A Beginner's Guide to Reinforcement Learning (for Java)

Neural networks have become well known for recent advances in such diverse fields as computer vision, machine translation and time series prediction – but reinforcement learning may be their killer app.

Reinforcement learning is goal-oriented. RL algorithms learn how to attain a complex objective or maximize along a dimension over many steps, starting from a blank slate, and under the right conditions they achieve superhuman performance.

Reinforcement algorithms with deep learning at their core are currently beating expert humans at numerous Atari video games (https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf). While that may sound trivial, it's a vast improvement over their previous accomplishments. Two reinforcement learning algorithms - Deep-Q learning and A3C - have been implemented in a Deeplearning4j library called RL4J (https://github.com/deeplearning4j/rl4j). They it can already play Doom (https://www.youtube.com/watch?v=Pgktl6PWa-o).

In time, we expect reinforcement learning to perform better in more ambiguous, real-life environments while choosing from an arbitrary number of possible actions, rather than from the limited options of a video game. When people talk about building robot armies, this is what they mean. :)

GET STARTED WITH DEEPLEARNING4J (quickstart)

Reinforcement Learning

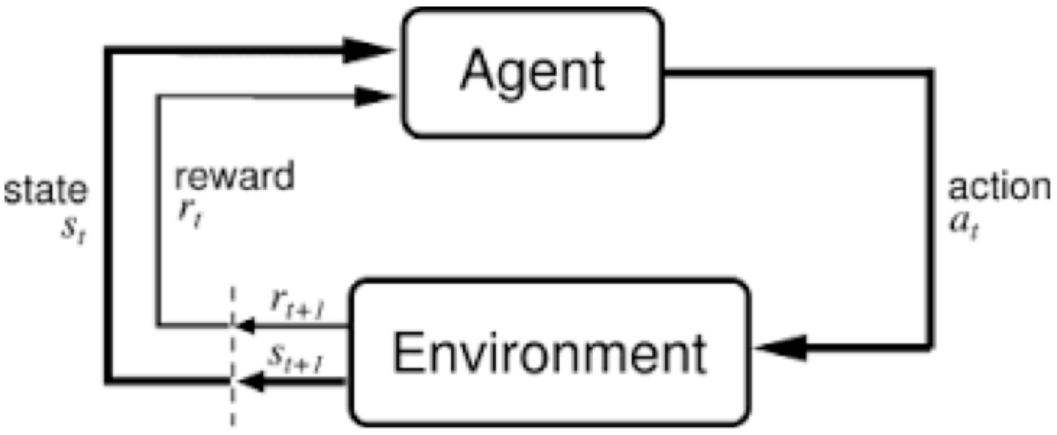
Reinforcement learning is based on agents, environments, states, actions and rewards, all of which we'll explain.

An agent takes actions; for example, a drone making a delivery, or Super Mario navigating a video game.

A state is the situation in which the agent finds itself; i.e. a specific place and moment, a configuration that puts the agent in relation to other significant things such as tools, obstacles, enemies or prizes.

An action is almost self-explanatory, but it should be noted that agents choose among a list of possible actions. In video games, the list might include running right or left, jumping high or low, crouching or standing still. In the stock markets, the list might include buying, selling or holding any one of an array of securities and their derivatives. When handling aerial drones, alternatives would include many different velocities and accelerations in 3D space.

A reward is the feedback by which we measure the success or failure of an agent's actions. For example, in a video game, when Mario touches a coin, he wins points. An agent sends output in the form of actions to the environment, and the environment returns the agent's new state as well as rewards.



In the feedback loop above, the subscripts denote time steps t and t+1, each of which refer to different states: the state at moment t, and the state at moment t+1. Unlike other forms of machine learning – such as supervised and unsupervised learning – reinforcement learning can only be thought about sequentially in terms of state-action pairs that occur one after the other.

Reinforcement learning judges actions by the results they produce. It is goal oriented, and its aim is to learn sequences of actions that will lead it to achieve its goal. In video games, the goal is to finish the game with the most points, so each additional point obtained throughout the game will affect the agent's subsequent behavior; i.e. the agent may learn that it should shoot battleships, touch coins or dodge meteors to maximize its score.

In the real world, the goal might be for a robot to travel from point A to point B, and every inch the robot is able to move closer to point B could be counted like points.

RL differs from both supervised and unsupervised learning by how it interprets inputs. We can illustrate their difference by describing what they learn about a "thing."

Unsupervised learning: That thing is like this other thing. (Similarities w/o names) Supervised learning: That thing is a "double bacon cheese burger". (Labels) Reinforcement learning: Eat that thing because it tastes good and will keep you alive. (Actions based on short- and long-term rewards.)

DL4J
DEEPLEARNING4J
(../index.html)

Deep Learning Textbook (https://www.amazon.com/Deep-Learning-Practitioners-Adam-Gibson/dp/1491914254)

Request Corporate Training (https://skymind.ai/academy)

Getting Started ›

Tutorials ›

Introduction to Deep Learning ›

Neural Networks ›

Data & ETL ›

Tuning & Training ›

Deployment ›

Open-Source Community ›

Natural Language Processing ›

ND4J: Numpy for the JVM ›

Resources ⌄

- Eigenvectors, PCA, Covariance and Entropy (eigenvector)
- Thought Vectors, AI and NLP (thoughtvectors)
- Monte Carlo, Markov Chains and Deep Learning (markovchainmontecarlo)
- Unsupervised Learning: Use Cases (unsupervised-learning)
- DL and Reinforcement Learning (reinforcementlearning)
- Scala, Spark and Deep Learning (scala)
- DL4J, Torch7, Theano and Caffe (compare-dl4j-torch7-pylearn)
- Glossary of Terms for Deep Learning and Neural Nets (glossary)
- Free Online Courses, Tutorials and Papers (deeplearningpapers)
- Deep Learning in Other Languages (deeplearningtranslated)

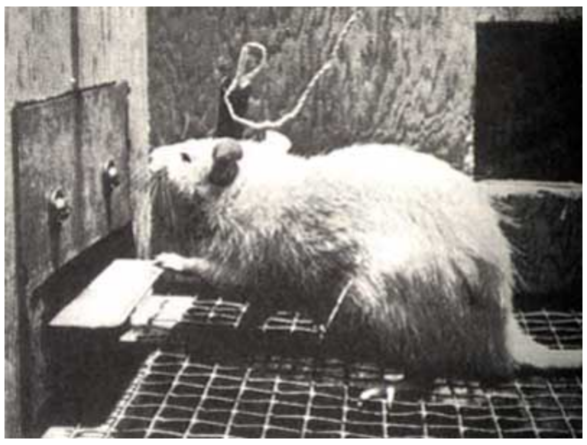
Other Languages ›

Subscribe to our mailing list

Keep me updated!

One way to imagine an autonomous RL agent would be as a blind person attempting to navigate the world with their ears and a white cane. Agents have small windows that allow them to perceive their environment, and those windows may not even be the most appropriate way for them to perceive what's around them.

(In fact, deciding *which types* of feedback your agent should pay attention to is a hard problem to solve, and glossed over by algorithms that are learning how to play video games, where the kinds of feedback are limited and well defined. These video games are much closer to the sterile environment of the lab, where ideas about reinforcement learning were initially tested.)



The goal of reinforcement learning is to pick the best known action in any state, which means the actions have to be ranked, assigned values relative to one another.

Since those actions are state dependent, what we are really gauging is the value of state-action pairs; i.e. an action taken from a certain state, something you did somewhere.

If the action is marrying someone, then marrying a 35-year-old when you're 18 should mean something different than marrying a 35-year-old when you're 90.

If the action is yelling "Fire!" Performing the action a crowded theater should mean something different from performing the action next to a squad of men with rifles. We can't predict an action's outcome without knowing the context.

We map state-action pairs to the values we expect them to produce with the Q function.

The Q function takes as its input an agent's state and action, and maps them to probable rewards. Reinforcement learning is the process of running the agent through sequences of state-action pairs, observing the rewards that result, and adapting the predictions of the Q function to those rewards until it accurately predicts the best path for the agent to take. That prediction is known as a policy.

Reinforcement learning is iterative. In its most interesting applications, it doesn't begin by knowing which rewards state-action pairs will produce. It learns those relations by running through states again and again, like athletes or musicians iterate through states in an attempt to improve their performance.

Reinforcement learning is Groundhog Day for algorithms. And since most humans never experience their Groundhog Day, that means reinforcement learning gives algorithms the potential to learn more, and better, than humans. In fact, that's the gist of the last several papers published by DeepMind, since their algorithms now show superhuman performance on most of the video games they've trained on. LINK

Neural Networks and Reinforcement Learning

Where do neural networks fit in? Neural networks are the agent that learns to map state-action pairs to rewards. Like all neural networks, they use coefficients to approximate the function relating inputs to outputs, and their learning consists to finding the right coefficients, or weights, by iteratively adjusting those weights along gradients that promise less error.

In reinforcement learning, convolutional networks can be used to recognize an agent's state; e.g. the screen that Mario is on, or the terrain before a drone. That is, they perform their typical task of image recognition.

But convolutional networks derive different interpretations from images in reinforcement learning than in supervised learning. In supervised learning, the network applies a label to an image; that is, it matches names to pixels.

Chat with us on Gitter

DL4J
DEEPLEARNING4J
(../index.html)

Deep Learning Textbook (https://www.amazon.com/Deep-Learning-Practitioners-Adam-Gibson/dp/1491914254)

Request Corporate Training (https://skymind.ai/academy)

Getting Started ›

Tutorials ›

Introduction to Deep Learning ›

Neural Networks ›

Data & ETL ›

Tuning & Training ›

Deployment ›

Open-Source Community ›

Natural Language Processing ›

ND4J: Numpy for the JVM ›

Resources ⌄

Eigenvectors, PCA, Covariance and Entropy (eigenvector)

Thought Vectors, AI and NLP (thoughtvectors)

Monte Carlo, Markov Chains and Deep Learning (markovchainmontecarlo)

Unsupervised Learning: Use Cases (unsupervised-learning)

DL and Reinforcement Learning (reinforcementlearning)

Scala, Spark and Deep Learning (scala)

DL4J, Torch7, Theano and Caffe (compare-dl4j-torch7-pylearn)

Glossary of Terms for Deep Learning and Neural Nets (glossary)

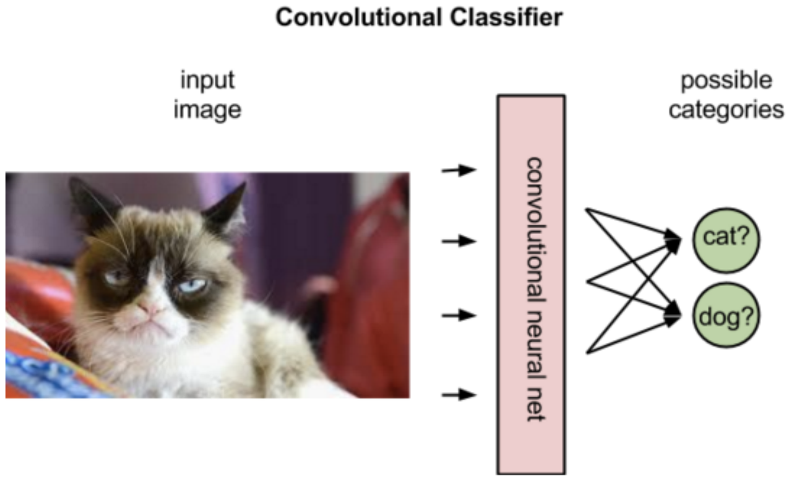
Free Online Courses, Tutorials and Papers (deeplearningpapers)

Deep Learning in Other Languages (deeplearningtranslated)

Other Languages ›

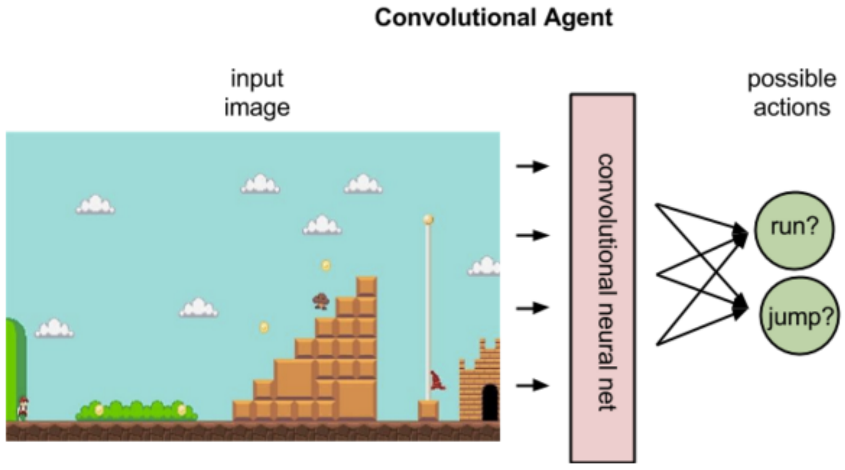
Subscribe to our mailing list

Keep me updated!



In fact, it will rank the labels that best fit the image in terms of their probabilities. Shown an image of a donkey, it might decide the picture is 80% likely to be a donkey, 50% likely to be a horse, and 30% likely to be a dog.

In reinforcement learning, given an image that represents a state, a convolutional net can rank the actions possible to perform in that state; for example, it might predict that running right will return 5 points, jumping 7, and running left none.



Having assigned values to the expected rewards, the Q function simply selects the state-action pair with the highest so-called Q value.

At the beginning of reinforcement learning, the neural network coefficients may be initialized stochastically, or randomly. Using feedback from the environment, the neural net can use the difference between its expected reward and the ground-truth reward to adjust its weights and improve its interpretation of state-action pairs.

This feedback loop is analogous to the backpropagation of error in supervised learning. However, supervised learning begins with knowledge of the ground-truth labels the neural network is trying to predict. Its goal is to create a model that maps different images to their respective names.

Reinforcement learning relies on the environment to send it a scalar number in response to each new action. The rewards returned by the environment can be varied, delayed or affected by unknown variables, introducing noise to the feedback loop.

This leads us to a more complete expression of the Q function, which takes into account not only the immediate rewards produced by an action, but also the delayed rewards that may be returned several time steps deeper in the sequence.

Like human beings, the Q function is recursive. Just as calling the wetware method human() contains within it another method human(), of which we are all the fruit, calling the Q function on a given state-action pair requires us to call a nested Q function to predict the value of the next state, which in turn depends on the Q function of the state after that, and so forth.

Further Reading

- RL4J: Reinforcement Learning in Java (https://github.com/deeplearning4j/rl4j)
- Richard S. Sutton and Andrew G. Barto's Reinforcement Learning: An Introduction (https://webdocs.cs.ualberta.ca/~sutton/book/the-book.html)
- Andrej Karpathy's ConvNetJS Deep Q Learning Demo (https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html)
- Brown-UMBC Reinforcement Learning and Planning (BURLAP) (http://burlap.cs.brown.edu/)(Apache 2.0 Licensed as of June 2016)
- Glossary of Terms in Reinforcement Learning (http://www-anw.cs.umass.edu/rlr/terms.html)
- Reinforcement Learning and DQN, learning to play from pixels (https://rubenfiszel.github.io/posts/rl4j/2016-08-24-Reinforcement-Learning-and-DQN.html)