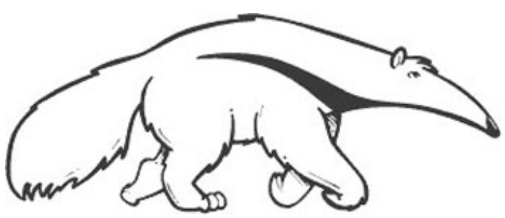
Reinforcement Learning

Machine Learning & Data Mining

Prof. Alexander Ihler



BREN:ICS INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

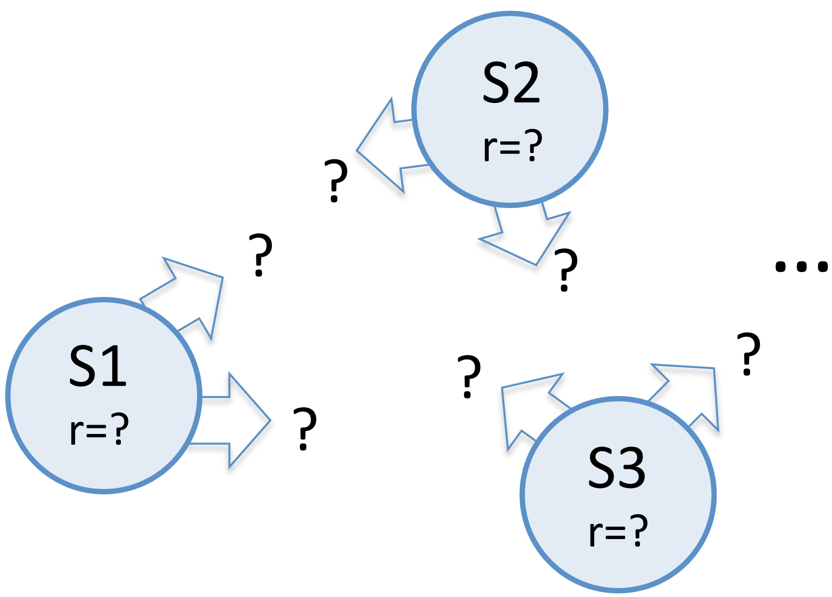
Notes

- Due
  - HW5 due Friday
  - Kaggle uploads close Sunday
  - Reports due Tuesday
- Discussion Thursday & Lecture Friday
  - Review for Final
- Bonus points
  - Peformance on Kaggle final standings (by decile)
  - Course evaluation (closes Sunday) – 1 point final grade

Planning

- Markov Decision Processes
  - Known states, actions, transitions, rewards
  - Find optimal policy

- What about learning?
  - Ex: know states, actions; not transitions or rewards?



Multi-armed bandits

- Very simple reinforcement learning problem
  - Several slot machines ("one-armed bandits"); we have T=1000 plays
  - "Payoff" of each slot is unknown
  - "Bernoulli Bandit": fixed $ value, unknown probability of winning
  - Which should we play, to maximize our reward?
- This is a *multi-armed bandit* problem:
  - Want to play only the best machine
  - But each play gives only noisy info!
- Trivial MDP: one state, several actions
  - Unknown reward r(s,a); no transitions
- *Explore* vs. *exploit* tradeoff
  - How can we balance actions to learn system vs actions to exploit known quantities?

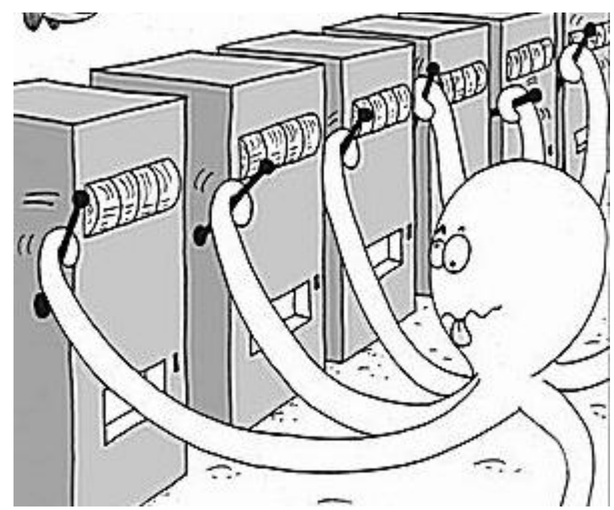


Image from Microsoft Research

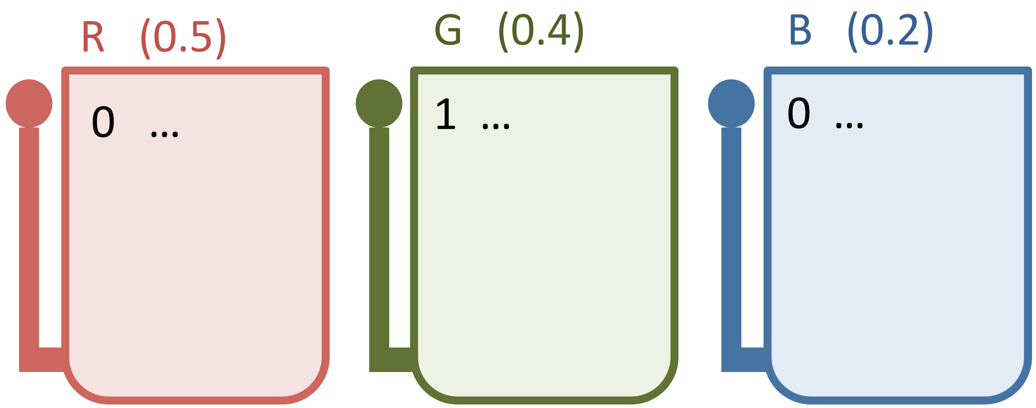
Some basic strategy ideas

- Greedy actions
  - Always take the action we currently think is best

Ex: Bernoulli Bandit

- Strategy: greedy
  - Always play currently estimated best slot

T=1000



Play:
Red => 0
Green => 1
(Blue => 0)

Now choose green forever?

Explore vs *Exploit* tradeoff!

We are "exploiting" a resource we think is good (green has won at least once)
But we are not taking any actions to "explore" the potential rewards of other actions!

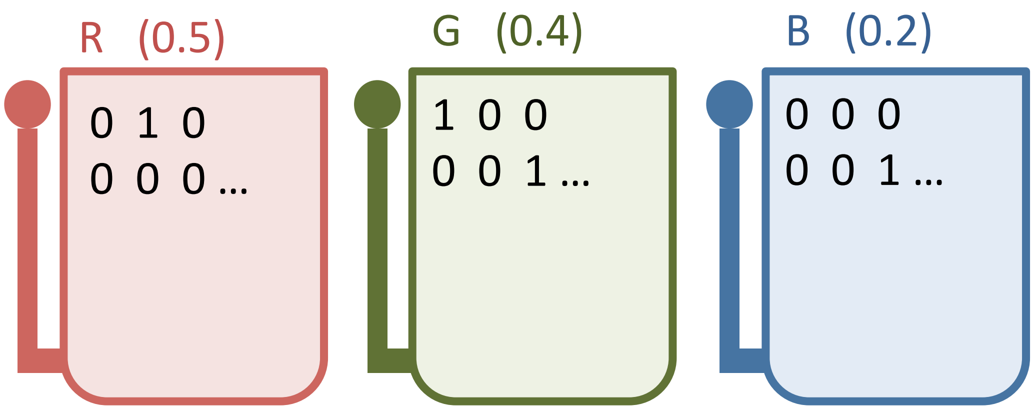
Some basic strategy ideas

- Greedy actions
  - Always take the action we currently think is best
- Greedy + random exploration
  - Interleave some simple exploration into greedy actions

Ex: Bernoulli Bandit

- Strategy: epsilon greedy
  - Play greedily, except with small probability epsilon, play randomly
  - Simplified “batch” version: play K randomly, then T-K greedily

T=1000



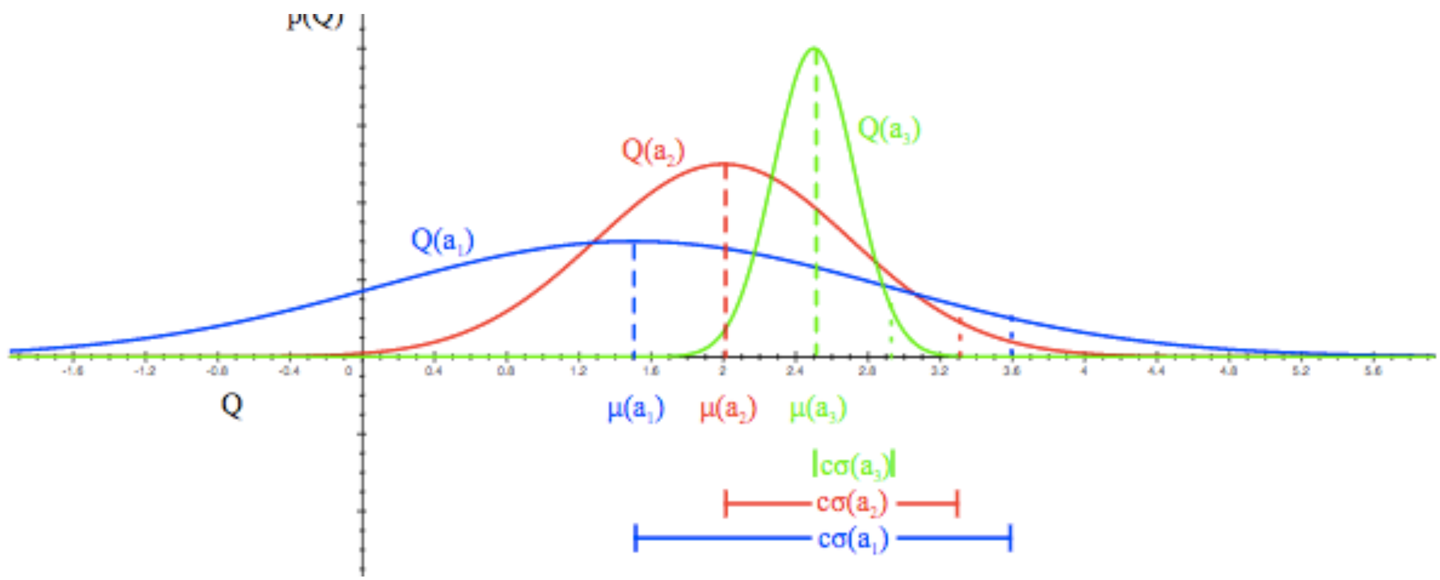
K=6: $\hat{\theta}_R = \frac{1}{6}$ $\quad \hat{\theta}_G = \frac{2}{6}$ $\quad \hat{\theta}_B = \frac{1}{6}$ $\quad \Rightarrow$ Greedy = always play Green
Suboptimal policy!

K=600:

$\hat{\theta}_R = 0.49$ $\quad \hat{\theta}_G = 0.43$ $\quad \hat{\theta}_B = 0.22$ $\quad \Rightarrow$ Greedy = play Red
Better final policy
But, too long playing randomly!

Some basic strategy ideas

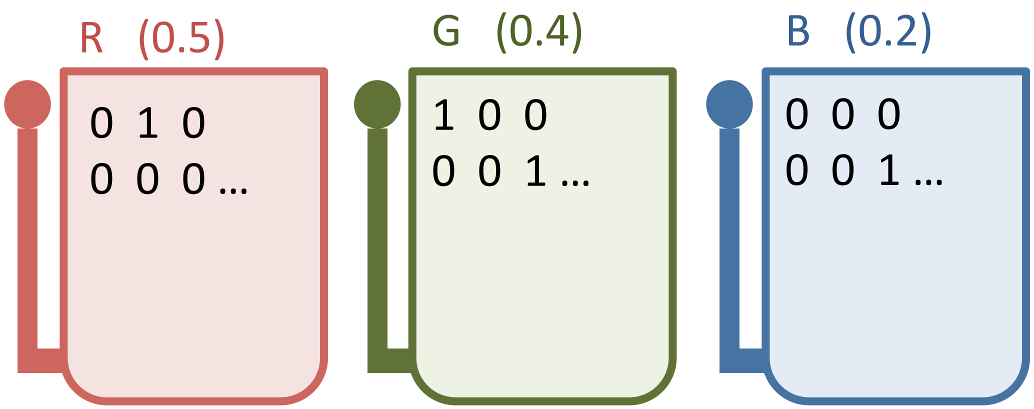
- Greedy actions
  - Always take the action we currently think is best
- Greedy + random exploration
  - Interleave some simple exploration into greedy actions
- Optimism under uncertainty
  - Balance observed rewards with number of observations
  - Ex: Upper confidence bound (UCB) methods



Ex: Bernoulli Bandit

- Strategy: Upper Confidence Bound (UCB)
  - Play best policy, estimated optimistically

T=1000



$$\arg\max_a \; \hat{r}_a + \sqrt{\frac{2\log t}{n_t(a)}}$$

Estimated reward from experience

Confidence given n_t attempts in t steps

Play:
Red => 0
Green => 1
Blue => 0

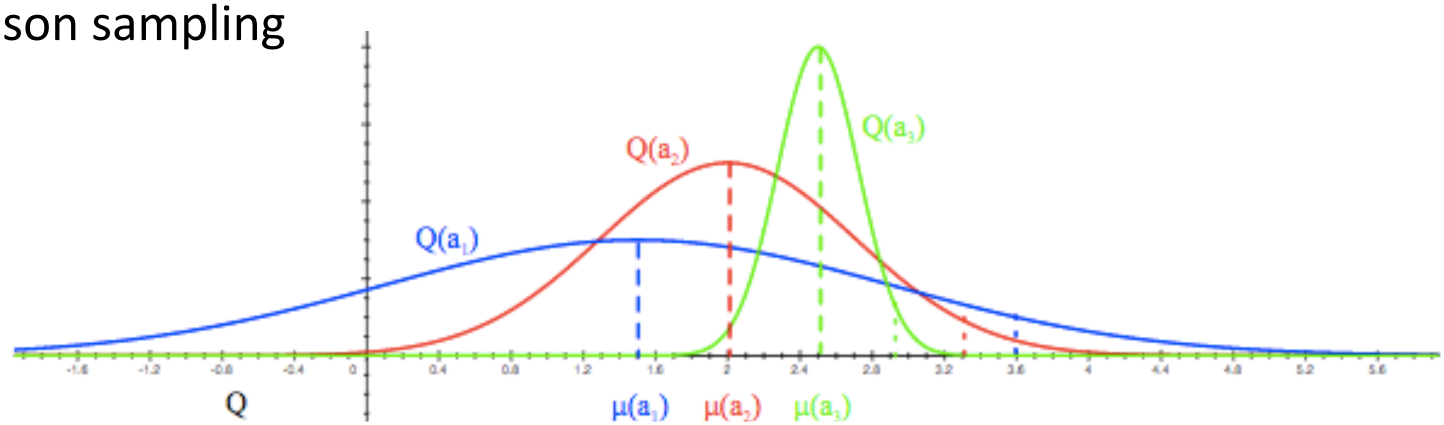
$r_R = 0$ $r_G = 1$ $r_B = 0$
$u_R = 0$ $u_G = 0$ $u_B = 0$
Play Green => 0

$r_R = 0$ $r_G = 0.5$ $r_B = 0$
$u_R = 1$ $u_G = 0.5$ $u_B = 1$
Play Red => 1

…

Some basic strategy ideas

- Greedy actions
  - Always take the action we currently think is best
- Greedy + random exploration
  - Interleave some simple exploration into greedy actions
- Optimism under uncertainty
  - Balance observed rewards with number of observations
  - Ex: Upper confidence bound (UCB) methods
- Sampling methods
  - Intuition: choose actions according to probability of being best
  - Ex: Thompson sampling



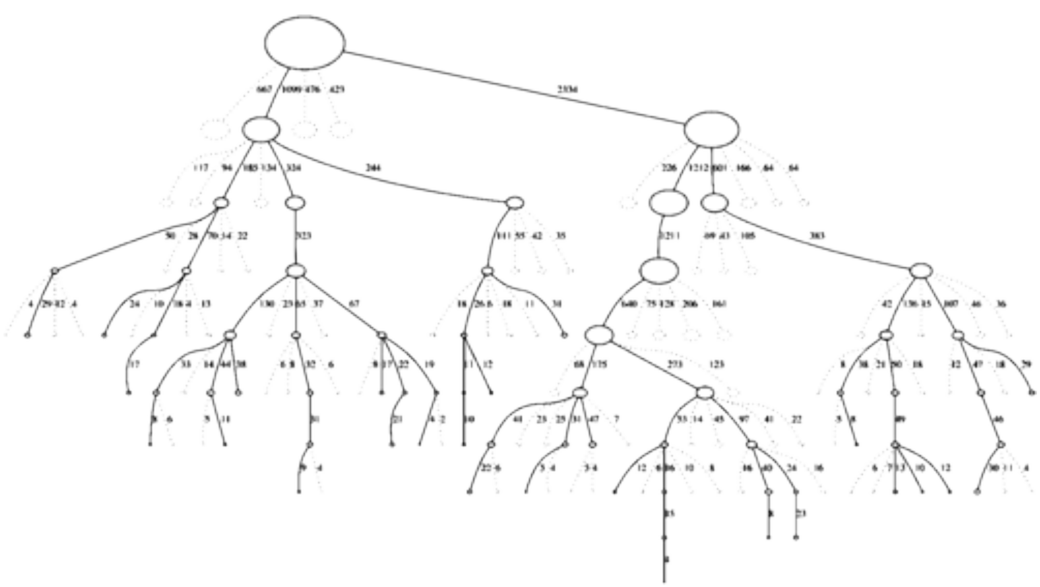
Contextual Bandit problems

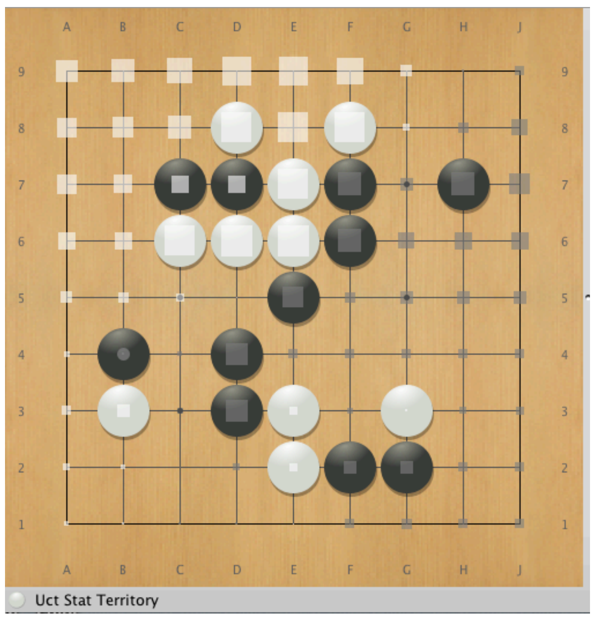
- Observe “context” features x
- Model p(reward | action, x)

- Ex: Online advertising
  - Observe user features, website content, etc.
  - Action = select ad to show to user
  - Payoff = clicks; purchases, etc.
  - Explore vs Exploit
    - show ads that have done well in the past, or try something new?

Monte Carlo Tree Search

- Key technique for many game search algorithms (e.g., AlphaGo)
- At each level of the tree, keep track of
  - Number of times we've explored a path
  - Number of times we won on that path
- Follow winning (from max/min perspective) strategies more often, but also explore others that are not well-explored
- "UCT" algorithm = UCB applied to trees

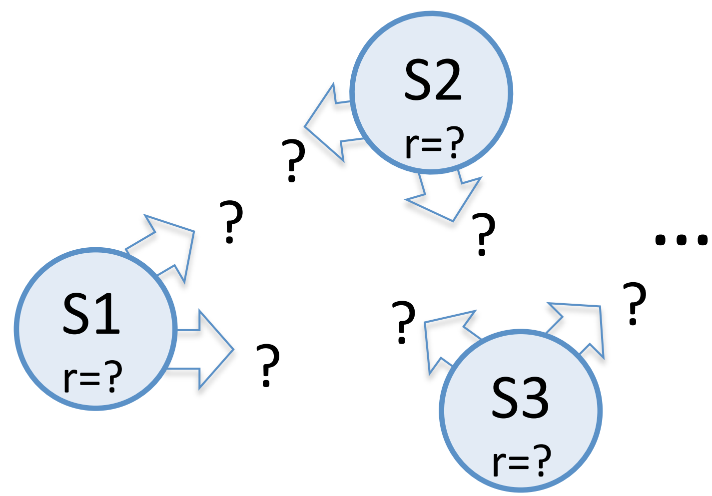




Back to MDPs

- One approach
  - Estimate rewards r(s); transitions p(s' | s,a)
  - Use dynamic programming to evaluate (estimated) J*(s)
  - Take actions according to explore / exploit tradeoff

- Alternate approach
  - Can we estimate the optimal policy / its value directly?
  - Q-learning: powerful technique for MDPs



Q-learning

- Define Q*(s,a) = expected discounted future reward, given start in state s, take action a, proceed optimally afterwards

- Can define recursively (as J* in MDP lecture):

$$Q^*(s,a) = r_a + \sum_{s'} p(s'|s,a) \max_{a'} Q^*(s',a')$$

  - Similar to Bellman Eq'n for J*, but specifies first action a

- Optimal policy $\pi(s) = \arg\max_a Q^*(s,a)$

- Can we estimate Q* directly?

Q-learning

- Initialize Q(s,a)
- Each step:
  - In state s, take some action a, observe r, s’
  - Update

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r_s + \gamma \max_{a'} Q(s',a') \right)$$

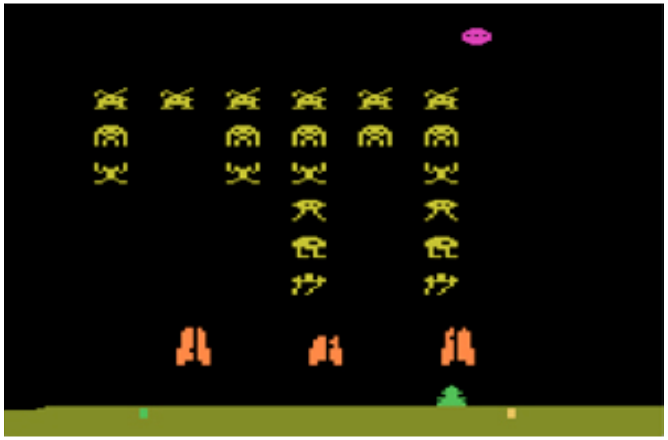
(learning rate)

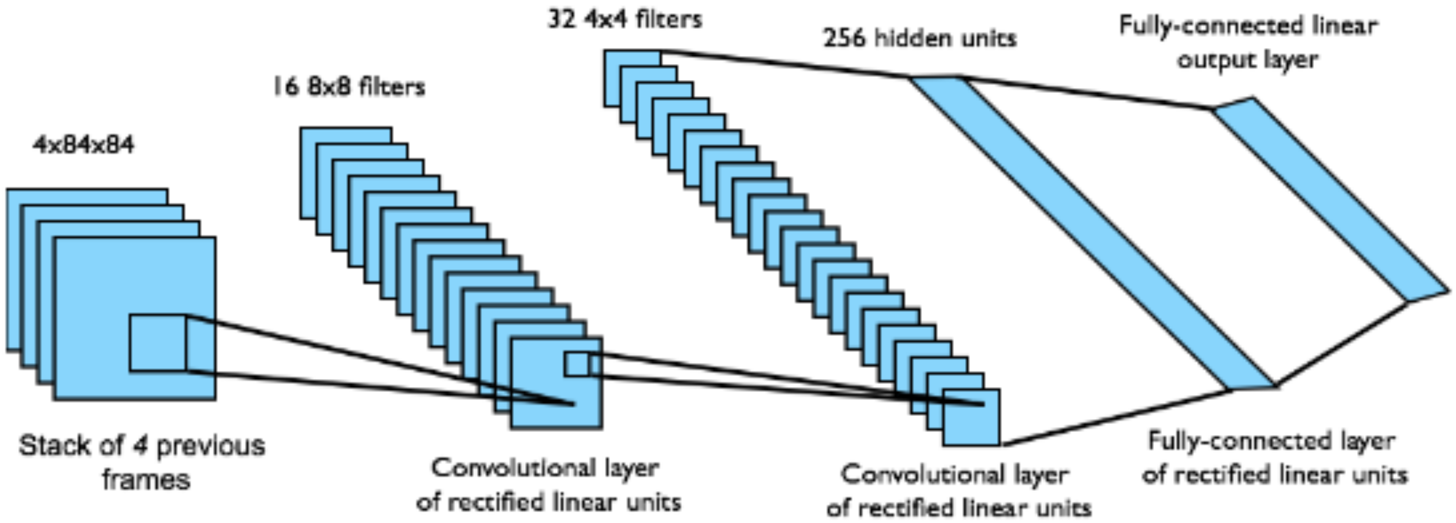
Assuming Q is correct a s’, this is the value of Q(s,a)

- Under model assumptions, this converges to Q*
  - Need decreasing learning rate schedule;
  - Take all actions infinitely often; …
- Can use online, or revisit past experience
  - Store “experience” (s,a,r,s’) & revisit during training

Deep-Q learning

- Use deep neural network architectures for Q(s,a)
- Ex: Atari game playing (DeepMind)
  - Input: pixel images of current state
  - Output: joystick actions





Conclusions

- Reinforcement learning
  - Learn policy (state -> action) based on indirect feedback
  - Fundamental explore / exploit tradeoff
- Multi-armed bandit problems
  - Reward only
  - Various strategies (greedy, optimistic, …)
- Q-learning
  - Directly estimate the value of the optimal policy

See also e.g., David Silver's course on RL:
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html