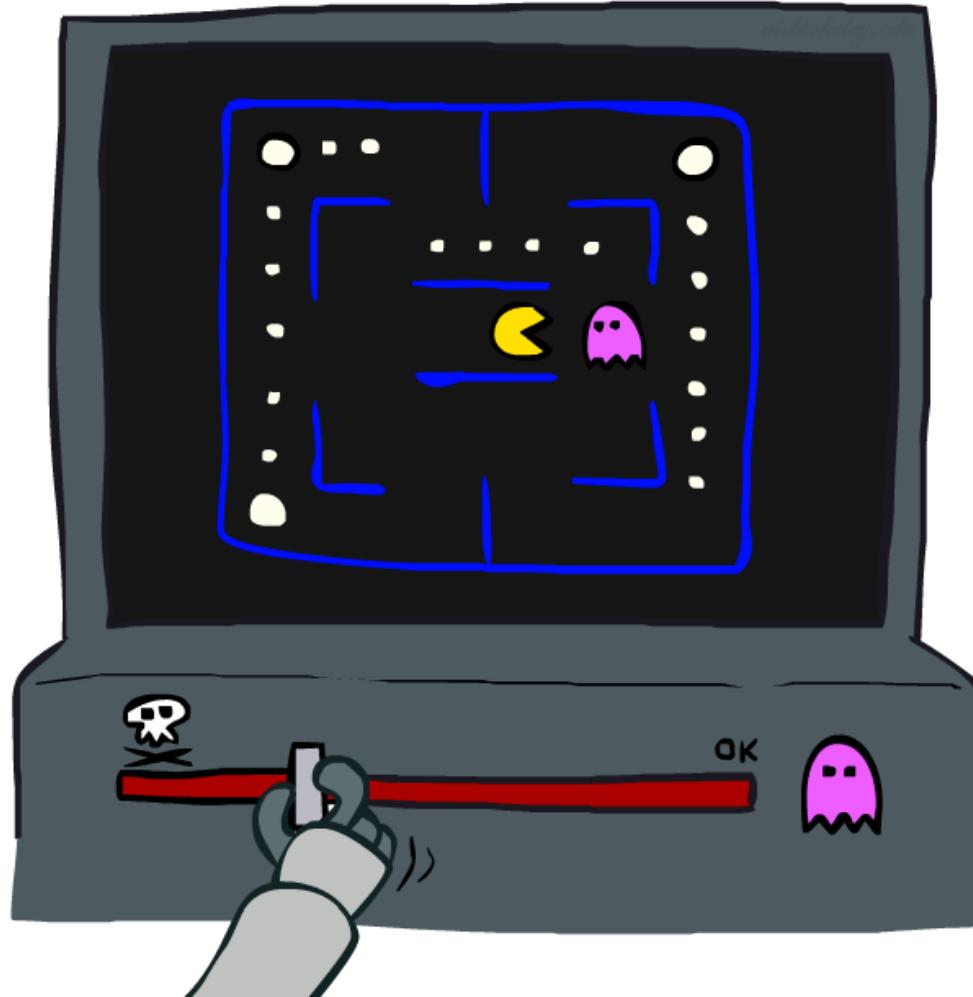


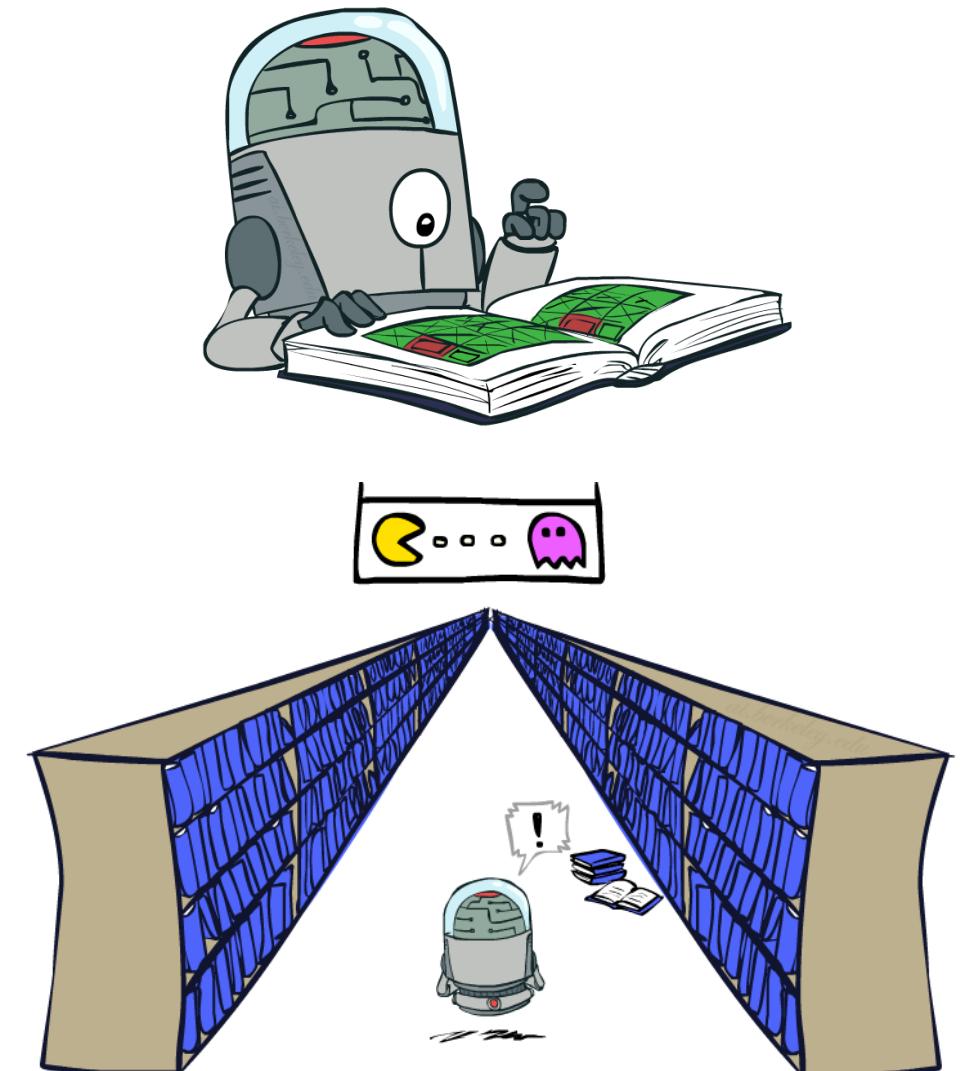
# Approximate Q-Learning

---



# 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



[demo – RL pacman]

# 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)
    - ..... etc.
    - Is it the exact state on this slide?
  - 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 (or value function) for any state using a few weights:

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

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

- 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

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

- Q-learning with linear Q-functions:

transition =  $(s, a, r, s')$

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

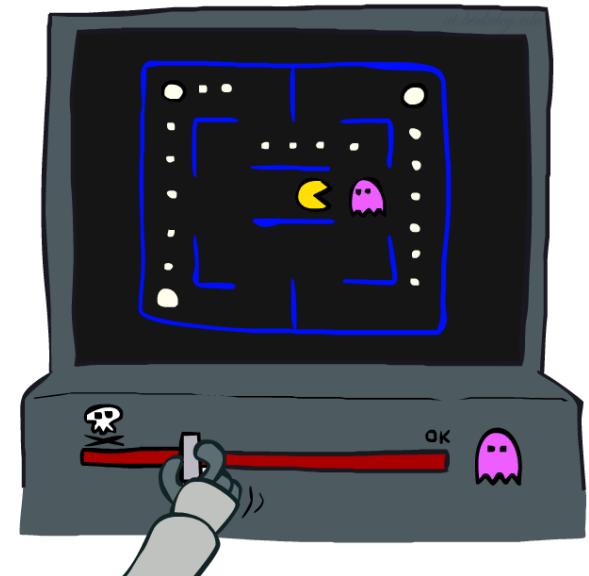
$Q(s, a) \leftarrow Q(s, a) + \alpha \text{ [difference]}$       Exact Q's

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

- Intuitive interpretation:

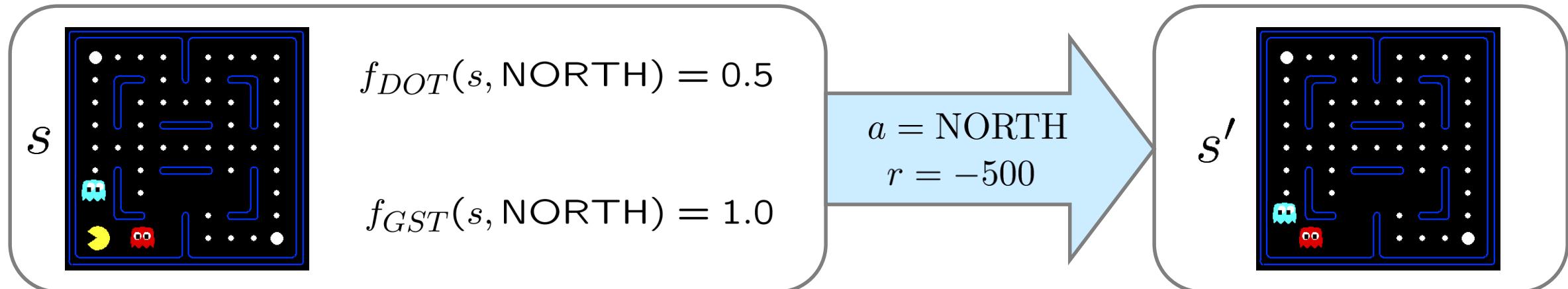
- Adjust weights of active features
- E.g., if something unexpectedly bad happens, blame the features that were on:  
disprefer all states with that state's features

- Formal justification: online least squares



# Example: Q-Pacman

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

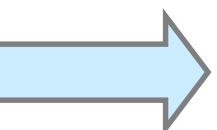


$$Q(s, \text{NORTH}) = +1$$

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

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

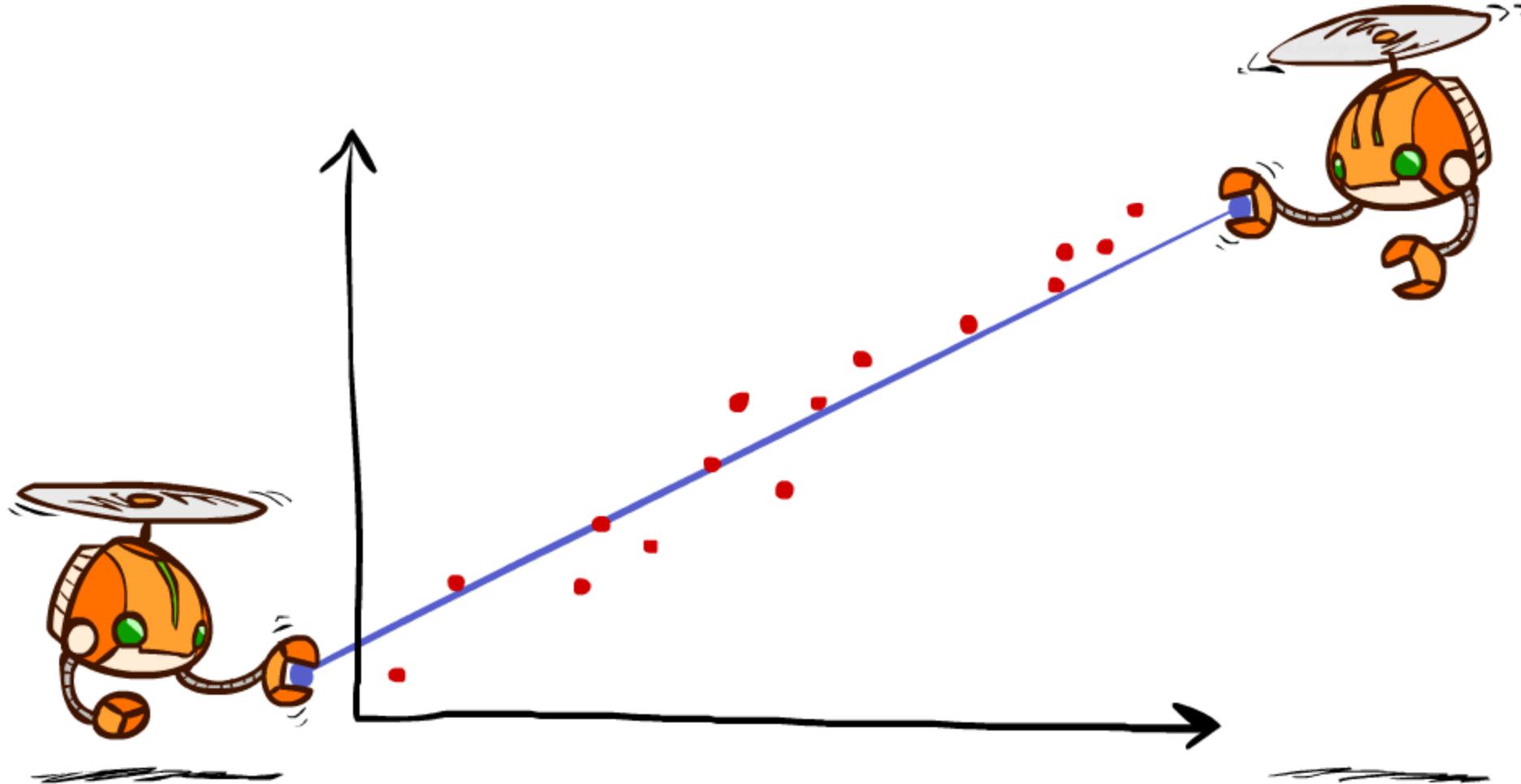
$$\text{difference} = -501$$



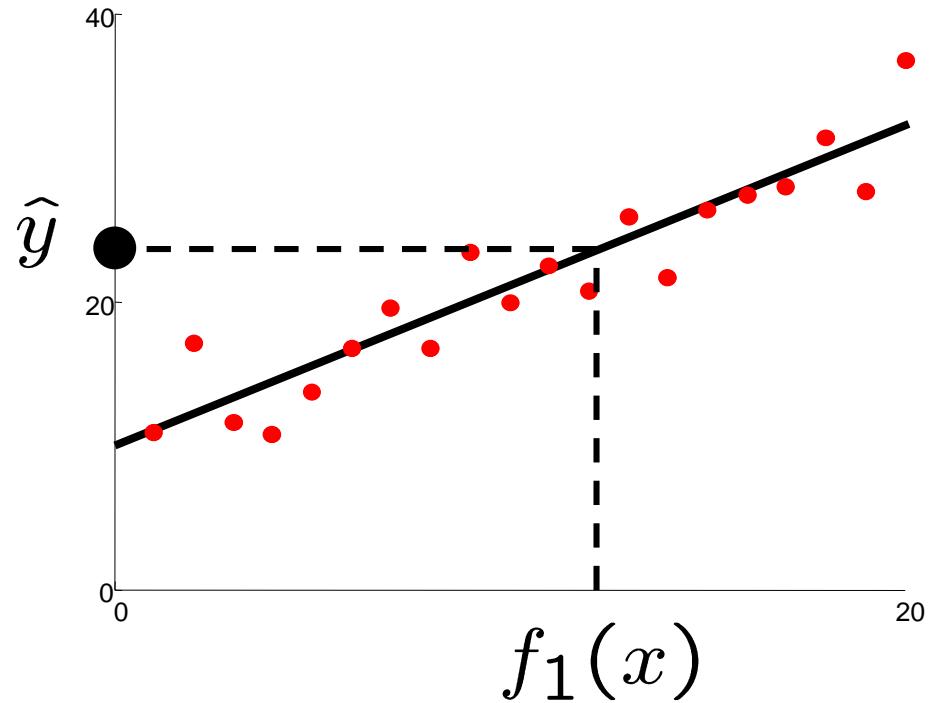
$$w_{DOT} \leftarrow 4.0 + \alpha [-501] 0.5$$
$$w_{GST} \leftarrow -1.0 + \alpha [-501] 1.0$$

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

# Q-Learning and Least Squares

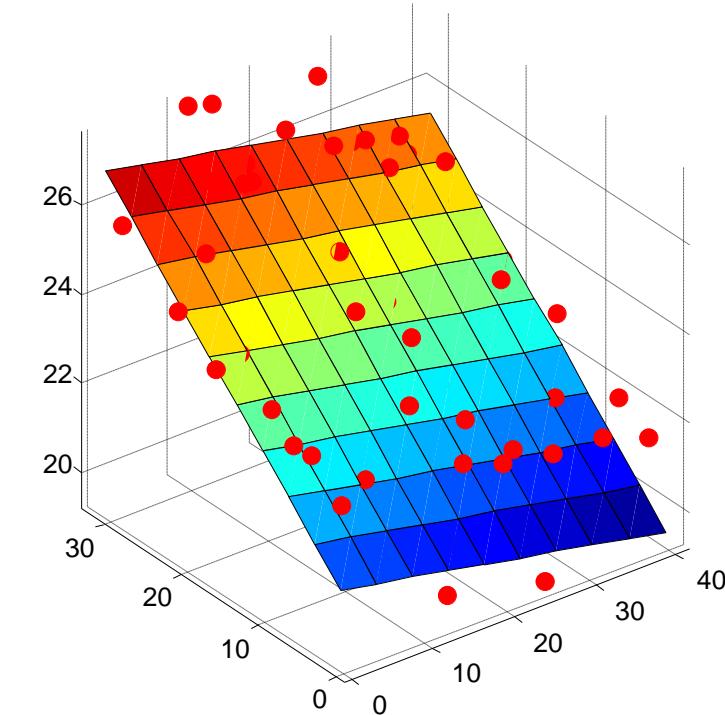


# Linear Approximation: Regression



Prediction:

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

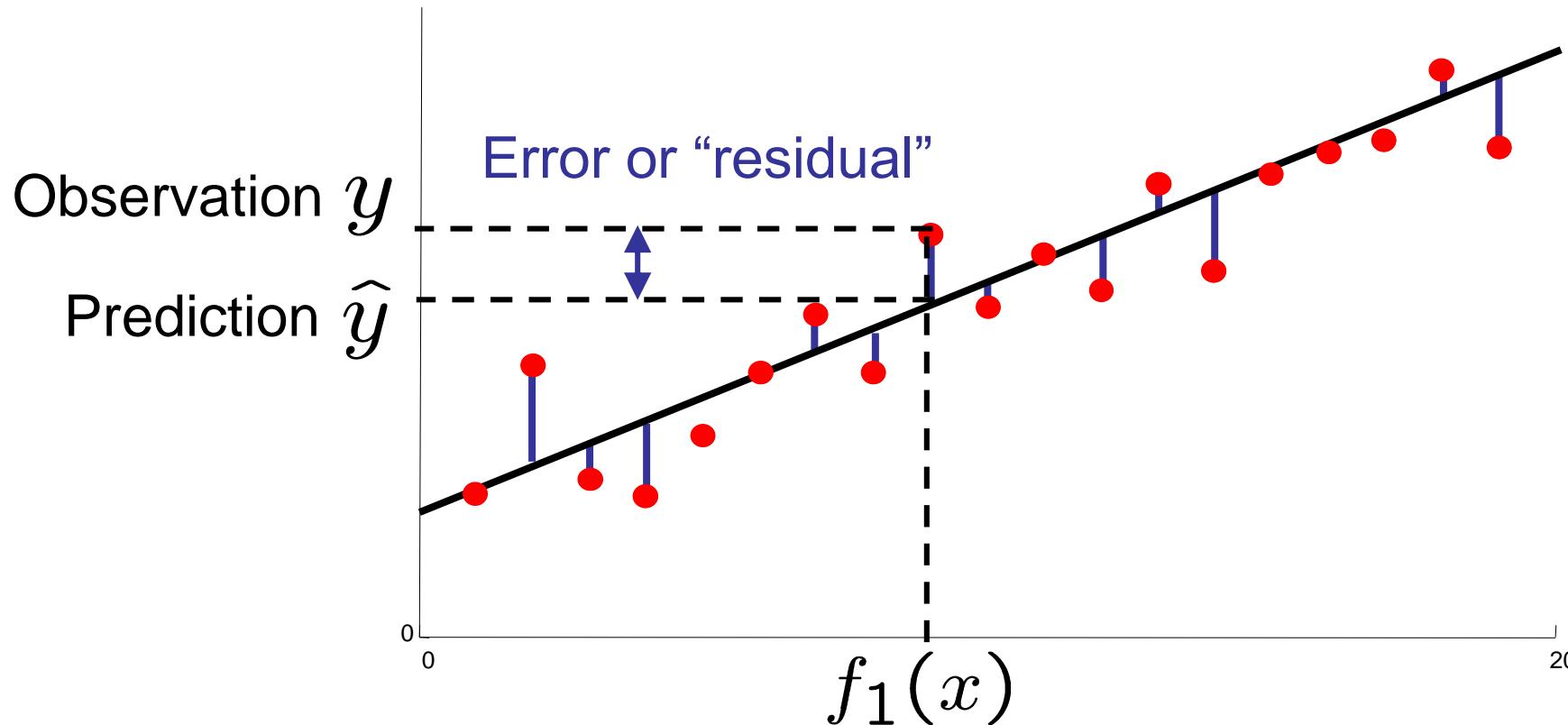


Prediction:

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

# Optimization: Least Squares

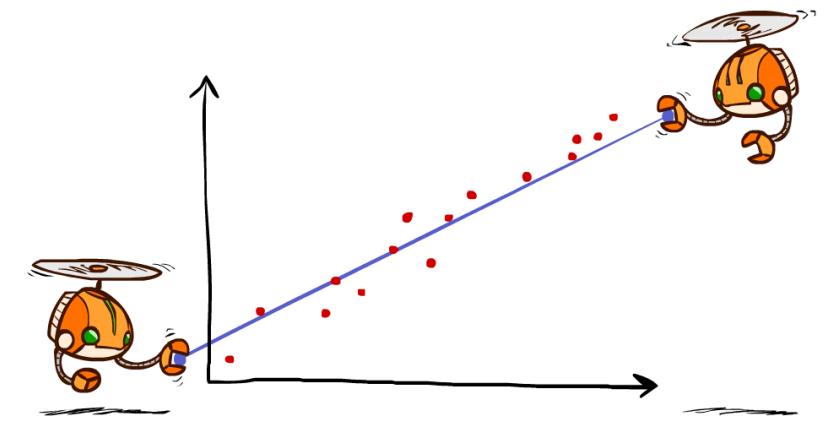
$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left( y_i - \sum_k w_k f_k(x_i) \right)^2$$



# Minimizing Error

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

$$\text{error}(w) = \frac{1}{2} \left( y - \sum_k w_k f_k(x) \right)^2$$
$$\frac{\partial \text{error}(w)}{\partial w_m} = - \left( y - \sum_k w_k f_k(x) \right) f_m(x)$$
$$w_m \leftarrow w_m + \alpha \left( y - \sum_k w_k f_k(x) \right) f_m(x)$$



Approximate q update explained:

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

“target”

“prediction”

# More Powerful Function Approximation

- Linear:

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

- Polynomial:

$$Q(s, a) = w_{11} f_1(s, a) + w_{12} f_2(s, a)^2 + w_{13} f_3(s, a)^3 + \cdots$$

- Neural Network:

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

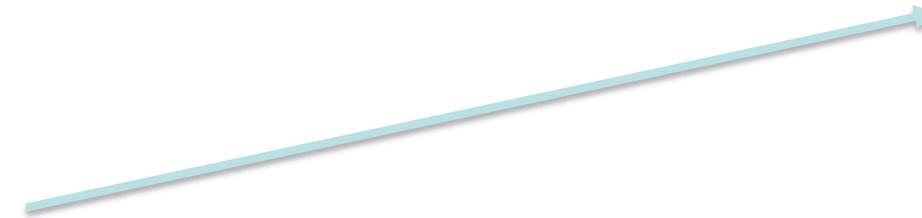


Learn These Too

# More Powerful Function Approximation

- Summary of iterative feature update equation!

$$w_m \leftarrow w_m + \alpha \left[ r + \gamma \max_a Q(s', a') - Q(s, a) \right] \frac{dQ}{dw_m}(s, a)$$


$$= f_m(s, a) \text{ when linear approximation}$$

# Overfitting: Why Limiting Capacity Can Help

