MACHINE LEARNING FOR HEALTHCARE

6.S897, HST.S53

Lecture 13: Finding optimal treatment policies

Prof. David Sontag

MIT EECS, CSAIL, IMES

(Thanks to Peter Bodik for slides on reinforcement learning)

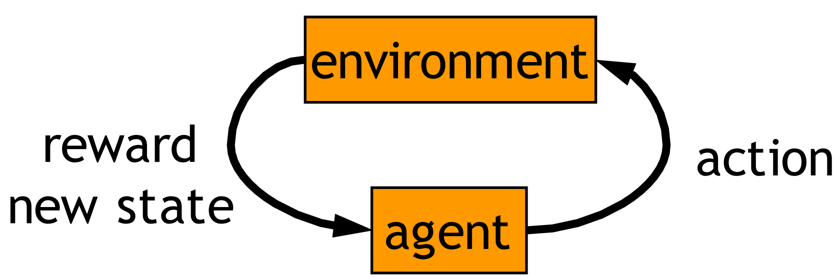
Massachusetts Institute of Technology

Outline for today's class

- **Finding optimal treatment policies**
  - "Reinforcement learning" / "dynamic treatment regimes"
  - What makes this hard?
- Q-learning (Watkins '89)
- Fitted Q-iteration (Ernst et al. '05)
  - Application to schizophrenia (Shortreed et al., 11)
  - Deep Q-networks for playing Atari games (Mnih et al. '15)

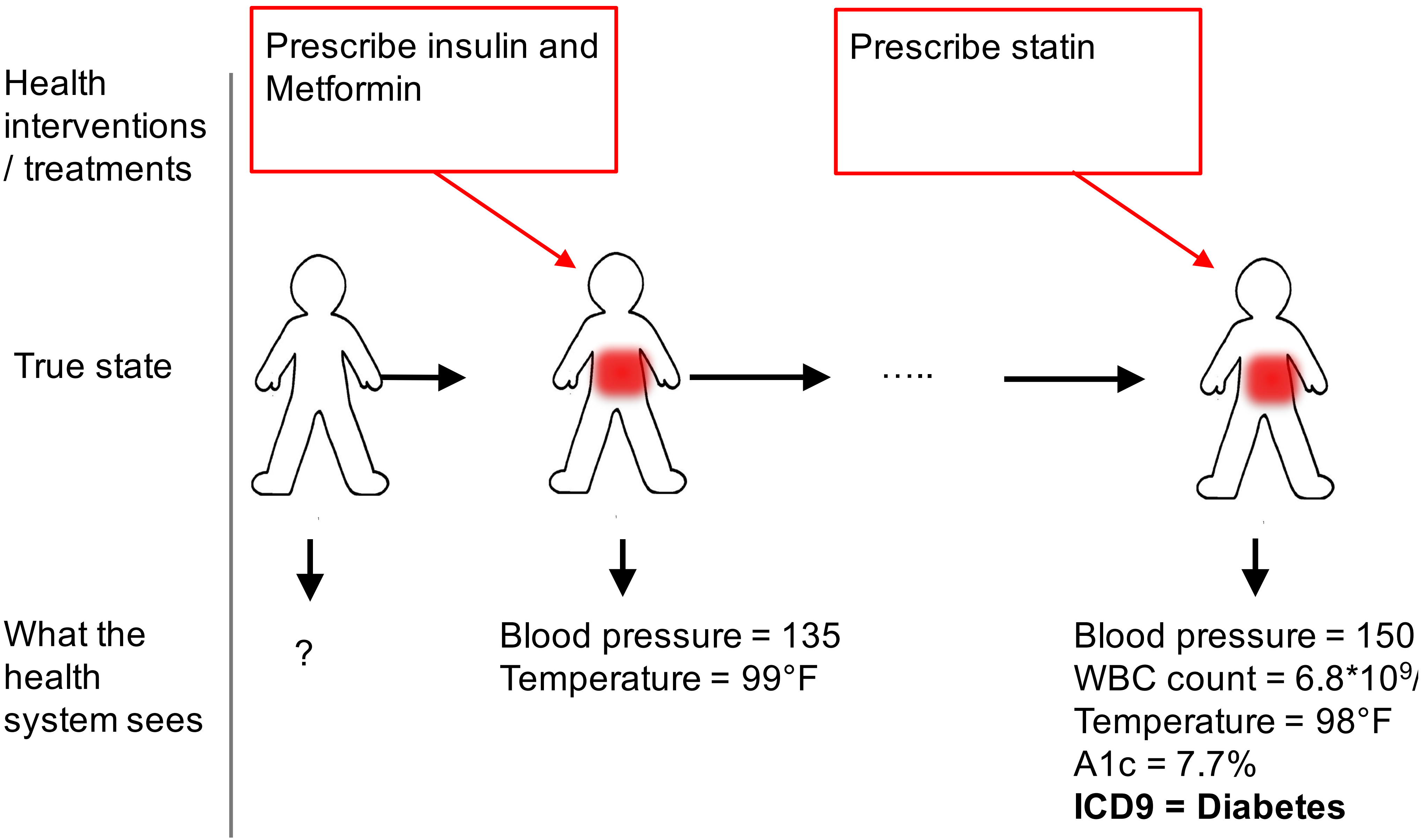
Previous Lectures

- Supervised learning
  - classification, regression
- Unsupervised learning
  - clustering
- **Reinforcement learning**
  - more general than supervised/unsupervised learning
  - learn from interaction w/ environment to achieve a goal

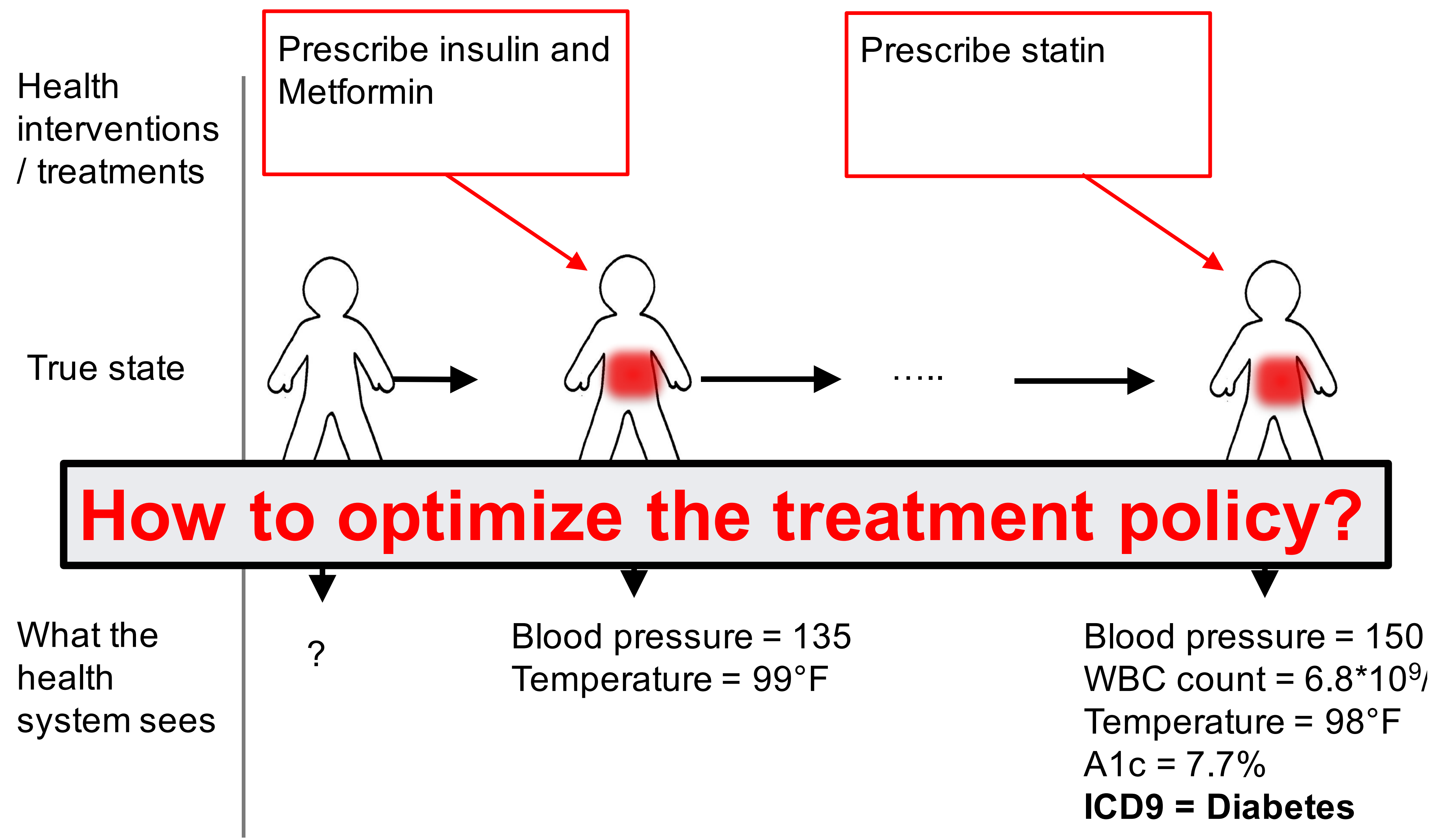


[Slide from Peter Bodik]

Finding optimal treatment policies



Finding optimal treatment policies



Key challenges

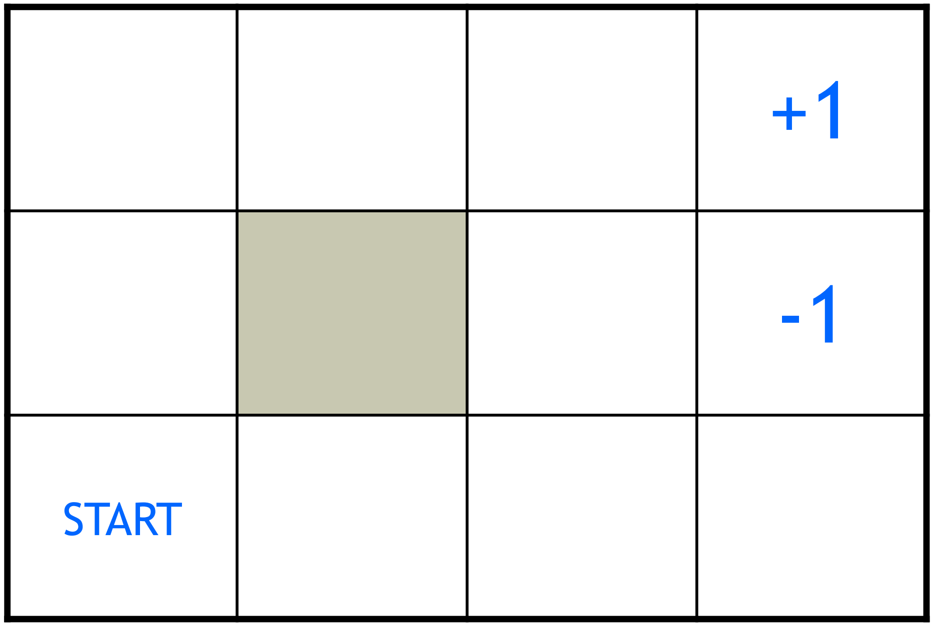
1. Only have observational data to learn policies from
   - **At least as hard as causal inference**
   - *Reduction:* just 1 treatment & time-step

2. Have to define outcome that we want to optimize (reward function)

3. Input data can be high-dimensional, noisy, and incomplete

4. Must disentangle (possibly long-term) effects of sequential actions and confounders → *credit assignment problem*

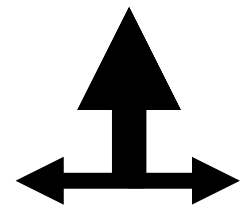
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80%	move UP
10%	move LEFT
10%	move RIGHT

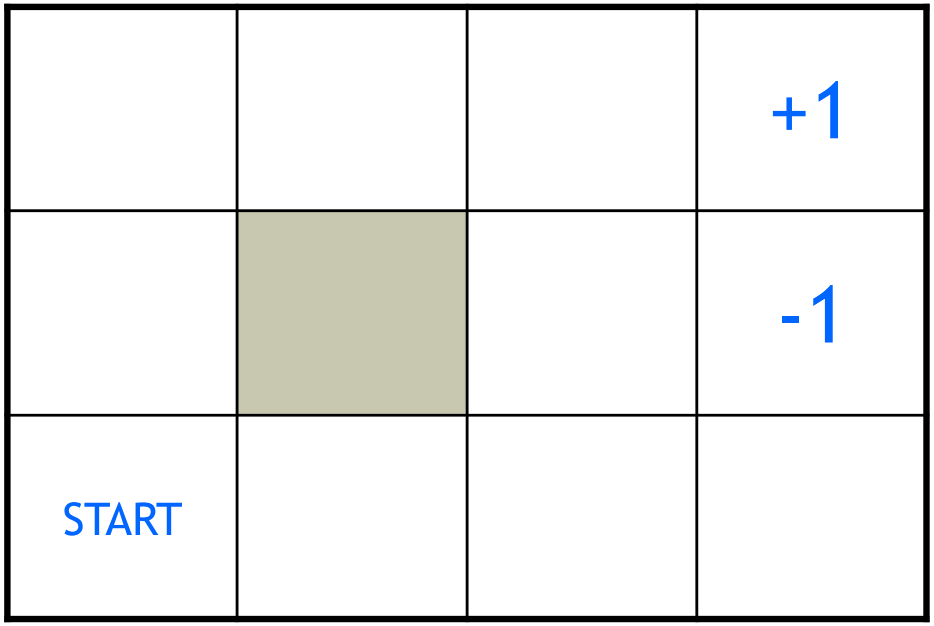


- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

- what's the strategy to achieve max reward?
- what if the actions were deterministic?

[Slide from Peter Bodik]

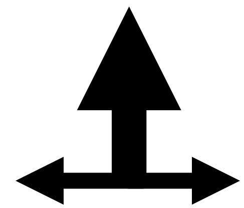
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT



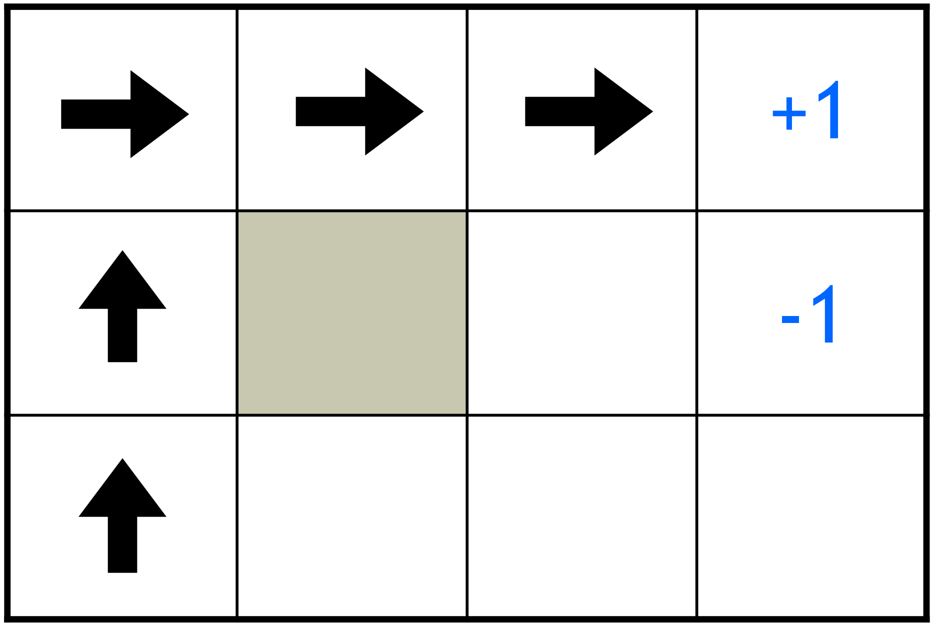
reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

- states
- actions
- rewards

- what is the solution?

[Slide from Peter Bodik]

Is this a solution?

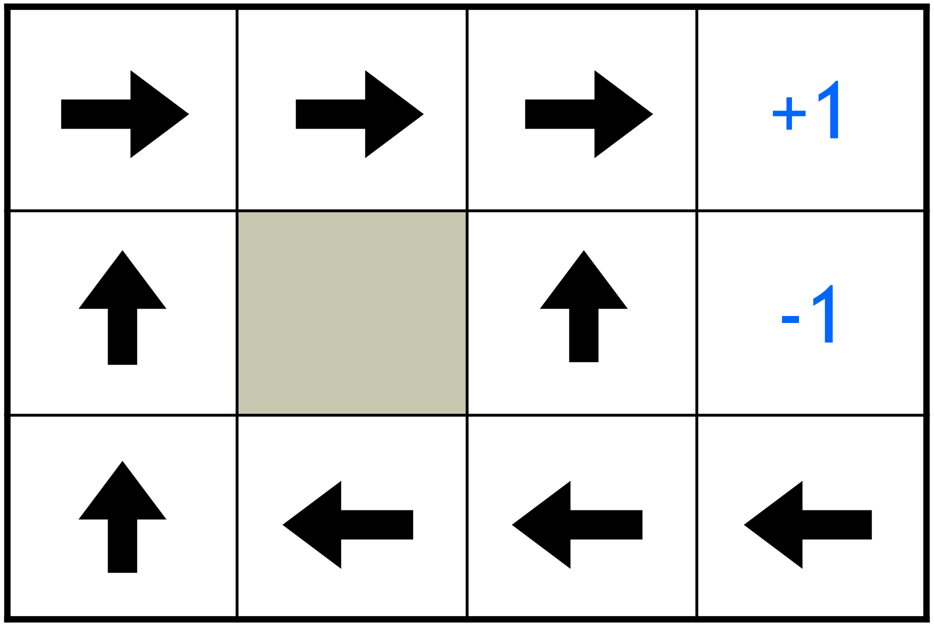


- only if actions deterministic
  - not in this case (actions are stochastic)

- solution/policy
  - mapping from each state to an action

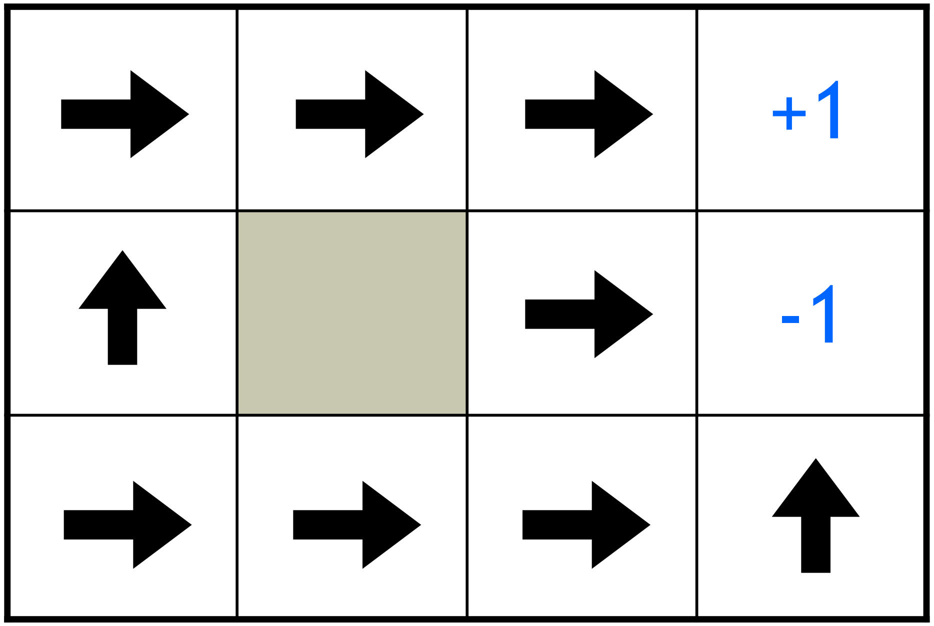
[Slide from Peter Bodik]

Optimal policy



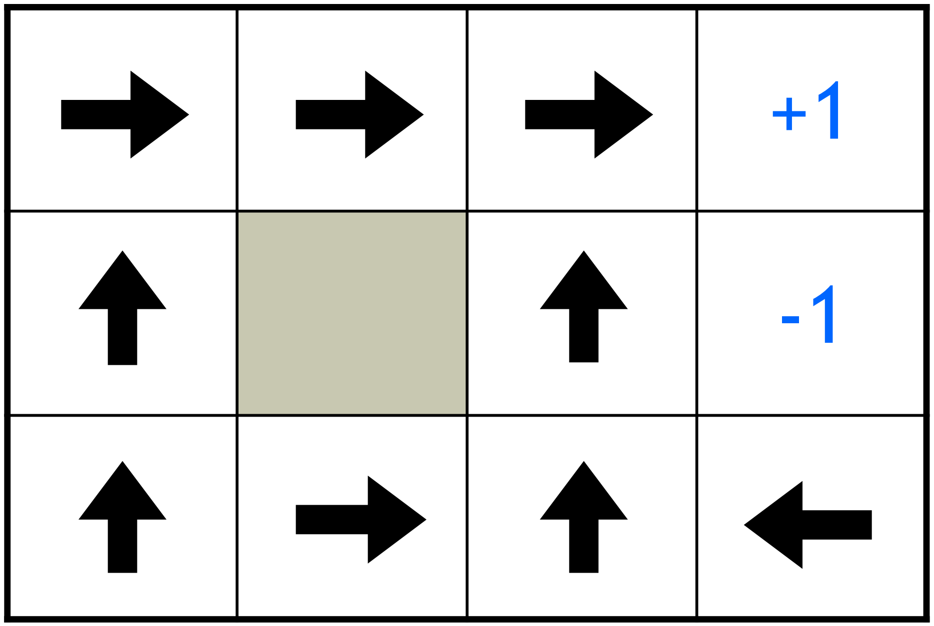
[Slide from Peter Bodik]

Reward for each step: -2



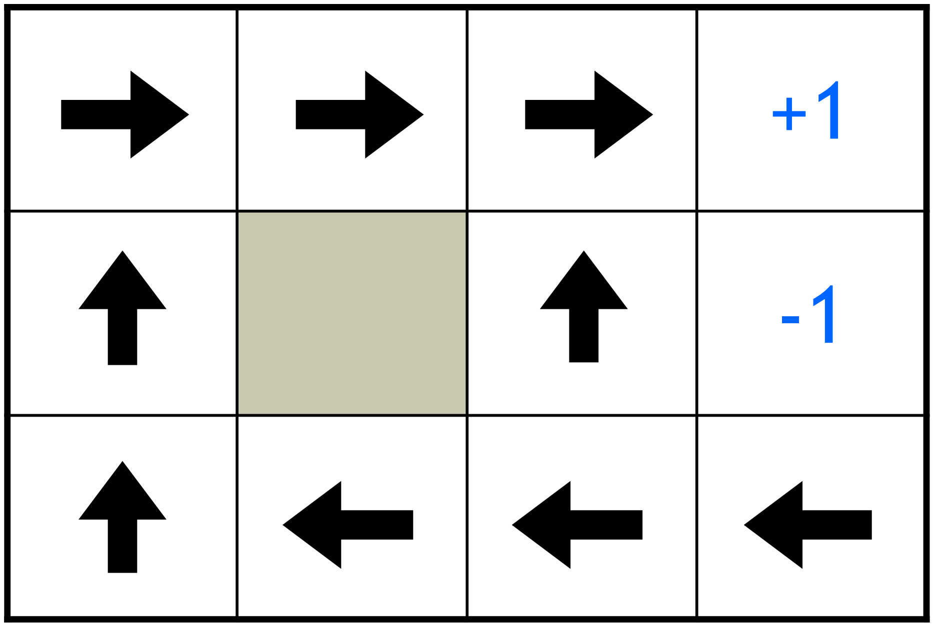
[Slide from Peter Bodik]

Reward for each step: -0.1



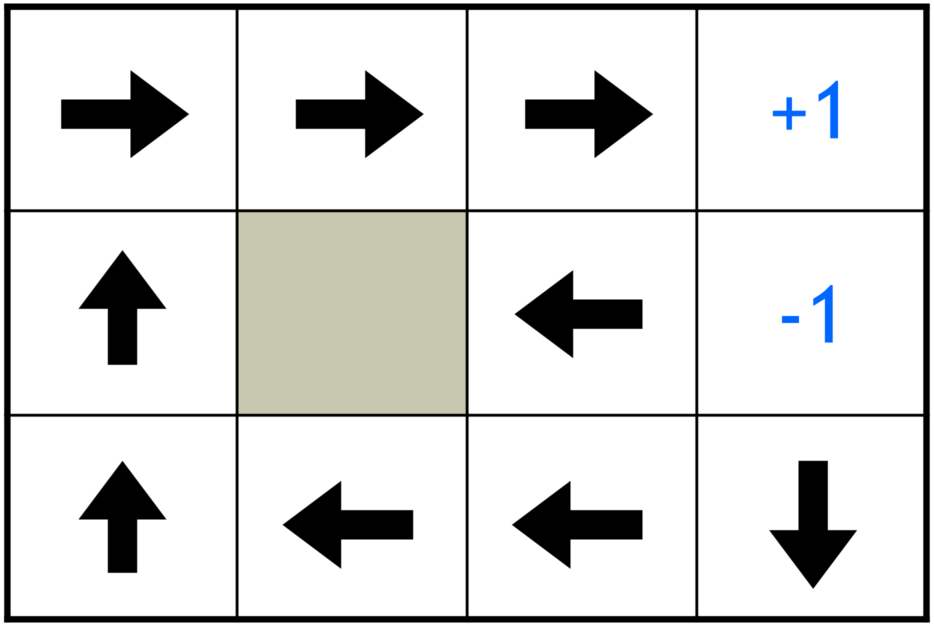
[Slide from Peter Bodik]

Reward for each step: -0.04



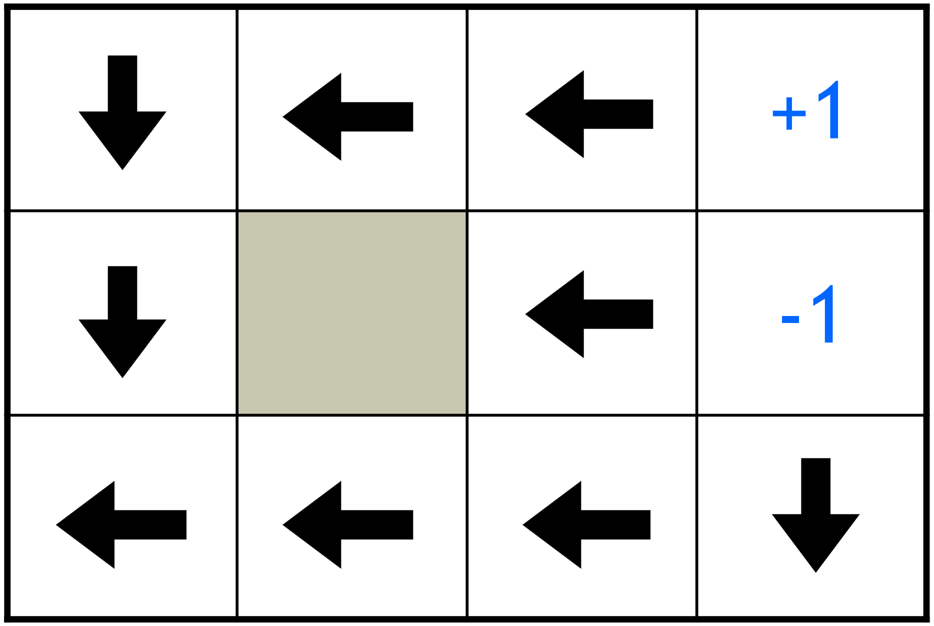
[Slide from Peter Bodik]

Reward for each step: -0.01



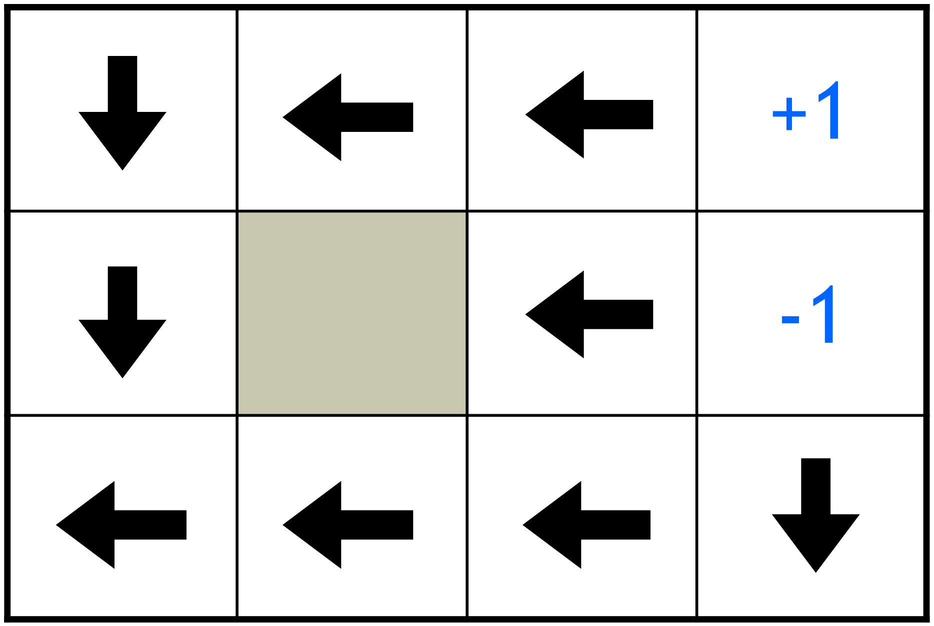
[Slide from Peter Bodik]

Reward for each step: ???



[Slide from Peter Bodik]

Reward for each step: +0.01



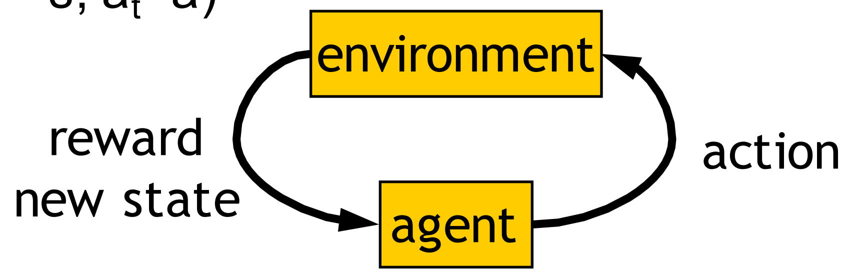
[Slide from Peter Bodik]

Outline for today's class

- Finding optimal treatment policies
  - "Reinforcement learning" / "dynamic treatment regimes"
  - What makes this hard?
- **Q-learning (Watkins '89)**
- Fitted Q-iteration (Ernst et al. '05)
  - Application to schizophrenia (Shortreed et al., 11)
  - Deep Q-networks for playing Atari games (Mnih et al. '15)

Markov Decision Process (MDP)

- set of states S, set of actions A, initial state S_0
- transition model $P(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t=a)$
  - P([1,1], up, [1,2]) = 0.8
- reward function r(s)
  - r([4,3]) = +1
- goal: maximize cumulative reward in the long run



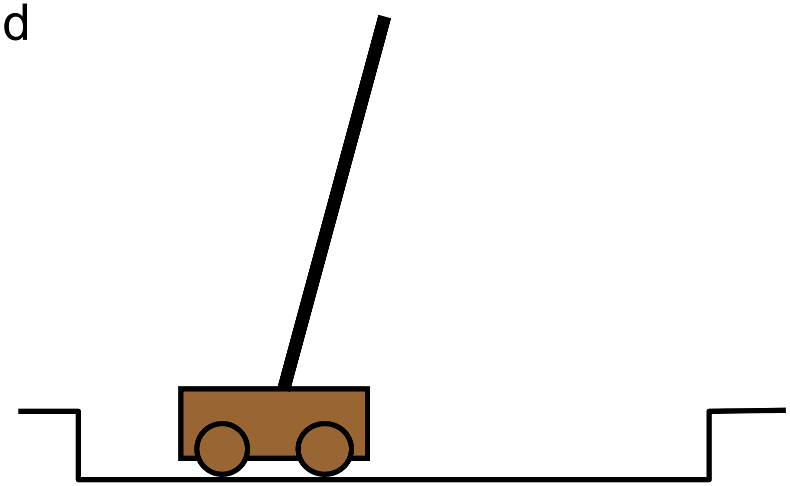
- policy: mapping from S to A
  - $\pi(s)$ or $\pi(s,a)$ (deterministic vs. stochastic)

- reinforcement learning
  - transitions and rewards usually not available
  - how to change the policy based on experience
  - how to explore the environment

[Slide from Peter Bodik]

State representation

- pole-balancing
  - move car left/right to keep the pole balanced
- state representation
  - position and velocity of car
  - angle and angular velocity of pole
- what about *Markov property*?
  - would need more info
  - noise in sensors, temperature, bending of pole
- solution
  - coarse discretization of 4 state variables
    - left, center, right
  - totally non-Markov, but still works



[Slide from Peter Bodik]

Designing rewards

- robot in a maze
  - episodic task, not discounted, +1 when out, 0 for each step
- chess
  - GOOD: +1 for winning, -1 losing
  - BAD: +0.25 for taking opponent's pieces
    - high reward even when lose
- rewards
  - rewards indicate what we want to accomplish
  - NOT how we want to accomplish it
- shaping

  - positive reward often very "far away"
  - rewards for achieving subgoals (domain knowledge)
  - also: adjust initial policy or initial value function

[Slide from Peter Bodik]

Computing return from rewards

- episodic (vs. continuing) tasks
  - "game over" after N steps
  - optimal policy depends on N; harder to analyze

- additive rewards
  - $V(s_0, s_1, \ldots) = r(s_0) + r(s_1) + r(s_2) + \ldots$
  - infinite value for continuing tasks

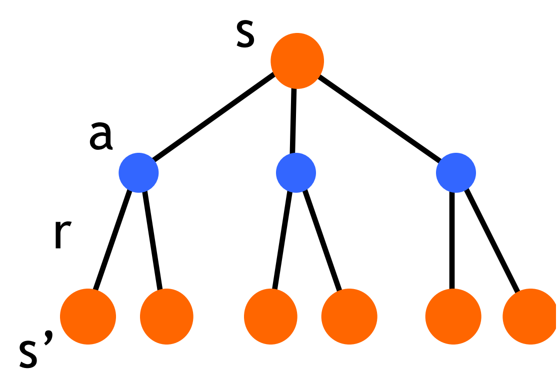
- discounted rewards
  - $V(s_0, s_1, \ldots) = r(s_0) + \gamma^* r(s_1) + \gamma^{2*} r(s_2) + \ldots$
  - value bounded if rewards bounded

[Slide from Peter Bodik]

Finding optimal policy using value iteration

- state value function: $V^{\pi}(s)$
  - expected return when starting in *s* and following π
  - optimal policy π* has property:

$$V^{\pi^*}(s) = \max_a \sum_{s'} P^a_{ss'}[r^a_{s,s'} + \gamma V^{\pi*}(s')]$$



- Learn using fixed point iteration:

$$V_{k+1}(s) = \max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V_k(s') \right]$$

- Equivalent formulation uses state-action value function:

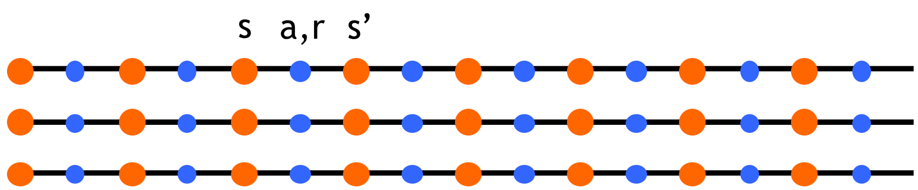
$$Q^{\pi}(s,a) = r^a_{s,s'} + \gamma V^{\pi}(s') \qquad V^{\pi}(s) = \max_a Q^{\pi}(s,a)$$

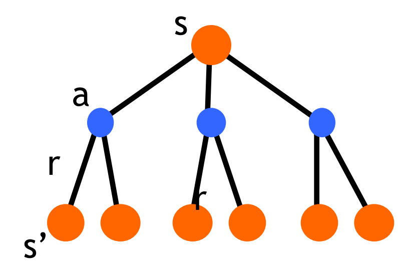
(expected return when starting in s, performing a, and following π)

$$Q_{k+1}(s,a) = \sum_{s'} P^a_{ss'}[r^a_{s,s'} + \gamma \max_{a'} Q_k(s',a')] \qquad \pi^*(s) = \arg\max_a Q^*(s,a)$$

Q-learning

- Same as value iteration, but rather than assume Pr(s' | s, a) is known, estimate it from data (i.e. episodes)
- **Input:** sequences/episodes from some *behavior policy*





- Combine data from all episodes into a set of *n* tuples (*n* = # episodes * length of each): $\{(s, a, s')\}$
- Use these to get empirical estimate $\hat{P}^a_{ss'}$ and use this instead
- In reinforcement learning, episodes are created as we go, using current policy + randomness for exploration

Where can Q-learning be used?

- need complete model of the environment and rewards
  - robot in a room
    - state space, action space, transition model
- can we use DP to solve
  - robot in a room?
  - back gammon, or Go?
  - helicopter?
  - optimal treatment trajectories?

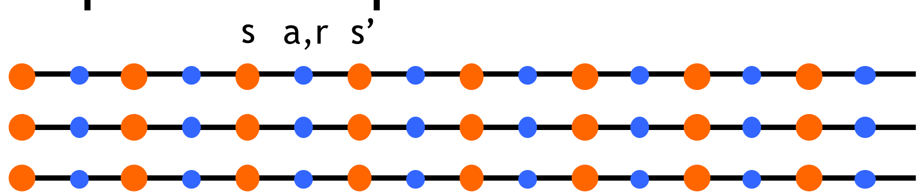
[Slide from Peter Bodik]

Outline for today's class

- Finding optimal treatment policies
  - "Reinforcement learning" / "dynamic treatment regimes"
  - What makes this hard?
- Q-learning (Watkins '89)
- **Fitted Q-iteration (Ernst et al. '05)**
  - Application to schizophrenia (Shortreed et al., 11)
  - Deep Q-networks for playing Atari games (Mnih et al. '15)

Fitted Q-iteration

- **Challenge**: in infinite or very large state spaces, very difficult to estimate Pr(s' | s, a)
- Moreover, this is a harder problem than we need to solve!
  - We only need to learn how to *act*
  - Can we learn the Q function directly, i.e. a mapping from s,a to expected cumulative reward? **("model-free" RL)**
  - Reduction to supervised machine learning (*exactly the same as we did in causal inference using regression*)
- **Input is the same:** sequences/episodes from some *behavior policy:*



- First let's create a dataset $\mathcal{F} = \{(\langle s_t^n, a_t^n \rangle, r_{t+1}^n, s_{t+1}^n), n = 1, \ldots, |\mathcal{F}|\}$ and learn $\hat{Q}(s_t, a_t) \rightarrow r_{t+1}$

Fitted Q-iteration

- First let's create a dataset $\mathcal{F} = \{(\langle s_t^n, a_t^n \rangle, r_{t+1}^n, s_{t+1}^n), n = 1, \ldots, |\mathcal{F}|\}$ and learn $\hat{Q}(s_t, a_t) \rightarrow r_{t+1}$
- **Trick:** to predict the **cumulative reward**, we *iterate this process*

- Initialize $\hat{Q}_0(s_t^n, a_t^n) = r_{t+1}^n$ using $\mathcal{F}$
- For k=1, …
  1. Create training set for *k*th learning problem:
  $$\mathcal{TS}_k = \{(\langle s_t^n, a_t^n \rangle, \hat{Q}_{k-1}(s_t^n, a_t^n)), \ \ \forall \langle s_t^n, a_t^n \rangle \in \mathcal{F}\}$$
  2. Use supervised learning to estimate function $\hat{Q}_{k-1}(s_t^n, a_t^n)$ from $\mathcal{TS}_k$
  3. Update Q values for each observed tuple in $\mathcal{F}$ using Bellman equation:
  $$\hat{Q}_k(s_t^n, a_t^n) = r_{t+1}^n + \gamma \max_a \hat{Q}_{k-1}(s_{t+1}^n, a)]$$

***GOAL:** extrapolate* to actions other than a^n_t (i.e., compute counterfactuals)

Example of Q-iteration

- Adaptive treatment strategy for treating psychiatric disorders

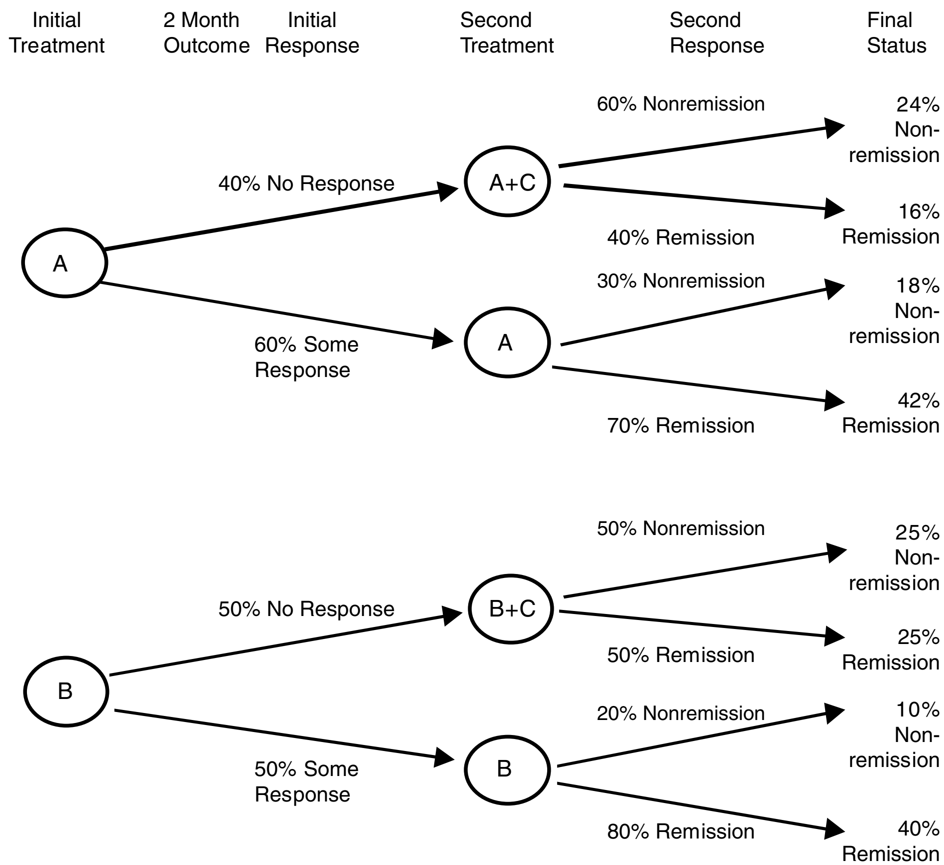


Figure 2 A comparison of two strategies. The strategy beginning with medication A has an overall remission rate at 4 months of 58% (16 + 42%). The strategy beginning with medication B has an overall remission rate at 4 months of 65% (25 + 40%). Medication A is best if considered as a stand-alone treatment, but medication B is best initially when considered as part of a sequence of treatments.

[Murphy et al., Neuropsychopharmacology, 2007]

Example of Q-iteration

- **Goal:** minimize average level of depression over 4-month period; only 2 decisions (initial and second treatment)
- Y_2 = summary of depression weeks 9 through 12
- S_8 = summary of side-effects up to end of 8^{th} week
- *First*, regress onto Y_2 using:

$$\beta_0 + \beta_1 S_8 + (\beta_2 + \beta_3 S_8) T_2$$

  learn decision rule that recommends switching treatment for patient if $\beta_2 + \beta_3 S_8$ is less than zero
- *Then consider initial decision T_1*, regressing on $Y_1 + \min(\beta_0 + \beta_1 S_8 + (\beta_2 + \beta_3 S_8), \beta_0 + \beta_1 S_8)$

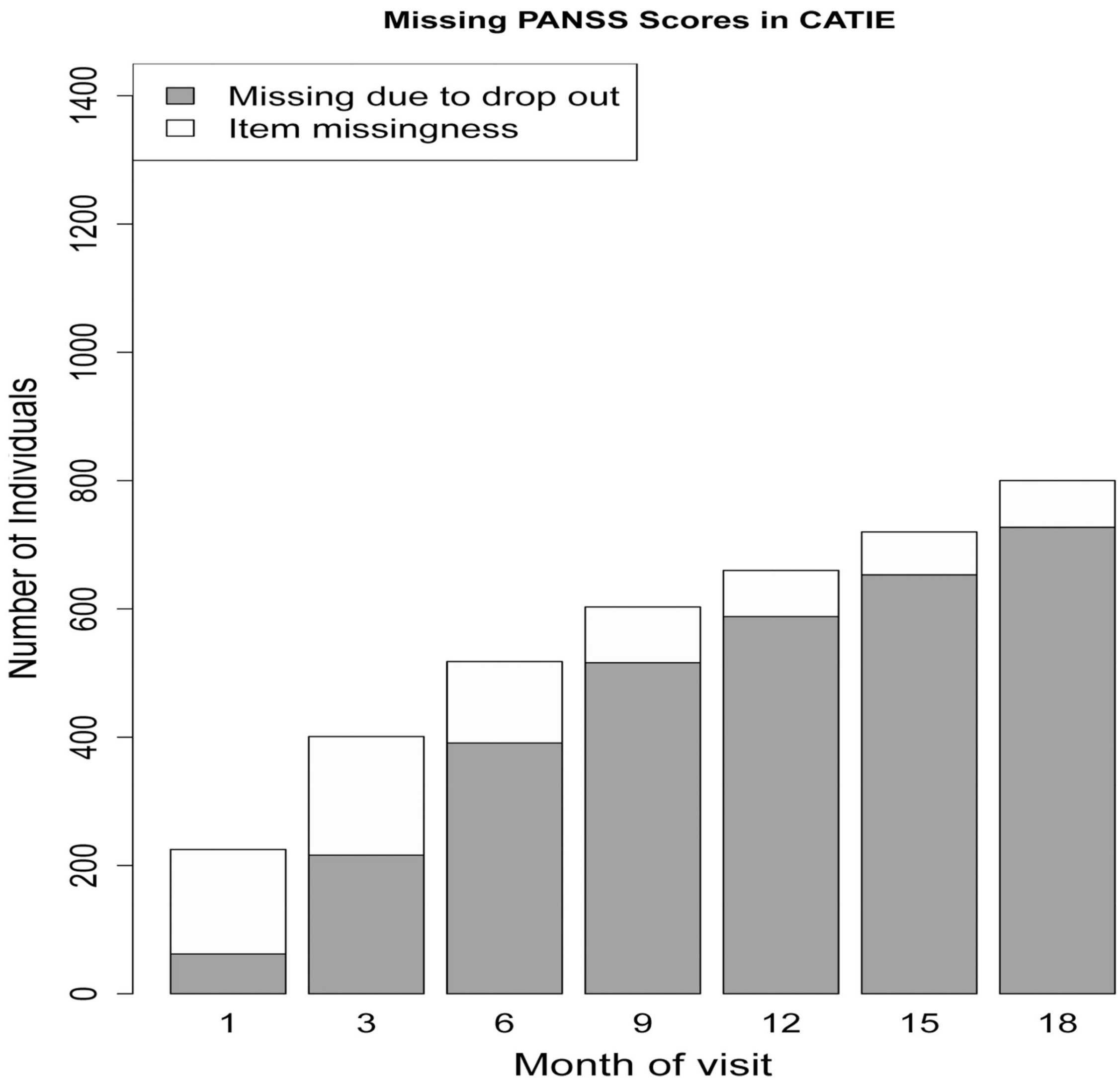
[Murphy et al., Neuropsychopharmacology, 2007]

Empirical study for schizophrenia

- Clinical Antipsychotic Trials of Intervention Effectiveness: 18 month multistage clinical trial of 1460 patients with schizophrenia – 2 stages
- Subjects randomly given a stage 1 treatment: olanzapine, risperidone, quetiapine, ziprasidone, and perphenazine
- Followed for up to 18 months; allowed to switch treatment if original was not effective:
  - Lack of *efficacy* (i.e., symptoms still high)
  - Lack of *tolerability* (i.e., side-effects large)
- Data recorded every 3 months (i.e., 6 time points)
- Reward at each time point: (negative) PANSS score (low PANSS score = few psychotic symptoms)

[Shortreed et al., *Mach Learn*, 2011]

Empirical study for schizophrenia



Most of the missing data is due to people dropping out of study prior to that month

[Shortreed et al., *Mach Learn*, 2011]

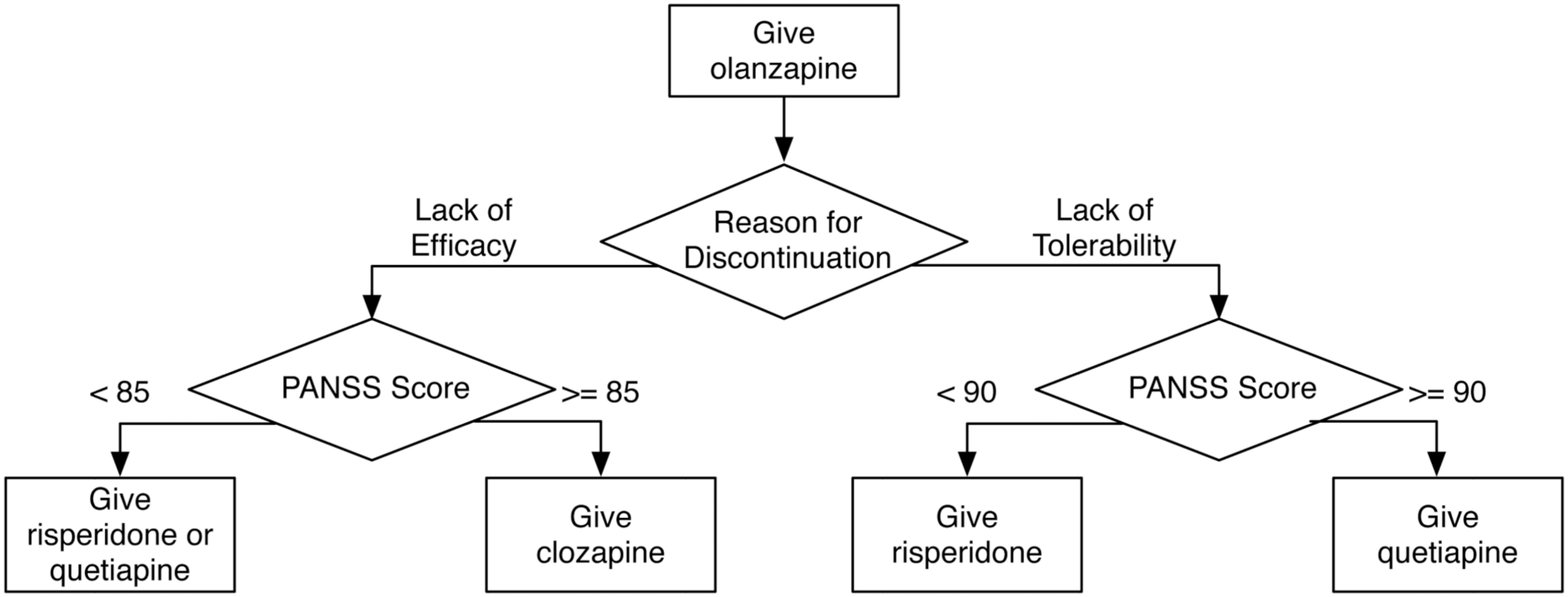
Empirical study for schizophrenia

- Data pre-processing:
  - Multiple imputation for the features (i.e. state)
  - Bayesian mixed effects model for PANSS score (i.e. reward)
- Fitted Q-iteration performed using *linear regression*
  - Different weight vector for each action (allows for *nonlinear* relationship between state and action)
  - Different weight vectors for each of the two time points
  - Weight sharing for variables not believed to be action specific but just helpful for estimating Q–function (e.g., tardive dyskinesia, recent psychotic episode, clinic site type)
- Bootstrap voting to get confidence intervals for treatment effects

[Shortreed et al., *Mach Learn*, 2011]

Empirical study for schizophrenia

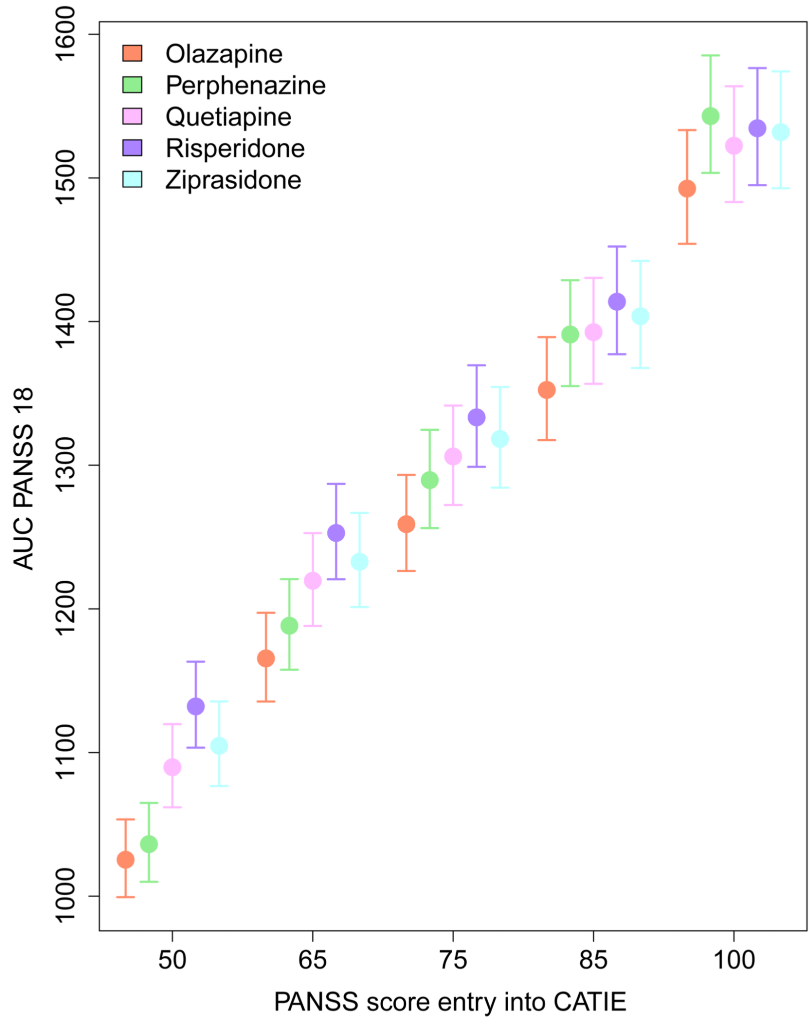
- Optimal treatment policy:



[Shortreed et al., *Mach Learn*, 2011]

Empirical study for schizophrenia

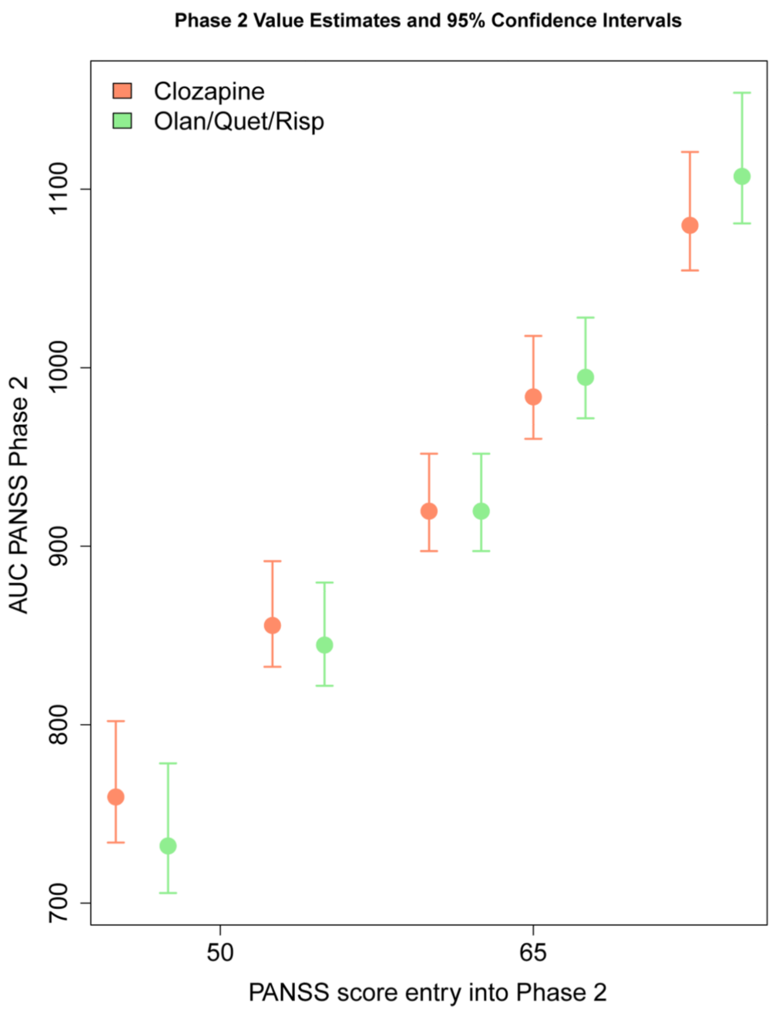
- Stage 1 stage-action value-function:

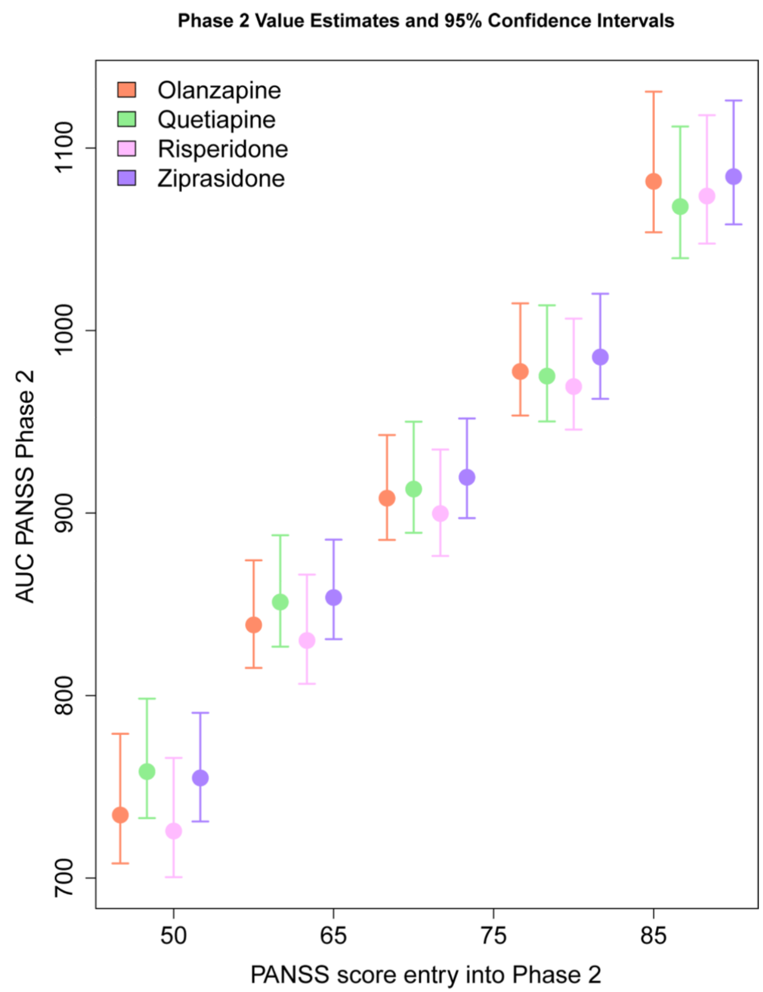


[Shortreed et al., *Mach Learn*, 2011]

Empirical study for schizophrenia

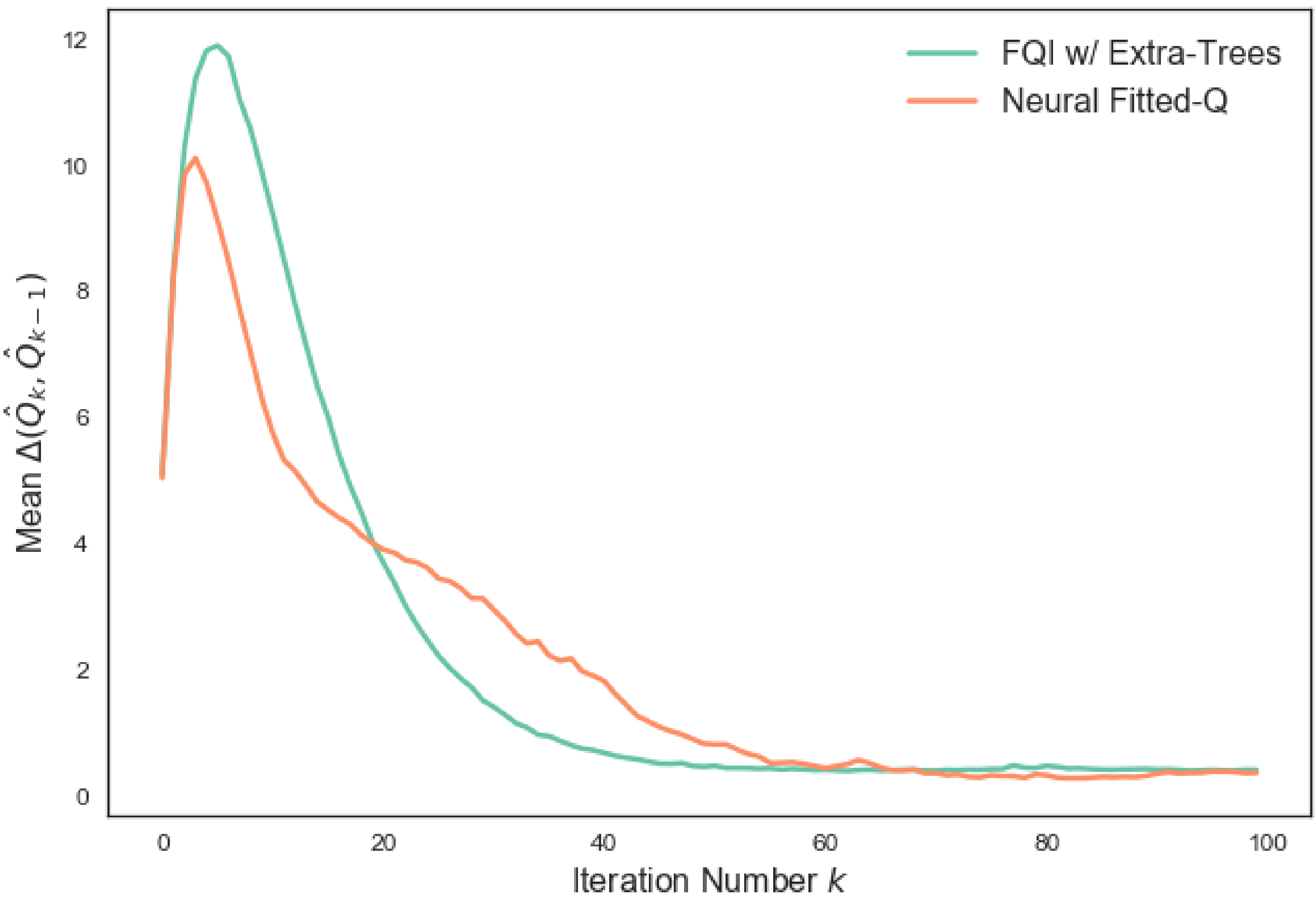
- Stage 2 stage-action value-function:





[Shortreed et al., *Mach Learn*, 2011]

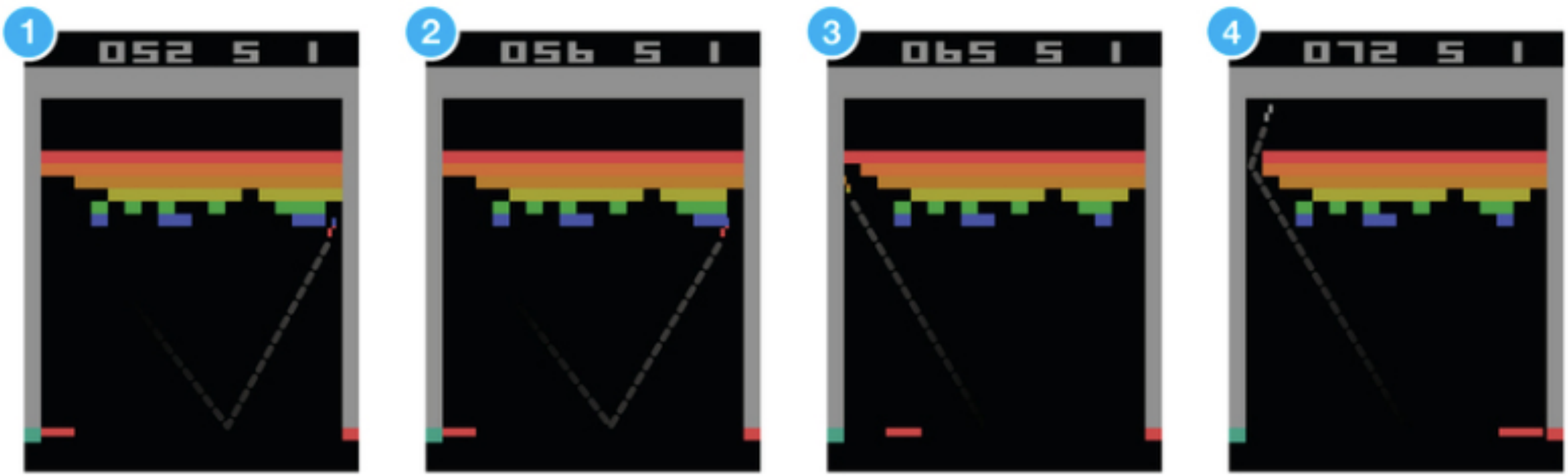
Measuring convergence in fitted Q-iteration



[Prasad et al., 2017]

Playing Atari with deep reinforcement learning

Game "Breakout": control paddle at bottom to break all bricks in upper half of screen



- Do fitted Q-iteration using deep convolutional neural networks to model the Q function
- Use eps-greedy algorithm to perform exploration
- Infinite time horizon

[Mnih et al., 2015]