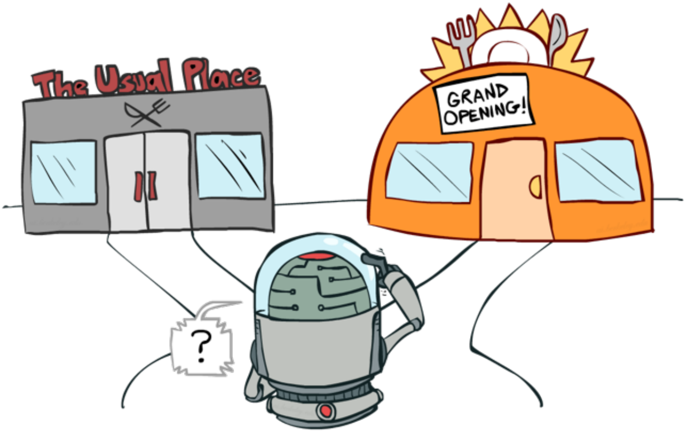
Reinforcement Learning II

CSE 4711: Artificial Intelligence

Md. Bakhtiar Hasan

Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology

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 $1 - \epsilon$, act on current policy

Videos: q-bridge, q-epsilon

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 $1 - \epsilon$, 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

Videos: q-bridge, q-epsilon

Exploration Functions

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

Video: q-explorer-crawler

Exploration Functions

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

Video: q-explorer-crawler

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. $f(u, n) = u + k/n$

Video: q-explorer-crawler

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: $Q(s,a) \leftarrow_a R(s,a,s') + \gamma \max_{a'} Q(s',a')$

Video: q-explorer-crawler

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: $Q(s, a) \leftarrow_a R(s, a, s') + \gamma \max_{a'} Q(s', a')$

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

Video: q-explorer-crawler

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. $f(u,n) = u + k/n$

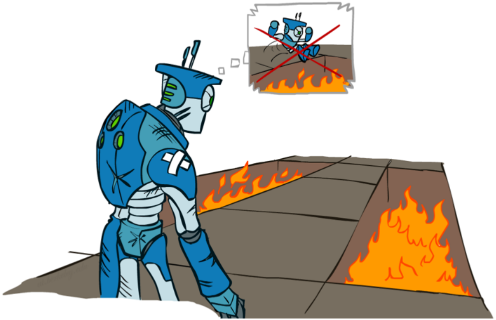
    Regular Q-Update: $Q(s,a) \leftarrow_a R(s,a,s') + \gamma \max_{a'} Q(s',a')$

    Modified Q-Update: $Q(s,a) \leftarrow_a R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$
  - Note: this propagates the "bonus" back to states that lead to unknown states as well!

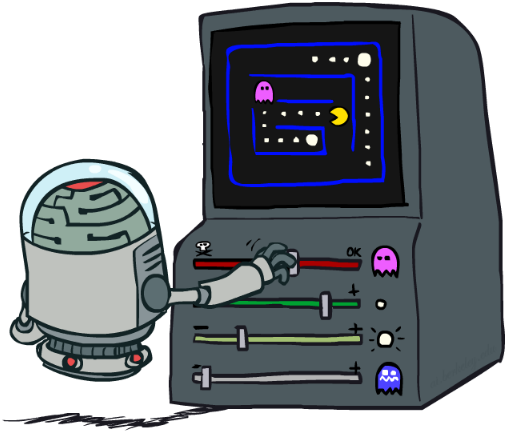
Video: q-explorer-crawler

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



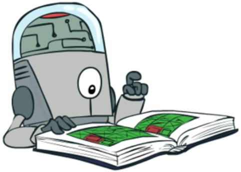
Generalizing Across States

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

Video: q-pacman, q-silent, q-tricky

Generalizing Across States

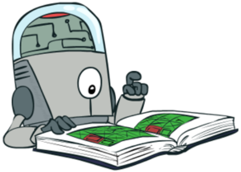
- Basic Q-Learning keeps a table of all q-values



Video: q-pacman, q-silent, q-tricky

Generalizing Across States

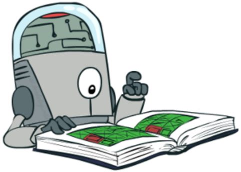
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory

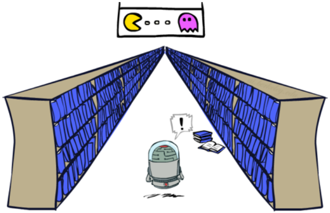


Video: q-pacman, q-silent, q-tricky

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

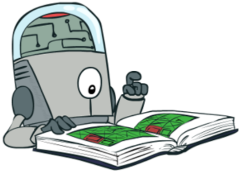


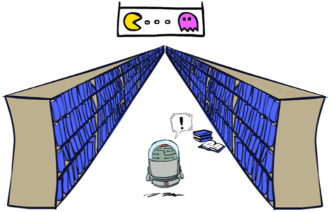


Video: q-pacman, q-silent, q-tricky

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

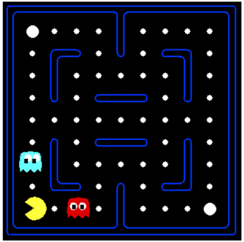




Video: q-pacman, q-silent, q-tricky

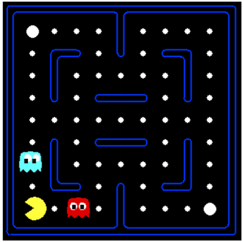
Example: Pacman

Let's say we discover
through experience
that this state is bad:

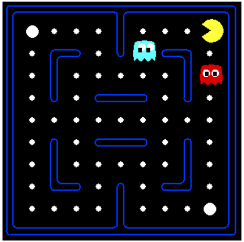


Example: Pacman

Let's say we discover through experience that this state is bad:

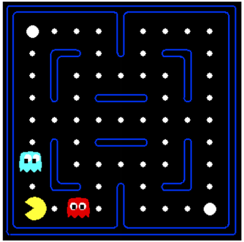


In naïve q-learning, we know nothing about this state:

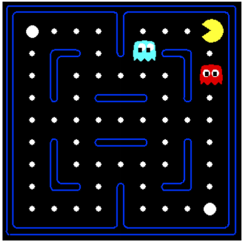


Example: Pacman

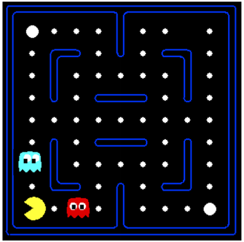
Let's say we discover through experience that this state is bad:



In naïve q-learning, we know nothing about this state:

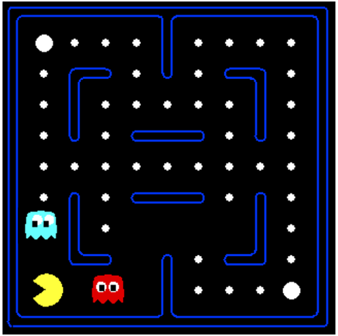


Or even this one!



Feature-Based Representations

- Solution: describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - $1/(\text{dist to dot})^2$
    - Is Pacman in a tunnel? (0/1)
  - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Linear Value Functions

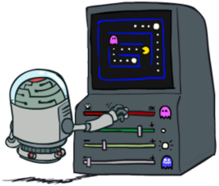
- Using a feature representation, we can write a Q-function for any state using a few weights:

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- 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

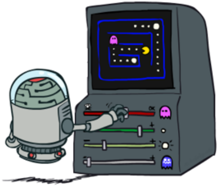
$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$



Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

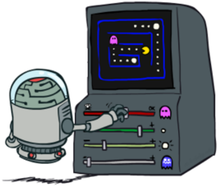


Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

  Transition = (s, a, r, s')



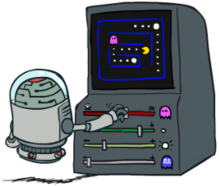
Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

  Transition = (s, a, r, s')

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



Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

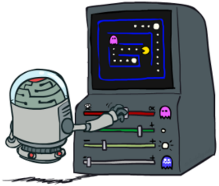
$Q(s,a) - \hat{Q}(s,a)$

- Q-learning with linear Q-functions:

  Transition = (s, a, r, s')

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

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



Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

  Transition = (s, a, r, s')

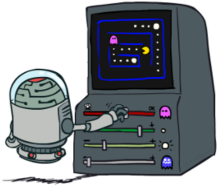
  Difference = $\left[r + \gamma \max_{a'} Q(s', a')\right] - \hat{Q}(s,a)$

  sample − $Q(s)$

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

  Approximate Q's $\quad w_i \leftarrow w_i + \alpha[\text{difference}]\, f_i(s,a)$

$W = W +$



Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

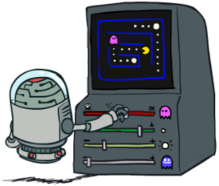
  Transition = (s, a, r, s')

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

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

  Approximate Q's $\quad w_i \leftarrow w_i + \alpha[\text{difference}] f_i(s,a)$

- 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



Approximate Q-Learning

$$\hat{Q}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

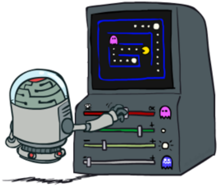
  Transition = (s, a, r, s')

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

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

  Approximate Q's $\quad \boxed{w_i \leftarrow w_i + \alpha[\text{difference}] f_i(s,a)}$

- 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



Example: Q-Pacman

$$\hat{Q}(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

Video: q-approx-pacman

Example: Q-Pacman

$$\hat{Q}(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$Q(s, \text{NORTH}) = +1$

Video: q-approx-pacman

Example: Q-Pacman

$$\hat{Q}(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$Q(s, \text{NORTH}) = +1$

$a = \text{NORTH}$
$r = -500$

s'

$Q(s', \cdot) = 0$

Video: q-approx-pacman

Example: Q-Pacman

$$\hat{Q}(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$a = \text{NORTH}$
$r = -500$

s'

$Q(s, \text{NORTH}) = +1$

$Q(s', \cdot) = 0$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

Video: q-approx-pacman

Example: Q-Pacman

$$\hat{Q}(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$a = \text{NORTH}$

$r = -500$

s'

$Q(s, \text{NORTH}) = +1$

$Q(s', \cdot) = 0$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

difference $= -501$

$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$

$w_{GST} \leftarrow -1.0 + \alpha\,[-501]\,1.0$

Video: q-approx-pacman

Example: Q-Pacman

$$\hat{Q}(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$a = \text{NORTH}$
$r = -500$

s'

$Q(s, \text{NORTH}) = +1$

$Q(s', \cdot) = 0$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

difference $= -501$

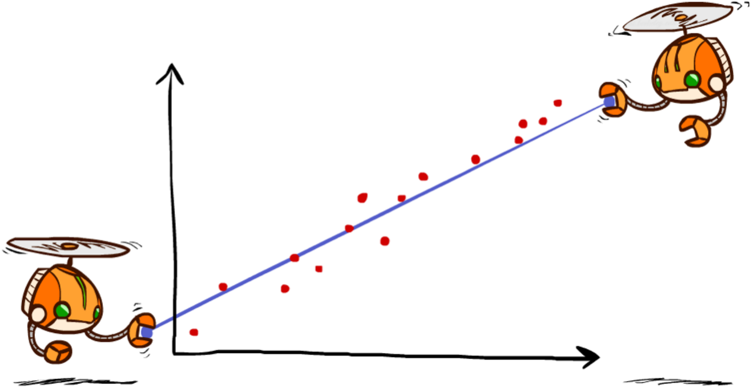
$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$

$w_{GST} \leftarrow -1.0 + \alpha\,[-501]\,1.0$

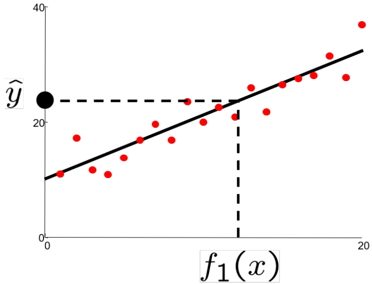
$$\hat{Q}(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$

Video: q-approx-pacman

Q-Learning and Least Squares



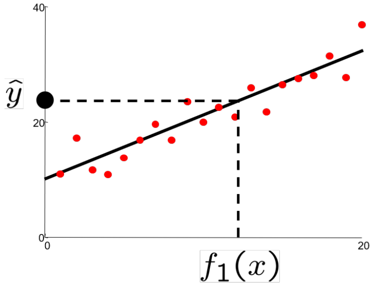
Linear Approximation: Regression

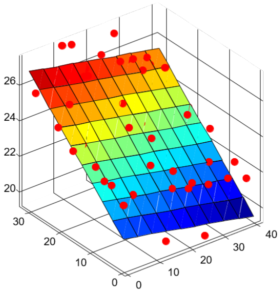


Prediction:

$$\hat{y} = w_0 + w_1 f_1(x)$$

Linear Approximation: Regression





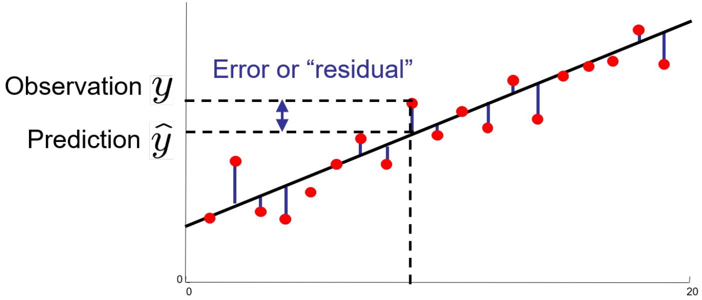
Prediction:

$$\hat{y} = w_0 + w_1 f_1(x)$$

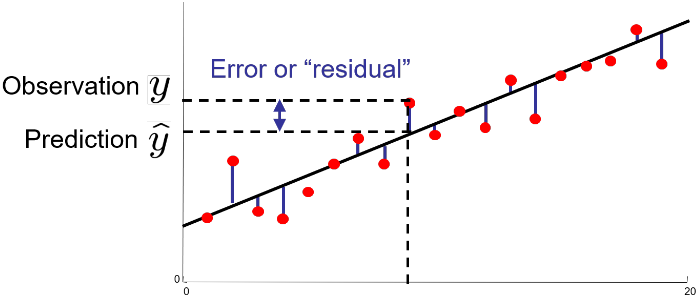
Prediction:

$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

Optimization: Least Squares

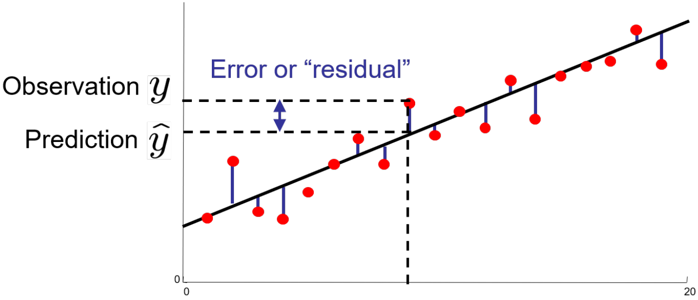


Optimization: Least Squares



total error = $\sum_i (y_i - \widehat{y}_i)^2$

Optimization: Least Squares



total error = $\sum_i (y_i - \widehat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k f_k(x_i) \right)^2$

Minimizing Error

Imagine we had only one point x, with features $f(x)$, target value y, and weights w:

Minimizing Error

Imagine we had only one point x, with features $f(x)$, target value y, and weights w:

$$error(w) = \frac{1}{2}\left(y - \sum_k (w_k f_k(x)\right)^2$$

Minimizing Error

Imagine we had only one point x, with features $f(x)$, target value y, and weights w:

$$error(w) = \frac{1}{2}\left(y - \sum_k (w_k f_k(x)\right)^2$$

$$\frac{\partial error(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

Minimizing Error

Imagine we had only one point x, with features $f(x)$, target value y, and weights w:

$$error(w) = \frac{1}{2}\left(y - \sum_k (w_k f_k(x))\right)^2$$

$$\frac{\partial error(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

$$w_m \leftarrow w_m + \underline{\alpha}\left(\underline{y} - \underbrace{\sum_k (w_k f_k(x))}_{\text{exp}}\right) \underline{f_m(x)}$$

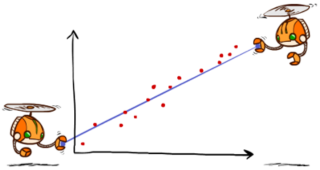
Minimizing Error

Imagine we had only one point x, with features $f(x)$, target value y, and weights w:

$$error(w) = \frac{1}{2}\left(y - \sum_k (w_k f_k(x))\right)^2$$

$$\frac{\partial error(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

$$w_m \leftarrow w_m + \alpha \left(y - \sum_k (w_k f_k(x))\right) f_m(x)$$



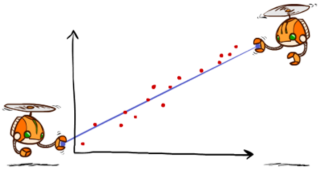
Minimizing Error

Imagine we had only one point x, with features $f(x)$, target value y, and weights w:

$$error(w) = \frac{1}{2}\left(y - \sum_k (w_k f_k(x))\right)^2$$

$$\frac{\partial error(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

$$w_m \leftarrow w_m + \alpha \left(y - \sum_k (w_k f_k(x))\right) f_m(x)$$

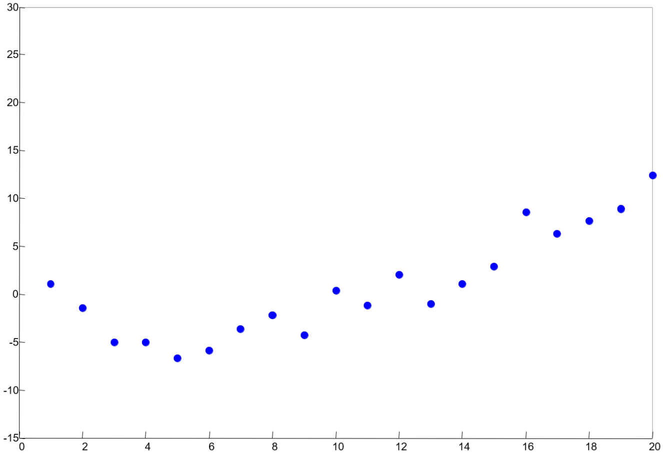


Approximate q-update:

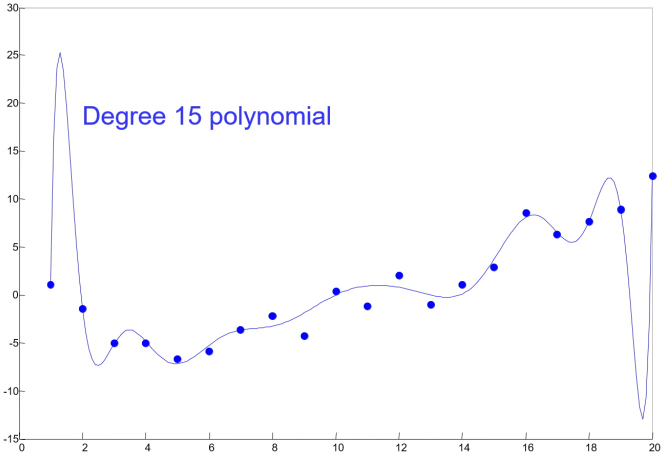
$$w_m \leftarrow w_m + \alpha \left[r + \gamma \max_a Q(s', a') - Q(s, a) \right] f_m(s, a)$$

Q- same

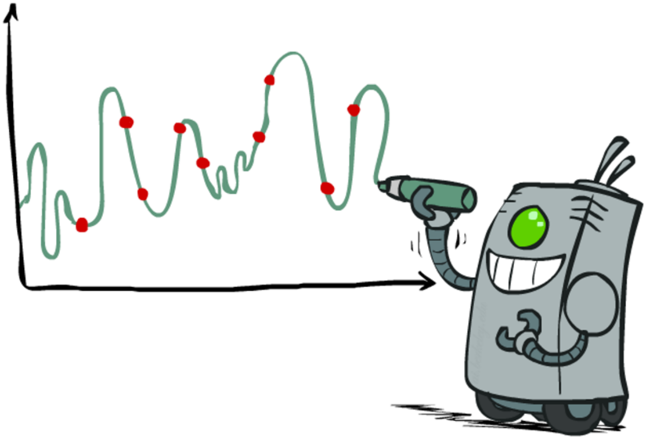
Overfitting: Why Limiting Capacity Can Help



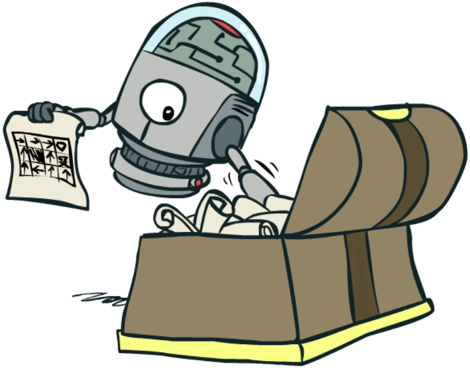
Overfitting: Why Limiting Capacity Can Help



Overfitting: Why Limiting Capacity Can Help



Policy Search



Policy Search

- Often the feature-based policies that work well aren't the ones that approximate V/Q best

Policy Search

- Often the feature-based policies that work well aren't the ones that approximate V/Q best
- Solution: Policy Search
  - Start with an initial linear value function or Q-function
  - Nudge each feature weight up and down and see if your policy is better than before

Policy Search

- Often the feature-based policies that work well aren't the ones that approximate V/Q best
- Solution: Policy Search
  - Start with an initial linear value function or Q-function
  - Nudge each feature weight up and down and see if your policy is better than before
- Problems
  - How do we tell the policy got better?
  - Need to run many sample episodes!
  - If there are a lot of features, this can be impractical

Policy Search

- Often the feature-based policies that work well aren't the ones that approximate V/Q best
- Solution: Policy Search
  - Start with an initial linear value function or Q-function
  - Nudge each feature weight up and down and see if your policy is better than before
- Problems
  - How do we tell the policy got better?
  - Need to run many sample episodes!
  - If there are a lot of features, this can be impractical
- Better methods exploit lookahead structure, sample wisely, change multiple parameters...

Conclusion

- We are done with Search and Planning!
- We have seen how AI methods can solve problems in:
  - Search
  - Constraint Satisfaction Problems
  - Games
  - Markov Decision Problems
  - Reinforcement Learning
- Next? Uncertainty and Learning

Suggested Reading

- Russell & Norvig: Chapter 21