Artificial Intelligence: Reinforcement Learning in Python

By: The LazyProgrammer

Reinforcement Learning

- When people talk about AI, they don't usually think about supervised/unsupervised learning
- These seem trivial compared to what we want an AI to do: play Chess/Go, drive cars, beat video games at superhuman level
- RL does all that and more
- Like Deep Learning, a lot of theory developed in 70s/80s, but not popular until now
- Now AIs can play games like Doom and Super Mario
- Law of accelerating returns: this improvement will grow exponentially!

Reinforcement Learning

- Supervised/Unsupervised Learning is not easy! I have 16+ courses on them already!
- Yet reinforcement learning opens up a whole new world
- RL paradigm is more different from Supervised/Unsupervised than they are from each other

What's in this course?

- In RL: huge gap between theory and code
- Theory is very abstract, important to continually think about how it applies to AI tasks like controlling a robot / playing a game
- So we start the course with some games:
- Slot machines
- Picks up where my course on Bayesian ML (A/B testing) left off
- Explore-exploit dilemma
- Epsilon-greedy, UCB1, Bayesian methods
- We can go beyond A/B testing, e-commerce stores, ads → how to behave in the real world

What's in this course?

- Tic-tac-toe
- How would you code an agent to play tic-tac-toe as a first-year comp-sci student?
- We'll introduce the RL way of doing things

What's in this course?

- Markov Decision Process (MDP)
- 3 techniques for solving MDPs:
  - Dynamic Programming (DP)
  - Monte Carlo (MC)
  - Temporal Difference Learning (TD)
- They all look very similar. The hard part is differentiating these 3!

What's in this course?

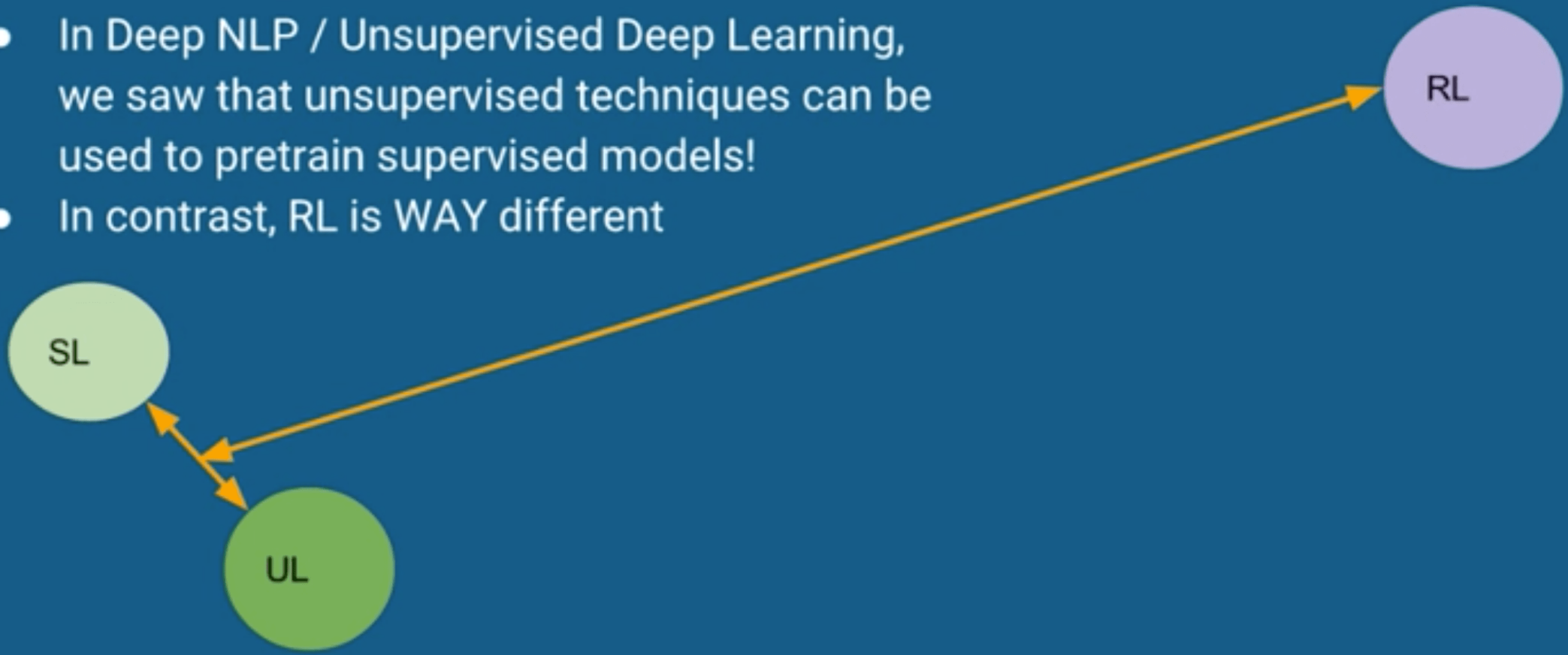
- Last section: Approximation methods
- Is a teaser into Deep Reinforcement Learning
- 3 solutions in this course won't scale to large problems, we need to modify them
- What's good for function approximation? Deep learning!
- This course will show you where you can "plug-in" a deep neural network, or any other approximator
- Without this course, Deep RL won't make much sense
- My original idea was to include these prerequisites into Deep RL, but there was just too much info!

What is reinforcement learning?

- What is reinforcement learning?
- How is it different from supervised and unsupervised learning?
- What are its applications?

What is reinforcement learning?

- In Deep NLP / Unsupervised Deep Learning, we saw that unsupervised techniques can be used to pretrain supervised models!
- In contrast, RL is WAY different



What is reinforcement learning?

- In supervised/unsupervised ML, we always imagine the same interface:

```
class SupervisedModel:
  def fit(X, Y): …
  def predict(X): …

class UnsupervisedModel:
  def fit(X): …
  def transform(X): … // PCA, Autoencoders, RBMs
                      // K-Means, GMM, …don't really transform data
```

What is reinforcement learning?

- Common theme with Unsupervised/Supervised Learning:
- The interface is the training data (matrices of numbers)
- In supervised learning we can make predictions on unseen/future data
- Useful by itself:
- Ex. app where you can take a photo at the grocery store, it classifies the item and shows its nutritional information

What is reinforcement learning?

- RL is different
- RL can guide an agent on how to act in the real world
- Interface is much more broad than training vectors, it's the entire environment
- Environment can be real world or simulated world (video game)
- Ex. real world: robot that vacuums your house, robot that learns to walk
- Military is interested in this tech: RL agents can replace soldiers. Not just walk, but fight, diffuse bombs, make important decisions

What is reinforcement learning?

- RL agents also train in a completely different way
- Many references to psychology
- Agents are used to model animal behavior
- Objective is a goal
- Supervised objective: maximize accuracy or likelihood / minimize cost
- RL agents get feedback as they interact with environment
- Feedback signals (rewards) are automatically given to the agent by the environment
- vs. Supervised learning: labels need to be made by humans (time-consuming and costly)

What is reinforcement learning?

- Phrasing objective in terms of goal allows us to solve a wide variety of problems
- AlphaGo's goal is to win Go
- Goal of video game AI: win the game / get highest score
- What about animals and humans?
- "Selfish Gene" theory (Richard Dawkins)
- AlphaGo found unique ways of winning: unusual and unexpected by observers
- Does that apply to animals/humans?

What is reinforcement learning?

- Why do we do anything we do?
- Selfish Gene theory says everything we do is designed to serve our genes' desire to multiply
- Ex. Why do people want to be rich?
- Perhaps led to better healthcare or social status, led to genes maximizing their goal
- Richness has no physical relationship to genes, yet it's a novel solution to the problem

What is reinforcement learning?

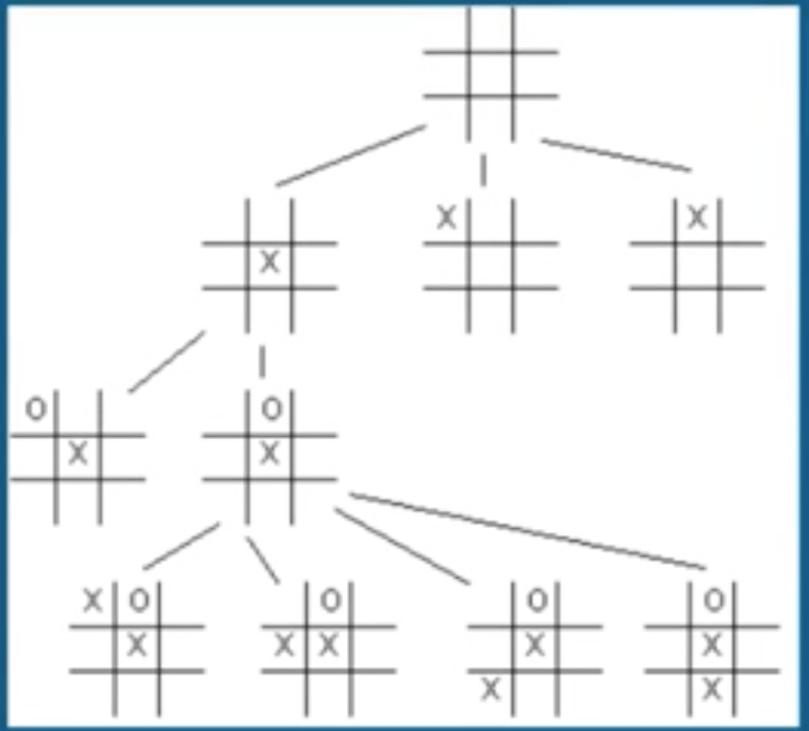
- “Desiring money” just a random example
- Replace with anything you like: “being healthy and strong”, “having good analytic skills”
- Studying those factors is a social scientist’s job
- For us, it’s more interesting that there is just one main objective to maximize but various novel ways to achieve it
- At one point in history, seeking as much sugar as possible would give you energy (hence improving survival). Today, it kills you!
- Genes use mutation and natural selection
- AI uses reinforcement learning

What is reinforcement learning?

- Humans / AIs alike never sense the entire world/universe at once
- We have sensors which feed signals to our brain from the environment
- We don’t even know everything that’s going on in a room
- Thus the sensors limit the amount of information we get
- The measurements we get from these sensors (e.g. sight, sound, touch) make up a “state”
- We’ll only discuss finite state spaces
- State spaces with an infinite number of states are possible too

What is reinforcement learning?

- If you don't know how tic-tac-toe works, do a google search, you can play right in the browser!
- What's the # of states?
  [if we simplify the problem so that
  we can keep adding x's and o's
  even after a player gets 3 in a row]



What is reinforcement learning?

- Each location on the board has 3 possibilities: empty, X, O
- 9 locations on the board
- # states = 3x3x...x3 = 3^9

Recap so far

- 3 important terms (I may have snuck in a few more)

Agent: thing that senses the environment, thing we're trying to code intelligence/learning into.

Environment: Real world or simulated world that the agent lives in.

State: Different configurations of the environment that the agent can sense.

What is reinforcement learning?

- Another concept: “Rewards”
- This is what differentiates RL from other types of ML
- An agent not only tries to maximize its immediate reward, but future rewards as well
- RL algorithms will find novel ways of accomplishing this
- AlphaGo: learned unique / unpredictable strategies that led to beating a world champion
- Not intuitive to humans, but RL can figure it out

Unintended consequences

- Possible danger of RL: unintended consequences
- Commonly repeated idea: AI could wipe out humanity if it decides that's the best thing for us
- Ex. Minimize human deaths
- AI decides that since # humans grows exponentially, that more people will die in the future, then best to destroy everyone now to minimize dying in the future

Unintended consequences

- Lower level example: robot trying to solve a maze
- Reasonable goal: solve the maze
- Reward = 1 if solved, reward = 0 if not solved
- Possible solution: move randomly until maze is solved

Unintended consequences

- Lower level example: robot trying to solve a maze
- Reasonable goal: solve the maze
- Reward = 1 if solved, reward = 0 if not solved
- Possible solution: move randomly until maze is solved
- Is that a good strategy? No!
- We never told the AI that it needs to solve the maze efficiently (we always get the reward in the end)
- What about this: Reward of -1 for every step taken
- In order to maximize total reward, must minimize # steps
- Note: reward is always a real number

Terms

So far: agent, environment, state, reward

Next: actions

Actions are what an agent does in its environment.

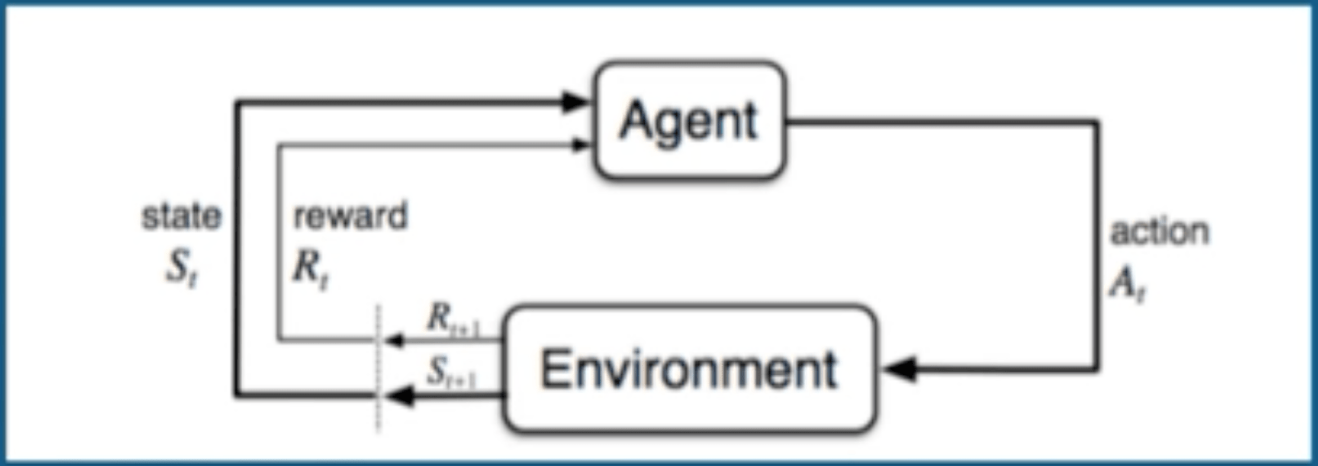
Ex. agent = a 2-D video game character. Actions = { up, down, left, right, jump }

We look at finite sets of actions only.

SAR triples

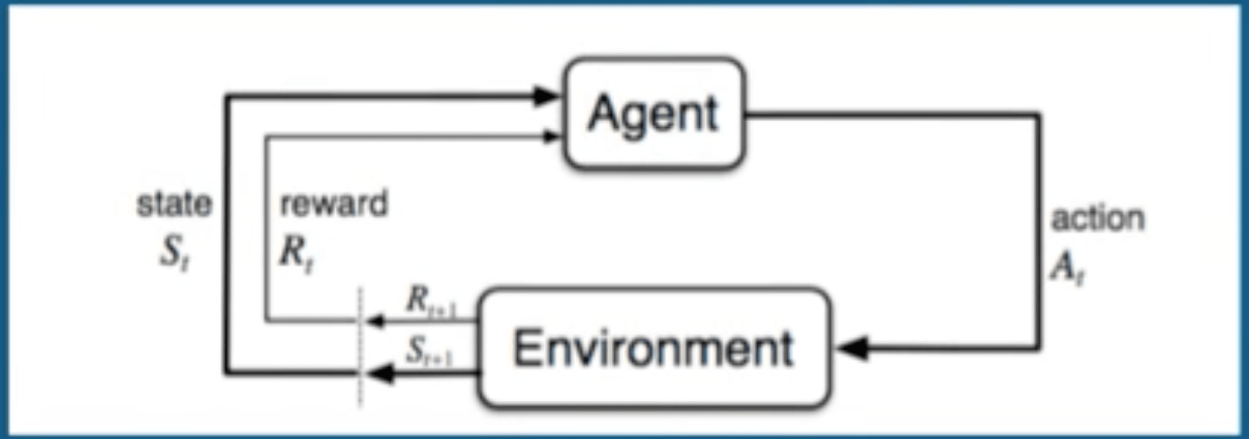
We often think about (state, action, reward) as a triple

Notation: (s, a, r)



Timing

- Timing is important in RL
- Every game is a sequence of states, actions, rewards
- Convention: Start in state S(t), take action A(t), receive a reward of R(t+1)
- Reward always results from (s, a) you took at previous time
- S(t), A(t) also brings you to a new state, S(t+1)
- This also makes a triple:
- [S(t), A(t), S(t+1)]
- Also denoted as:
- (s, a, s')



What is reinforcement learning?

- Summary
- Program the agent to be intelligent
- Agent interacts with its environment by being in a state, taking action based on that state, which brings it to a new state
- Environment gives the agent a reward, can be +ve or -ve (but must be a number)
- Reward is received in next state

Where to get the code

https://github.com/lazyprogrammer/machine_learning_examples

Folder: rl

```
git clone git@github.com:lazyprogrammer/machine_learning_examples.git
```

- [don't need type to it out manually, copy & paste from github]
- Use SSH url so you don't need to type your Github password every time
- Run "git pull" periodically so you always have the latest version
- Data will be mentioned in relevant lecture, if needed
- Check environment setup lecture in appendix if you need help installing libs

Code by yourself first

- Code by yourself first!
- Github code has already been verified to work, you can always check your work against that
- Common question: “What are the practice activities?”
- Implementing the algorithms you learned about!
- Theory will give you the algorithm at a high level
- You want to be able to understand it so well you can implement them without any reference
- That’s what being a “computer scientist” is
- If you try to watch the videos straight through, you will not retain anything

Code by yourself first

- Especially important in reinforcement learning
- Why?
- All the algorithms look the same
- You might wonder “why did the instructor do the same thing 3 times?”
- But if you are proactive enough to write the code, you will understand the differences
- Struggle through the details
- Spend endless nights debugging
- This is what makes a good coder

How to maximize your returns

- No exams or grades
- Be the kind of person that takes initiative and is proactive about learning
- This means:
- Understanding all the equations that get written down
- Being able to do derivations independently
- Writing the code yourself before looking at my code
- Ask questions on the discussion board! Stupid questions are ok!

How to maximize your returns

- ML isn’t like physics or other school-type course
- No 20 practice problems at the end of a chapter for you to do
- Nice thing about ML:
- It automatically self-checks
- Your code runs, or not
- Your code behaves as expected, or not

How to maximize your returns

- Be a good debugger, don’t give up the instant you get an error
- People think spending 1 hour debugging is too much
- Try all night or multiple days
- Strive to make things work on your own, check vs. Github, your code can even be better than mine!

How to maximize your returns

- Write as much of the code on your own as you can, ideally only using pseudocode from lectures
- As an example:
- In supervised ML, we covered Naive Bayes, KNN, Decision Trees, Perceptron
- These are clearly 4 distinct algorithms
- This course: Everything looks the same
- The real challenge will be telling them apart
- Understanding differences between each method is key to this course

How to maximize your returns

- You've heard me say in previous courses "all data is the same"
- Easy to achieve this level of perspective in ML
- Supervised/Unsupervised learning: we are always dealing with inputs X and targets Y
- Don't need to specify what X or Y is, you still understand
- You already know X can represent any feature: age/weight/height/number of pages visited in your store/DNA sequence, etc.
- Works in all different fields: finance, bio, advertising, etc.
- We don't need to specify those things, saying "X and Y" is good enough
- Does this apply to RL?

How to maximize your returns

- Does “all data is the same” apply to RL?
- Eventually I believe it will, but probably not at first
- RL is more difficult, interface to the agent is much more broad than just features + targets, Xs + Ys
- Once you get used to environments, states, actions, rewards, … you’ll get better at abstracting these

How to maximize your returns

- In Deep Learning especially, experience has shown me many people want to jump into it because it's popular
- But they have no idea about prerequisites
- They "don't know what they don't know"
- # of messages I get saying "it's too hard" / "too much math" / "why can't I program a self-driving car now?" has decreased to almost 0

How to maximize your returns

With RL especially, it's a big jump in knowledge to get to the "interesting" stuff. I have 2 goals. You:

1) "Know what you don't know"

2) "Know what you need to know in order to know the things you want to know"

If you think you'll program an AI to play Super Mario tomorrow, you currently "don't know what you don't know". By the end of this course, you'll have a better idea of what is required.