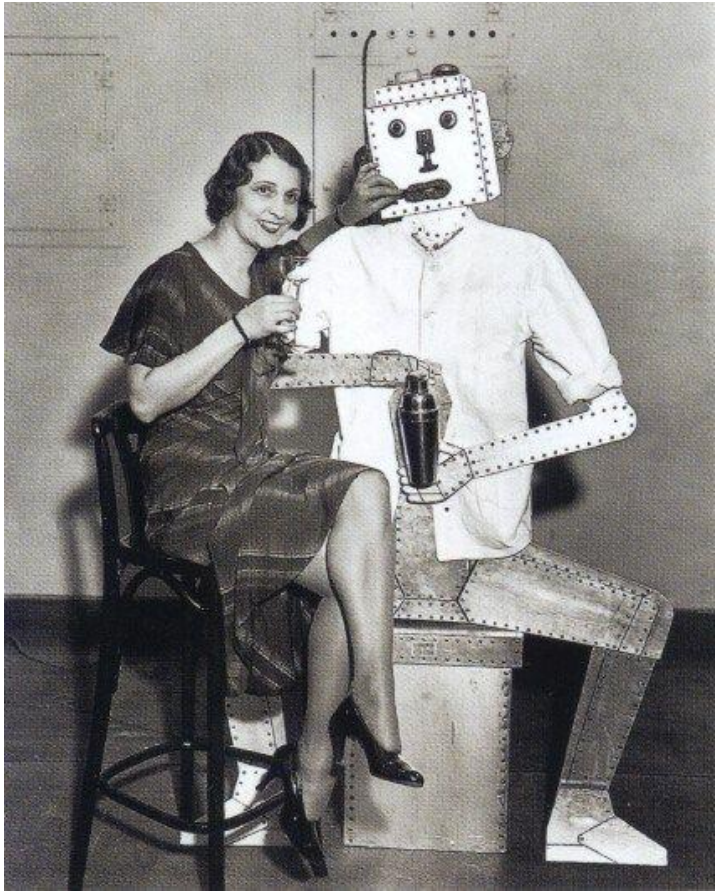
Intro to RL in Julia

RL == Reinforcement Learning (not Rugby League)

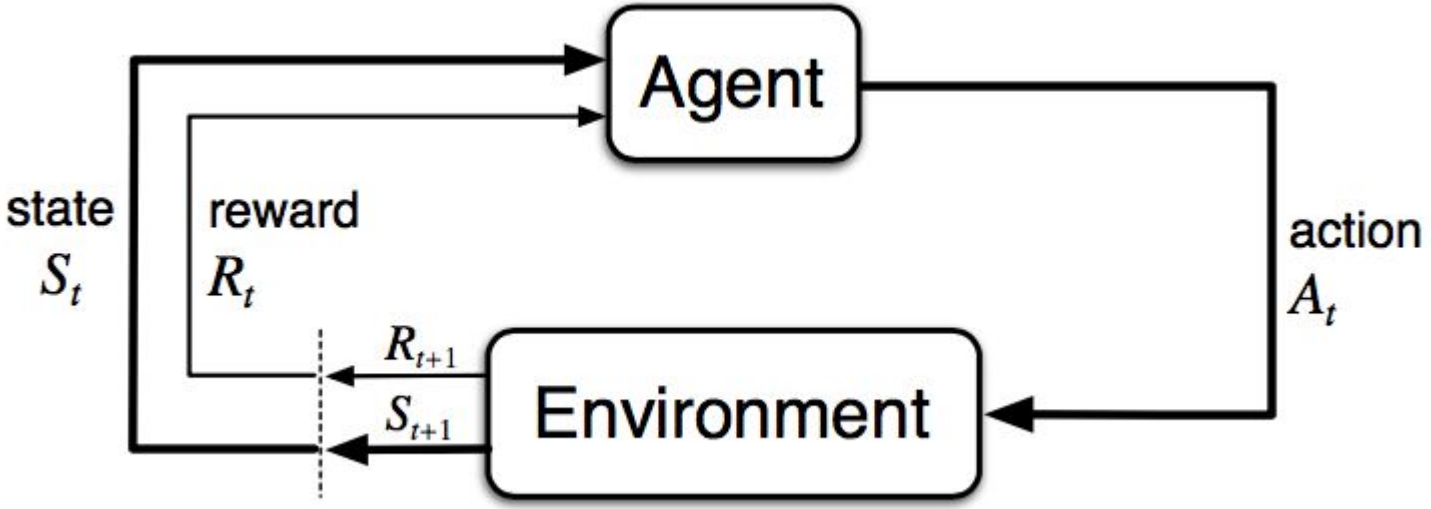
In the beginning...

Programmers




IBM 7090

Reinforcement Learning



Key Features of RL

- Agent interacting with an environment (Trial and Error)
  - sequence of states, actions, rewards

Key Features of RL

- Agent interacting with an environment (Trial and Error)
  - sequence of states, actions, rewards
- Feedback not necessarily immediate (Delayed Rewards)

Key Features of RL

- Agent interacting with an environment (Trial and Error)
  - sequence of states, actions, rewards
- Feedback not necessarily immediate (Delayed Rewards)
- Vs Supervised ML:
  - single decision
  - immediate/“dense” feedback

Key Features of RL

- Agent interacting with an environment (Trial and Error)
  - sequence of states, actions, rewards
- Feedback not necessarily immediate (Delayed Rewards)
- Vs Supervised ML:
  - single decision
  - immediate/“dense” feedback
- Vs Unsupervised ML: there is *some* feedback

Examples where RL is used

- **Robotics**
  - Walking
  - Picking things up, Pouring a coffee, Making a sandwich
  - State: sensory input
  - Action: torques to apply to motors
- **A/B Testing**
  - State: URL, search history, age, interests, etc.
  - Action: which ad to show, where to place it

Where RL is used

- **Data Center Cooling*:**
  - State: temperatures, power, pump speeds, etc.
  - Action: where to pump coolant
- **Taxi/Ride Share Dispatch**
  - State: locations of taxis and riders
  - Action: dispatch taxi X to location Y
- **Game Playing AI**: AlphaGo/Zero, Backgammon, Atari Games, DOTA2, StarCraft2

https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/

Key Idea: Value & Credit Assignment

- Rewards are often delayed, so reward in next timestep != value of action
  - I.e. hard to know which action(s) to credit for a particular reward
  - E.g. hitting a good shot in tennis - don’t score immediately
  - Extreme case: decisive move in chess, reward only received at end of game
- Key component of RL methods is estimating the **Value** of an action
- Defined as
  - V(s): Sum of (expected) future rewards starting in state s
  - Q(s,a): Sum of (expected) future rewards starting in state s, taking action a
  - (and then continuing with current policy)

Model Based vs Model Free

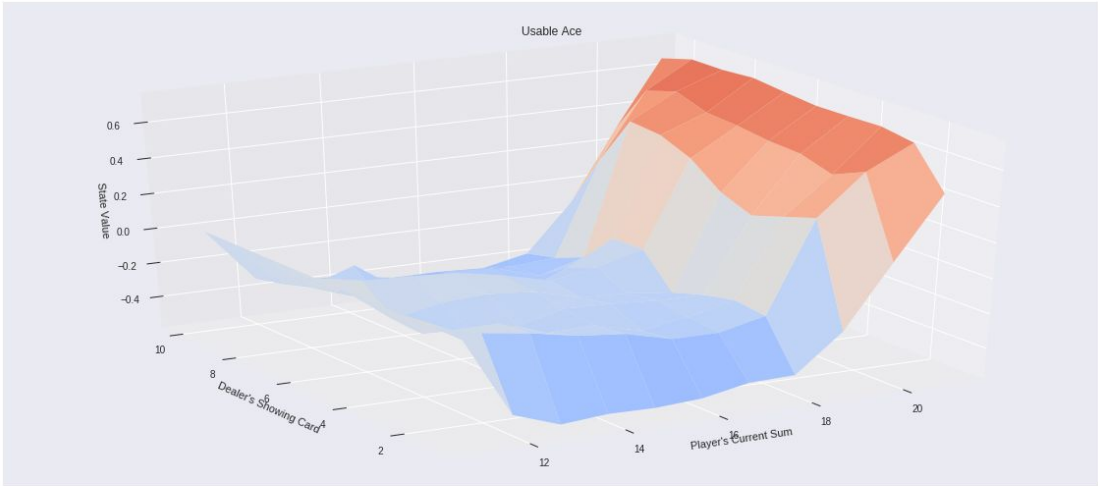
- Model Free
  - Learn value of states, or state, action pairs
- Model Based
  - explicit environment model
  - model errors accumulate
  - less explored, much potential

Tabular RL

Table storing, for every state, action pair (e.g. all possible player/dealer hand combinations in blackjack):

- Action to take in that state (hit, stay, double, split)
- (Expected) Value of each state: (prob of winning * amount that will be won) - stake

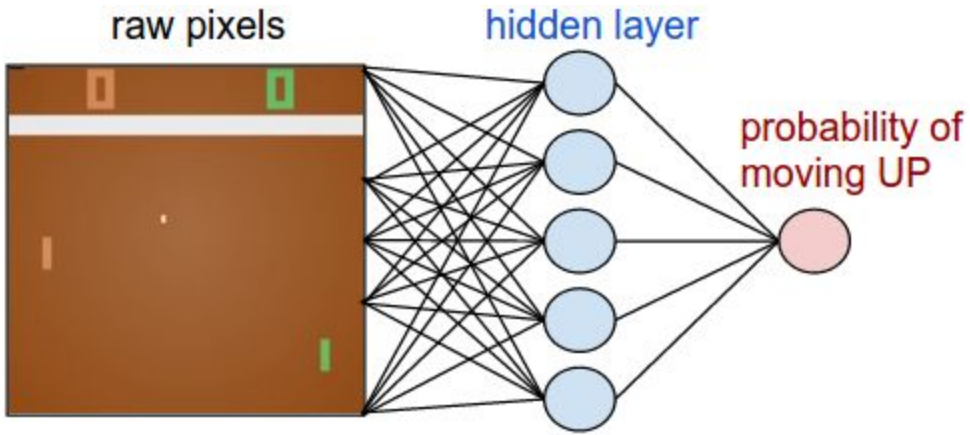
Value Iteration: run a lot of simulations, being sure to try all actions in each state - will give you a “monte carlo” estimate of value.


Usable Ace
State Value
0.6
0.4
0.2
0.0
-0.2
-0.4
Dealer's Showing Card
10
8
6
4
2
Player's Current Sum
12
14
16
18
20

Tabular RL

There are other algorithms that are more efficient (than Monte-Carlo Value Iteration), but they’re not generally scalable to large state/action spaces.

Deep RL



Limitations and Research Frontiers

- Simulators required
  - sample inefficient
  - unsafe
- Exploration (vs Exploitation)
  - Intrinsic reward - e.g. curiosity based
- Model Based
- Low level control
- Multi-task learning
- ...

Why Julia

- Speed: https://github.com/JuliaML/OpenAIGym.jl/pull/13
  - FFI: use C, Python libs easily and performantly
  - C++ not as straightforward but do-able and improving
- Same language for research and deployment
- Distributed/Concurrent/Parallel execution in the stdlib
  - Great GPU support: CuArrays, CUDAnative
- Package ecosystem:
  - JuliaDiffEq.jl
  - Optim.jl and other Optim packages (NLOpt.jl, BlackBoxOptim.jl, ...)
  - Distributions.jl - Probability distributions
  - Enabled by: multiple dispatch and AbstractArray

Julia Reinforcement Learning

(Code) https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl

(Docs) https://juliareinforcementlearning.github.io/ReinforcementLearning.jl/latest/

RL in Julia Roadmap

- ReinforcementLearning.jl v0.3.0 - see https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl/issues/24
- Distributed Execution
- More environments
  - Integrate with JuliaRobotics Stack
    - Faster simulations
    - Differentiable environments with Zygote
  - Wrap more environments with C/C++

Resources

- Julia RLish stuff:
  - https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl
  - Julia Robotics talk @ JuliaCon 2018 (not RL, but related + we hope to integrate as RL envs) https://www.youtube.com/watch?v=dmWQtI3DFFo
- Recommended RL reading/resources:
  - http://karpathy.github.io/2016/05/31/rl/
  - David Silver (DeepMind) Deep RL lectures: https://www.youtube.com/playlist?list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ
  - OpenAI spinning up in Deep RL: https://spinningup.openai.com (project idea: translate this into Julia)
  - *The* RL Textbook: http://incompleteideas.net/book/the-book-2nd.html