks. Carnegie Mellon University, Ph.D. Thesis Lin:1992: RLR:168871 (1992)
44. Seno, T.: Welcome to deep reinforcement learning, towardsdatascience.com. <https://towardsdatascience.com/welcome-to-deep-reinforcement-learning-part-1-dqn-c3cab4d41b6b>. Accessed Aug 2018
45. Juliani, A.: Simple reinforcement learning with tensorflow, medium.com. <https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e2b8df>. Accessed Aug 2018
46. van Hasselt, H., Guez, A., Silver, D.: Deep Reinforcement Learning with Double Q-Learning (2015). <https://arxiv.org/abs/1509.06461>
47. Wang, Z., de Freitas, N., Lanctot, M.: Dueling Network Architectures for Deep Reinforcement Learning (2015). <https://arxiv.org/abs/1511.06581>
48. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the 12th International Conference on Neural Information Processing Systems, NIPS'99, pp. 1057–1063 (1999)
49. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: Proceedings of the 31st International Conference on Machine Learning, pp. 387–395 (2014)
50. Silver, D.: Lecture 7—Policy Gradient, University College London. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/pg.pdf. Accessed Aug 2018
51. Li, F.-F., Johnson, J., Yeung, S.: Lecture 14, CS231—Stanford. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture14.pdf. Accessed Aug 2018
52. Williams, R.J.: A class of gradient-estimating algorithms for reinforcement learning in neural networks. In: Proceedings of the IEEE First International Conference on Neural Networks, San Diego, CA (1987)
53. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**(3), 229–256 (1992)
54. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning (2016). <https://arxiv.org/abs/1602.01783>
55. Degris, T., Pilarski, P.M., Sutton, R.S.: Model-free reinforcement learning with continuous action in practice. In: 2012 American Control Conference (ACC), pp. 2177–2182 (2012)
56. Bhatnagar, S., Sutton, R.S., Ghavamzadeh, M., Lee, M.: Natural actor-critic algorithms. *Automatica* **45**(11), 2471–2482 (2009)
57. Sutton, R.S., Barto, A.G.: Reinforcement learning—an introduction. In: Adaptive Computation and Machine Learning. MIT Press (1998)
58. Wu, Y., Mansimov, E., Liao, S., Radford, A., Schulman, J.: Openai baselines: Acktr a2c (2017). <https://blog.openai.com/baselines-acktr-a2c>
59. Konda, V.R., Tsitsiklis, J.N.: On actor-critic algorithms. *SIAMJ. Control Optim.* **42**(4), 1143–1166 (2003)
60. Abadi, M., et al.: Model Sub-classing, TensorFlow Guide: High Level API—Keras (2019). https://www.tensorflow.org/guide/keras#model_subclassing
61. Abadi, M., et al.: Functional API, TensorFlow Guide: High Level API—Keras (2019). https://www.tensorflow.org/guide/keras#model_subclassing
62. Abadi, M., et al.: Gradient Tapes, TensorFlow Tutorial: Automatic Differentiation and Gradient Tapes (2019). https://www.tensorflow.org/tutorials/eager/automatic_differentiation
63. Abadi, M., et al.: apply_gradient(), TensorFlow API Docs: tf.train.Optimizer Class (2019). https://www.tensorflow.org/api_docs/python/tf/train/Optimizer

64. Yuan, R.: Deep reinforcement learning: playing CartPole through asynchronous advantage actor critic (A3C) with tf.keras and eager execution, Medium.com (2018). <https://medium.com/tensorflow/deep-reinforcement-learning-playing-cartpole-through-asynchronous-advantage-actor-critic-a3c-7eab2eea5296>
65. Daoust, M.: A3C Blog Post, GitHub repository: TensorFlow/Models/Research (2018). https://github.com/tensorflow/models/tree/master/research/a3c_blogpost
66. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 32, pp. 387–395, Beijing, China, 22–24 Jun 2014
67. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015). <https://arxiv.org/abs/1509.02971>
68. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). <https://arxiv.org/abs/1502.03167>
69. Weng, L.: Policy Gradient Algorithms (2018). <https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#off-policy-policy-gradient>. Accessed Jan 2019

Index

A

Accumulating the rewards
value, reward, discounting factor, 21
Act Humanly. *See* Intelligent-behavior
Action adaptive epsilon algorithms
epsilon, 62
Action–value
Bellman equation, 21
Q-function, 16, 18
Activation
deep learning, 76, 80
Actor-critic methods
actor-critic, 141
Actor-critic model class
actor-critic, code, 153
Actor-critic models
DPG, 173
Adaptive Moment Estimation (ADAM)
optimizers, 83
Additional Target Q-Network
Double DQN. *See* Deep Q Network
Advantage
actor-critic, 148
REINFORCE algorithm, 138
Advantage network
Dueling DQN, 106
Agent, 14
 α
learning rate, 57
AlphaGo
DeepMind, game, DQN. *See* Google
DeepMind
Anaconda
platform requirements, 65
Annealing epsilon (ϵ)

epsilon. *See* Time Adaptive “epsilon”
Architecture
actor-critic, 144
Argmax, 26
Artificial Intelligence, 1
Artificial Intelligence agents
Artificial Intelligence systems, 1
Artificial Intelligence systems
Artificial Intelligence, 1
Artificial Neural Network (ANN)
artificial neurons, deep learning, 77
Artificial neurons
deep learning, 75
Asynchronous Advantage Actor-Critic
Implementation (A3C)
actor-critic, 149
Asynchronous mode
value iteration, 26
Atari
Gym, 97
Atari2600, 152
Atari2600 games, 91
Attribution of rewards
reward, 4

B

Backgammon
game, environment, 96
Backpropagation
deep learning, training, 78
Backward view
TD (λ), 57
Balance a pole on a cart
game. *See* CartPole
Bandit Algorithm

- Bandit Algorithm (*cont.*)
 - epsilon, 62
 - Q-Learning, epsilon, 59, 60
- Baseline
 - OpenAI, 93
 - REINFORCE algorithm, 138
- Baseline agents
 - OpenAI, 90
- Batch normalization
 - Deep Deterministic Policy Gradient, 181
- Behavior policy
 - epsilon, 65
- Behavior policy class
 - Double DQN, Code, 119
- Bellman equation, 19, 21–25, 27
- Bellman equation for value-function
 - Bellman equation, value function, 55
- Box2D
 - OpenAI Gym, 91
- Building a custom environment class
 - environment, 31
- BURLAP, 93
 - reinforcement learning wrapper libraries, 93
- C**
- CartPole
 - environment, 96
 - game, environment, 10
- CartPole balancing problem
 - CartPole, 11
- CartPole-v0
 - environment, 153
- CartPole-v1
 - environment, 109
- Chain rule of derivatives
 - optimization, 78
- Challenges
 - classical DP, 51
- Classical Dynamic Programming (Classical DP)
 - dynamic programming, 51, 52
 - classical DP, 51–54
- Classical Reinforcement Learning (Classical RL)
 - classical RL, 52, 53
 - reinforcement learning, RL. *See* Classical DP
- Classic Control
 - OpenAI Gym, 91
- Clipping rewards and penalties
 - Deep Q Network, 103
- Coach
 - Nervana Systems, 94
 - reinforcement learning wrapper libraries, 94
- Code
 - Q-Learning, 65
- Color-channels
 - Convolutional Neural Network, 84
- Combining the solutions to these two subproblems
 - dynamic programming, 24
- Conceptual design
 - actor-critic, 143
- Conditional probability distribution, 19
- Constructing the environment
 - grid-world, 31
- Continuous-action agent
 - agent, 11
- Continuous action space, 130
- Continuous-tasks, 54
- Control problems of reinforcement learning, 54
- Convergence-assurance, 130
- Convolutional layer
 - Convolutional Neural Network, 85
- Convolutional-map
 - Convolutional Neural Network, 86
- Convolutional Neural Networks (CNN)
 - Convolutional Neural Network, 75, 84–88
 - deep learning, 84
- Counter-Strike
 - video game, 96
- Covariate shifts
 - batch normalization. *See* ELU
- CPU threads, 154
- Crossentropy loss
 - loss functions, SoftMax, 82
- Custom environment class
 - environment. *See* Gym
- Custom exceptions classes
 - code, 125
- D**
- Decreasing epsilon
 - epsilon, 61
- Deep Deterministic Policy Gradient
 - deterministic policy gradient, 178
- Deep learning for vision
 - Convolutional Neural Network, 84
- DeepMind Control Suite
 - DeepMind, 92
 - reinforcement learning wrapper libraries, 92
- DeepMind Lab
 - DeepMind, 92
- Deep Neural Networks (DNN)
 - deep learning, 77
- Deep Q Networks, 97
- Deep Reinforcement Agents. *See* Artificial Intelligence agent

- Deployment. *See* Production
 - Deque
 - experience replay, 155
 - Deterministic policy gradient
 - policy gradient, 173
 - Different types of Reward
 - reward, 6
 - Discounted rewards
 - reward, 21–23
 - Done (boolean)
 - custom environment class, 34
 - Double DQN. *See* DDQN
 - code, 111
 - DQN algorithm
 - Deep Q Network, 98
 - Dueling DQN, 105
 - Dynamic programming. *See* Bellman Equation; Value Iteration, Policy Iteration
- E**
- Eager execution
 - TensorFlow, 154
 - Eligibility Traces
 - TD (λ), 56, 57
 - Exponential Linear Unit (ELU)
 - activation, 82
 - Environment, 1, 2, 7, 8, 10, 14, 17
 - Envs
 - environment. *See* Custom Environment Class
 - Epsilon (ϵ)
 - epsilon. *See* Exploration policy, 17
 - Epsilon (ϵ)-greedy
 - epsilon. *See* Epsilon-greedy
 - Epsilon first
 - epsilon, 61
 - Epsilon-greedy algorithm
 - epsilon, behavior policy, 65
 - Epsilon soft
 - action adaptive epsilon algorithms, 62
 - Equal attribution. *See* Attribution of rewards
 - Essential recipes
 - environment. *See* Gym
 - Estimating the action–value
 - Bellman equation, 23
 - Estimating the value function
 - Bellman equation, value, 22
 - Estimation sub-problem, 54
 - Example of state formulation
 - state, 13
 - Experience replay memory class
 - Double DQN, Code, 123
 - Experience replays
 - Deep Deterministic Policy Gradient, 180
 - Deep Q Network, 100
 - Experience trail
 - Deep Q Network, experience replay, 100
 - Experience-tuples
 - experience replay, 100
 - Exploit
 - policy, 16, 17
 - Exploration
 - policy, 16, 17
 - Explore
 - policy, 16, 17
- F**
- Feasibility of application of dynamic programming
 - dynamic programming, 24
 - FC networks. *See* Fully Connected Layers
 - Feed-Forward
 - deep learning, 77
 - Feed-Forward Deep Neural Network
 - deep learning, 79
 - Feed-Forward mechanism
 - deep learning, 79
 - Flattened layer
 - Convolutional Neural Network, 86
 - Formulation of
 - state, 8, 9, 14
 - Fortnite
 - video game, 96
 - Forward view
 - TD (λ), 57
 - Fully connected layer
 - Convolutional Neural Network, 85, 86
 - Functional API
 - TensorFlow, 154
 - Function-approximators
 - model, 52
 - Future rewards
 - reward, 3
- G**
- Gamma
 - discounting factor, 21, 22, 25
 - Garage, 92
 - reinforcement learning wrapper libraries, 92
 - General Artificial Intelligence
 - Artificial Intelligence, 95
 - Geoffrey Hinton
 - deep learning, 79
 - GitHub, 153
 - Global Interpreter Lock (GIL)

Python. *See* Global Interpreter Lock
 Graphical games
 game. *See* CNN
 Grid-World
 environment, 29, 30, 36, 41, 44
 Gym
 environment, 29, 31, 32
 OpenAI, 91

H

Half-Life
 video game, 96
 High variance
 REINFORCE algorithm, 135

I

Identity
 activation, 81
 Info (dict)
 custom environment class, 34
 Intelligent-behavior, 1
 Image frames
 vision, 13
 Inheriting an environment class
 environment, 31
 Intractability, 174
 Introduction to deep learning
 deep learning, DL. *See* DRL

J

Java
 programming language, 93

K

Keras
 platform dependencies, 109
 Keras-RL, 93
 platform dependencies, 185
 reinforcement learning wrapper libraries, 93
 Kernel filters
 Convolutional Neural Network, 85

L

Large action space, 130
 Linear
 activation, 81
 L1 loss
 loss functions, 82
 L2 loss
 loss functions, 82
 Logit layer
 actor network, 154
 Loss functions
 deep learning, 82

M

Malmo, 92
 Mario
 game, 11, 12
 Markov Chain
 MDP, 20, 21
 Markov Decision Process (MDP), 19–22, 25, 27
 MDP, 19, 20, 25, 27
 Markov Property
 MDP, 19, 20, 22
 Mathematical objective, 21
 MATLAB
 programming language, 93
 MDP model
 MDP, model, 52
 Mean shift
 ReLU, 82
 Mean Square Error
 loss functions, 82
 Memory buffer
 Deep Deterministic Policy Gradient, 180
 Memory class
 code, 155
 Miniconda
 platform requirements, 65
 MLP-DNN
 deep learning, 75, 77, 84, 86, 88
 Model
 MDP, 51
 Model-based approach
 MDP, 52, 53
 Model-free approaches
 MDP, 52
 Monte Carlo Policy Gradient
 REINFORCE algorithm, 134
 stochastic policy gradient, 134
 Monte Carlo simulation
 MDP, 53, 56
 Mountain Car Continuous
 environment, 186
 MountainCarContinuous-v0
 environment, 191
 MSE
 loss functions, 82
 MuJoCo
 OpenAI Gym, 91
 Multilayer Perceptron (MLP)
 deep learning, DNN, artificial neurons. *See*
 MLP-DNN

N

Negative reward
 reward. *See* Penalty

- Nervana Systems, [94](#)
- Neural-networks
 - function-approximators, ANN, DNN, CNN, MLP, [51](#)
- Non-episodic tasks, [54](#)
- NumPy
 - platform dependencies, [185](#)
- O**
- Observation. *See* State
- Observation (object)
 - custom environment class, [33](#)
- Observation, reward, done, info
 - step () method. *See* Custom Environment Class
- Off-policy
 - policy, agent, [16](#)
- On-policy, [1](#), [16](#), [17](#), [173](#)
- OpenAI. *See* Gym
- OpenAI baselines
 - reinforcement learning wrapper libraries, [93](#)
- OpenAI Gym
 - OpenAI, [91](#)
- OpenAI Universe
 - OpenAI, [91](#)
- Optimal Substructure
 - dynamic programming, [24](#), [25](#)
- Optimistic-initial-condition
 - State-Action-Reward-State-Action (SARSA), [58](#)
- Optimizers
 - deep learning, [83](#)
- Overlapping Subproblems
 - dynamic programming, [24](#)
- P**
- $P_a(s, s')$
 - state transition probability, [20](#), [22](#)
- Parametrized Leaky ReLU
 - activation, [81](#)
- Penalty
 - reward. *See* Negative reward
- Perfectly “model” the environment
 - MDP, [52](#)
- Performance Function (J), [174](#)
- Permissible state transitions
 - grid-world. *See* State Transition Probability
- Performance of policy
 - performance function, [132](#)
- π
 - policy, [16](#), [20](#), [21](#)
- Pip
 - platform requirements, [65](#)
- Pointwise multiplication of the two functions
 - Convolutional Neural Network, [85](#)
- Policy, [1](#), [16](#), [17](#)
- Policy approximation, [127](#)
 - policy approximation, [128](#)
- Policy-based approaches, [173](#)
- Policy Evaluation
 - dynamic programming, [27](#)
- Policy gradient-based approaches
 - policy approximation, [128](#)
- Policy-Iteration, [52](#)
 - dynamic programming, [19](#), [25](#), [27](#)
 - grid-world, [36](#)
- Pooling layer
 - Convolutional Neural Network, [85](#), [86](#)
- Pooling technique
 - pooling layer, [86](#)
- Pre-requisites
 - dynamic programming, [25](#)
- Prioritized experience replay
 - Deep Q Network, experience replay, [101](#)
- Probability of reaching different states
 - state transition probability, [23](#)
- Probable next state
 - state, [19](#)
- Problems with calculating the policy gradient
 - stochastic policy gradient, [132](#)
- Production. *See* Deployment
- Profitable state
 - value, state, reward, [15](#)
- Pseudo-code
 - Deep Deterministic Policy Gradient, [182](#)
 - REINFORCE algorithm, [137](#)
- PUBG
 - video game, [96](#)
- PyCharm IDE
 - platform dependencies, [109](#)
- Python 3.6.5 runtime
 - platform requirements, [65](#)
- Python 3.x
 - platform requirements, [34](#)
- Q**
- Q-function. *See* Action-Value Function
 - Bellman equation, [21](#)
 - Q-Learning, [58](#)
 - Q. *See* Action-Value SARSA, [57](#)
- Q-Learning, [51](#), [58](#), [59](#), [62](#), [63](#), [65](#)
 - model-free approach, [53](#)
- Q Table
 - Q-Learning, [65](#)

R

- $R_a(s, s')$**
 - reward function. *See* Reward
- Random action
 - explore, 60
- Rational
 - act ‘Humanly’. *See* Intelligent-behavior
- Randomized Leaky ReLU
 - activation, 81
- Readings from all the sensors
 - state, 8
- Rectifier linear unit
 - activation, 81
- REINFORCE
 - stochastic policy gradient, 133
- REINFORCE algorithm
 - stochastic policy gradient, 133
- Reinforcement Learning. *See* Artificial Intelligence
- Reinforcement learning agent
 - reinforcement learning, agent, 8, 11, 12
 - Artificial Intelligence agents, 1
- REINFORCE with baseline, 138
- ReLU
 - activation, 81
- Reset
 - environment. *See* Gym
- Reset () method
 - custom environment class, 34
- Retro
 - OpenAI, 91
- Reward, 1–9, 12, 17
- Reward (float)
 - custom environment class, 34
- Reward function
 - reward, 3
- RGB intensity
 - Convolutional Neural Network, 84
- Richard Bellman
 - Bellman Equation, 21
- rl4j, 93
 - reinforcement learning wrapper libraries, 93
- RLlab. *See* Garage
 - reinforcement learning wrapper libraries, 92
- RLlib
 - reinforcement learning wrapper libraries, 94
- RMSProp
 - optimizers, deep learning, 83
- Ronald J. Williams
 - REINFORCE algorithm, 133
- S**
- Secondary reward
 - reward, 12
- Sequence of convolutional maps
 - CNN, 14
- Shared network architecture, 154
- Sigmoid
 - activation, 79, 81
- Soft updates
 - Deep Deterministic Policy Gradient, 180
- Stochastic model
 - stochastic. *See* Probability Distribution
- Skipping frames
 - Deep Q Network, 102
- SoftMax
 - activation, 81
- Solve an MDP problem
 - MDP, 29
- Standardized training environments, 91
- State, 7, 11, 15
- State-Action-Reward-State-Action (SARSA), 51, 53, 54, 57–59, 63
 - ‘model-free’ approach. *See* Classical RL
 - SARSA, 57
- State transition probability, 20, 22
- Step
 - environment. *See* Gym
- Step () method
 - environment, 33
- Stochastic events
 - stochastic, event, 20
- Stochastic policy
 - policy, 132
- Stride
 - pooling layer, 86
- Structure for the code
 - grid-world, 34
- Sub-Classing feature
 - TensorFlow, 153
- SVM
 - function-approximators, 52
- Synchronous Advantage Actor-Critic (A2C)
 - actor-critic, 150
- Synchronous mode
 - value iteration. *See* Dynamic Programming
- T**
- Tanh
 - activation, 79
- TD target
 - Temporal Difference Learning, 55, 56
- ToyText
 - OpenAI Gym, 92
- Tau (τ)
 - trajectory, 132
- TD (0)

- Temporal Difference Learning. *See* TD Lambda
 - TD (λ)
 - Temporal Difference. *See* TD (0)
 - Temporal Difference Learning, [54](#), [56](#)
 - Temporal Difference
 - model-based approach, [53](#)
 - Temporal Difference (TD) Learning
 - classical RL, [51](#), [53](#), [55](#), [56](#), [58](#)
 - Temporal Difference Model (TDM)
 - Temporal Difference Learning. *See* Classical RL
 - TensorFlow
 - platform dependencies, [153](#)
 - Terminal state
 - state, [29](#), [34](#)
 - Test. *See* Deployment
 - Test our agents
 - deployment, [31](#), [32](#)
 - Theorem
 - deterministic policy gradient, [176](#)
 - Thought-process. *See* Intelligent-behavior
 - Tic-tac-toe
 - game, [8](#), [9](#), [11](#)
 - Time adaptive “epsilon”
 - epsilon. *See* Epsilon-greedy
 - Training environment. *See* Deployment
 - Trajectory, [132](#)
 - Transformations
 - pre-processing, [6](#), [14](#)
 - TRFL
 - DeepMind, [93](#)
 - reinforcement learning wrapper libraries, [93](#)
- U**
- Uncertain rewards
 - reward, [4](#)
 - Utility
 - value, [15](#)
- Unidentifiability**
- Dueling DQN, [107](#)
- V**
- Validation. *See* Deployment
 - Value, [15–18](#)
 - Value adaptive epsilon algorithms
 - epsilon, [62](#)
 - Value Difference Based Exploration
 - value adaptive epsilon algorithms. *See* VDBE
 - Value function . *See* Reward
 - Value iteration
 - dynamic programming, [19](#), [25](#), [26](#)
 - grid-world, [35](#)
 - Vanishing gradient
 - deep learning, [77](#), [79](#)
 - VDBE
 - value adaptive epsilon algorithms, [62](#)
 - Vision challenge
 - vision. *See* CNN
- $V_{(s)}$
- value function, [15](#)
- W**
- Wrapper libraries
 - reinforcement learning wrapper libraries, [185](#)
- Y**
- \hat{y} , [77](#)
- Z**
- Z
 - artificial neurons, activation, [76](#)