CS287 Fall 2019 – Lecture 2

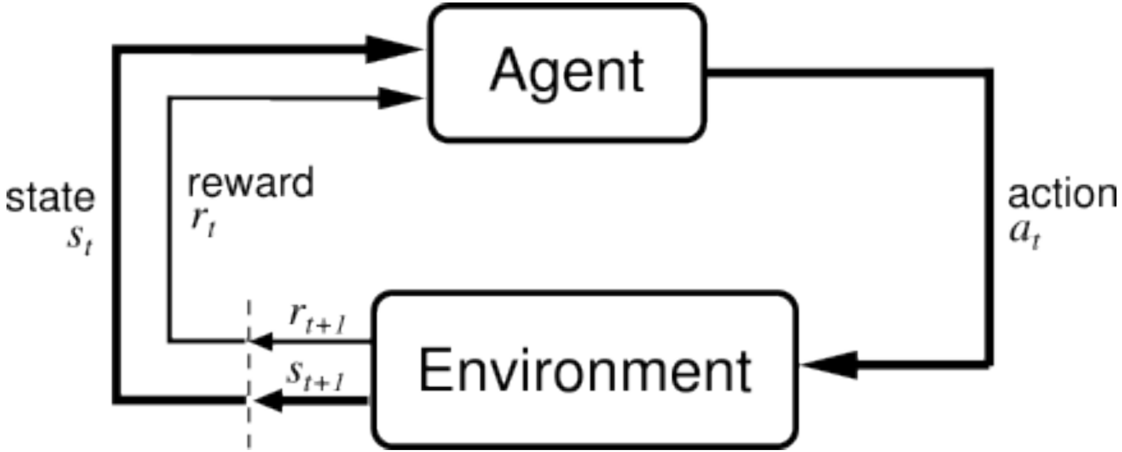
Markov Decision Processes
and
Exact Solution Methods

Pieter Abbeel
UC Berkeley EECS

Outline for Today's Lecture

- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - Value Iteration
  - Policy Iteration
  - Linear Programming
- Maximum Entropy Formulation
  - Entropy
  - Max-ent Formulation
  - Intermezzo on Constrained Optimization
  - Max-Ent Value Iteration

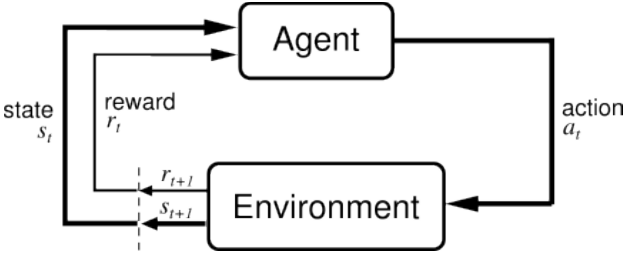
Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Markov Decision Process (S, A, T, R, γ, H)



Given:

- S: set of states
- A: set of actions
- T: S x A x S x {0,1,…,H} → [0,1] $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R: S x A x S x {0, 1, …, H} → $\mathbb{R}$ $R_t(s,a,s')$ = reward for $(s_{t+1} = s', s_t = s, a_t = a)$
- γ in (0,1]: discount factor H: horizon over which the agent will act

Goal:

- Find π*: S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg\max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R_t(S_t, A_t, S_{t+1}) | \pi]$$

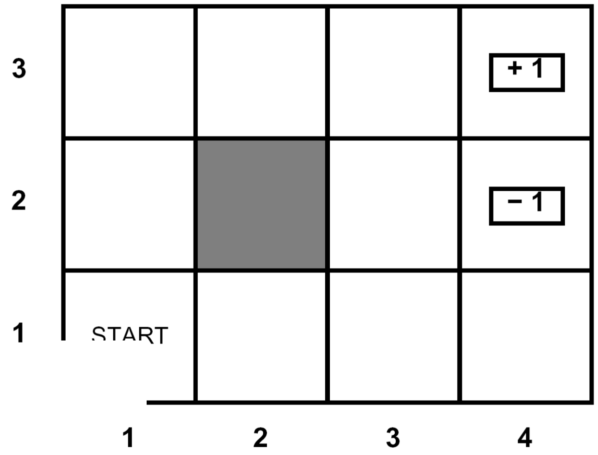
Examples

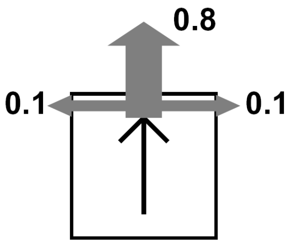
MDP (S, A, T, R, γ, H), goal: $max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$

- Cleaning robot
- Walking robot
- Pole balancing
- Games: tetris, backgammon
- Server management
- Shortest path problems
- Model for animals, people

Canonical Example: Grid World

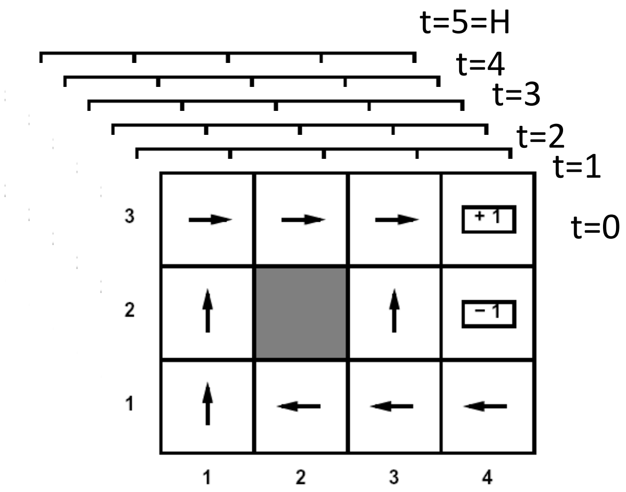
- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Big rewards come at the end





Solving MDPs

- In an MDP, we want to find an optimal policy π^*: S x 0:H → A
  - A policy π gives an action for each state for each time



  - An optimal policy maximizes expected sum of rewards

- Contrast: If environment were deterministic, then would just need an optimal plan, or sequence of actions, from start to a goal

Outline for Today's Lecture

- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - ***Value Iteration***
  - Policy Iteration
  - Linear Programming
- Maximum Entropy Formulation
  - Entropy
  - Max-ent Formulation
  - Intermezzo on Constrained Optimization
  - Max-Ent Value Iteration

For now: discrete state-action spaces as they are simpler to get the main concepts across.

We will consider continuous spaces next lecture!

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For i = 1, … , H

For all states s in S:

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^*(s')\right]$$

$$\pi_{i+1}^*(s) \leftarrow \arg\max_{a \in A} \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^*(s')\right]$$

This is called a value update or Bellman update/back-up

$V_i^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for i steps

$\pi_i^*(s)$ = optimal action when in state s and getting to act for i steps

Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

Value Iteration in Gridworld

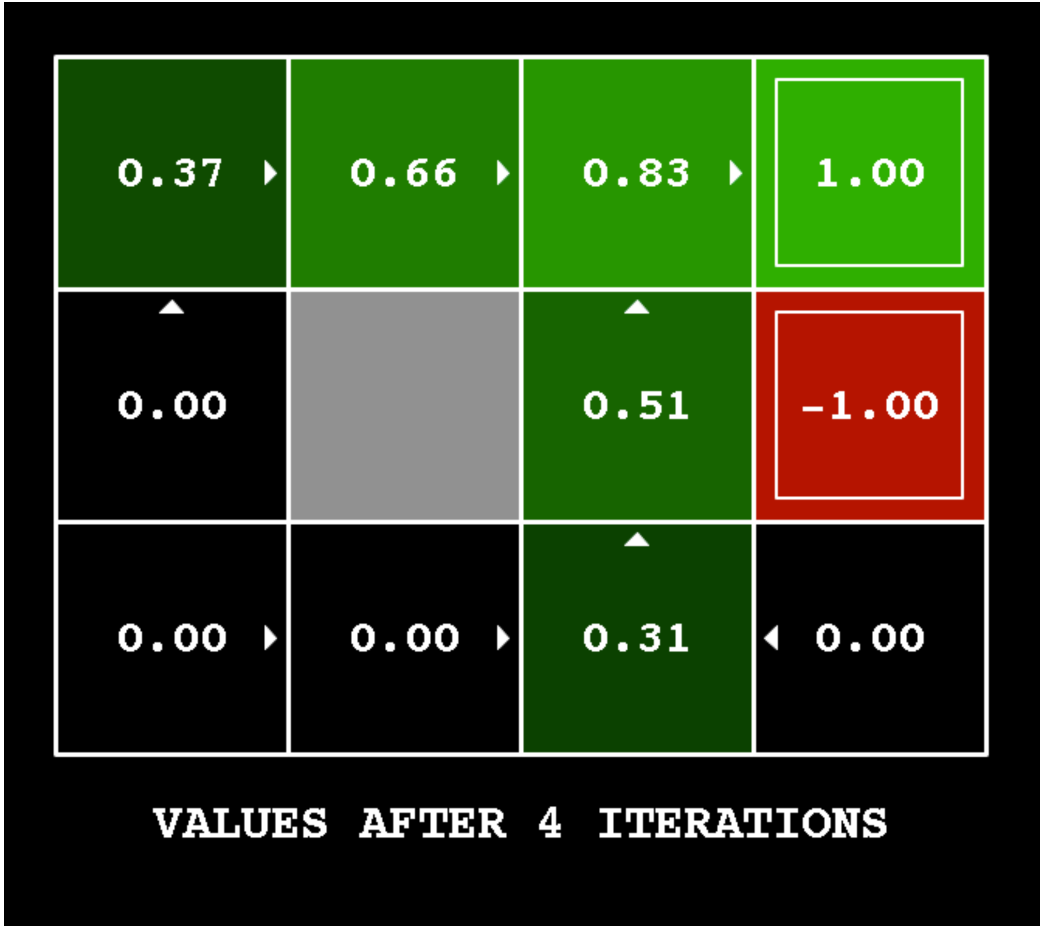
noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

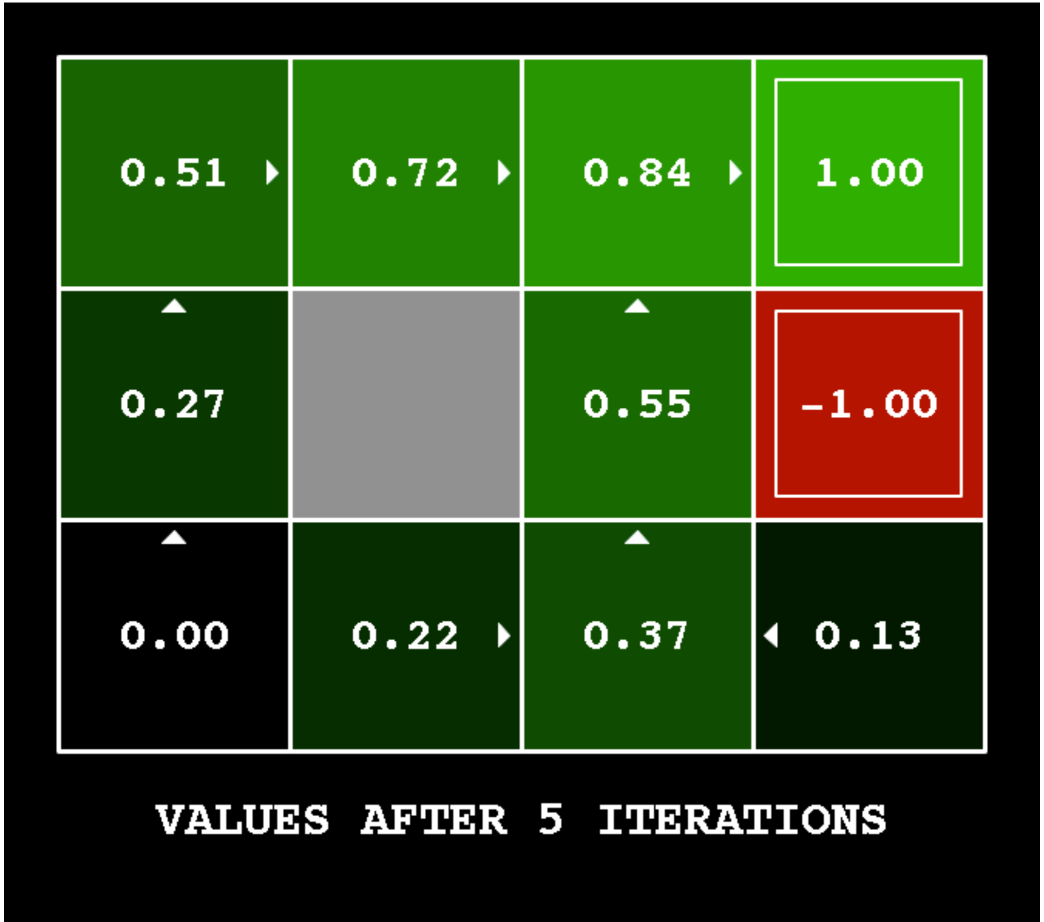
Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1



Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

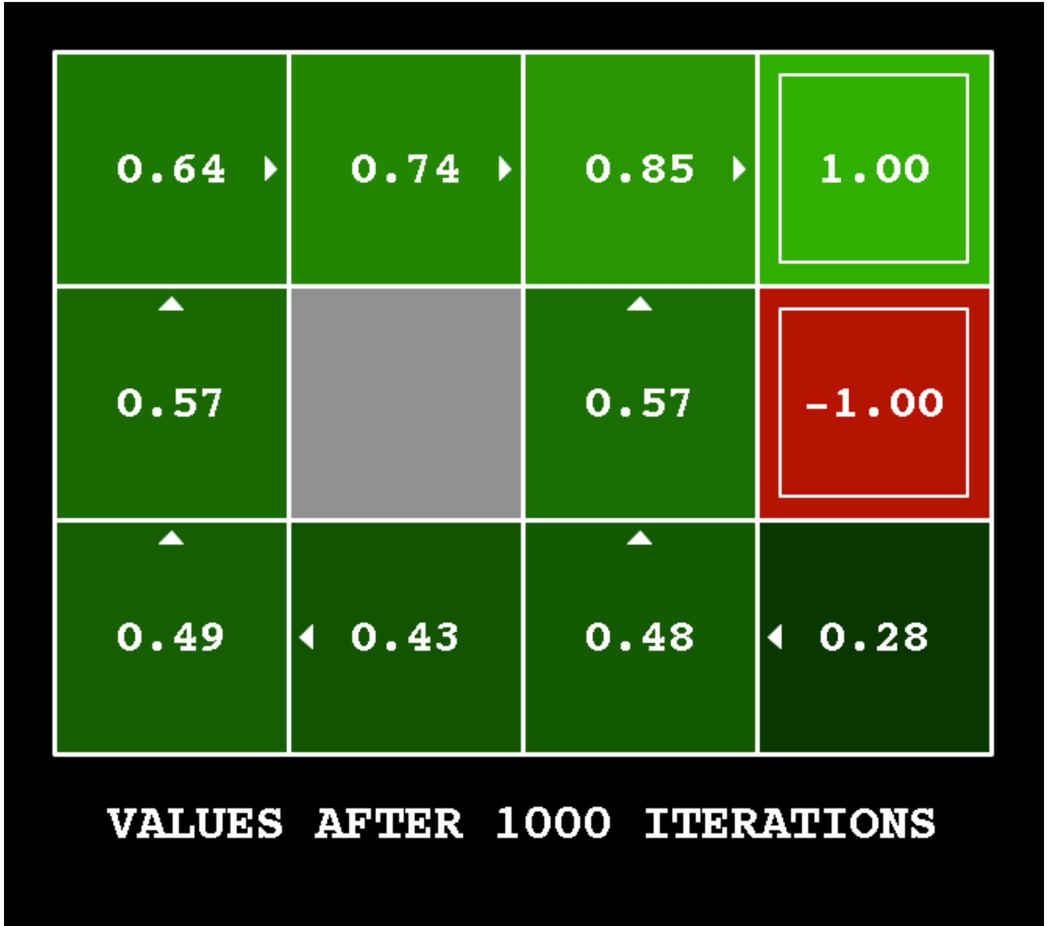


Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1



Value Iteration Convergence

Theorem. Value iteration converges. At convergence, we have found the optimal value function V* for the discounted infinite horizon problem, which satisfies the Bellman equations

$$\forall S \in S: \quad V^*(s) = \max_A \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

- Now we know how to act for infinite horizon with discounted rewards!
  - Run value iteration till convergence.
  - This produces V*, which in turn tells us how to act, namely following:

$$\pi^*(s) = \arg\max_{a \in A} \textstyle\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- Note: the infinite horizon optimal policy is stationary, i.e., the optimal action at a state s is the same action at all times. (Efficient to store!)

Convergence: Intuition

- $V^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for ∞ steps
- $V_H^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for H steps

- Additional reward collected over time steps H+1, H+2, …

$$\gamma^{H+1}R(s_{H+1}) + \gamma^{H+2}R(s_{H+2}) + \ldots \leq \gamma^{H+1}R_{max} + \gamma^{H+2}R_{max} + \ldots = \frac{\gamma^{H+1}}{1-\gamma}R_{max}$$

goes to zero as H goes to infinity

Hence $V_H^* \xrightarrow{H\to\infty} V*$

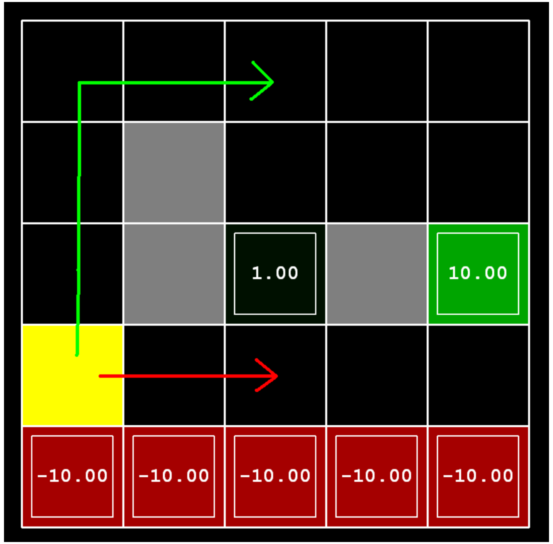
For simplicity of notation in the above it was assumed that rewards are always greater than or equal to zero. If rewards can be negative, a similar argument holds, using max |R| and bounding from both sides.

Convergence and Contractions

- Definition: max-norm: $||U|| = \max_s |U(s)|$
- Definition: An update operation is a γ-contraction in max-norm if and only if

  for all U_i, V_i: $||U_{i+1} - V_{i+1}|| \leq \gamma \, ||U_i - V_i||$
- Theorem: A contraction converges to a unique fixed point, no matter initialization.
- Fact: the value iteration update is a γ-contraction in max-norm
- Corollary: value iteration converges to a unique fixed point
- Additional fact: $||V_{i+1} - V_i|| < \epsilon, \Rightarrow ||V_{i+1} - V^*|| < 2\epsilon\gamma/(1-\gamma)$
  - I.e. once the update is small, it must also be close to converged

Exercise 1: Effect of Discount and Noise



(a) Prefer the close exit (+1), risking the cliff (-10)

(b) Prefer the close exit (+1), but avoiding the cliff (-10)

(c) Prefer the distant exit (+10), risking the cliff (-10)

(d) Prefer the distant exit (+10), avoiding the cliff (-10)

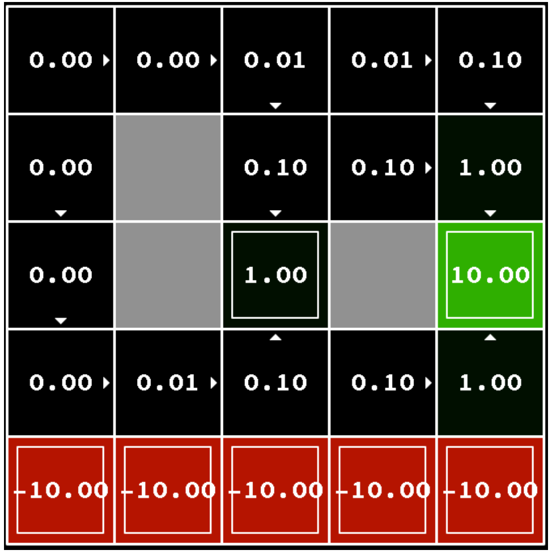
(1) $\gamma = 0.1$, noise = 0.5

(2) $\gamma = 0.99$, noise = 0

(3) $\gamma = 0.99$, noise = 0.5

(4) $\gamma = 0.1$, noise = 0

Exercise 1 Solution

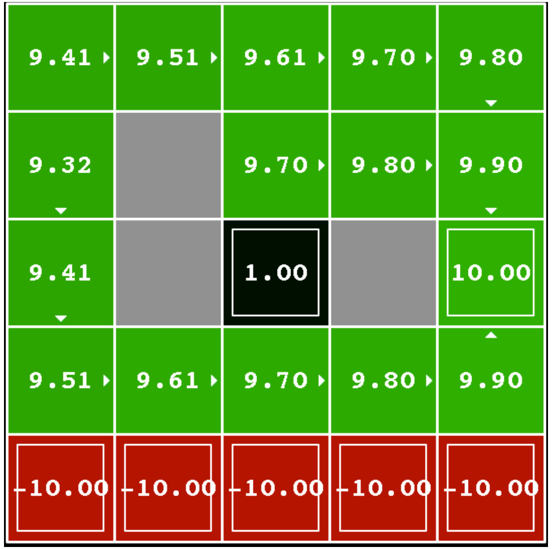


(a) Prefer close exit (+1), risking the cliff (-10) --- (4) γ = 0.1, noise = 0

Exercise 1 Solution

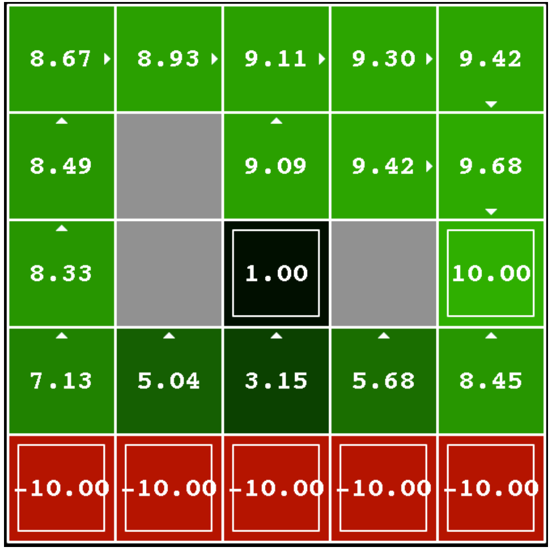
(b) Prefer close exit (+1), avoiding the cliff (-10) --- (1) $\gamma = 0.1$, noise = 0.5

Exercise 1 Solution



(c) Prefer distant exit (+1), risking the cliff (-10) --- (2) $\gamma = 0.99$, noise = 0

Exercise 1 Solution



(d) Prefer distant exit (+1), avoid the cliff (-10) --- (3) $\gamma = 0.99$, noise = 0.5

Outline for Today's Lecture

- ✓ Markov Decision Processes (MDPs)
- Exact Solution Methods
  - ✓ Value Iteration
  - ***Policy Iteration***
  - Linear Programming
- Maximum Entropy Formulation
  - Entropy
  - Max-ent Formulation
  - Intermezzo on Constrained Optimization
  - Max-Ent Value Iteration

For now: discrete state-action spaces as they are simpler to get the main concepts across.

We will consider continuous spaces next lecture!

Policy Evaluation

- Recall value iteration iterates:

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_i^*(s')]$$

- **Policy evaluation:**

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^\pi(s')]$$

At convergence:

$$\forall s \quad V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

Exercise 2

Consider a stochastic policy $\mu(a|s)$, where $\mu(a|s)$ is the probability of taking action a when in state s. Which of the following is the correct update to perform policy evaluation for this stochastic policy?

1. $V_{i+1}^{\mu}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')(R(s,a,s') + \gamma V_i^{\mu}(s'))$
2. $V_{i+1}^{\mu}(s) \leftarrow \sum_{s'} \sum_a \mu(a|s) T(s,a,s')(R(s,a,s') + \gamma V_i^{\mu}(s'))$
3. $V_{i+1}^{\mu}(s) \leftarrow \sum_a \mu(a|s) \max_{s'} T(s,a,s')(R(s,a,s') + \gamma V_i^{\mu}(s'))$

Policy Iteration

One iteration of policy iteration:

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates:
  - Iterate until values converge

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} T(s, \pi_k(s), s') \left[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

- Repeat until policy converges

- At convergence: optimal policy; and converges faster under some conditions

Policy Evaluation Revisited

- *Idea 1:* modify Bellman updates

$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^\pi(s')]$$

- Idea 2: it is just a linear system, solve with Matlab (or whatever)

  variables: $V^\pi(s)$

  constants: T, R

$$\forall s \quad V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

Policy Iteration Guarantees

Policy Iteration iterates over:

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates:
  - Iterate until values converge

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} T(s, \pi_k(s), s') \left[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s') \right]$$

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s') \right]$$

Theorem. Policy iteration is guaranteed to converge and at convergence, the current policy and its value function are the optimal policy and the optimal value function!

Proof sketch:

(1) *Guarantee to converge*: In every step the policy improves. This means that a given policy can be encountered at most once. This means that after we have iterated as many times as there are different policies, i.e., (number actions)$^{\text{(number states)}}$, we must be done and hence have converged.

(2) *Optimal at convergence*: by definition of convergence, at convergence $\pi_{k+1}(s) = \pi_k(s)$ for all states s. This means $\forall s \;\; V^{\pi_k}(s) = \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_i^{\pi_k}(s') \right]$

Hence V^{π_k} satisfies the Bellman equation, which means V^{π_k} is equal to the optimal value function V*.

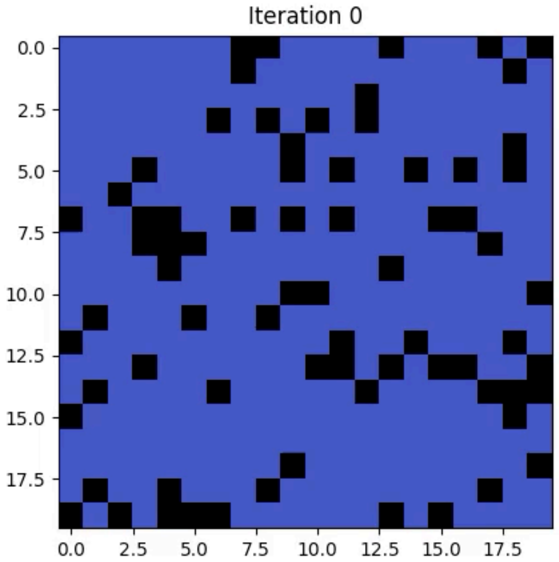
Outline for Today's Lecture

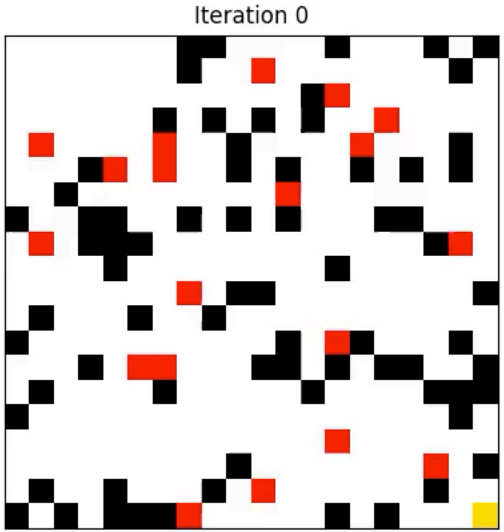
- ✓ Markov Decision Processes (MDPs)
- Exact Solution Methods
  - ✓ Value Iteration
  - ✓ Policy Iteration
  - Linear Programming
- ***Maximum Entropy Formulation***
  - Entropy
  - Max-ent Formulation
  - Intermezzo on Constrained Optimization
  - Max-ent Value Iteration

For now: discrete state-action spaces as they are simpler to get the main concepts across.

We will consider continuous spaces next lecture!

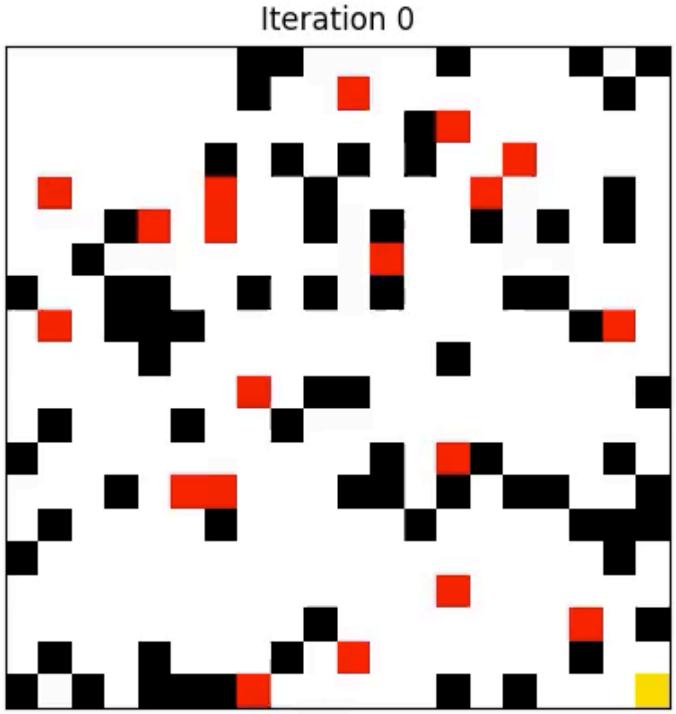
Obstacles Gridworld





- What if optimal path becomes blocked? Optimal policy fails.
- Is there any way to solve for a distribution rather than single solution? → more robust

What if we could find a “set of solutions”?



Entropy

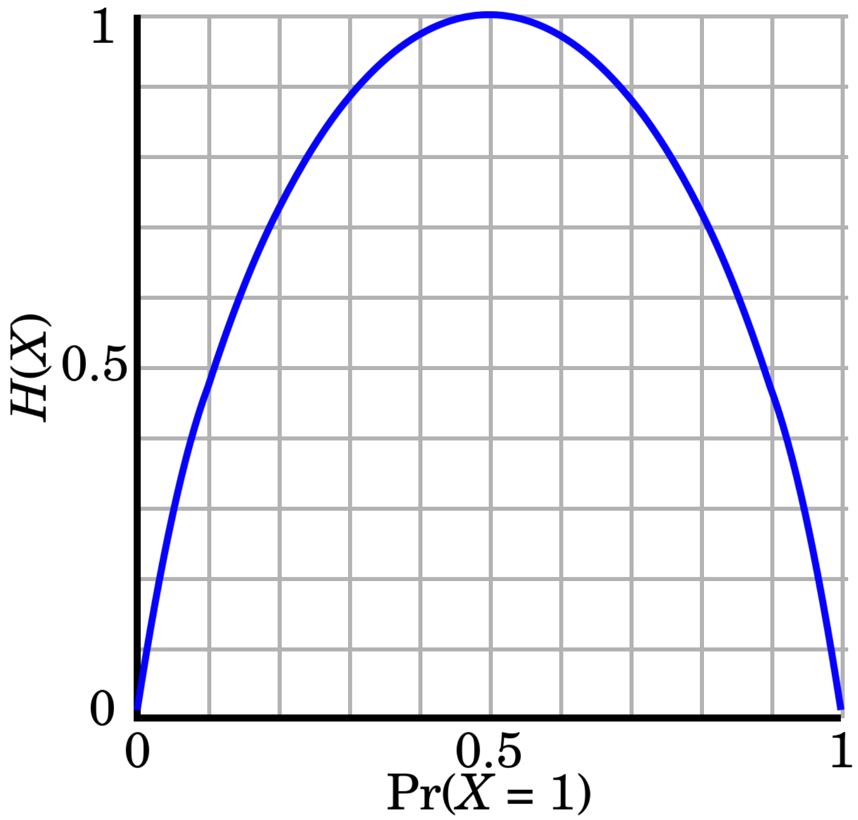
- Entropy = measure of uncertainty over random variable X

  = number of bits required to encode X (on average)

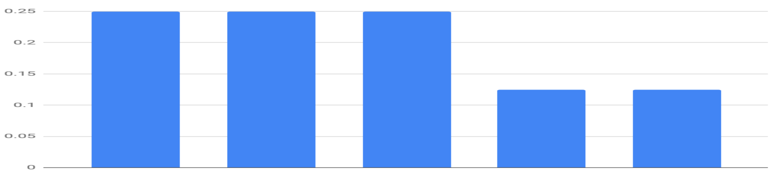
$$\mathcal{H}(X) = \sum_i p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_i p(x_i) \log_2 p(x_i)$$

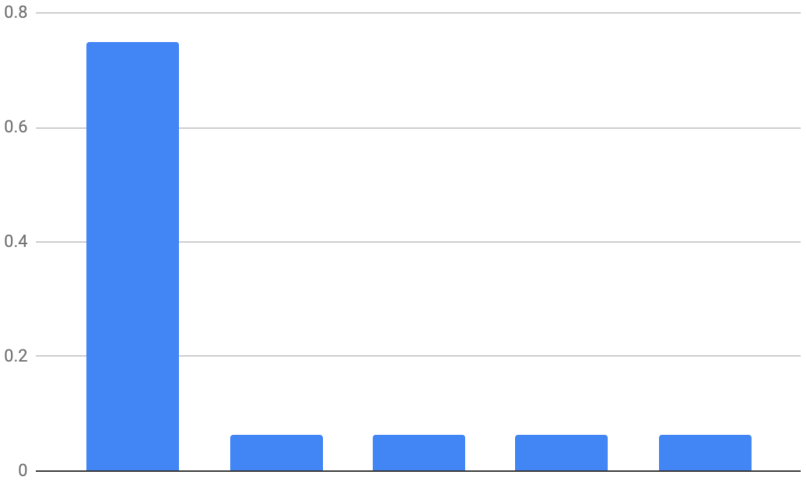
Entropy

E.g. binary random variable



Entropy





$$
\begin{aligned}
p(S) &= \{0.25, 0.25, 0.25, 0.125, 0.125\} \\
H &= 3 \times 0.25 \times \log_2 4 + 2 \times 0.125 \times \log_2 8 \\
&= 1.5 + 0.75 \\
&= 2.25
\end{aligned}
$$

$$
\begin{aligned}
p(s) &= \{0.75, 0.0625, 0.0625, 0.0625, 0.0625\} \\
H &= 0.75 \times \log_2(\frac{4}{3}) + 4 \times 0.0625 \times \log_2 16 \\
&= 0.3 + 1 \\
&= 1.3
\end{aligned}
$$

Maximum Entropy MDP

- Regular formulation:

$$\max_{\pi} E\left[\sum_{t=0}^{H} r_t\right]$$

- Max-ent formulation:

$$\max_{\pi} E\left[\sum_{t=0}^{H} r_t + \beta \mathcal{H}(\pi(\cdot \mid s_t))\right]$$

Max-ent Value Iteration

- But first need intermezzo on constrained optimization…

Constrained Optimization

- Original problem:

$$\max_{x} \quad f(x)$$
$$\text{s.t.} \quad g(x) = 0$$

- Lagrangian:

$$\max_{x} \min_{\lambda} \mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$$

- At optimum:

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial x} = 0$$

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial \lambda} = 0$$

Max-ent for 1-step problem

$$\max_{\pi(a)} \; E\left[r(a)\right] + \beta \mathcal{H}(\pi(a))$$

$$\max_{\pi(a)} \; \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a)$$

$$\max_{\pi(a)} \min_{\lambda} \; \mathcal{L}(\pi(a), \lambda) = \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a) + \lambda(\sum_a \pi(a) - 1)$$

$$\tfrac{\partial}{\partial \pi(a)} \mathcal{L}(\pi(a), \lambda) = 0$$

$$\tfrac{\partial}{\partial \pi(a)} \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a) + \lambda(\sum_a \pi(a) - 1) = 0$$

$$r(a) - \beta \log \pi(a) - \beta + \lambda = 0$$

$$\beta \log \pi(a) = r(a) - \beta + \lambda$$

$$\pi(a) = \exp\left[\tfrac{1}{\beta}(r(a) - \beta + \lambda)\right]$$

$$\tfrac{\partial}{\partial \lambda} \mathcal{L}(\pi(a), \lambda) = 0$$

$$\textstyle\sum_a \pi(a) - 1 = 0$$

$$\pi(a) = \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \qquad Z = \sum_a \exp(\frac{1}{\beta} r(a))$$

Max-ent for 1-step problem

$$\max_{\pi(a)} \; E\left[r(a)\right] + \beta \mathcal{H}(\pi(a))$$

$$\max_{\pi(a)} \; \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a)$$

$$\pi(a) = \tfrac{1}{Z} \exp(\tfrac{1}{\beta} r(a)) \qquad Z = \sum_a \exp(\frac{1}{\beta} r(a))$$

$$\begin{aligned}
V = & \quad \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) r(a) - \beta \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \log \left(\frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \right) \\
= & \quad \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \left(r(a) - \beta \log \left(\exp(\frac{1}{\beta} r(a)) \right) \right) - \beta \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \log \frac{1}{Z} \\
= & \quad 0 - \beta \log \textstyle\frac{1}{Z} \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \\
= & \quad -\beta \log \textstyle\frac{1}{Z} \\
= & \quad \beta \log \textstyle\sum_a \exp(\frac{1}{\beta} r(a))
\end{aligned}$$

= softmax

Max-ent Value Iteration

$$\max_{\pi} E\left[\sum_{t=0}^{H} r_t + \beta \mathcal{H}(\pi(\cdot \mid s_t))\right]$$

$$V_k(s) = \max_{\pi} E\left[\sum_{t=H-k}^{H} r(s_t, a_t) + \beta \mathcal{H}(\pi(a_t|s_t)\right]$$

$$\begin{aligned} V_k(s) = & \quad \max_{\pi} E\left[r(s,a) + \beta \mathcal{H}(\pi(a|s) + V_{k-1}(s')\right] \\ = & \quad \max_{\pi} E\left[Q_k(s,a) + \beta \mathcal{H}(\pi(a|s)\right] \end{aligned}$$

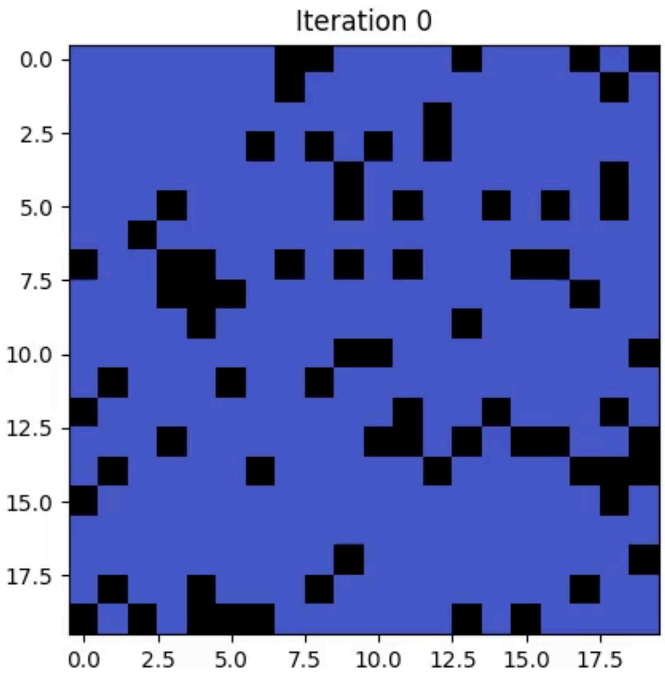
$$Q_k(s,a) = E\left[r(s,a) + V_{k-1}(s')\right]$$

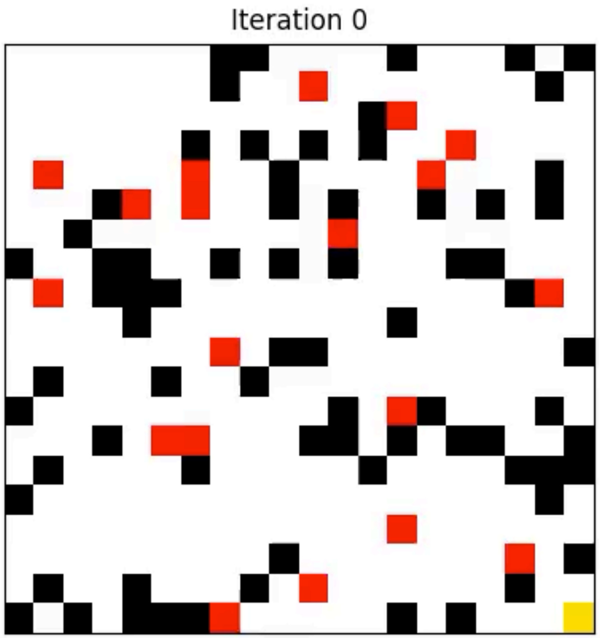
= 1-step problem (with Q instead of r), so we can directly transcribe solution:

$$V_k(s) = \beta \log \sum_a \exp(\frac{1}{\beta} Q_k(s,a))$$

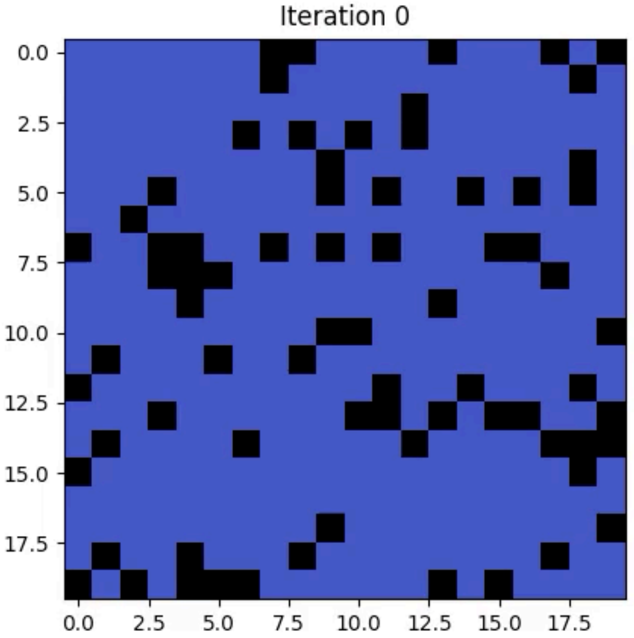
$$\pi_k(a|s) = \frac{1}{Z} \exp(\frac{1}{\beta} Q_k(s,a))$$

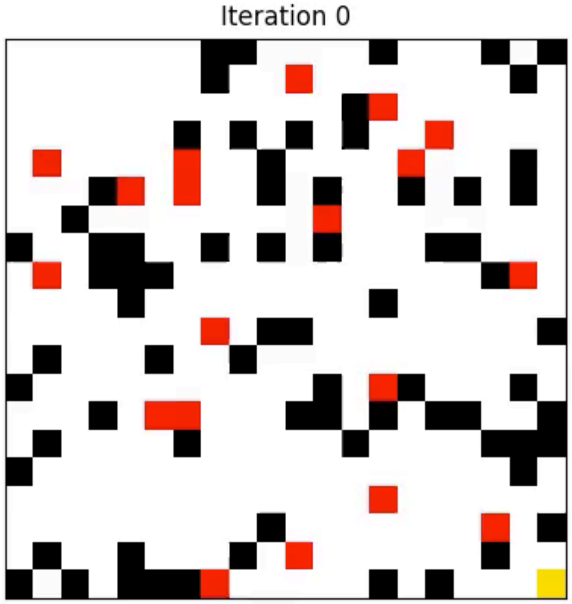
Maxent in Our Obstacles Gridworld (T=1)



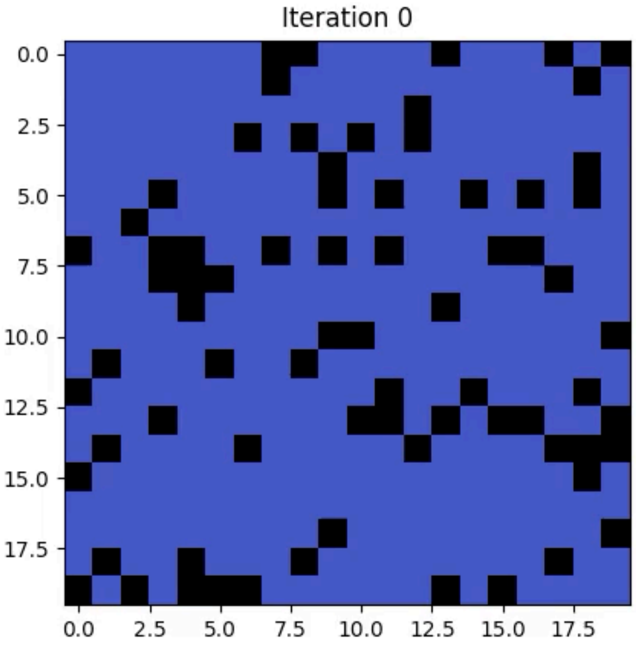


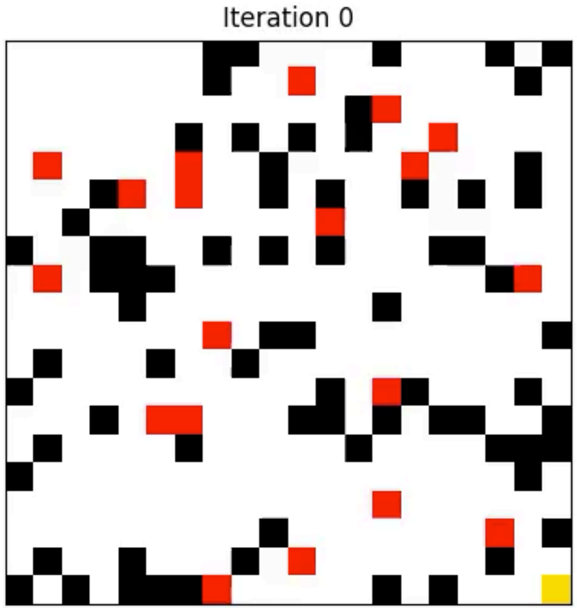
Maxent in Our Obstacles Gridworld (T=1e-2)





Maxent in Our Obstacles Gridworld (T=0)





Outline for Today's Lecture

- ✓ Markov Decision Processes (MDPs)
- Exact Solution Methods
  - ✓ Value Iteration
  - ✓ Policy Iteration
  - Linear Programming
- ✓ ***Maximum Entropy Formulation***
  - ✓ Entropy
  - ✓ Max-ent Formulation
  - ✓ Intermezzo on Constrained Optimization
  - ✓ Max-ent Value Iteration

For now: discrete state-action spaces as they are simpler to get the main concepts across.

We will consider continuous spaces next lecture!

Infinite Horizon Linear Program

- Recall, at value iteration convergence we have

$$\forall s \in S: \quad V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

- LP formulation to find V^*:

$$\begin{aligned} \min_V \quad & \sum_s \mu_0(s) V(s) \\ \text{s.t.} \quad & \forall s \in S, \forall a \in A: \\ & V(s) \geq \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right] \end{aligned}$$

μ_0 is a probability distribution over S, with $\mu_0(s) > 0$ for all s in S.

Theorem. V^* is the solution to the above LP.

Theorem Proof

Let F be the Bellman operator, i.e., $V^*_{i+1} = F(V_i)$. Then the LP can be written as:

$$\begin{array}{ll} \min_V & \mu_0^\top V \\ \text{s.t.} & V \geq F(V) \end{array}$$

Monotonicity Property: If $U \geq V$ then $F(U) \geq F(V)$.
Hence, if $V \geq F(V)$ then $F(V) \geq F(F(V))$, and by repeated application, $V \geq F(V) \geq F^2V \geq F^3V \geq \ldots \geq F^\infty V = V^*$.
Any feasible solution to the LP must satisfy $V \geq F(V)$, and hence must satisfy $V \geq V^*$. Hence, assuming all entries in μ_0 are positive, V^* is the optimal solution to the LP.

Exercise 3

- How about:

$$\begin{aligned} \max_V \quad & \mu_0^\top V \\ \text{s.t.} \quad & V \leq F(V) \end{aligned}$$

Dual Linear Program

$$\max_\lambda \quad \sum_{s\in S}\sum_{a\in A}\sum_{s'\in S}\lambda(s,a)T(s,a,s')R(s,a,s')$$

$$\text{s.t.} \quad \forall s' \in S: \sum_{a'\in A}\lambda(s',a') = \mu_0(s) + \gamma\sum_{s\in S}\sum_{a\in A}\lambda(s,a)T(s,a,s')$$

- Interpretation:
  - $\lambda(s,a) = \sum_{t=0}^{\infty}\gamma^t P(s_t = s, a_t = a)$
  - Equation 2: ensures that λ has the above meaning
  - Equation 1: maximize expected discounted sum of rewards
- Optimal policy: $\pi^*(s) = \arg\max_a \lambda(s,a)$

Outline for Today's Lecture

- ✓ Markov Decision Processes (MDPs)
- Exact Solution Methods
  - ✓ Value Iteration
  - ✓ Policy Iteration
  - ✓ Linear Programming
- ✓ ***Maximum Entropy Formulation***
  - ✓ Entropy
  - ✓ Max-ent Formulation
  - ✓ Intermezzo on Constrained Optimization
  - ✓ Max-ent Value Iteration

For now: discrete state-action spaces as they are simpler to get the main concepts across.

We will consider continuous spaces next lecture!

Today and Forthcoming Lectures

- Optimal control: provides general computational approach to tackle control problems.
  - Dynamic programming / Value iteration
    - ✓ Discrete state spaces – Exact methods
    - Continuous state spaces – Approximate solutions through discretization
    - Large state spaces – Approximate solutions through function approximation
    - Linear systems – Closed form exact solution with LQR
    - Nonlinear systems – How to extend the exact solutions for linear systems:
      - Local linearization
      - iLQR, Differential dynamic programming
  - Optimal Control through Nonlinear Optimization
    - Shooting <> Collocation formulations
    - Model Predictive Control (MPC)
  - Examples: