

# Reinforcement Learning: An Introduction

Richard S. Sutton and Andrew G. Barto



The MIT Press

*From The MIT Press*



**MITCogNet**

© 1998 Richard S. Sutton and Andrew G. Barto

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This book was set in Times Roman by Windfall Software using Z<sub>z</sub>T<sub>E</sub>X and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Sutton, Richard S.

Reinforcement learning: an introduction / Richard S. Sutton and  
Andrew G. Barto.

p. cm. — (Adaptive computation and machine learning)

“A Bradford book.”

Includes bibliographical references and index.

ISBN 0-262-19398-1 (alk. paper)

I. Reinforcement learning (Machine learning) I. Barto, Andrew G.

II. Title. III. Series.

Q325.6.S88 1998

006.3'1—dc21

97-26416

CIP

# Index

Page numbers in italics are recommended to be consulted first.

- Absorbing state, 60
- Access-control queuing example, *154–155*, 157, 160
- Accumulating traces or Accumulate-trace methods, *173*, 176. *See also* Eligibility traces
  - vs. replacing traces, 186–189, 221(fig.)
- Acrobot: case study, 270–274
- Action nodes, 68
- Action preferences, *152*, 185
  - in the  $n$ -armed bandit problem, *41–42*, 43, 44
- Action-value functions, 69, 84, 257
  - and function approximation, 210–211
  - and Monte Carlo methods, 116–118
  - optimal, 75, 78, 84
  - and TD learning, *145–146*, 148, 152, 154
- Action-value methods, 27–30
- Action values, 69, 72–74, 116
  - in the  $n$ -armed bandit problem, 26, 27–28, 48
- Actions, 51–52
  - in the  $n$ -armed bandit problem, 26
- Actor–critic methods, 22, 41, *151–153*, 159–160
  - and Dyna, 237(fig.)
  - eligibility traces for, *185–186*, 192
- Afterstates, 156–157
- Agent, 5, *51–55*
- AI. *See* Artificial intelligence
- Andreae, John, 19, 84, 109
- Animal learning psychology. *See* Psychology
  - and reinforcement learning
- Approximation, 80–81
- Artificial intelligence (AI), 3, 5
  - and DP, 109
  - and reinforcement learning, 15, *18–23*, 83
- Associative learning, 18
- Associative memory network, 225
- Associative reinforcement learning, 49
- Associative reward-penalty algorithm, 226
- Associative search, *45–46*, 49
- Asynchronous dynamic programming, *103–104*, 107–108, 110, 254
- Average-reward case. *See* Undiscounted continuing tasks
- Averagers, 219
- Backgammon: program for. *See* TD-Gammon
- Backup diagrams, 71, 191
  - complex, 169
  - for DP, *70–71*, 76–77
  - for half-backups, 74
  - for Monte Carlo methods, 115, 117
  - for  $n$ -step TD methods, 165
  - for one-step methods, 243
  - for one-step Q-learning, 150

- Backup diagrams (*cont.*)
  - for  $Q(\lambda)$ , 183, 185
  - for Samuel's checker player, 269
  - for Sarsa( $\lambda$ ), 180
  - for TD(0), 135
  - for TD( $\lambda$ ), 170
- Backups, 12, 71
  - distribution of, 195–196, 201, 216–218, 246–249, 251, 252, 254
  - in dynamic programming, 91
  - full vs. sample, 242–246
  - function approximation and, 194–195, 196, 201
  - in heuristic search, 251–253
  - $n$ -step, 164–165, 166, 169–173
  - prioritized sweeping and, 239–240
  - trajectory sampling and, 248–250
  - in Samuel's checker player, 268–269
- Backward view of eligibility traces, 163, 175(fig.), 191, 192
  - equivalence with forward view, 176–179
  - with function approximation, 199, 211
  - in  $Q(\lambda)$ , 182
  - in TD( $\lambda$ ), 173–178
- Baird's counterexample, 216–218, 220, 224
- Bandit problems, 48, 128. *See also*  $N$ -armed bandit problem
- Batch updating or Batch training, 141–144, 159
- Bellman, Richard, 16, 21, 48, 49, 85, 109
- Bellman equations, 16, 85, 90
  - for  $V^\pi$ , 70
  - for  $Q^\pi$ , 72
  - for  $V^*$ , 76
  - for  $Q^*$ , 76
  - and backups, 108
  - and DP, 101, 108
  - solving the, 77–80
- Bellman error, 219–220, 224
- Binary bandit tasks, 33–36, 46, 49
- Binary features, 202, 205
  - vs. radial basis functions, 208
- Bioreactor: example, 54, 83
- Blackjack: examples, 112–114, 121, 131
- Blocking maze: example, 236–237
- Boltzmann distribution, 30
- Bootstrapping, 109
  - assessment of, 139–142, 220–223, 224
  - and DP, 109
  - and eligibility traces, 220–221
  - and function approximation, 195, 199, 201, 216–222, 222–223
  - and Monte Carlo methods, 115
  - and TD learning, 134, 138
- BOXES, 19, 131, 223
- Branching factor, 245
- Breakfast: example, 6–7, 23
- Bucket brigade, 159
- Car rental: example, 98–100, 157
- Cellular telephones: dynamic channel allocation in, 279–283
- Certainty-equivalence estimate, 144–145, 159
- Checkers, 56, 62, 251–252, 267–270
- Checkers player: Samuel's, 109, 225, 261, 267–270
- Chess, 6–7, 21, 56, 80, 84, 156, 226
- Classical conditioning models, 22
- Classifier systems, 20, 22, 225
- Cliff walking: example, 149, 150(fig.)
- CMAC. *See* Tile coding
- Coarse coding, 202–205, 208
- Complete knowledge, 17, 82
- Complex backups, 169
- Contingency space, 34(fig.), 49
- Continuing tasks, 58, 60–61, 83
- Continuous action, 89, 153, 193, 211, 226
- Continuous state, 63, 85, 89, 109, 193, 202, 205, 215
- Continuous time, 52, 85, 275, 276–277
- Credit assignment, 18, 163, 192
- Critic, 20, 151–152, 159
- Curse of dimensionality, 16, 107, 207
- Decision-tree methods, 197, 226
- Delayed reward, 4, 87, 191

- Direct reinforcement learning (Direct RL), 230–234, 254
- Dirichlet problem, 131
- Discount-rate parameter, 58, 277
- Discounting, 58–59, 61, 257
- Distribution models, 227–228, 244
- DP. *See* Dynamic programming
- Draw poker: example, 64–65
- Driving: example, 55
- Driving home: example, 135–138
- Dual control, 48
- Dyna agents, 230–238, 254, 258
  - architecture, 232
  - Dyna-AC, 237, 244
  - Dyna-Q, 230–237
  - Dyna-Q+, 237, 238
  - vs. prioritized sweeping, 240–241
- Dyna maze, 233–235, 241
- Dynamic channel allocation: in cellular telephone systems, 279–283
- Dynamic programming (DP), 16–17, 49, 89–110
  - and artificial intelligence, 109
  - backup diagrams for, 70–71, 76–77
  - efficiency of, 107–108
  - function approximation and, 194, 216–219, 222, 224, 225
  - incremental, 109
  - vs. Monte Carlo methods, 111–119, 129–131, 133, 256
  - reinforcement learning and, 9, 16–17, 23, 89, 109
  - temporal-difference learning and, 133–135, 138, 159, 256
- Elevator dispatching: case study, 274–279
- Eligibility traces, 163–192. *See also*
  - Accumulating traces, Replacing traces for actor–critic methods, 185–186, 192
  - performance with, 221(fig.)
  - and Q-learning, 182–185, 192
  - and Sarsa, 179–181, 192
  - with variable  $\lambda$ , 189–190, 192
- Environment, 3, 51
  - and agent, 51–44
  - Markov property and, 63
  - models of, 227
- Episodes, episodic tasks, 58–61, 83
- Error reduction property, 166
- Estimator algorithms, 48
- Evaluation vs. instruction, 25, 31–36
- Evaluative feedback, 25–49
- Evolution, 9, 18
- Evolutionary methods, 9, 11, 13, 20, 225, 228
- Experience, 111, 228
- Experience replay, 287
- Exploration–exploitation dilemma, 4–5, 130, 145, 236–238
  - in the  $n$ -armed bandit problem, 26–27, 30, 46–48, 49
- Exploratory moves or actions, 11, 12(fig.), 15, 182–184
- Exploring starts, 117, 120, 122, 130
- Farley and Clark, 18, 224
- Features, 200–213, 225
- Forward view of eligibility traces, 163, 171(fig.), 191
  - equivalence with backward view, 176–179, 192
  - with function approximation, 198–199
  - in  $Q(\lambda)$ , 182–185
  - in  $TD(\lambda)$ , 169–173
  - with variable  $\lambda$ , 190
- Full backups, 91, 108, 242–246, 255–256
- Function approximation, 193–225, 259
  - control with, 210–215
  - counterexamples to, with off-policy bootstrapping, 216–220
  - gradient-descent methods for, 197–200
  - linear methods for, 200–210
- Gambler’s problem, 101–103
- Gauss–Seidel-style algorithm, 110
- Gazelle calf: example, 6–7
- Generalization, 193–225

- Generalized policy iteration (GPI), 105–107, 108, 210, 211, 255
  - and Monte Carlo methods, 118, 122, 126, 130
  - and TD learning, 145, 154, 157–158
- Generalized reinforcement, 22
- Genetic algorithms, 8, 11, 20, 48, 225
- Gibbs distribution, 30–31
- Gittins indices, 48, 49
- Global optimum, 196
- Goal state, 239
- Goals, 4, 6, 56–57
- Golf: example, 72–73, 75–76, 80
- GPI. *See* Generalized policy iteration
- Gradient-descent methods, 197–201, 210–213, 222, 223–225
- Greedy or  $\epsilon$ -greedy action selection methods, 28–30, 48
- Greedy actions, 26
- Greedy policies, 77, 96, 119
  - $\epsilon$ -greedy policies, 122
- Grid tasks or Gridworld examples
  - cliff walking, 149
  - and DP, 92–94
  - and eligibility traces, 190–181
  - and value functions, 71–72, 78–79
  - windy, 146–148
- Hamilton, William, 16, 84
- Hamilton-Jacobi-Bellman equation, 85
- Hamming distance, 210
- Hashing, 207, 224
- Heuristic dynamic programming, 109
- Heuristic search, 250–253, 256
  - as expanding Bellman equations, 79
  - as sequence of backups, 252–253(fig.)
  - in TD-Gammon, 266
- Hierarchy and modularity, 259
- History of reinforcement learning, 16–23
- Holland, John, 20, 22, 48, 158–159, 225
- In-place algorithms, 91, 110
- Incomplete knowledge, 82
- Indirect reinforcement learning, 230–231, 254
- Information state, 47, 49
- Instruction vs. evaluation, 25, 31–36
- Interaction, 6
  - agent–environment, 52
  - learning from, 3
- Interval estimation methods, 47, 49
- Jack’s car rental: example, 98–100, 157
- Job-shop scheduling: case study, 283–290
- Kanerva coding, 209–210, 224
- Klopf, Harry, 20–22, 158, 192
- $\lambda$ -return, 170–173, 189–190, 191, 198
- $\lambda$ -return algorithm, 171–172, 176–180
- Law of Effect, 17–18, 49
- Learning. *See also* by type
  - from examples. *See* Supervised learning
  - from interaction, 3, 9, 13
  - and planning, 9, 227–254, 259–260
- Learning automata, 20, 34–35, 48, 49
- Least-mean-square (LMS) algorithm, 20, 223
- Linear methods, function approximation
  - using, 200–210
  - bound on prediction error, 201
  - convergence of, 201, 223–224
- LMS. *See* Least-mean-square algorithm
- Local optimum, 196, 198–199, 201
- Lookup tables. *See* tabular methods
- Machine learning, 3, 4, 23
  - special journal issues on reinforcement learning, 23
- Markov decision processes (MDPs), 16, 17, 23, 66–67, 82, 83–84. *See also* Partially observable MDPs, Semi-Markov decision process
- Markov property, 61–65, 82, 130, 258–259
- Maximum-likelihood estimate, 144
- Maze examples, 233–235, 236, 238, 240

- MC methods. *See also* Monte Carlo methods  
 constant- $\alpha$ , 134, 136, 139–142, 144, 171  
 every-visit, 112, 117, 131, 188  
 first-visit, 112–113, 117, 131, 188
- MDPs. *See* Markov decision processes
- Mean-squared error (MSE), 195–196, 201
- MENACE (Matchbox Educable Naughts and Crosses Engine), 19, 84
- Michie, Donald, 19–20, 83, 84, 131, 223
- Minimax, 10, 268–269
- Minsky, Marvin, 18, 21, 22, 109
- Model-free methods. *See* Direct RL
- Model-learning, 230–238, 254
- Models (of the environment), 9, 82, 227–228  
 incorrect, 235–238  
 and planning, 9, 227–235  
 types of, 227–228
- Modified policy iteration, 110
- Modified Q-learning, 159
- Monte Carlo methods, 111–131. *See also*  
 MC methods  
 advantages of, 129–131  
 backup diagrams for, 115, 117  
 and control, 118–121  
 convergence of, 112, 120–121, 124  
 and DP methods, 111–112, 129–131  
 eligibility traces and, 163, 172, 188, 190–191, 192  
 incremental implementation of, 128–129  
 $n$ -step backups and, 164–166, 170–171, 172  
 off-policy control by, 126–128  
 on-policy control by, 122–124  
 and TD learning, 133–145, 169, 175
- Monte Carlo with Exploring Starts (Monte Carlo ES), 120–121
- Mountain-car task, 214–215
- MSE. *See* Mean-squared error
- $N$ -armed bandit problem, 20, 26–27, 48  
 action-value methods and, 27–31  
 associative search and, 45–46  
 initial action-value estimates and, 39–41  
 evaluation vs. instruction in, 31–36
- Monte Carlo methods and, 128  
 nonstationary environments and, 38–39  
 pursuit methods for, 43–45  
 reinforcement comparison methods for, 41–43
- $N$ -step backups, 164–166, 191
- $N$ -step returns, 165–166, 191
- $N$ -step TD methods, 164–168  
 and Monte Carlo methods, 164–166, 170–171, 172  
 problems with, 167  
 and TD( $\lambda$ ), 169–172, 191
- Naive Q( $\lambda$ ), 184
- Neural networks, 21, 22, 23, 225, 263–266
- Neuroscience, 22, 192
- Non-Markov problems, 63, 64, 153, 190–191, 258–259
- Nonassociative problems and methods, 20, 25, 45, 284
- Nonbootstrapping methods, 220–222
- Nonstationarity, 30, 38–39, 128, 195
- Off-line updating, 166
- Off-policy methods, 126–127, 211  
 DP as, 216  
 Monte Carlo, 124–128  
 problems with bootstrapping in, 216–220, 222  
 Q-learning as, 182
- On-line updating, 166, 175, 192
- On-policy distribution, 196, 201, 216–217  
 vs. uniform distribution, 247–249
- On-policy methods, 122, 130  
 Monte Carlo, 122–124  
 vs. off-policy methods, 130, 148–150  
 Sarsa as, 145–148, 179–181
- One-step methods, 164  
 backups diagrams for, 243
- Optimal control, 16–17, 83–84
- Optimality,  
 and approximation, 80–81  
 of TD(0), 141–145  
 and value functions, 75–80
- Optimistic initial values, 39–41, 215

- Partially observable MDPs (POMDPs), 17, 49, 258–259
- Pattern recognition, 4, 18
- Peng's  $Q(\lambda)$ , 182, 183–185(fig.), 192
- Petroleum refinery: example, 6–7
- Planning, 9, 227–254, 259–260
  - deliberation vs. reaction in, 9, 231
  - and DP, 229
  - in Dyna, 230–238
  - and heuristic search, 250–252
  - incremental, 230, 253
  - integrated with action and learning, 230–238
  - and learning, 9, 227–254
  - partial-order, 228
  - by Q-learning, 229–230
  - and reinforcement learning, 5, 9
  - state-space vs. plan-space, 228
  - and trajectory sampling, 246–250
- Pleasure and pain, 7–8, 57
- Pole-balancing, 59, 64, 83, 202, 221(fig.).  
*See also* BOXES
- Policy, 52
  - behavior, 126
  - deterministic, 92, 291
  - equiprobable random, 93
  - estimation, 126
  - greedy, 77, 96, 119
  - $\epsilon$ -greedy, 122
  - optimal, 75
  - soft,  $\epsilon$ -soft, 100, 122
  - stochastic, 52
- Policy evaluation, 90–93, 94(fig.)
  - for action values, 116–117, 145–146
  - by DP, 90–93
  - iterative, 91–92
  - by Monte Carlo methods, 112–117,
  - of one policy while following another, 124–126
  - by TD methods, 133–146
  - termination of, 92
- Policy improvement, 93–98
  - and Monte Carlo methods, 118–119
  - theorem of, 95–97
- Policy iteration, 97–100. *See also*
  - Generalized policy iteration
  - by Monte Carlo methods, 118–119
- POMDPs. *See* Partially observable MDPs
- Prediction problem, 90
- Prior knowledge, using, 14, 56, 260
- Prioritized sweeping, 239–240, 241(fig.), 253, 254
- Prototype states, 210
- Psychology and reinforcement learning, 16, 48, 254. *See also* Classical conditioning models, Law of Effect, Secondary reinforcers
  - and shaping, 260
  - and stimulus–response associations, 7
  - and stimulus traces, 192
- Pursuit methods, 43–45
- Q-functions, 84. *See also* Action-value functions
- Q-learning, 23, 148–151, 159, 224, 277
  - convergence of, 148, 159, 192, 216, 218
  - and eligibility traces, 182–185, 192
  - and planning, 229–230
- Q-planning, 229–230, 231–232
- $Q(\lambda)$ , 182–185, 192
- Quasi-Newton methods, 223
- Queuing example, access-control, 154–155, 157
- R-learning, 153–155, 160
- Racetrack: example, 127–128, 131
- Radial basis functions (RBFs), 208, 224
- Random walk: examples,
  - batch updating on, 142
  - five state, 139–142, 166–167
  - $\lambda$ -return algorithm on, 172
  - with  $n$ -step methods on, 166–168
  - 19-state, 167–168, 172, 178–179
  - TD vs. MC methods on, 139–142
  - TD( $\lambda$ ) on, 178–179
  - values learned by TD(0) on, 140(fig.)
- Random-sample one-step tabular Q-planning, 229–230



- RBFs. *See* Radial basis functions
- Reactive decision-making, 231
- Real-time DP, 254
- Real-time heuristic search, 254
- Recursive-least-square methods, 223
- Recycling robot: examples, 6–7, 55, 56, 66–68, 78, 83
- Reference reward, 41–42
- REINFORCE algorithms, 226
- Reinforcement comparison methods, 41–43, 49, 152
- Reinforcement learning problem, 4, 51–85
- Replacing traces or Replace-trace methods, 186–189, 221(fig.). *See also* Eligibility traces
- Residual-gradient methods, 224
- Return, 57–60, 81, 257
  - $n$ -step, or corrected  $n$ -step truncated, 165
  - unified notation for, 60–61
  - and value functions, 68–69, 75
- Rewards, 7–8, 51–52, 53, 54, 81
  - and goals, 56–57
  - in the  $n$ -armed bandit problem, 26
  - reference, 41
- RMS error. *See* Root mean-squared error
- Robot examples, 52, 53, 56, 57, 59, 202, 270
  - pick and place, 54
  - recycling, 55, 56, 66–68, 78, 83
  - trash collecting, 6–7
- Rod maneuvering: example, 240–242
- Root mean-squared error, 141(fig.)
- Rosenblatt, Frank, 19
- Rubik's cube: example, 53
  
- Sample-average methods, 28
- Sample backups, 135, 242, 246, 255
- Sample models, 129, 227–228
- Samuel, Arthur, 21, 22, 109, 225. *See also*
  - Checkers player:
- Samuel's
- Sarsa, 145–148, 159, 210
- Sarsa( $\lambda$ ), 179–181, 192
  - linear, gradient-descent, 211–213, 224
  - and TD( $\lambda$ ), 179–180
- Search control, 232
- Secondary reinforcers, 21
- Selectional learning, 18, 19, 20
- Selective bootstrap adaptation, 20, 225–226
- Self-play, 14, 266, 268
- Semi-Markov decision process, 276–277, 280–281
- Sequential design of experiments, 48, 83
- Shannon, Claude, 21, 84, 267–268
- Shortcut maze: example, 236–238
- Signature tables, 225–270
- Simulated experience, 111
- SNARCs (Stochastic Neural-Analog Reinforcement Calculators), 18
- Soap bubble: example, 116, 131
- Softmax method, 30–31, 49
- Space shuttle payload processing (SSPP):
  - case study, 284–286
- Sparse distributed memory, 224
- State(s), 52–55, 61–65, 81, 83. *See also* State representations
- State-action eligibility traces, 179–180, 188
- State aggregation, 199–200, 202, 223, 224, 225, 259
- State nodes, 68
- State representations, 61–62
  - augmenting, 258–259
  - exploiting structure in, 260
- State values or State value functions, 68–73, 75–77, 84
- STeLLA system, 19
- Step-size parameters, 12–13, 37, 38–39
- “Steps Toward Artificial Intelligence” paper (Minsky), 18, 21, 22
- Stochastic approximation convergence
  - conditions, 39
- Stochastic approximation methods, 33
- Supervised algorithm, 33
- Supervised learning, 4, 19, 32, 193, 195
- Sweeps, 91, 246. *See also* Prioritized sweeping
- Symmetries: in Tic-tac-toe, 14–15
- Synchronous algorithms, 110

- System identification, 254. *See also* Model learning
- Tabular methods, 80–81
- Target or Target function, 37, 134, 164–165, 194–195
- TD error, 152, 174–165, 178, 180
- TD-Gammon, 14, 261–267
  - and heuristic search, 251
  - and Samuel’s checker player, 268
- TD learning. *See* Temporal-difference learning
- TD(0), 23, 134, 135(figs.), 159
  - vs. constant- $\alpha$  MC, 139–142
  - convergence of, 138, 159
  - function approximation and, 194, 223
  - $\lambda$ -return algorithm and, 171
  - optimality under batch updating of, 141–145
  - and Sarsa, 145
  - and TD( $\lambda$ ), 175
- TD(1), 175, 188, 216
- TD( $\lambda$ ), 22, 169–175, 191–192
  - backward view of, 173–175
  - convergence of, 191–192, 201
  - forward view of, 169–173
  - function approximation and, 198–200, 201, 223–224
  - and Sarsa( $\lambda$ ), 179–180
- TDNN. *See* Time-delay neural network
- Temperature, 31, 42, 49, 278, 285
- Temporal-difference error. *See* TD error
- Temporal-difference learning (TD learning), 133–160, 256
  - actor–critic methods and, 151–153
  - advantages of, 138–145
  - bootstrapping and, 138
  - convergence of, 138
  - DP and, 134, 138
  - eligibility traces and, 163, 190–191
  - and GPI, 157–158
  - Monte Carlo methods and, 133–145, 175, 188
  - $n$ -step, 164–168
  - one-step, 148, 158, 159, 164, 179
  - off-policy control and, 148–151
  - on-policy control and, 145–148
  - roots of, 21–23, 158–159
  - Tic-tac-toe example, 12–13
- Terminal state, 58
- Testbed, 10-armed, 28
- Thorndike, Edward, 17–18
- Tic-tac-toe: example, 10–15. *See also* MENACE
- Tile coding, 204–208, 215, 224, 272–273
- Time-delay neural network (TDNN), 287–289
- Time steps, 52
- Trace-decay parameter, 173, 189–190
- Trajectory sampling, 246–250, 253, 254
- Transient DP, 254
- Transition graphs, 67–68
- Transition probabilities, 66
- Trial-and-error learning, 16, 17–21
- Tsitsiklis and Van Roy’s counterexample, 219–220
- Undiscounted continuing tasks, 61, 153–155, 160, 224
- Value(s), 8, 69, 84
  - in the  $n$ -armed bandit problem, 26
  - relative, 154
- Value functions, 8–9, 68–80, 82, 229, 286
  - and DP, 89–90
  - function approximation and, 193
  - optimal, 75–80
  - policy evaluation and, 90–93
  - roots of, 16, 84
- Value iteration, 100–103
- Vision system: example, 65
- Watkins’s Q( $\lambda$ ), 182–184, 192, 211, 213, 224
- Werbos, Paul, 23, 84, 109, 159, 223
- Widrow, Bernard, 19, 20, 49, 131, 223, 225