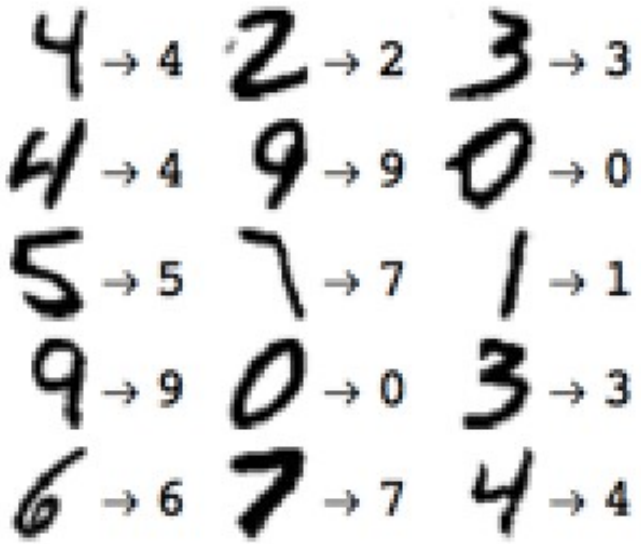
Dynamic Programming and Reinforcement Learning

Daniel Russo

Columbia Business School
Decision Risk and Operations Division

Fall, 2017

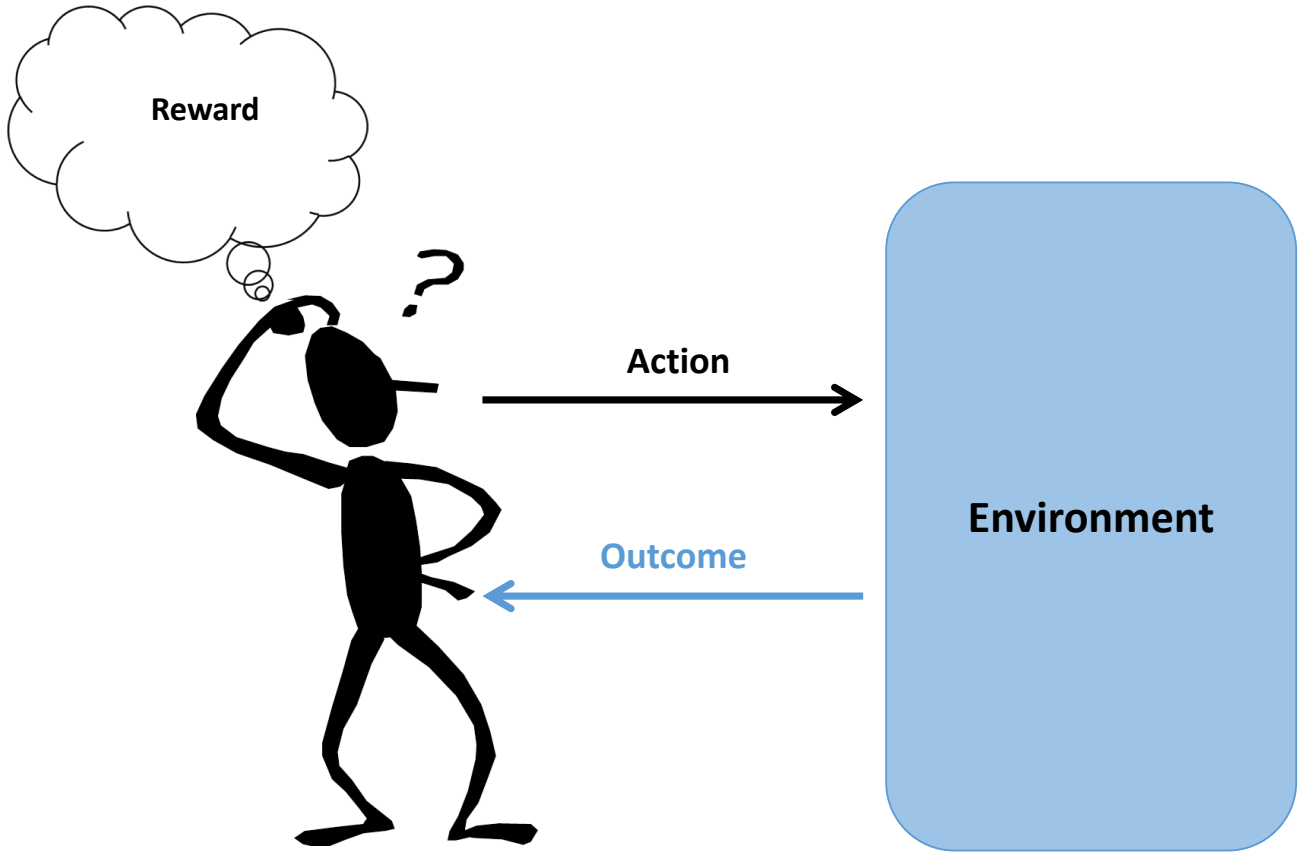
Supervised Machine Learning



Learning from datasets

- A passive paradigm
- Focus on pattern recognition

Reinforcement Learning



Learning to attain a goal through interaction with a poorly understood environment.

Canonical (and toy) RL environments

Cart Pole
Mountain Car

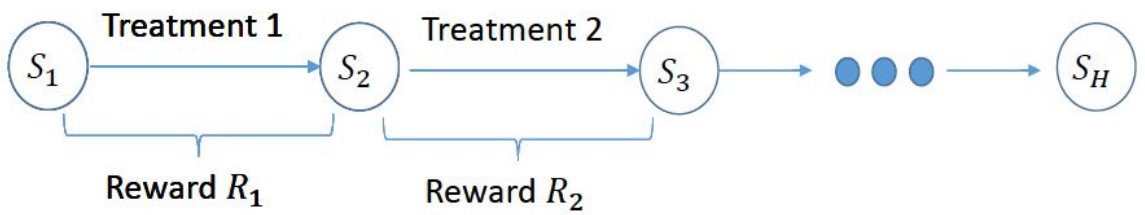
Impressive new (and toy) RL environments

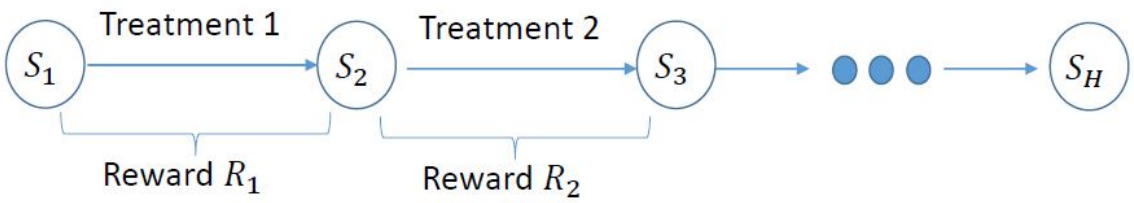
Atari from pixels

Challenges in Reinforcement Learning

- Partial Feedback
  - The data one gathers depends on the actions they take.
- Delayed Consequences
  - Rather than maximize the immediate benefit from the next interaction, one must consider the impact on future interactions.

Dream Application: Management of Chronic Diseases





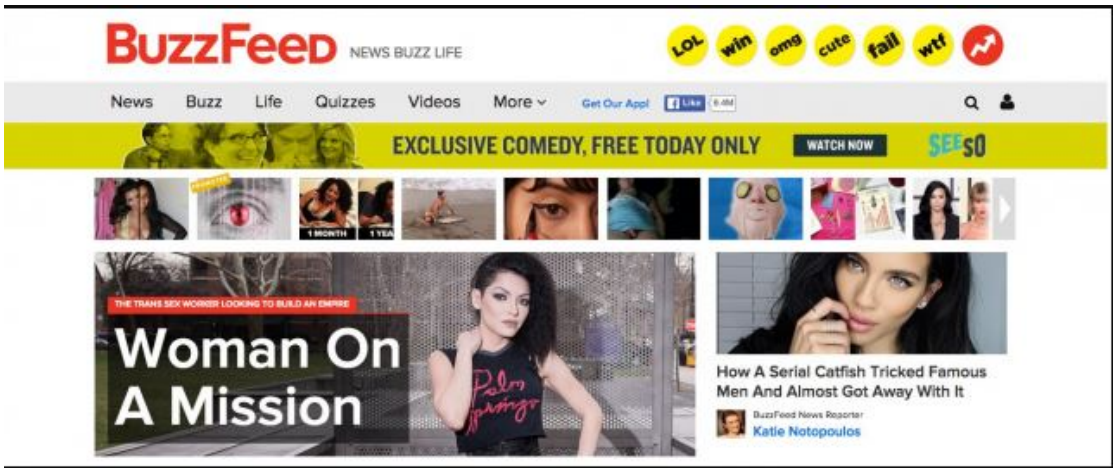
Various researchers are working on mobile health interventions

Dream Application: Intelligent Tutoring Systems

*Picture shamelessly lifted from a slide of Emma Brunskill's

Dream Application: Beyond Myopia in E-Commerce

- Online marketplaces and web services have repeated interactions with users, but are deigned to optimize the *next interaction*.
- RL provides a framework for optimizing the cumulative value generated by such interactions.
- How useful will this turn out to be?



Deep Reinforcement Learning

- RL where function approximation is performed using a deep neural network, instead of using linear models, kernel methods, shallow neural networks, etc.

Deep Reinforcement Learning

- RL where function approximation is performed using a deep neural network, instead of using linear models, kernel methods, shallow neural networks, etc.
- Justified excitement
  - Hope is to enable direct training of control systems based on complex sensory inputs (e.g. visual or auditory sensors)
  - DeepMind's DQN learns to play Atari from pixels, without learning vision first.

Deep Reinforcement Learning

- RL where function approximation is performed using a deep neural network, instead of using linear models, kernel methods, shallow neural networks, etc.
- Justified excitement
  - Hope is to enable direct training of control systems based on complex sensory inputs (e.g. visual or auditory sensors)
  - DeepMind's DQN learns to play Atari from pixels, without learning vision first.
- Also a lot of less justified hype.

Warning

1. This is an advanced PhD course.
2. It will be primarily theoretical. We will prove theorems when we can. The emphasis will be on precise understand of why methods work and why they may fail completely in simple cases.
3. There are tons of engineering tricks to Deep RL. I won't cover these.

My Goals

1. Encourage great students to do research in this area.
2. Provide a fun platform for introducing technical tools to operations PhD students.
   - Dynamic programming, stochastic approximation, exploration algorithms and regret analysis.
3. Sharpen my own understanding.

Tentative Course Outline

1. Foundational Material on MDPs
2. Estimating Long Run Value
3. Exploration Algorithms

* Additional topics as time permits
  - Policy gradients and actor critic
  - Rollout and Monte-Carlo tree search.

Markov Decision Processes: A warmup

On the white-board

Shortest path in a directed graph

Markov Decision Processes: A warmup

On the white-board

Shortest path in a directed graph

Imagine while traversing the shortest path, you discover one of the routes is closed. How should you adjust your path?

Example: Inventory Control

Stochastic demand
Orders have lead time
Non-perishable inventory
Inventory holding costs
Finite selling season

Example: Inventory Control

Periods $k = 0, 1, 2, \ldots, N$

$x_k \in \{0, \ldots, 1000\}$	current inventory
$u_k \in \{0, \ldots, 1000 - x_k\}$	inventory order
$w_k \in \{0, 1, 2, \ldots\}$	demand (*i.i.d w/ known dist.*)
$x_{k+1} = \lfloor x_k - w_k \rfloor + u_k$	Transition dynamics

Example: Inventory Control

Periods $k = 0, 1, 2, \ldots, N$

$$x_k \in \{0, \ldots, 1000\} \quad \text{current inventory}$$
$$u_k \in \{0, \ldots, 1000 - x_k\} \quad \text{inventory order}$$
$$w_k \in \{0, 1, 2, \ldots\} \quad \text{demand } (\textit{i.i.d w/ known dist.})$$
$$x_{k+1} = \lfloor x_k - w_k \rfloor + u_k \quad \text{Transition dynamics}$$

Cost function

$$g(x, u, w) = \underbrace{c_H x}_{\textit{Holding cost}} + \underbrace{c_L \lfloor w - x \rfloor}_{\textit{Lost sales}} + \underbrace{c_O(u)}_{\textit{Order cost}}$$

Example: Inventory Control

Objective:

$$\min \mathbb{E}\left[\sum_{k=0}^{N} g(x_k, u_k, w_k)\right]$$

Example: Inventory Control

Objective:

$$\min \mathbb{E}\left[\sum_{k=0}^{N} g(x_k, u_k, w_k)\right]$$

- Minimize over what?

Example: Inventory Control

Objective:

$$\min \mathbb{E}\left[\sum_{k=0}^{N} g(x_k, u_k, w_k)\right]$$

- Minimize over what?
  - Over fixed sequences of controls $u_0, u_1, \ldots$?
  - No, over policies (adaptive ordering strategies).

Example: Inventory Control

Objective:

$$\min \mathbb{E}\left[\sum_{k=0}^{N} g(x_k, u_k, w_k)\right]$$

- Minimize over what?
  - Over fixed sequences of controls $u_0, u_1, \ldots$?
  - No, over policies (adaptive ordering strategies).
- Sequential decision making under uncertainty where
  - Decisions have delayed consequences.
  - Relevant information is revealed during the decision process.

Further Examples

- Dynamic pricing (over a selling season)
- Trade execution (with market impact)
- Queuing admission control
- Consumption-savings models in economics
- Search models in economics
- Timing of maintenance and repairs

Finite Horizon MDPs: formulation

A discrete time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k) \qquad k = 0, 1, ..., N$$

where

$$\begin{array}{cc} x_k \in \mathcal{X}_k & \text{state} \\ u_k \in U_k(x_k) & \text{control} \\ w_k & (\textit{i.i.d w/ known dist.}) \end{array}$$

Assume state and control spaces are finite.

Finite Horizon MDPs: formulation

A discrete time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k) \qquad k = 0, 1, ..., N$$

where

$$\begin{array}{cc} x_k \in \mathcal{X}_k & \text{state} \\ u_k \in U_k(x_k) & \text{control} \\ w_k & (\textit{i.i.d w/ known dist.}) \end{array}$$

Assume state and control spaces are finite.

The total cost incurred is

$$\sum_{k=0}^{N} \underbrace{g_k(x_k, u_k, w_k)}_{\textit{cost in period } k}$$

Finite Horizon MDPs: formulation

A policy is a sequence $\pi = (\mu_0, \mu_1, ..., \mu_N)$ where

$$\mu_k : x_k \mapsto u_k \in U_k(x_k).$$

Expected cost of following π from state x_0 is

$$J_\pi(x_0) = \mathbb{E}\left[\sum_{k=0}^{N} g_k(x_k, u_k, w_k)\right]$$

where $x_{k+1} = f_k(x_k, u_k, w_k)$ and $\mathbb{E}[\cdot]$ is over the $w_k's$.

Finite Horizon MDPs: formulation

The optimal expected cost to go from x_0 is

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)$$

where Π consists of all feasible policies.

We will see the same policy π^* is optimal for all initial states. So

$$J^*(x) = J_{\pi^*}(x) \qquad \forall x$$

Minor differences with Bertsekas Vol. I

Bertsekas

- Uses a special terminal cost function $g_N(x_N)$
  - Can always take $g_N(x, u, w)$ to be independent of u, w.
- Lets the distribution of w_k depend on k and x_k.
  - This can be embedded in the functions f_k, g_k.

Principle of Optimality

Regardless of the consequences of initial decisions, an optimal policy should be optimal in the sub-problem beginning in the current state and time period.

Principle of Optimality

Regardless of the consequences of initial decisions, an optimal policy should be optimal in the sub-problem beginning in the current state and time period.

- **Sufficiency:** Such policies exist and minimize total expected cost from any initial state.
- **Necessity:** A policy that is optimal from some initial state must behave optimally in any subproblem that is reached with positive probability.

The Dynamic Programming Algorithm

Set

$$J_N^*(x) = \min_{u \in U_N(x)} \mathbb{E}[g_N(x, u, w)] \qquad \forall x \in \mathcal{X}_n$$

For $k = N-1, N-2, \ldots 0$, set

$$J_k^*(x) = \min_{u \in U_k(x)} \mathbb{E}[g_k(x, u, w) + J_{k+1}^*(f_k(x, u, w))] \quad \forall x \in \mathcal{X}_k.$$

The Dynamic Programming Algorithm

Proposition

For all $x \in \mathcal{X}_0$, $J^*(x) = J_0^*(x)$. The optimal cost to go is attained by a policy $\pi^* = (\mu_0, ..., \mu_N)$ where

$$\mu_N(x) \in \arg\min_{u \in U_N(x)} \mathbb{E}[g_N(x, u, w)] \qquad \forall x \in \mathcal{X}_N$$

and for all $k \in \{0, \ldots, N-1\}$, $x \in \mathcal{X}_k$

$$\mu_k^*(x) \in \arg\min_{u \in U_k(x)} \mathbb{E}[g_k(x, u, w) + J_{k+1}^*(f_k(x, u, w))].$$

The Dynamic Programming Algorithm

Class Exercise

Argue this is true for a 2 period problem (N=1).
Hint, recall the tower property of conditional expectation.

$$\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y|X]]$$

A Tedious Proof

For any policy $\pi = (\mu_0, \mu_1)$ and initial state x_0,

$$
\begin{aligned}
& \mathbb{E}_\pi[g_0(x_0, \mu_0(x_0), w_0) + g_1(x_1, \mu_1(x_1), w_1)] \\
= \; & \mathbb{E}_\pi[g_0(x_0, \mu_0(x_0), w_0) + \mathbb{E}[g_1(x_1, \mu_1(x_1), w_1)|x_1]] \\
\geq \; & \mathbb{E}_\pi[g_0(x_0, \mu_0(x_0), w_0) + \min_{u \in U(x_1)} \mathbb{E}[g_1(x_1, u, w_1)|x_1]] \\
= \; & \mathbb{E}_\pi[g_0(x_0, \mu_0(x_0), w_0) + J_1^*(x_1)] \\
= \; & \mathbb{E}_\pi[g_0(x_0, \mu_0(x_0), w_0) + J_1^*(f_0(x_0, \mu_0(x_0), w_0))] \\
\geq \; & \min_{u \in U(x_0)} \mathbb{E}[g_0(x_0, u, w_0) + J_1^*(f_0(x_0, u, w_0)] \\
= \; & J_0^*(x_0)
\end{aligned}
$$

Under π^*, every inequality is an equality.

Markov Property

Markov Chain

A stochastic process $(X_0, X_1, X_2, \ldots)$ is a Markov chain if for each $n \in \mathbb{N}$, conditioned on X_{n-1}, X_n is independent of $(X_0, \ldots, X_{n-2})$. That is

$$\mathbb{P}(X_n = \cdot | X_{n-1}) = \mathbb{P}(X_n = \cdot | X_0, \ldots, X_{n-1})$$

Markov Property

Markov Chain

A stochastic process $(X_0, X_1, X_2, \ldots)$ is a Markov chain if for each $n \in \mathbb{N}$, conditioned on X_{n-1}, X_n is independent of $(X_0, \ldots, X_{n-2})$. That is

$$\mathbb{P}(X_n = \cdot | X_{n-1}) = \mathbb{P}(X_n = \cdot | X_0, \ldots, X_{n-1})$$

Without loss of generality we can view a Markov chain as the output of a stochastic recursion

$$X_{n+1} = f_n(X_n, W_n)$$

for an i.i.d sequence of disturbances $(W_0, W_1, \ldots)$.

Markov Property

Our problem is called a Markov decision process because

$$\begin{aligned} & \mathbb{P}(x_{n+1} = x | x_0, u_0, w_0, \ldots, x_n, u_n) \\ = \; & \mathbb{P}(f_n(x_n, u_n, w_n) = x | x_n, u_n) \\ = \; & \mathbb{P}(x_{n+1} = x | x_n, u_n) \end{aligned}$$

Requires the encoding of the state is sufficiently rich.

Inventory Control Revisited

- Suppose that inventory has a lead time of 2 periods.
- Orders can still be placed in any period.
- Is this an MDP with state=current inventory?

Inventory Control Revisited

- Suppose that inventory has a lead time of 2 periods.
- Orders can still be placed in any period.
- Is this an MDP with state=current inventory?
  - No!
  - Transition probabilities depend on the order that is currently in transit.

Inventory Control Revisited

- Suppose that inventory has a lead time of 2 periods.
- Orders can still be placed in any period.
- Is this an MDP with state=current inventory?
  - No!
  - Transition probabilities depend on the order that is currently in transit.

This is an MDP if we augment the state space so

$x_n = (\text{current inventory},\ \text{inventory arriving next period}).$

State Augmentation

In the extreme, choosing the state to be the full history

$$\tilde{x}_{n-1} = (x_0, u_0, \ldots, u_{n-2}, x_{n-1})$$

suffices since

$$\mathbb{P}(\tilde{x}_n = \cdot | \tilde{x}_{n-1}, u_{n-1}) = \mathbb{P}(\tilde{x}_n = \cdot | x_0, u_0, \ldots, x_{n-1}, u_{n-1}).$$

State Augmentation

In the extreme, choosing the state to be the full history

$$\tilde{x}_{n-1} = (x_0, u_0, \ldots, u_{n-2}, x_{n-1})$$

suffices since

$$\mathbb{P}(\tilde{x}_n = \cdot | \tilde{x}_{n-1}, u_{n-1}) = \mathbb{P}(\tilde{x}_n = \cdot | x_0, u_0, \ldots, x_{n-1}, u_{n-1}).$$

- For the next few weeks we will assume the Markov property holds.

State Augmentation

In the extreme, choosing the state to be the full history

$$\tilde{x}_{n-1} = (x_0, u_0, \ldots, u_{n-2}, x_{n-1})$$

suffices since

$$\mathbb{P}(\tilde{x}_n = \cdot | \tilde{x}_{n-1}, u_{n-1}) = \mathbb{P}(\tilde{x}_n = \cdot | x_0, u_0, \ldots, x_{n-1}, u_{n-1}).$$

- For the next few weeks we will assume the Markov property holds.
- Computational tractability usually requires a compact state representation.

Example: selling an asset

An instance of optimal stopping.

- Deadline to sell within N periods.
- Potential buyers make offers in sequence.
- The agent chooses to accept or reject each offer
  - The asset is sold once an offer is accepted.
  - Offers are no longer available once declined.
- Offers are statistically independent.
- Profits can be invested with interest rate $r > 0$ per period.

Example: selling an asset

An instance of optimal stopping.

- Deadline to sell within N periods.
- Potential buyers make offers in sequence.
- The agent chooses to accept or reject each offer
  - The asset is sold once an offer is accepted.
  - Offers are no longer available once declined.
- Offers are statistically independent.
- Profits can be invested with interest rate $r > 0$ per period.

Class Exercise

1. Formulate this as a finite horizon MDP.
2. Write down the DP algorithm.

Example: selling an asset

- Special terminal state t (costless and absorbing)
- $x_k \neq t$ is the offer considered at time k.
- $x_0 = 0$ is fictitious null offer.
- $g_k(x_k, \text{sell}) = (1 + r)^{N-k} x_k$.
- $x_k = w_{k-1}$ for independent $w_0, w_1, \ldots$

Example: selling an asset

DP Algorithm

$$\begin{aligned} J_k^*(t) &= 0 \qquad \forall k \\ J_N^*(x) &= x \\ J_k^*(x) &= \max\{(1+r)^{N-k}x, \mathbb{E}[J_{k+1}^*(w_k)]\} \end{aligned}$$

A threshold policy is optimal:

$$\textit{Sell} \iff x_k \geq \frac{\mathbb{E}[J_{k+1}^*(w_k)]}{(1+r)^{N-k}}$$