# Reminders

- 1 week until the American election. I voted. Did you? If you haven't returned your PA mail-in ballot yet, drop it off at one of these locations:
  https://www.votespa.com/Voting-in-PA/pages/drop-box.aspx

- Today is the last day to vote early!
  https://www.votespa.com/Voting-in-PA/Pages/Early-Voting.aspx

- The extra credit for voting / civic engagement is now available (due before 8pm on election day).   If you're a foreign student, you have two options:
  1) Visit Independence Hall in Philadelphia
  2) Watch a documentary about the history of voting in the USA.

- Midterm is due tomorrow before 8am Eastern.

- You can opt in to having a partner on future HWs. Partners will be randomly assigned, and you'll get a new partner each HW assignment.
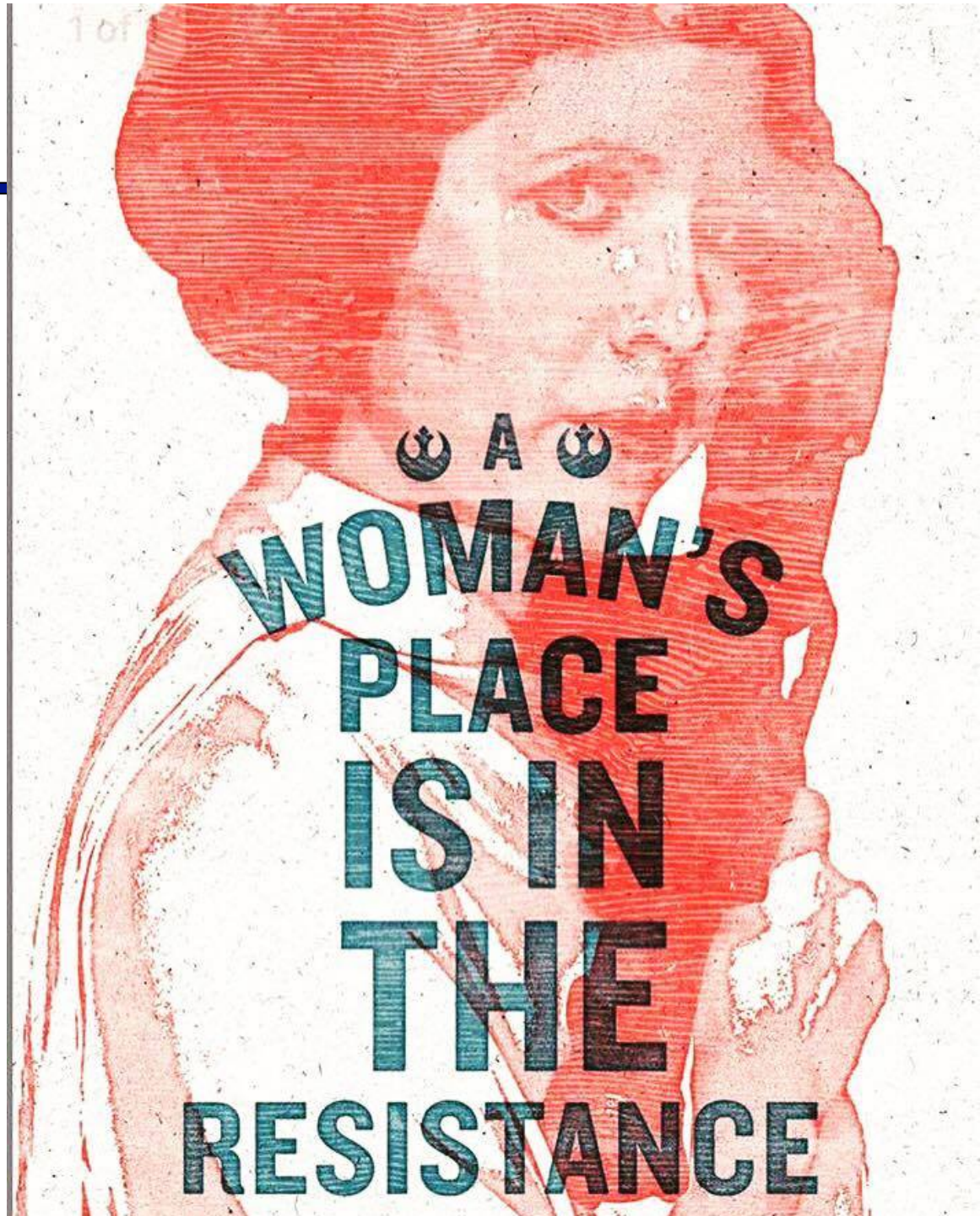
Reinforcement Learning
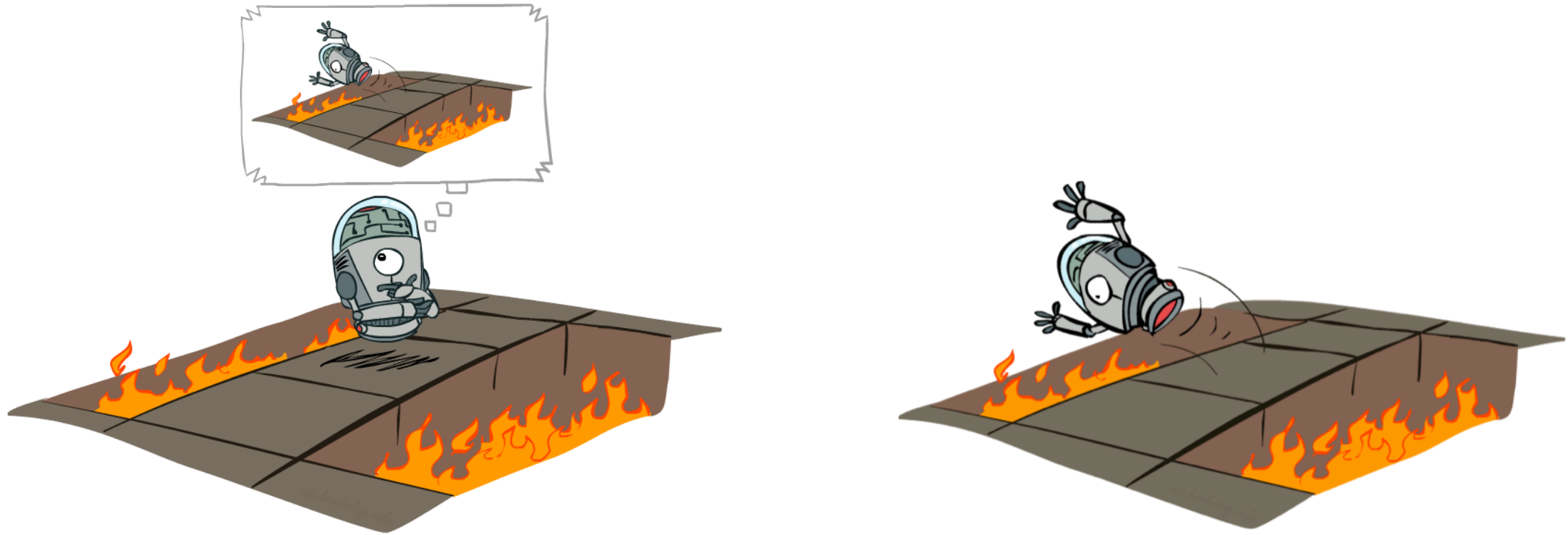
Slides courtesy of Dan Klein and Pieter Abbeel

University of California, Berkeley

Active Reinforcement Learning

- **Full reinforcement learning: optimal policies (like value iteration)**
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You choose the actions now
  - Goal: learn the optimal policy / values

- **In this case:**
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning! You actually take actions in the world and find out what happens…
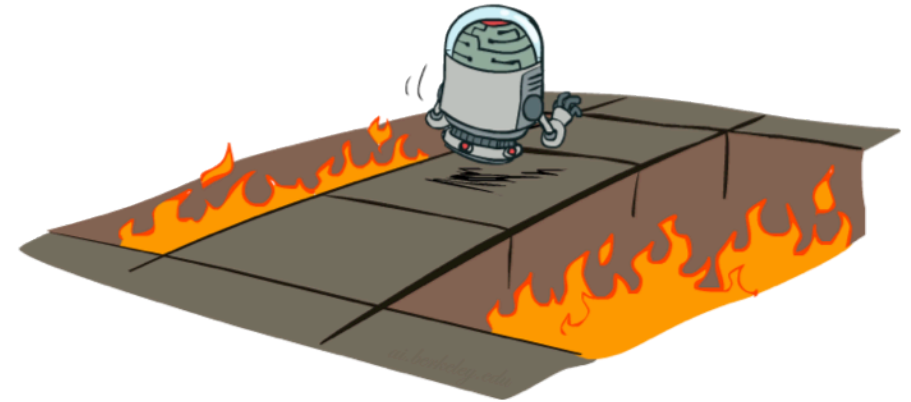
Detour: Q-Value Iteration

- Value iteration: find successive (depth-limited) values
  - Start with $V_0(s) = 0$, which we know is right
  - Given V_k, calculate the depth k+1 values for all states:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s') \right]$$

- But Q-values are more useful, so compute them instead
  - Start with $Q_0(s,a) = 0$, which we know is right
  - Given Q_k, calculate the depth k+1 q-values for all q-states:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Q-Learning

- Q-Learning: sample-based Q-value iteration

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$
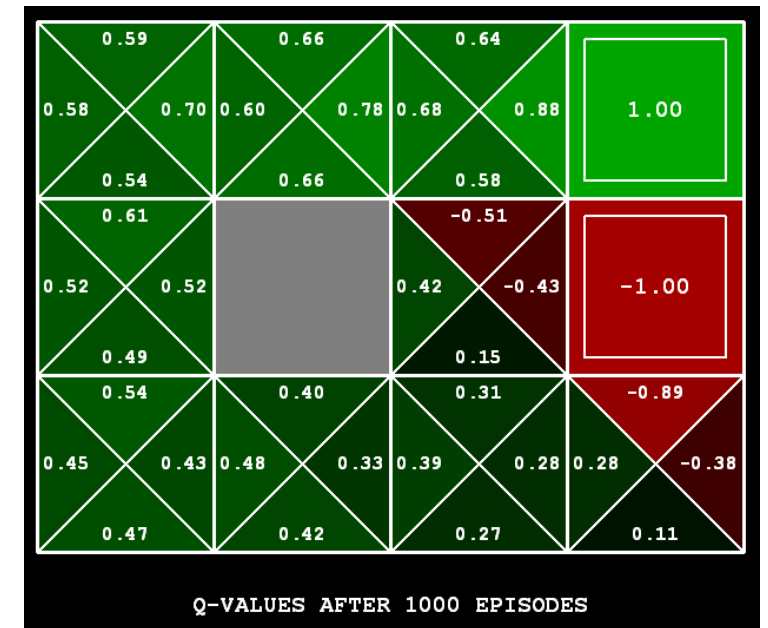
- Learn Q(s,a) values as you go

  - Receive a sample (s,a,s',r)

  - Consider your old estimate: $Q(s,a)$

  - Consider your new sample estimate:

  $$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

  $$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[sample\right]$$


Q-VALUES AFTER 1000 EPISODES

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you're acting suboptimally!

- This is called off-policy learning

- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - ... but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions (!)
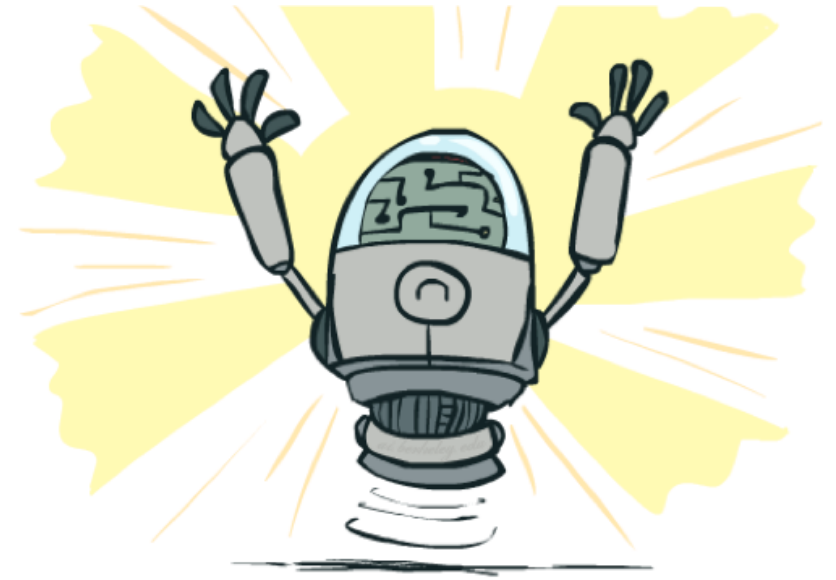
Exploration vs. Exploitation
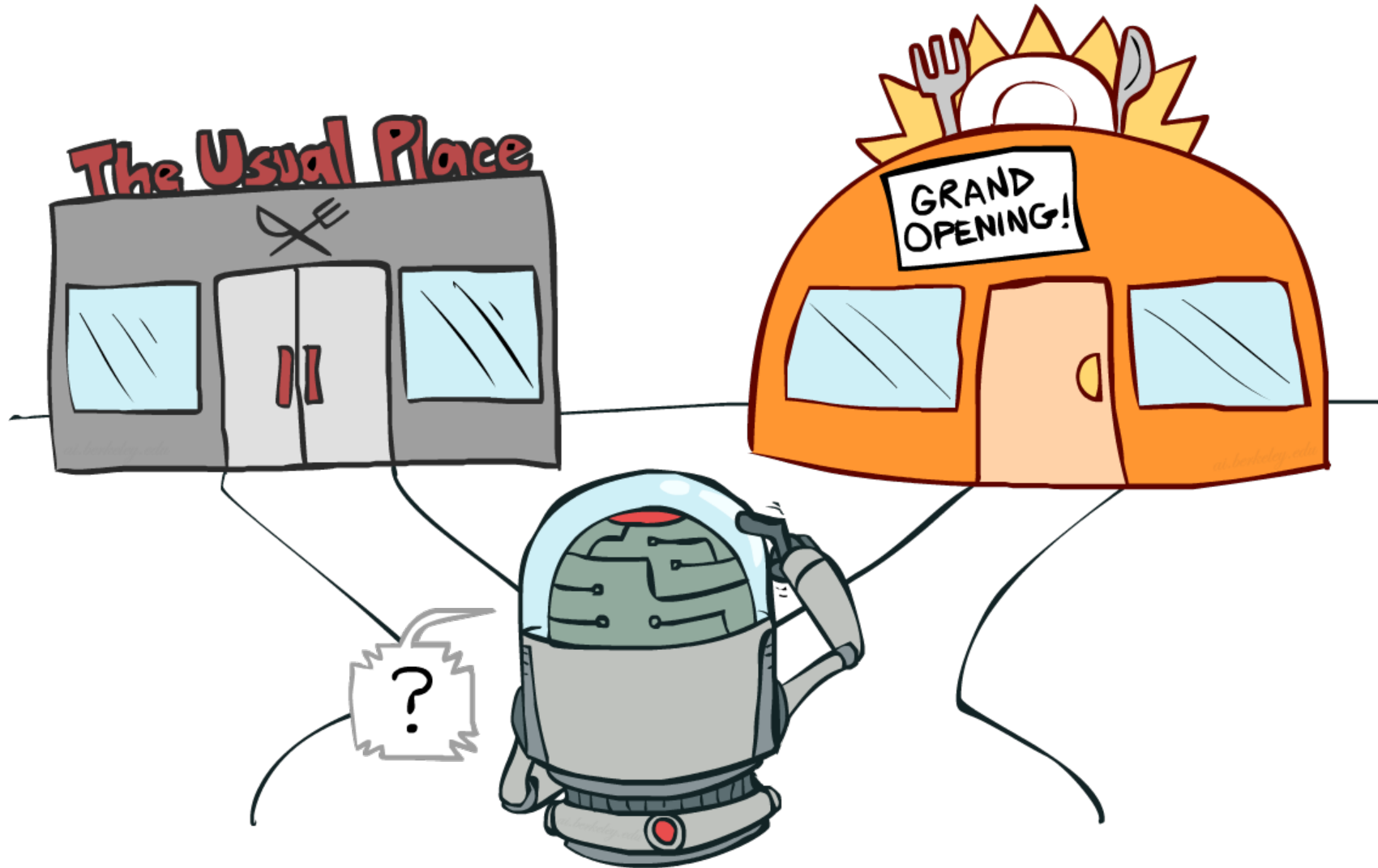
How to Explore?

- **Several schemes for forcing exploration**
  - Simplest: random actions (ε-greedy)
    - Every time step, flip a coin
    - With (small) probability ε, act randomly
    - With (large) probability $1-\varepsilon$, act on current policy

How to Explore?

- **Several schemes for forcing exploration**
  - Simplest: random actions (ε-greedy)
    - Every time step, flip a coin
    - With (small) probability ε, act randomly
    - With (large) probability $1-\varepsilon$, act on current policy
  - Problems with random actions?
    - You do eventually explore the space, but keep thrashing around once learning is done
    - One solution: lower ε over time
    - Another solution: exploration functions

Exploration Functions

- ## When to explore?
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring

- ## Exploration function
  - Takes a value estimate u and a visit count n, and returns an optimistic utility, e.g. $f(u, n) = u + k/n$

   Regular Q-Update: $\quad Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} Q(s',a')$

    Modified Q-Update: $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

  - Note: this propagates the "bonus" back to states that lead to unknown states as well!

Regret

- Even if you learn the optimal policy, you still make mistakes along the way

- Regret is a measure of your total mistake cost: the difference between your (expected) rewards, including youthful suboptimality, and optimal (expected) rewards

- Minimizing regret goes beyond learning to be optimal – it requires optimally learning to be optimal

- Example: random exploration and exploration functions both end up optimal, but random exploration has higher regret
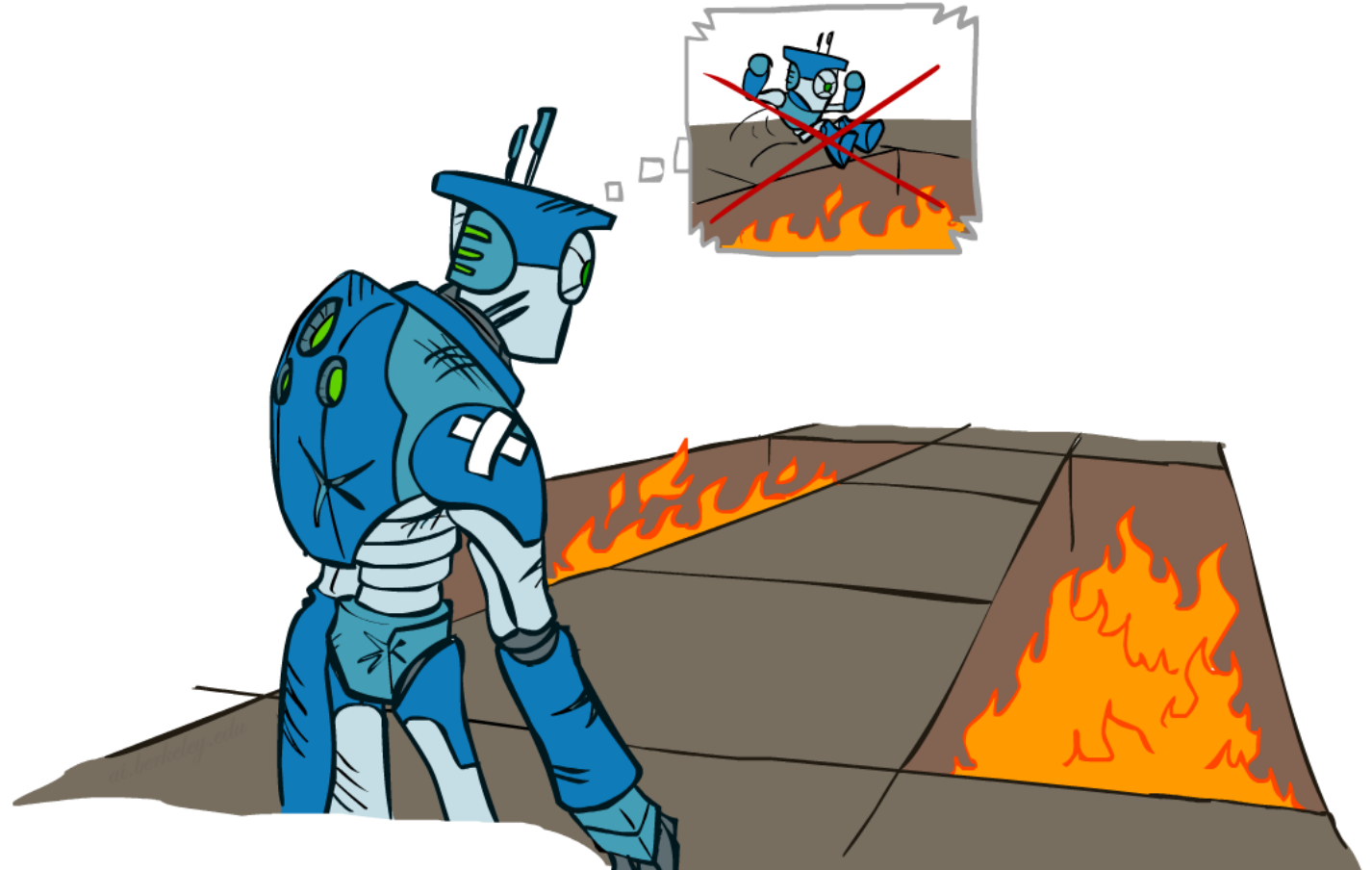
Approximate Q-Learning
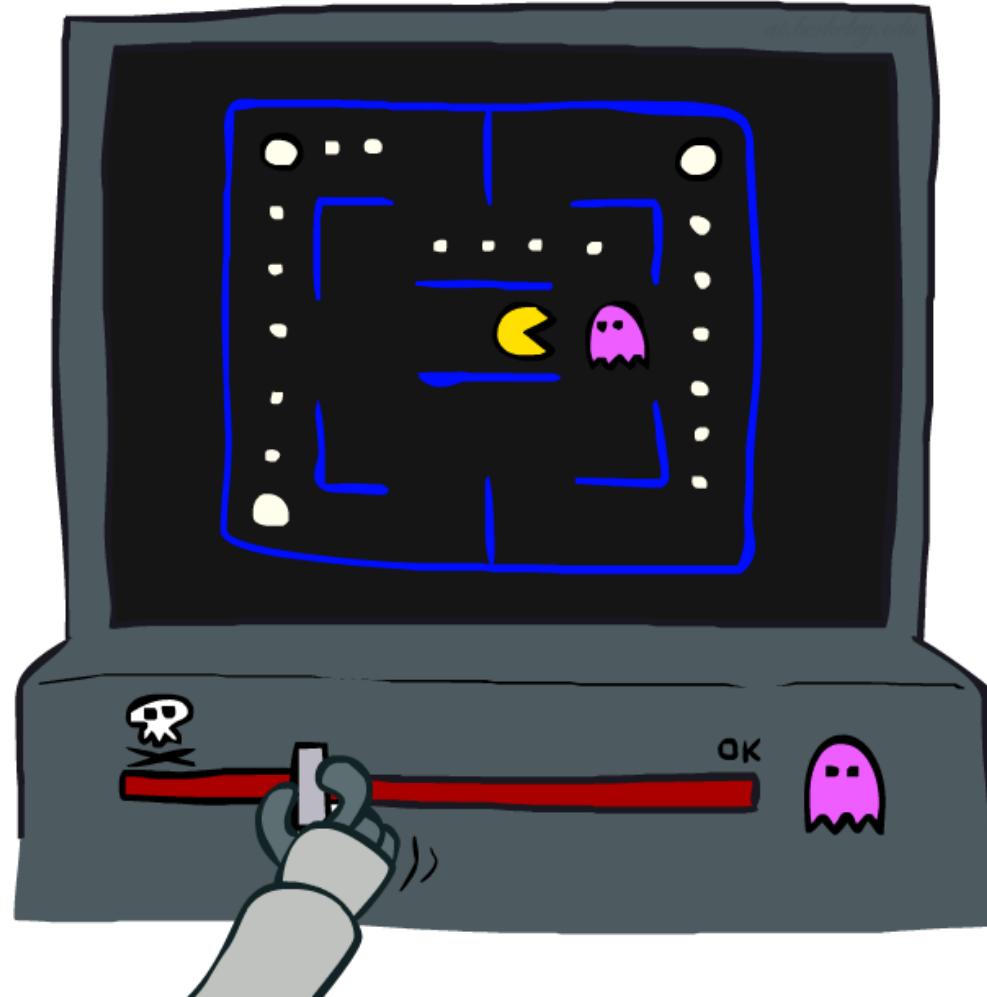
Generalizing Across States

- Basic Q-Learning keeps a table of all q-values

- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory

- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations
  - This is a fundamental idea in machine learning, and we'll see it over and over again
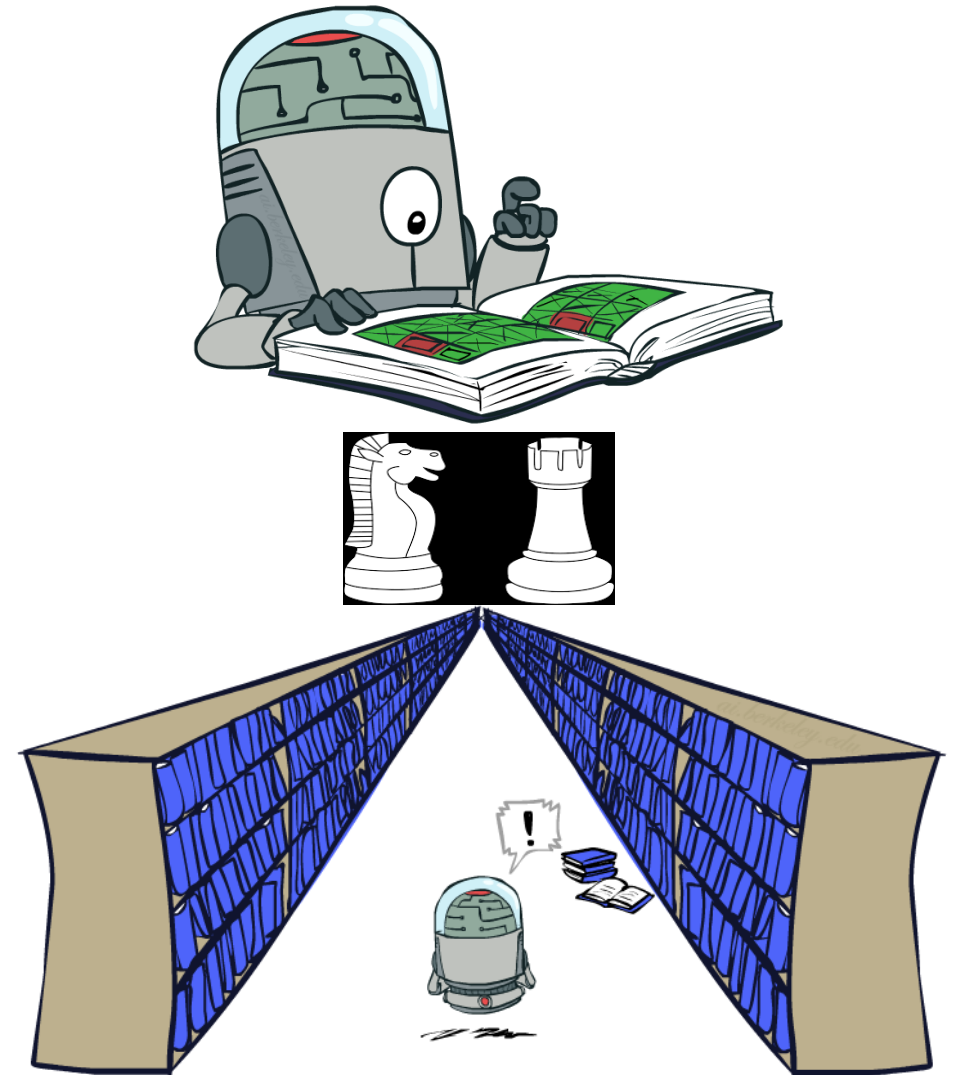
Flashback: Evaluation Functions

- Evaluation functions score non-terminals in depth-limited search

Black to move

White slightly better

White to move

Black winning
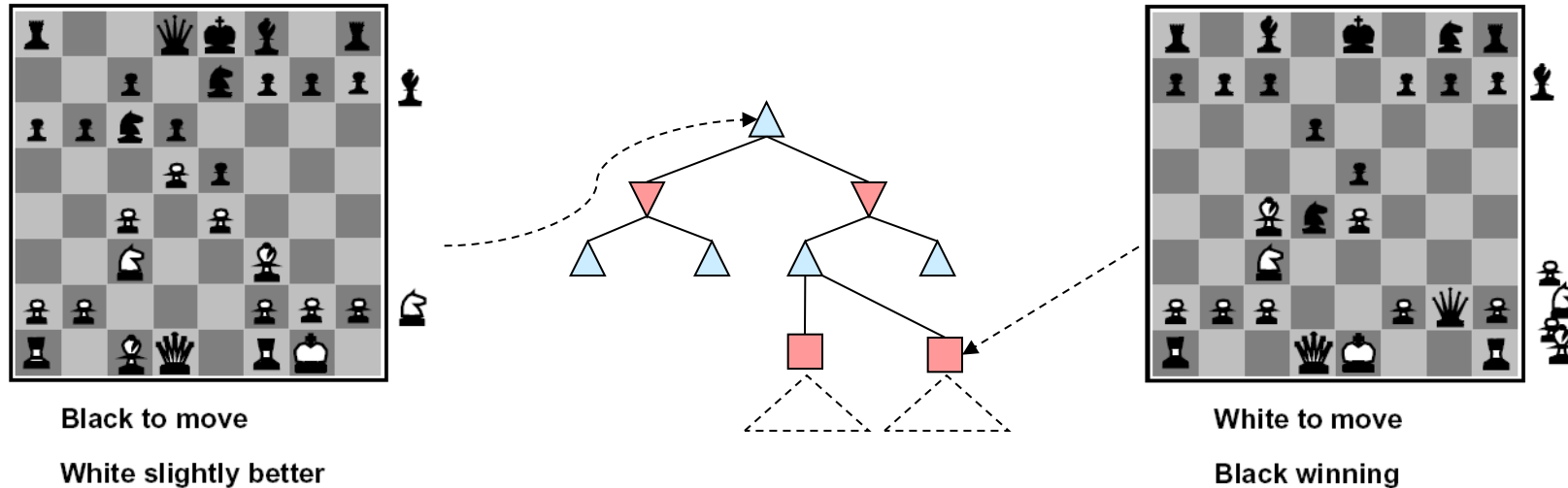
- Ideal function: returns the actual minimax value of the position
- In practice: typically weighted linear sum of features:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

- e.g. $f_1(s)$ = (num white queens – num black queens), etc.

Linear Value Functions

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Advantage: our experience is summed up in a few powerful numbers

- Disadvantage: states may share features but actually be very different in value!

Approximate Q-Learning

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \ldots + w_n f_n(s, a)$$

- Q-learning with linear Q-functions:

  $$\text{transition} = (s, a, r, s')$$

  $$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a') \right] - Q(s, a)$$

  $$Q(s, a) \leftarrow Q(s, a) + \alpha \, [\text{difference}] \qquad \text{Exact Q's}$$

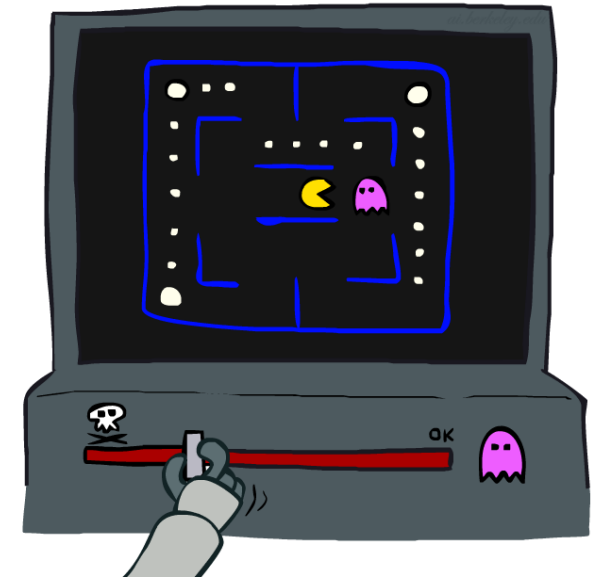 $$w_i \leftarrow w_i + \alpha \, [\text{difference}] \, f_i(s, a) \qquad \text{Approximate Q's}$$
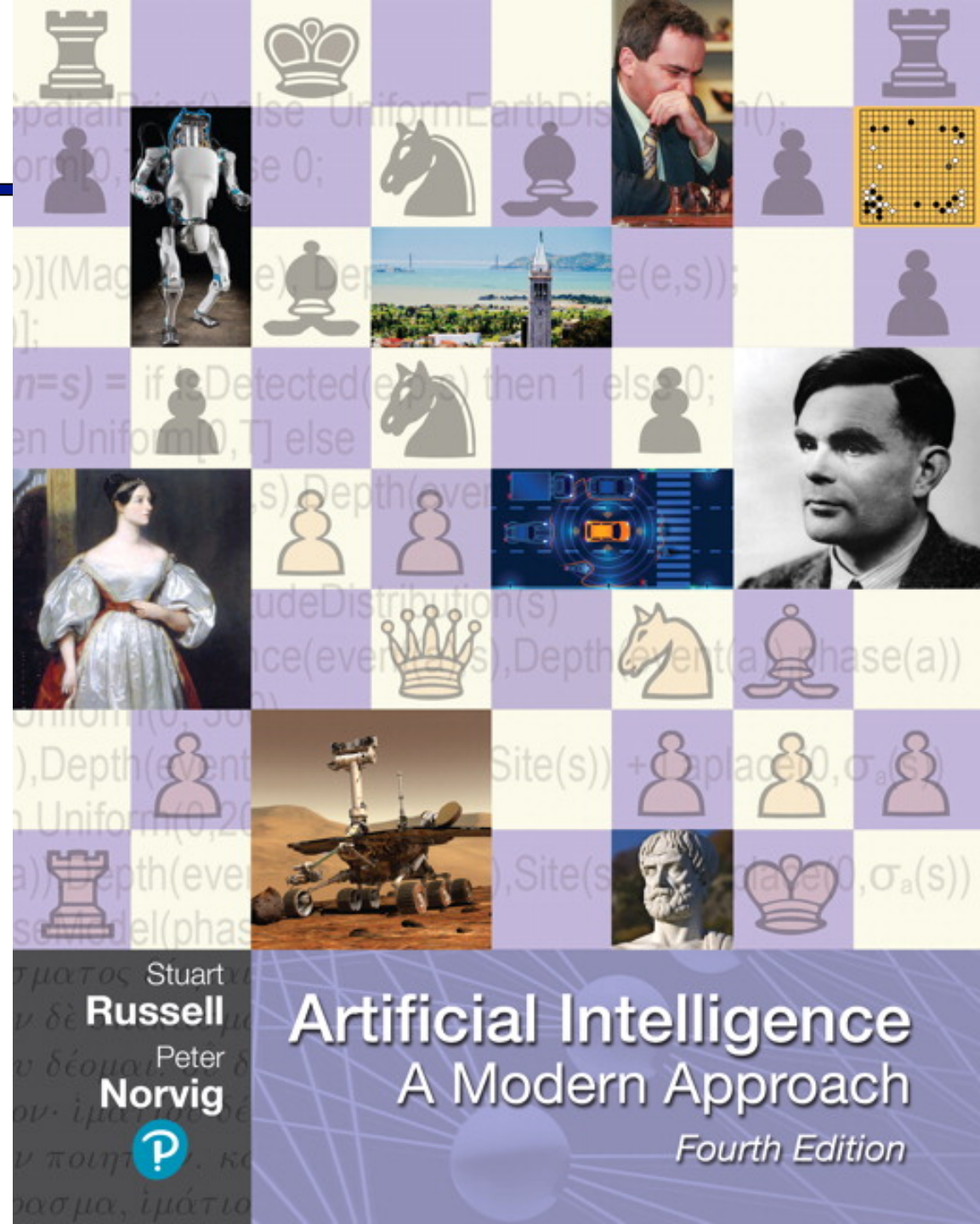
- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features

- Formal justification: online least squares

Reading

Chapter 22 – Reinforcement Learning
Sections 22.1-22.5

Stuart **Russell**
Peter **Norvig**

Artificial Intelligence
A Modern Approach

Fourth Edition

Penn Engineering