Can't Artificial Intelligence Already Do That?

How to Train Your Robot

Chapter 1

Brandon Rohrer

About This Project

How to Train Your Robot is a long term side project. I've been working on it in some form for 20 years. I don't know if I'll ever finish it. But I find it deeply satisfying to share progress as I go, and who knows, maybe someone will find it useful. This chapter is the first installment.

Share and enjoy!

Brandon
Boston, USA
August 18, 2022

Can't AI Already Do That?

Chapter 1

In which we answer this question with a no,
but provide a running start on an alternative.

What do we even mean when we talk about training a robot? There are radically different ways to interpret it, so let's spell out what we're talking about here. The best mental picture I can offer you is that of training a puppy.

If I want to teach my pup to sit, I give her a cue, like a hand gesture or a verbal command, and then encourage her to sit down. As soon as she does I reward her with praise or her favorite treat. Then I repeat this process dozens of times. In my puppy's brain, the pattern of hearing me say "sit" and then doing the action of sitting down reliably leads to getting her treat of choice, a fresh blueberry.

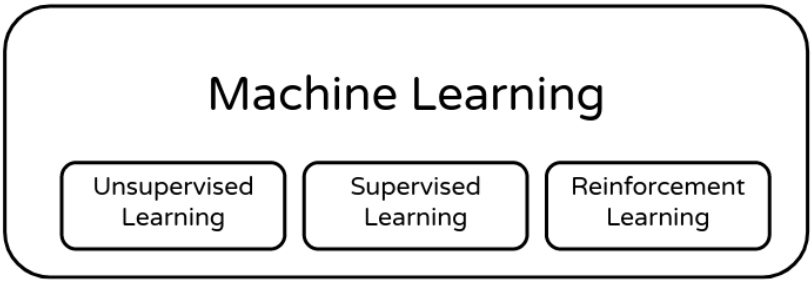
I would like to be able to train a robot the same way. I want to provide a cue and, when the robot does what I want, give it a reward. Once that pattern is established, I can back off on the

reward and just provide it occasionally. Like puppies, the robot should learn that doing what I want *might* get it a reward, so it's worthwhile to do the action even if I don't have a treat in hand.

Sadly, my Shih Tzu is smarter than me in this respect. If there's no treat to be had she lets me know that she couldn't care less what I want.

It's only natural to ask: Can't AI already do this? Surely any technology that can outsmart the world's most brilliant chess players and categorize billions of spam emails per minute can learn to sit. Right?

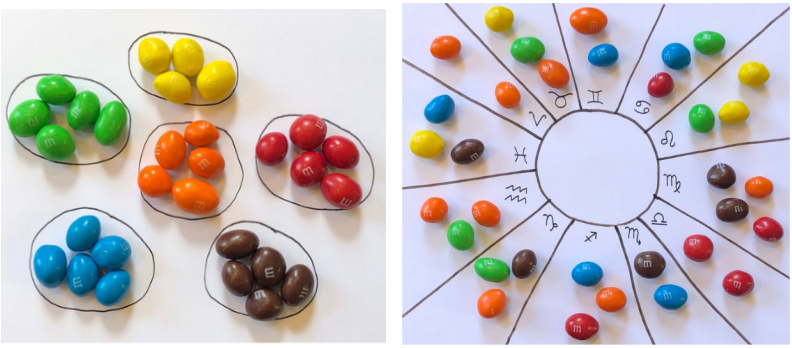
The short answer is no. The longer answer requires a quick walking tour through the field of artificial intelligence.



When we talk about artificial intelligence in 2022, we are most likely referring to a set of statistical learning methods known collectively as machine learning. Machine learning methods can be broken down into three broad categories, unsupervised learning, supervised learning, and reinforcement learning.

Unsupervised learning

Unsupervised learning is a grab bag of algorithms. They have names like **clustering** or **embedding** or **dimensionality reduction**, but what they have in common is that they can sort things into groups. They are the Marie Kondo of machine learning. If someone hands you a bag of M&Ms and says "Organize these," it's likely that you'd sort them by color, and you might even rank the colors based on how many candies of each color were present. But it's equally valid to sort them by weight, or by how perfectly round they are, or by the patterns of imperfections in the printed **m**.



M&M candies clustered by color (left) and astrological sign (right).

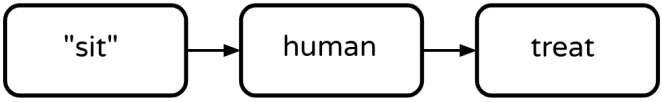
This is an example of unsupervised learning. There's no objectively right or wrong way to do it, it all depends on what you're trying to accomplish. Different methods give different results and you get to choose which results you like best.

In the context of training a robot, unsupervised learning is quite useful, but it doesn't get us all the way there. Robots are often forced to process a large amount of information. A robot can have wheels with odometers and tachometers, motors with current and torque sensors, all manner of bumpers and whiskers, rangefinders and proximity detectors. Alexa, Siri, and Google Home can be thought of as robots with microphones, giving them a continuous pipeline of audio information to process. Self-driving cars are often equipped with LIDAR, which produces an entire array of laser-measured distances many times per second. And thanks to all of the fantastic work in miniaturizing video cameras, many robots ingest a fat stream of data from the millions of pixel sensors they have on board.

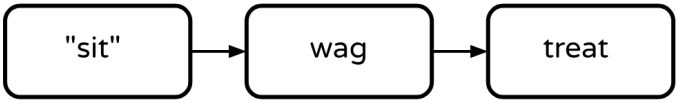
Unsupervised learning simplifies this problem for the robots. By learning naturally occurring patterns in the data stream, a robot can learn to operate on a simplified version of its world. If a robot's purpose is to identify ripe tomatoes on a conveyor belt, then a carefully designed unsupervised learning method can reduce the billions of bits of visual information it receives every second to just a few that help it make a specific decision: Is this a tomato? Is it ripe? Unsupervised learning is a way of going from way too much data to just the right amount.

By including the element of time, variants of unsupervised learning can even learn useful sequences. In our puppy training example, they could pick up on the fact that a verbal command to "sit," when followed by the action of sitting, is always followed by a treat.

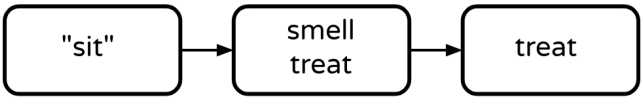
This feels like it's getting very close to what we need in order to train a robot. And we will revisit it later as part of a useful approach. But unsupervised learning is missing a critical piece – it doesn't inherently have a way to choose an action. While it can in theory learn the pattern we are trying to train, it is just as likely to learn a number of other patterns that are less helpful, such as the verbal command to "sit," detecting the presence of a human, and receiving a treat:



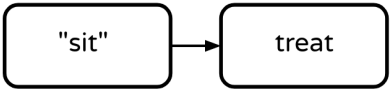
Also



or



or even just



There are any number of sequences that a robot puppy could learn that would not be helpful in generating the desired action. As a trainer, it's good practice to repeat this procedure in as many different contexts as possible to eliminate confusion and draw attention to the critical part, the action of sitting. But because the pup has access to such rich sensory information, the number of possible patterns is enormous, and it's impossible to cleanly eliminate all of the competing options.

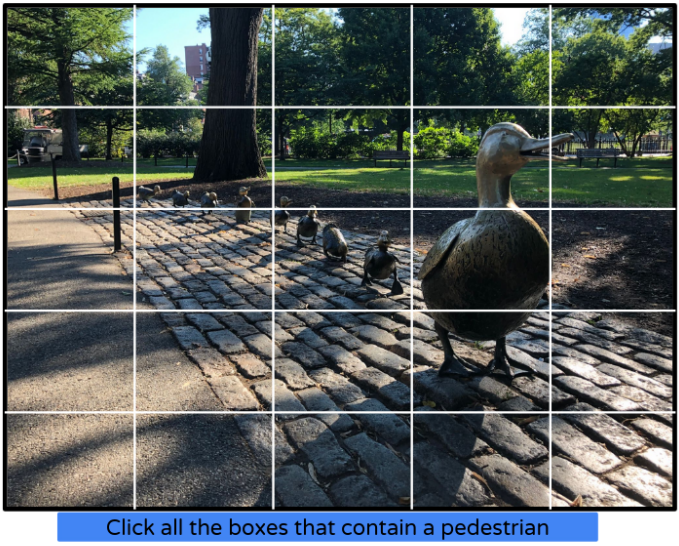
Learning the direct relationship between the act of sitting in response to the verbal command and receiving a treat is a **causal modeling** problem. The pup needs to learn that her decision to sit causes the reward (or at least starts a chain of events that reliably leads to a reward). This is also called building a world model. The pup has to build a mental picture of how the world works in order to choose the right strategies to get what she wants. Puppies are pretty good at this. For robots to catch up to

them they will have to rely on more than just unsupervised learning.

Unsupervised learning is so named because it treats all the data equally. It ingests sensor data in the form of video, verbal commands, and reward signals and organizes it according to its whims. It doesn't give preferential treatment to any of the data channels. It doesn't treat any of them as special or privileged. But sometimes it's helpful to give a signal preferential treatment, particularly when that signal comes from a human supervisor.

Supervised learning

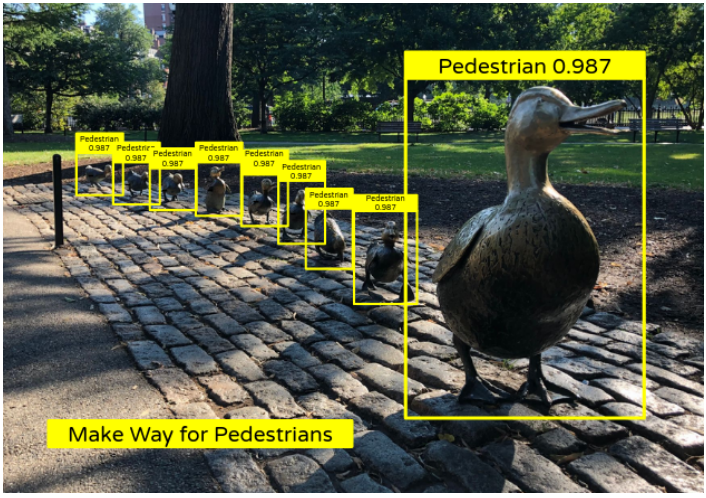
Supervised learning does exactly this. It has a privileged data channel, called a **label**. If you want to train a machine learning algorithm to recognize a pedestrian so that your self-driving car can avoid them, you do this by providing thousands of images, some of which contain pedestrians. You also provide a label for each image. This label is a separate categorical data channel that the algorithm treats as absolute truth.



This label is most often generated by having humans look at the images and determine whether or not they contain a pedestrian. (If you're clever, you can get unsuspecting Internet users to do this work for you by calling it a test to prove they are not robots.

The irony here is that after sufficient training, robots can become better at this task than humans.)

This categorical, true or false, one or zero piece of information helps the supervised learning algorithm to know what the right answer should be. It's the cheat sheet. Then the algorithm literally works backward and adjusts its decision making process so that it's more likely to give the right answer the next time it sees that particular image. After repeating this process millions of times, the algorithm gets good enough that you can show it an image it has never seen before, and it can tell you whether there's a pedestrian in it.



This is a two phase process. Phase one is **training**, where the labels are used to adjust the algorithm and teach it to make good

predictions. Phase two is **testing**, where the algorithm is given unfamiliar images and asked to classify them and then has its answers compared against the ground truth label.

Supervised learning comes in many forms. Labels can be categories, as in our pedestrian example, or they can be numbers, as in models that learn to predict temperatures or prices. Supervised learning with categorical labels is called **classification**, and with numerical labels it's called **regression**. But they are both variants of the same underlying mechanism–generating a decision process that gives predictions that are as close as possible to the known right answers.

Supervised learning is far and away the most popular class of algorithms in use. Your email spam filter is trained using supervised learning. The headline-grabbing large language models that write plausible sounding articles and movie scripts are all based on a very clever supervised learning approach called a transformer. The cashier-less supermarkets that watch you shop and charge you automatically run on supervised learning too. Unsupervised learning often plays a supporting role in these applications, but supervised learning, with its curated collections of human labeled examples and training and testing procedures, are the star of the show.

With a little imagination we can adopt our robot training project to be a supervised learning problem. Instead of assigning a label of "pedestrian" or "not pedestrian" to an image, we could have

the algorithm instead assign a best action label to every possible set of robot sensor readings.

In a self-driving car this might mean assigning one of the available actions {turn_right, turn_left, accelerate, brake, do_nothing} to every moment's collection of car sensor data. This approach assumes that in every situation the robot will encounter, there is a single best thing to do, one correct action to take.

The privileged channel of information here, the human labels, require a little bit of work to construct. When my puppy sits after I tell her to sit, I give her a blueberry to let her know she took the right action at the right time. The combination of sitting-plus-blueberry creates a label for that situation. When the self-driving car is supposed to turn left, and then does turn left, a human can hit a button to reward it. The reward tells the algorithm that its most recent action was the right one for that situation. The reward singles out and elevates that particular action. The combination of action-plus-reward creates a label.

This is technically a valid way to train a robot. A sufficiently patient human trainer can wait until the robot chooses the right action enough times in enough situations that it learns the patterns of how to behave. Eventually a robot dog would learn, no matter what else it is sensing, that when it hears a verbal command to "sit" it should sit down.

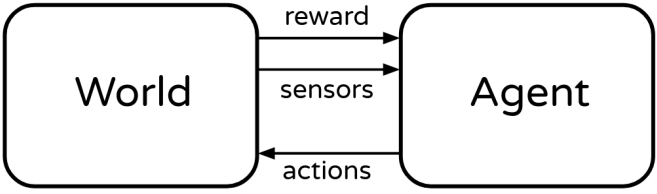
The barriers to training a robot in this way are mostly practical. It can take a very long time for a robot to choose the right action by chance when it has many actions to choose from. With the two classes of {pedestrian, not_pedestrian} this problem isn't so pronounced, but if your robot has a lot of actions to choose from, and it's trying to sort every situation into dozens or hundreds of action categories by trial and error, the amount of experience it needs to sort them correctly goes up accordingly.

This approach is also less than ideal because it discards a lot of information. Every time an action is taken and not rewarded, you and I can conclude that the action was wrong for that situation. There's useful information in the absence of a reward. Unfortunately, supervised learning algorithms don't have a natural way to make use of these negative examples. For any real robot, the large majority of its actions will be wrong and end up unrewarded and unlabeled. This leaves a lot of valuable information unused.

Reinforcement Learning

You may be wondering whether there's a way to fix or extend supervised learning methods so that, instead of just having one best answer for each situation, a single correct action with all the rest being wrong, that we could somehow rate and rank all the available actions. The answer is yes! This line of reasoning brings us to reinforcement learning, or RL as it's called by those in the know.[1]

Rather than force-feeding our robot training project into a supervised learning problem, reinforcement learning fits what we are trying to do nicely. It handles sensors, actions, and rewards separately, acknowledging that they are fundamentally different in character.



The concept of reinforcement learning was born out of psychologists, mathematicians, and engineers trying to understand why humans and animals do what we do and how we can get machines to learn and behave in similar ways. One big difference between reinforcement learning and supervised learning is that in reinforcement learning the thing making the

decisions, the **agent**, interacts with the world. In supervised learning, classifying an image as having a pedestrian or not has no effect on the next image the algorithm will be asked to classify. But in reinforcement learning, every time the agent takes an action it can have a large effect on what it experiences next. If a self-driving car turns right, its cameras will see something very different than if it decides to turn left. This interdependence between the actions taken and the sensory experience is also called **interaction**. In the diagram above, it's captured as the loop in which actions are passed from the Agent to the World and sensory experiences are passed back from the World to the Agent. The ability to capture and reason about interaction is a fundamental difference between reinforcement learning and supervised learning.

Rewards

Another important difference between reinforcement learning and supervised learning is that, while there is still a privileged channel of information, in reinforcement learning it's a number rather than a category. Reward is how the agent learns how useful its actions were.

A large reward is equivalent to telling a dog "Good girl!" and giving her a treat.

Rewards can also be smaller, letting the agent know that the action they took was OK, but that another action might work better in the future.

Rewards can even be negative, in which case they are actually punishments. Negative rewards tell the agent that what they did was not good and please don't do it again.

By being able to handle rewards of varying magnitudes and signs, reinforcement learning can observe and learn subtle distinctions between actions. Supervised learning would be hard-pressed to do the same.

Labels vs. Rewards

Reinforcement learning promises to solve a longstanding problem in supervised learning, namely that human generated labels are hard to get.

Consider our pedestrian/not pedestrian model. Most machine learning algorithms would require tens of thousands of labeled images to achieve good performance on this task. For each one of those labels, a human has to look at an image, make a judgment, and indicate whether it contains a pedestrian. If the image quality is poor or if the pedestrian is occluded or only partly in the frame, then the human has to spend additional time making a judgment call. Experience has shown that humans get tired of this very quickly. Even graduate students. Workarounds have been developed, including pressing people into service via captchas and even paying them. In fact, there is an entire industry built around hiring out data labelers for your company's particular machine learning use case.

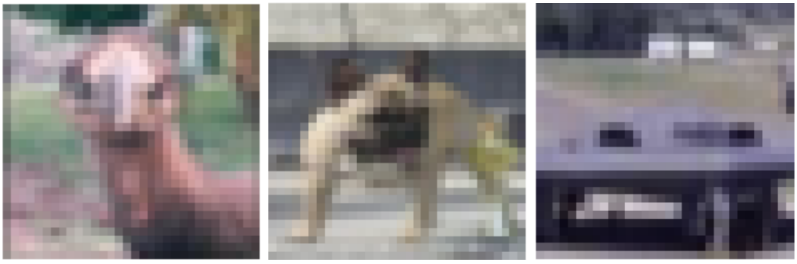
This problem is compounded when you need to learn many different categories. The broad class of "pedestrian" could be further subdivided into "cyclist," "child," "scooter," "person with baby stroller," "person walking a ferret," etc. For every single one of these classes, there would need to be a large number of labeled examples. Most of the time this means that machine learning applications are limited to a handful of pre-trained classes, because the resources to collect and label the data are simply out of reach.

This problem becomes yet more pronounced when the classification requires expert knowledge. Anyone can learn to distinguish pedestrians from non-pedestrians, but for a model intended to distinguish frogs from toads, or species of penguin,[4] the pool of people from which labels can be collected is much smaller and their time tends to be much more expensive.

Puppy or Ewok?

It is an open secret in the machine learning community that even within benchmark data sets, label quality is often low. Mysterious handwritten squiggles in the MNIST Digits data set are confidently labeled as being the number three. There are blurry pixelated images in the CIFAR-10 image data set labeled as trucks and birds that I struggle to see even after I'm told what I'm looking for. Other popular data sets have similar stories, with labels that are mysterious, or just plain wrong.

8, 1, and 6, according to MNIST handwritten digits data set.[5]



"bird", "cat", and "airplane," according to CIFAR-10 data set.[5]

Supervised learning is great in theory, but in practice its insatiable thirst for labeled data is limiting. The massive amount of human effort needed to train machine learning models takes some of the shine from their reputation as being intelligent, scalable, automated solutions to classification and prediction problems.

On its surface, reinforcement learning seems to be a solution to the need for large amounts of human hand-holding. A carefully defined reward signal encapsulates all of the minutiae of judgment that go into classifying data points. A reinforcement learning algorithm has the much simpler task of learning to maximize reward. It eliminates the need for all of those human labelers to suffer the expensive drudgery of flipping through images or audio clips and tagging them.

Imagine teaching a humanoid robot to walk. Under the paradigm of supervised learning, every moment's collection of body position and velocity, and that of every joint and limb, would be evaluated by a human and assigned an appropriate action. This would be infeasibly tedious.

However, if you set up the robot with a reinforcement learning algorithm and give it a reward for every meter of forward motion, then the algorithm sorts out the rest. It coordinates the timing of actuating a dozen interconnected joints and linkages to move that robot forward. It's an amazing thing to see. In early trials, the robot flails and flops in ineffective and

sometimes hilarious ways. But over time it figures out how to propel itself forward with increasing effectiveness.

Reward engineering

However, as you may have guessed, it's not always quite that simple. It turns out that choosing a good reward is hard. With a capital H. There's a delightful genre of RL research failures where the algorithm successfully learned to maximize the reward signal it was given, but in doing so learned a highly undesirable behavior.

Another wrinkle is that the whole process of learning relies on randomness. You can restart the same robot simulation 100 times, and find that 80 times the RL algorithm learns to move one way, 13 times another, six times yet another, and one of those hundred it learns to do something inexplicably effective.

For example, the humanoid robot doesn't always learn to run. It turns out that with slight variations in the reward signal, the robot finds that it can make faster forward progress by flinging itself head over heels and tumbling through space in what appears to be an extremely painful manner.

I got to experience this firsthand, when training a simulated 7 degree-of-freedom robot arm to pick up a virtual salt shaker. I carefully constructed a reward function that rewarded pressure in the robot's grippers (which we would expect to see when the robot grabs the salt shaker) and upward motion of the shaker

(which we would expect to see when the shaker was lifted by the robot).

I expected the robot to learn to reach through the air toward the object, grasp it, and lift it. Instead, the robot found it much easier to reach down into the table, which was spongy and slippery due to the bugs in the physics simulation I wrote, and then come up from underneath the object, grasp it, and lift it from there. The strategy was entirely incompatible with anything the robot could have done in the physical world. But it was completely in tune with the nature of the simulation I gave it and the particular reward function I chose. It is a testament to the performance of the reinforcement learning algorithm that it learned to exploit the quirky simulation physics in a way I hadn't anticipated. But it was quite unsatisfying for this new researcher to fail to make the robot do what I wanted.

Misinterpretations of reward functions are the algorithmic manifestation of Goodhart's Law: That once a metric becomes a measure of success, it ceases to be a good measure. People will hack it and game it. As an example, I once worked in a laboratory that set a top priority goal for zero accidents. Company leaders made speeches about how critical this was to the laboratory's future and how every effort should be made and no expense spared to bring it about. I don't know whether it had occurred to them that the surest way to have zero accidents on the job is for no one to do any work. Although it's possible it did occur to them, because the policies they put into place served primarily to limit the amount of work we completed.

The problem of choosing a reward function that gets the robot to settle on a behavior that you want is common and has its own name, **reward engineering**. If you're able to corner reinforcement learning researchers at a conference, you can get good war stories about their reward engineering struggles. So far, we have no principles to guide this process. It is entirely an intuition-driven exercise in trial and error. In short, it is an art.

The more complex the behavior and the more complex the robot involved, the harder it is to choose a good reward function. It's telling that reinforcement learning demonstrations that don't rely on pre-programmed information about the environment often demonstrate relatively simple behaviors. The larger the number of sensors, the larger the number of actions available, the larger the number of steps required to successfully complete the desired behavior, the harder it is for the algorithm to stumble onto a successful strategy, one deserving of reward. If you think of all of the actuator motions that need to be coordinated for a walking robot to stand, let alone take a step forward without falling over, it's surprising that these simulations produce any positive results at all.

This is called the **curse of dimensionality**. In a mathematical representation of the problem, each sensor, each action, each step in the process adds another dimension, another column to the matrix calculations that take place. The challenge of working in a larger number of dimensions is that the number of possibilities go up not linearly, not polynomially, but

exponentially. That means that by going from 10 sensors to 11, learning a good robot behavior becomes not just 10% harder or 50% harder. It might become twice as hard. This pattern drives research examples toward simple cases, where the curse has less sway. But if we are looking to train a robot dog with dozens of joints and many possible actions to sit or heel or fetch, we are going to be exposed to the curse of dimensionality in full force. Designing a good reward function will be more important and much more difficult than in any toy example.

By moving from supervised learning to reinforcement learning, we've replaced the manual effort of labeling examples with the manual effort of guessing a reward function and testing it out over and over again until we end up with a good one.

There is a running joke among machine learning practitioners that if you dig deep enough, behind every algorithm there is an army of human beings. In the case of supervised learning this can be literally true. In the case of reinforcement learning, it may be one human being, exerting heroic effort through trial and error to find the right incantation, the exact formulation of a reward function, that produces the desired behavior. Reinforcement learning exchanges one form of manual craft for another.

So far, our tour of existing machine learning methods has shown some tantalizing possibilities, but none of them seems to be a really good fit for training a robot. This leads us to the

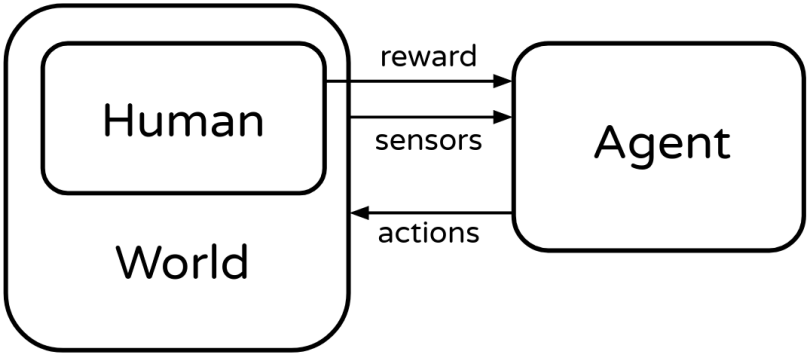
whole point of this chapter, introducing a somewhat new way of doing things that has a chance of getting us what we want.

Human-Directed Reinforcement Learning

Existing reinforcement learning methods almost without exception assume the existence of a reward signal that can be generated automatically. It might be a desired sensor value, a desired position on a map, a desired state for the robot, or some sophisticated function of these. But whatever it is, it's generated automatically.

What we want to do is different. We want to have a human trainer give the robot rewards. By hand. One at a time. It will be up to the human to decide whether any particular robot action is desirable or not, is worthy of reward or punishment, and just how strong that reward should be. This is distinct enough from existing robotics and machine learning work that we'll give it a separate name, **human-directed reinforcement learning**.

Human-directed reinforcement learning (HDRL) is just like regular reinforcement learning, except that a human trainer delivers each reward by hand. Artisanal RL, if you prefer.



There's another distinction we need to make when talking about human-directed reinforcement learning: issuing verbal commands as opposed to directly controlling the robot's actions. In industrial robotics it's common practice to train a robot by directly taking control of its motors and joints, making it do precisely what you want, and then having the robot replay that sequence of actions on command. This is effective for assembling automobiles, but it's not what we want to do here.

We want a robot we can train by speaking a command and having the robot learn the correct response through trial and error. This means that the human never has direct control of the robot agent. The human can never force the robot to take a particular action. The human's command to "sit" is simply one of many inputs that the stream the robot receives. In our project, these commands are not commands in the traditional sense at all. They're just cues to the robot; they are hints and

breadcrumbs that help the robot discover the actions that result in reward.

So have we done it? Can we train a robot using reinforcement learning with a human issuing rewards?

Sadly, no. At least not using existing methods. To be successful, we'll need to do something different than what's been done before. We're working under a severe limitation imposed by the problem we want to solve: there's a person involved.

HDRL Eliminates Reward Engineering

Human-directed reinforcement learning changes the training game. There is no reward function. There is no reward engineering. In every case a human manually observes the situation and the robot behavior and assesses its worthiness of reward.

Human directed reinforcement learning is no exception to the rule that there is always a person calling the shots behind the scenes. There is a puppeteer. There is a wizard behind the curtain pulling the levers and flipping the toggles. What HDRL does is to make this explicit. The role of the human is clear. There is no false pretense of intelligent machines going it alone.

This approach has a couple advantages over reward engineering. It removes the need to think like a machine or to imagine how every possible scenario will be reflected in the

robot's sensors. It doesn't require any domain expertise whatsoever, other than knowing what you want the robot to do.

Sample efficiency

The reinforcement learning algorithms in vogue today rely on a large amount of training information. They require a lot of repetition, many instances of choosing the best action and being rewarded for it. In the case of self-driving cars, they may need millions of examples. Even in cases where researchers try to get that number as low as possible, there are still tens of thousands of rewarded examples necessary to learn even a simple task.

When the reward is being calculated automatically from measured sensor values, this can happen hundreds of times per second. Racking up ten thousand examples can be done in minutes.[2] But when a human has to see what the robot is doing, decide how reward-worthy it is, and hit a key to generate a reward, this process slows way down. And when you add in the time for the human to issue verbal commands in each iteration, this process gets slower still. For a human to train a physical robot to do something useful using existing methods, it could take many lifetimes.[3]

It's possible to cheat and pre-program some basic behaviors and knowledge about the world and shorten that time considerably, but that is not what we want to do here, either. (That is called model-based RL and a lot of good robotics work has been done with it, most notably the captivating demonstrations from Boston Dynamics.[6]) This is a way to show some cool-looking

results quickly, but every pre-programmed shortcut we build in would narrow the field of tasks that could be learned and the types of robots that could learn to do them. We are aiming very high, for a solution that is as general as possible.

We want to make it so that a human we've never met using a robot we've never seen can teach it a task we've never thought of. We won't have the luxury of pre-programming shortcuts to speed this up. We won't get to subtly bake in things that we know about how the robot works or how to get around its environment. We won't be able to bias how the robot interprets the world or how it executes actions to make some tasks come naturally. We are trying to cover a very broad family of cases, and in so doing, we are forcing ourselves to play in expert mode.

It's true, we haven't yet solved the problem of needing lots of examples of reward. That's coming in future chapters. But we have placed that problem front and center. We've acknowledged that a human being will be behind every single reward assigned to this robot. Because the robot's behavior can't be known ahead of time, that human needs to be present or able to observe the robot in some way. This will be a time-intensive activity, and it is of the highest importance that it be efficient – that the algorithm learn acceptable behaviors as quickly as possible, so as to avoid taxing its human trainer. **Sample efficiency** will be our most stringent design constraint.

What could possibly go wrong?

If this approach turns out to be fruitful, human-directed reinforcement learning could end up being a very powerful tool, and it is a sad truth that any powerful tool can be used to help or harm. Wheels make both ambulances and military vehicles possible. A scalpel can be used to save a life or to take it. The difference is the intent of the user. Human-directed reinforcement learning would be no exception. We've already seen harmful machine learning applications, such as deepfakes, bigoted language models, and predictive policing tools which are horrible in every incarnation, but perform particularly poorly for members of already disadvantaged populations.

Should human-directed reinforcement learning ever realize its potential, thoughtful regulation will be absolutely necessary for preventing harm. One thing we can do to stack the deck in favor of minimizing harm is to design the algorithm from the bones out to be accessible to anyone and everyone with an interest. This ensures that its power won't be concentrated in the hands of a few corporations or states. Broad access and understanding will force conversations about transparency and accountability on a compressed timescale. These are the strongest protections we have for mitigating potential harms.

What could go right?

By taking on the challenge of HDRL, we are setting the bar pretty high for ourselves, but it's worth it. Being able to train new robots on new tasks from scratch will open up possibilities

that were never available before. We are so used to thinking about robots that are either pre-programmed to do a narrow set of tasks, or are trained by an army of researchers using warehouses full of computers, that it is difficult at first to imagine all the new avenues this opens. The ability for an individual hobbyist or researcher to train a robot on whatever task they choose would be game changing.

Throughout history, we have trained animals to do things for us that are either too tedious, too time-consuming, too dangerous, or beyond the reach of our capabilities entirely. Occasionally this is a mutually beneficial arrangement, but more often it comes at the expense of the animals' health and well-being. Having robots with the same abilities would keep those animals out of harm's way. It would be a very good thing to train robots to replace explosive-sniffing rats and the dogs who search for survivors of disasters, braving the aftermath of avalanches and collapsed buildings.

Robots also introduce the possibility of extending these capabilities far beyond what nature intended. What if a guide dog could also hear radio waves, see in the infrared, and read street signs? Or picture a flock of airborne drones that could pool their inputs to quickly search a large area for a lost hiker or for a hard to spot member of an endangered species. A legged environmental remediation robot could be trained to detect and track down very specific sources of chemical and radiological contamination.

What makes these use cases unattainable right now is that there aren't enough resources dedicated to accumulating the massive collection of training data needed to build and test them. In some cases the data required may not even exist or be prohibitively expensive to obtain. The goal of human-directed reinforcement learning is to enable training on match sparser and smaller data sets. It's hard to overstate how dramatically this would expand the set of use cases for which robots could be trained.

There's no reason the approach needs to be confined to mobile electromechanical devices. A smart home with environmental control looks just like a robot as far as the algorithm is concerned. It could be trained to maintain and adapt the temperature and humidity room by room based on the presence and activities of their occupants. They could even learn to coordinate the activities of appliances and cleaning robots the same way.

And although there is no substitute for puppy snuggles, a little bit of interaction and responsiveness goes a long way toward taking the rough edges off loneliness and stress. Just think about how comforting it is to be ignored by your cat. An HDRL powered robot could provide a similar level of companionship without the expense of litter and kibble. As a bonus, it could also be trained to roam about a child's play room and put abandoned Legos back in their bin.

What's next?

We've reached an exciting point in our journey. Now is when we step to the frontier, the boundary between problems that are solved and problems that are still looking for a solution.

The next step is to start developing a human-directed reinforcement learning method that can accommodate the severe constraints imposed by relying on a human. The next few chapters are going to kick off a very detailed walk through the development process, algorithmic methods, and code for this.

As we should expect from something this big, it's going to be a long road. And to be honest, we might not get all the way there. But the prize at the end is large, and the journey is guaranteed to be fun.

Recap

We're looking for a way to train a robot with verbal commands and rewards the way we would train a dog. The current collection of popular machine learning methods can't do this yet.

Unsupervised learning finds groups and sequences, but isn't suited to learning the logic of taking actions.

Supervised learning can learn to classify situations according to the action they call for, but does so slowly and clumsily.

Reinforcement learning is a good model for training a robot. It handles sensors, actions, and rewards just the way we want. However, existing methods need too many rewarded examples to be practical.

Human directed reinforcement learning is what we've named a variant of reinforcement learning where a human is manually providing all the rewards. They may also be providing cues or interacting with the robot.

If successful, HDRL would open up a lot of possibilities for using robots and automation to get things done.

Resources

1. The original and authoritative reference on reinforcement learning.
Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, Second Edition*. MIT Press, Cambridge, MA, 2018
Website: http://incompleteideas.net/book/the-book-2nd.html
Full PDF: http://incompleteideas.net/book/RLbook2020.pdf

2. A rare example of a physical robot without pre-programmed knowledge about its environment (model-free) learning a useful behavior in a small amount of time. The reward signal is automatic here, not human generated, so it doesn't quite meet our needs, but it's a tour de force regardless.
Laura Smith, Ilya Kostrikov, Sergey Levine. "A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning"
arXiv:2208.07860v1 [cs.RO] 16 Aug 2022
Blog: https://sites.google.com/berkeley.edu/walk-in-the-park
Paper: https://arxiv.org/abs/2208.07860

3. A more nuanced and detailed discussion of all the reasons reinforcement learning is hard.
Alex Irpan. "Deep Reinforcement Learning Doesn't Work Yet" 2018
Blog: https://www.alexirpan.com/2018/02/14/rl-hard.html

4. There really is a dataset full of penguins, delightful in its organization and its presentation. It's worth a look, if only to enjoy the custom illustrations.
Allison Marie Horst, Alison Presmanes Hill, Kristen B Gorman.
"palmerpenguins: Palmer Archipelago (Antarctica) penguin data," R package version 0.1.0. doi:10.5281/zenodo.3960218 2020
Blog: https://allisonhorst.github.io/palmerpenguins/

5. There is a well-designed display of questionable labels in popular benchmark data sets, and a paper that goes with it explaining the methodology and significance. This team marked the cracks in the foundations of ML research methods with bright orange spray paint.

Curtis G. Northcutt, Anish Athalye, Jonas Mueller. "Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks" Proceedings of the 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks. Dec 2021
Website of examples: https://labelerrors.com/
Paper: https://arxiv.org/pdf/2103.14749.pdf

6. Boston Dynamics is a robotics company that grew out of research from the MIT Leg Lab. Their crowning achievement is **Atlas**, a humanoid robot that performs truly impressive feats of coordination and choreography that were firmly in the realm of science fiction 10 years ago.
Website: https://www.bostondynamics.com/

About the Author

Robots made their way into Brandon's imagination while he watched The Empire Strikes Back as a child in a packed and darkened movie theater, and they never left. He went on to study robots and their ways at MIT and has been puzzling over them ever since. His lifetime goal is to make a robot as smart as his dog.

To see more of his work, visit brandonrohrer.com