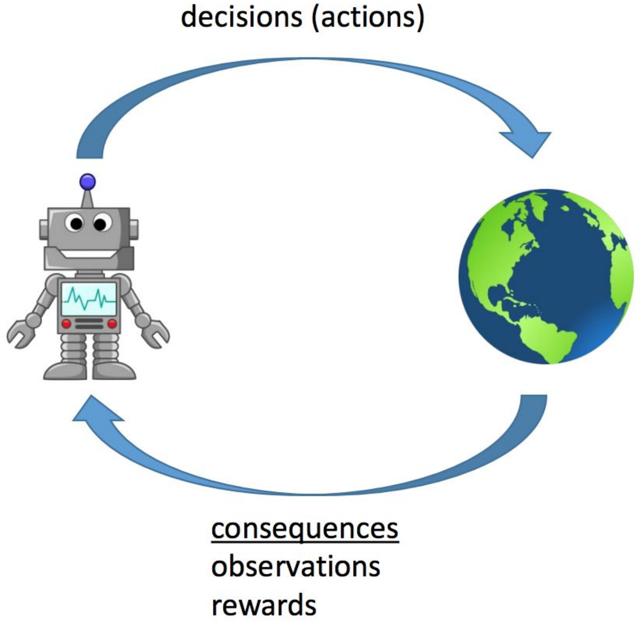
A short primer on RL

Aravind Rajeswaran

03/05/2018

What is Reinforcement Learning?



Some examples

Actions: muscle contractions
Observations: sight, smell
Rewards: food

Actions: motor current or torque
Observations: camera images
Rewards: task success measure (e.g., running speed)

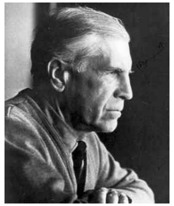
Actions: what to purchase
Observations: inventory levels
Rewards: profit

Which Reinforcement Learning?

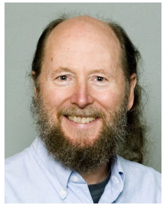
- Reinforcement Learning the "problem statement"
  Same as optimal control and dynamic programming
  decision making under uncertainty

- Reinforcement Learning the "method"
  Synonymous with "model-free" RL
  Techniques that can work with only sampling access

- Model-based RL = System ID + optimal control (in a loop)


Lev Pontryagin

Richard Bellman



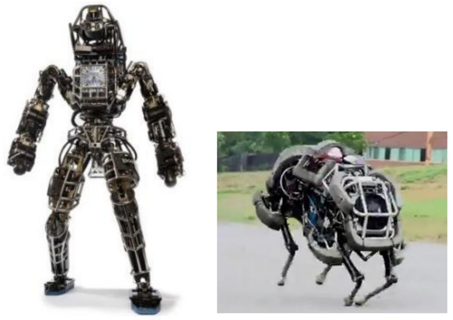
Richard Sutton and Andrew Barto

Why study Reinforcement Learning?

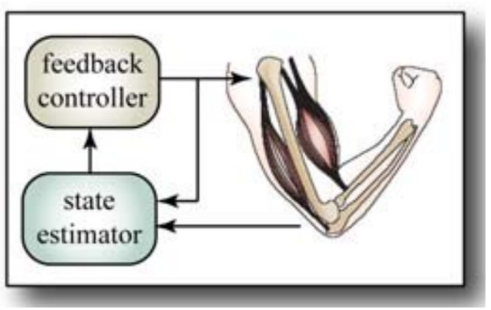
- **Algorithmic motivation:**
  Automatic iterative optimization based approach for finding near-optimal decision making rules

- **Scientific:**
  Understand how the brain makes decisions,
  help understand and address behavioral disorders

- **Applications:**
  Robotics, NLP, operations research, ad placement



Dynamic walking robots

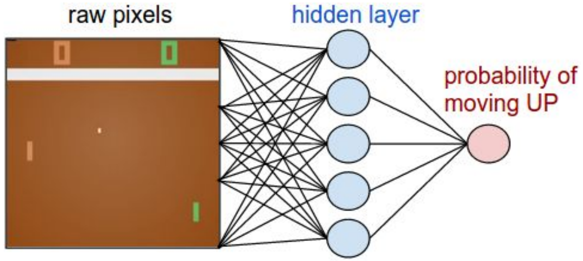


Optimal control in biology

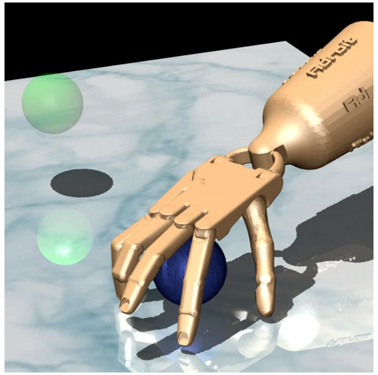
Some deep RL success stories

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO

AlphaGo:
Supervised learning + policy gradients + value functions + Monte Carlo tree search



Playing Atari Games:
Q-learning with convolutional networks



Dexterous manipulation:
Policy gradients + human demonstrations

Control from vision:
Guided Policy Search

Notations

- $s \in \mathcal{S}$ = state/observation of the world (e.g. object and robot positions/pose)
- $a \in \mathcal{A}$ = actions taken by the agent (e.g. motor torques at low level, turn steering left/right, take route A vs route B to airport etc.)
- $P(s'|s,a)$ = dynamics of the world
- $r(s,a)$ = immediate reward for choosing action a in state s
- $\pi(a|s)$ = policy or decision making rule – tells us what to do in every state. The optimization problem of interest is find ($r_t \equiv r(s_t, a_t)$):

$$\pi^*(a|s) = \text{argmax}_\pi \mathbb{E}\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots\right]$$

Goal: find "near-optimal" policy $\pi^*(a|s)$ which maximizes the long term reward.

Notations

- $Q^{\pi}(s,a)$: a function that summarizes long term reward for choosing a in s. Future actions will be taken according to policy π.

$$Q^{\pi}(s,a) = \mathbb{E}_{a_t \sim \pi(.|s_t)} \left[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \mid s_0 = s, a_0 = a \right]$$

- $V^{\pi}(s) = \mathbb{E}_{a \sim \pi(.|s)} Q(s,a)$ summarizes how good a state is under current policy

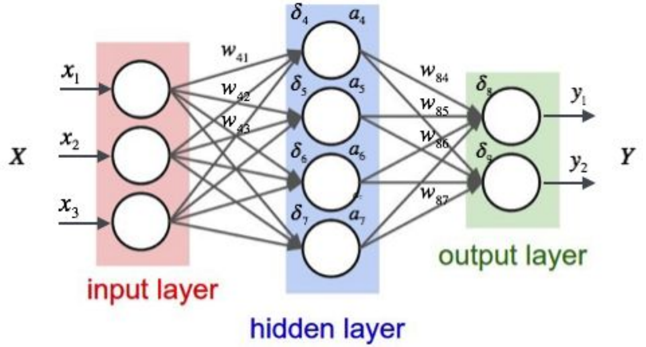
Taxonomy of RL methods

- **Model-free RL:** dynamics model, i.e. $P(s'|s,a)$, is not available and/or is not explicitly learned for solving the RL problem.
- **Model-based RL:** either model is given to us apriori (called planning) or is learned by the algorithm for the purposes of optimizing the policy.

Model-free RL

- **Direct policy search:** Pick a parameterized policy (e.g. neural network). Optimize these parameters directly to maximize long term performance.

$$\eta(\pi_\theta) = \mathbb{E}_{P, a \sim \pi_\theta(.|s)}\left[r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + \dots\right]$$



- **Indirect methods:** Pick a parameterized function approximator for the Q-function, i.e. Q_w and try to learn Q^{π^*}. Policy is the implicitly defined: either $\pi(.|s) \propto \exp(Q_w(s,a))$ (or) deterministic $\pi(s) = \mathrm{argmax}_a Q_w(s,a)$

Model-based RL

- **Optimize with given model:** Classical methods are value and policy iteration. Applications typically in operations research – scheduling, routing, finding optimal routes on maps etc.

- **Optimize with a learned model:** trajectory optimization methods like iLQG, CIO, as well as "sampling" based methods like A^* and MPPI.

Policy Gradient

We first derive the score function gradient estimator which forms the core of many policy gradient methods.

$$\eta(\pi_\theta) = \mathbb{E}_{P, a \sim \pi_\theta(.|s)} \left[r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + \ldots \right]$$

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x \sim h(x|\theta)}[f(x)] &= \nabla_\theta \int dx \ h(x|\theta) f(x) && (1) \\
&= \int dx \ \nabla_\theta h(x|\theta) f(x) && (2) \\
&= \int dx \ \frac{h(x|\theta)}{h(x|\theta)} \nabla_\theta h(x|\theta) f(x) && (3) \\
&= \int dx \ h(x|\theta) \nabla_\theta \ln h(x|\theta) f(x) && (4) \\
&= \mathbb{E}_{x \sim h(\theta)} [f(x) \ln h(x|\theta)] && (5)
\end{aligned}$$

REINFORCE: Monte Carlo estimate using score function

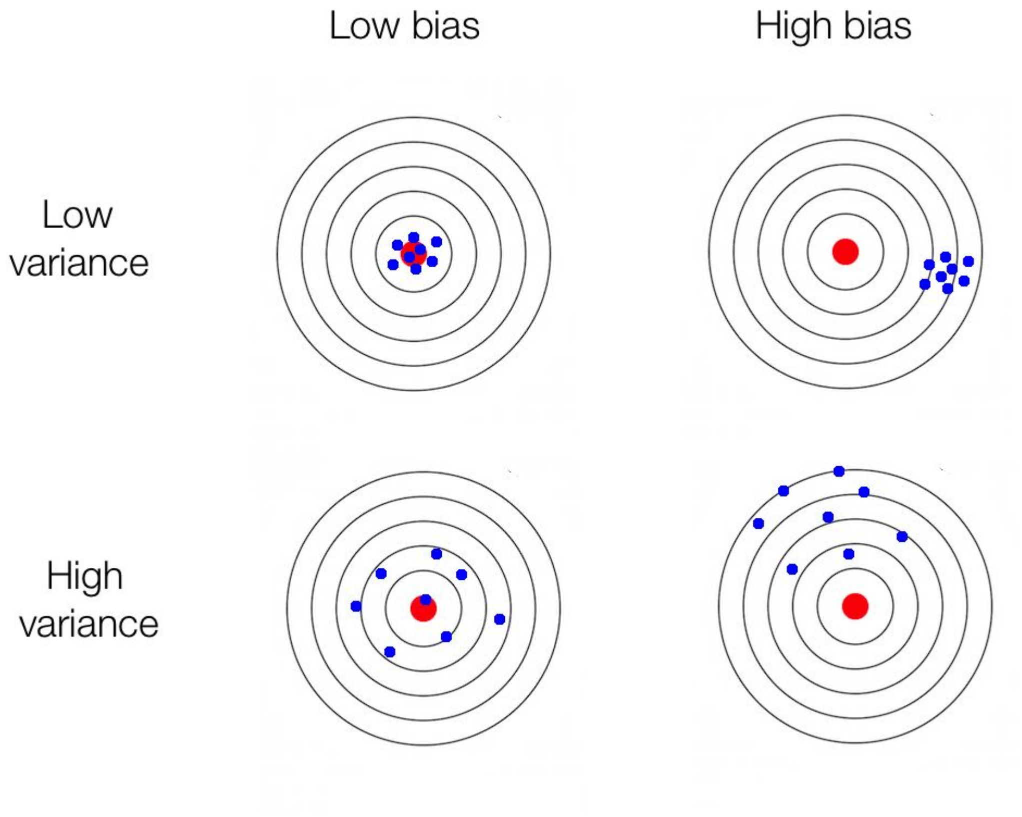
Score function provides unbiased estimates of the gradient through sampling.

$$\hat{g}_i = \nabla_\theta \ln h(x_i|\theta) f(x_i)$$

Increase the log probability of a sample in proportion to how good the sample is.

$$\nabla_\theta \hat{\eta}(\pi_\theta) = \sum_{t=0}^{T} \nabla_\theta \ln \pi(a_t|s_t;\theta) \left(\sum_{t'=t}^{T} \gamma^{t'-t} r_t \right)$$

How good is it in practice?



How good is it in practice?

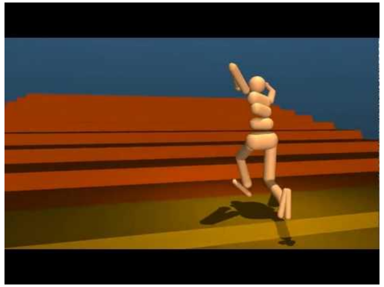
- Terrible! This suffers from extremely high variance.
- Need very large batch sizes (large number of rollouts).
- Problem of constant offset: suppose we re-define reward: $r_{new} = r + c$, variance of PG estimate will increase!
- Variance reduction using a baseline.

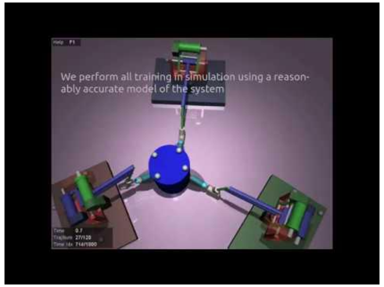
$$\nabla_\theta \hat{\eta}(\pi_\theta) = \sum_{t=0}^{T} \nabla_\theta \ln \pi(a_t | s_t; \theta) \left(\sum_{t'=t}^{T} \gamma^{t'-t} r_t - b(s_t) \right)$$

- **Intuition:** Increase probability of better than average actions and decrease probability of poorer than average actions.

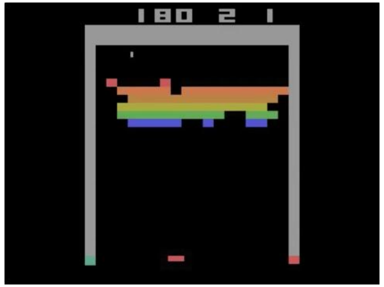
Some model-free RL success stories

- Policy gradients -- used primarily in sim2real (or direct real) for robotics
- Q-learning -- mostly used high dimensional inputs (e.g. images) and discrete actions





We perform all training in simulation using a reasonably accurate model of the system



Heess et al. Emergence of Locomotion Behaviours in Rich Environments, 2017

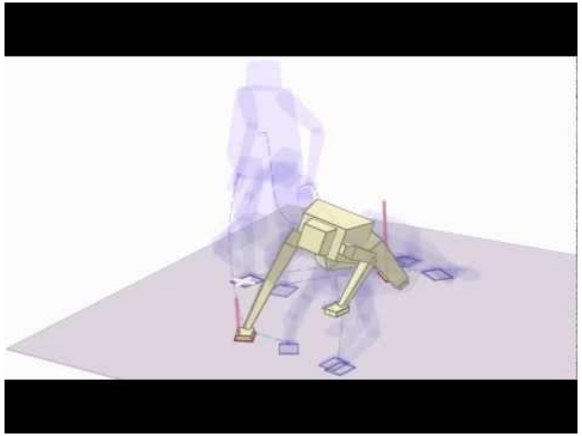
Rajeswaran et al. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations, 2017

Lowrey et al. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system, 2018

Mnih et al. Playing Atari with Deep Reinforcement Learning, 2013

A few model-based RL success stories

- System Identification: Can we learn the dynamics model of the world?
- Model-based optimization/control: Given a dynamics model, how to best use it?
- Can we couple the two so that one is aware of the mistakes of the other?

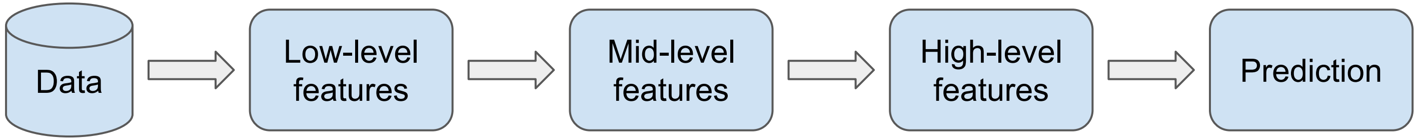


Mordatch et al. Discovery of complex behaviors through contact-invariant optimization, 2012

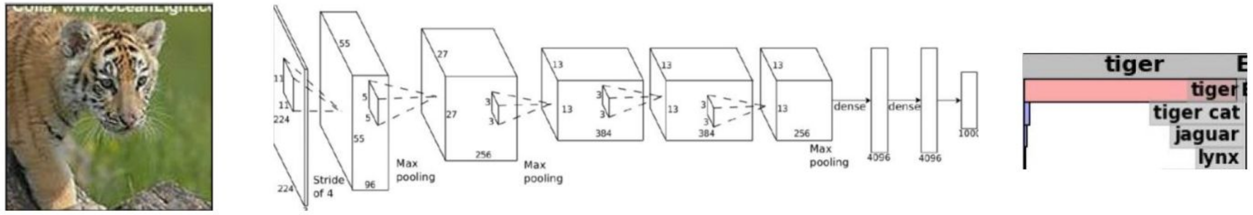
Kumar et al. Optimal Control with Learned Local Models: Application to Dexterous Manipulation, 2016

Why “Deep” RL?

Traditional paradigm is a modular, feature-based approach familiar in engineering



Deep Learning is an end-to-end approach



Why "Deep" RL?

End-to-end approaches are clearly working better for tasks associated with perception.

A large part of the brain is involved in even extremely simple movements, suggesting a tight coupling between perception, internal model of the world, and motor control.

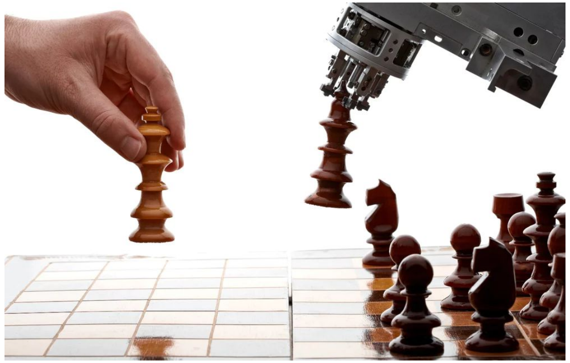
Complex animals learn most of their behavioral repertoire (instead of using genetically defined control strategies), thus their brain needs to have not only control machinery, but also machinery that builds control machinery (box of chocolates vs chocolate factory).

It would be nice (in practice, understanding, and pedagogy) to have a **single** learning algorithm to replicate the functionality of the brain.

Why “Deep” RL?

What must a single learning algorithm be capable of?

- Interpret rich sensory inputs (vision, speech, tactile etc.)
- Be capable of making complex decisions -- from low level muscle movements (how much to pull the muscle for walking) to high level plans (moves in a chess game)

Bayesian estimator
optimal controller


Combining deep learning with reinforcement learning is a step towards this direction

Thank you!!!

Acknowledgements: Kendall Lowrey, Emanuel Todorov, Sham Kakde