CS 440
Introduction to Artificial Intelligence

Lecture 21:

Cross-Validation – Linear Regression

May 7, 2020

POMDP

- **Agent cannot directly observe state of environment**
  - **Observations agent makes are limited and noisy**
- **Agent's actions can influence state of environment**
- **Results of actions not deterministic**
  - **Defined by probability distribution**

Formalization

- State space S
  - Agent does not know what state it is in
- Set of actions A
  - Actions can be noisy
  - Results of actions a probability distribution over other states
- Transition function T(s,a)
  - T(s,a) result of taking action a while in state s
    - Noisy actions:
      - T(s,a) is a probability distribution over state space
      - $T(s,a) = \{p(s'_1), p(s'_2), \ldots, p(s'_n)\}$
        - where $p(s'_i)$ is the probability that performing action a while in state s will result in state s'
    - Defined for all combinations of $s \in S$, $a \in A$
- Reward function R
- Immediate objective: Agent must determine what action to take in order to maximize future expected reward

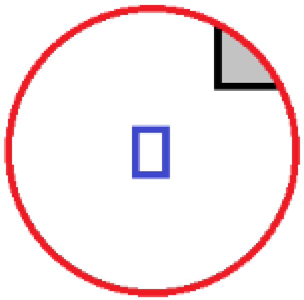
Belief

- **Belief defined as a probability distribution over state space**
- **$S = \{s_1, s_2, \ldots, s_n)$**
- **$b = \{p(s_1), p(s_2), \ldots, p(s_n)\}$**
  - **$p(s_i)$ is the agent's estimate of the likelihood it is in state s_i**
- **For discrete state space consists of a probability for each state**
  - **Often times probability of most states will be 0**
- **For non-discrete problem consist of a probability density function**
  - **Example: Gaussian**

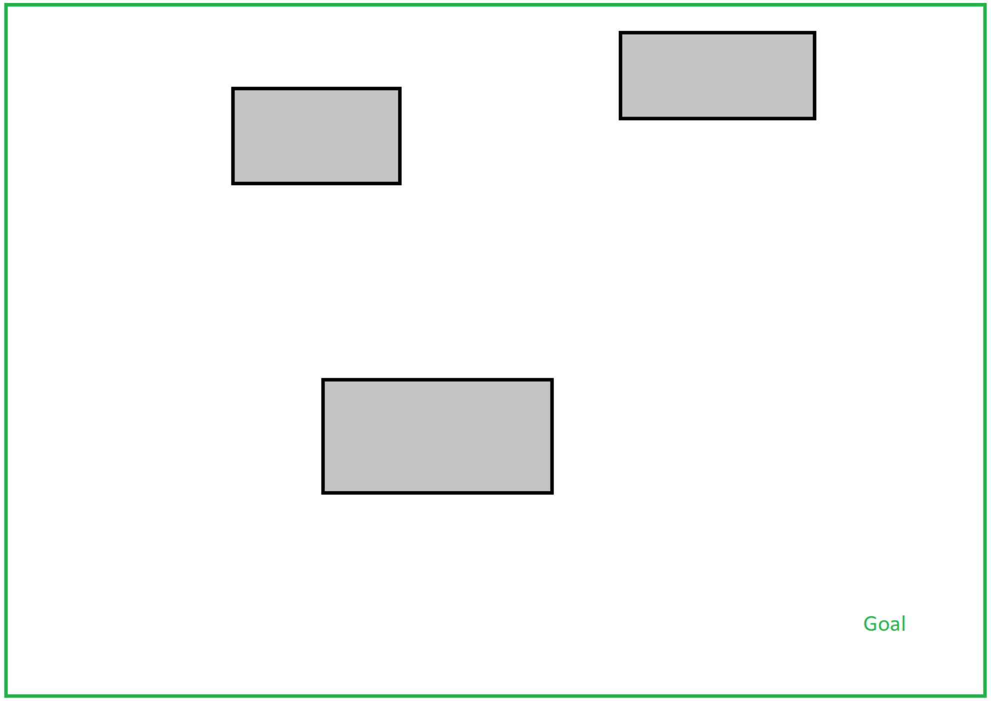
Exploration vs Exploitation

- **Exploration**
  - **Gain information about environment**
  - **Reduce uncertainty of belief**
- **Exploitation**
  - **Find a solution to the problem**
    - **Reach the goal**
    - **Get to a high reward state**
- **A solution to the pomdp needs to balance exploration and exploitation**

Example:Robot with limited sensor range

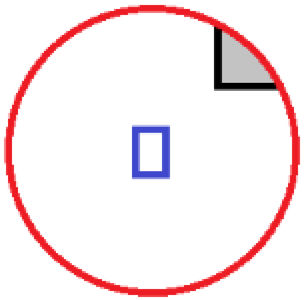


What robot sees

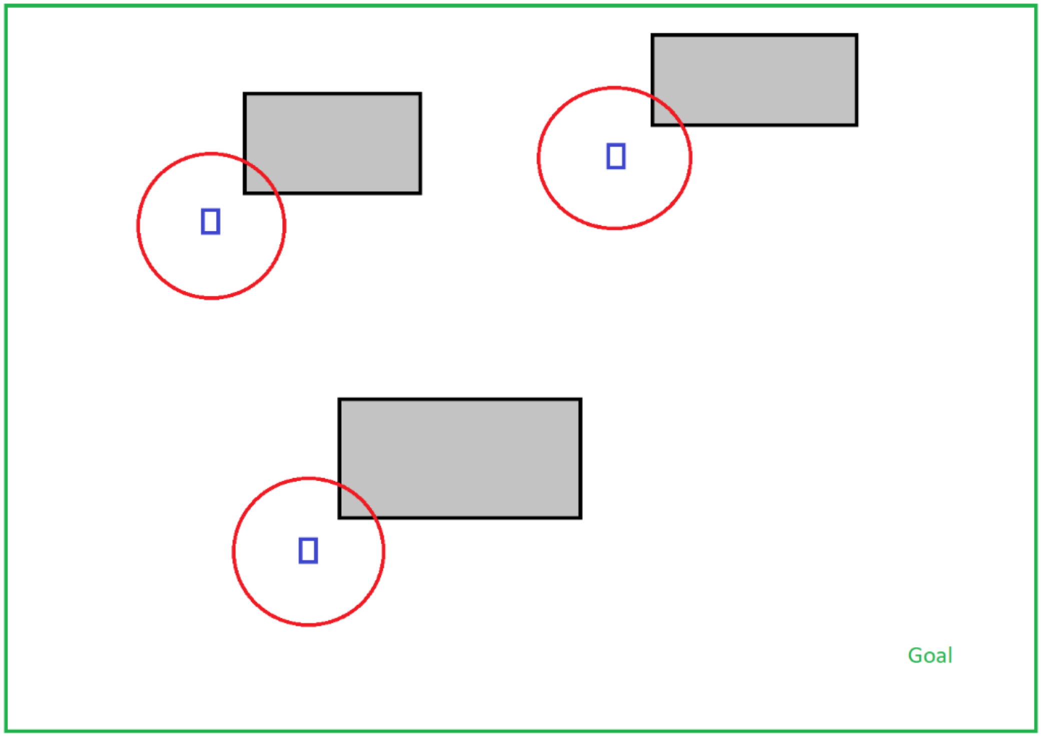


environment

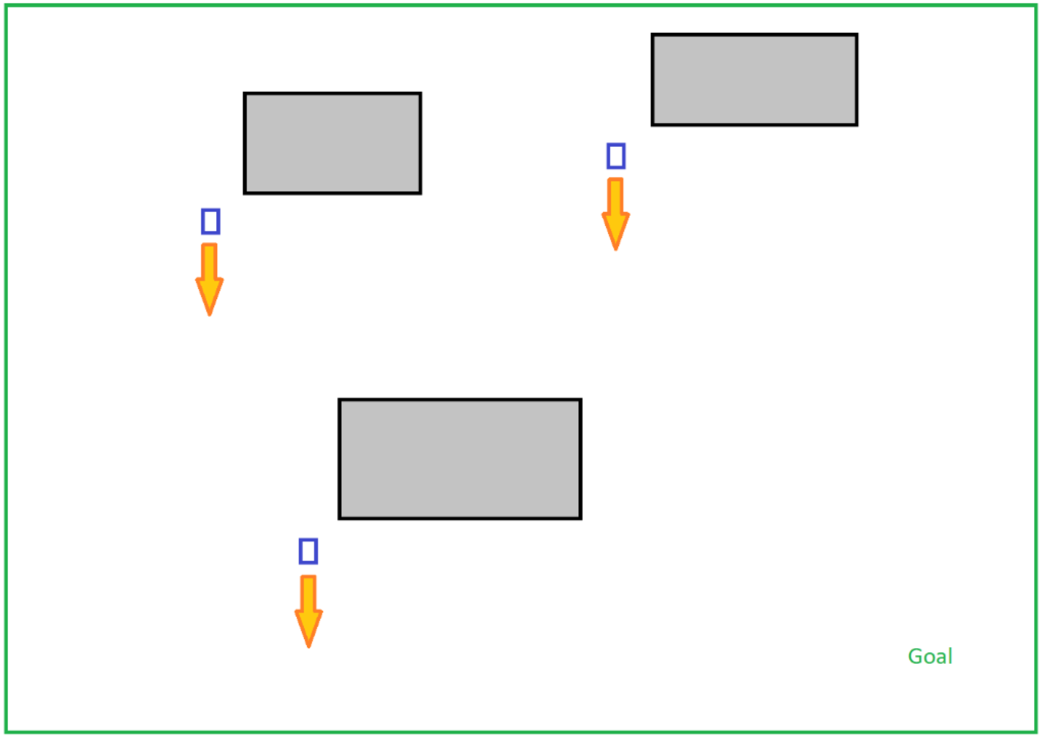
Example:Robot with limited sensor range



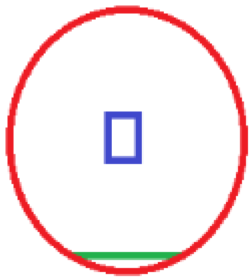
What robot sees



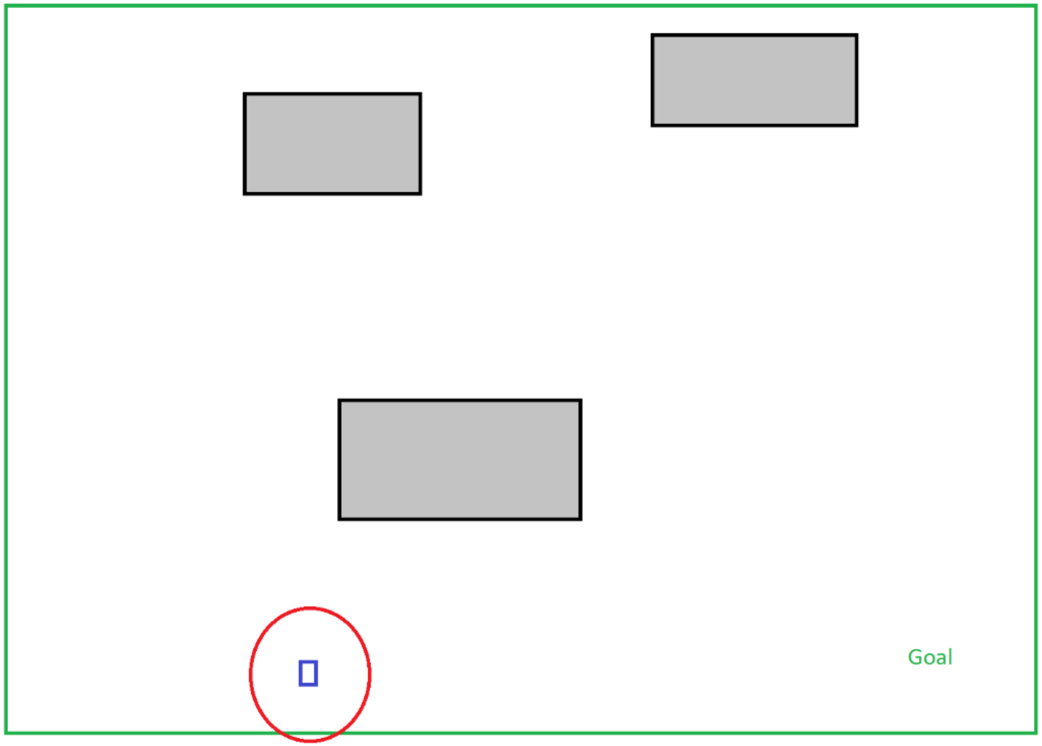
Example:Robot with limited sensor range



Example:Robot with limited sensor range



What robot sees



End of Reasoning Under Uncertainty

- **Probability**
- **Bayes theorem**
- **Bayesian Networks**
- **Markov Chains**
- **Markov Decision Processes**
- **Hidden Markov Models**
- **Partially Observable Markov Decision Processes**

- **What is Learning?**

- **What is Learning?**
  - **Use previous experience to solve problem**
  - **Example solutions to problems**
    - **Learning by demonstration**
  - **Examine trends in data to predict solution**
    - **Data mining**
  - **Train agent to perform task**
    - **Reinforcement learning**

Model

- Model
  - Representation of environment
  - Queried to find solutions to problems
- Models used in first two thirds of class
  - Examples:
    - State/Action/Transition/Reward models
    - Bayesian networks
    - Markov models
    - Logical statements
  - We defined these models
  - Agent used models we built
- What if we allowed the agent to build the models?

Learned Model

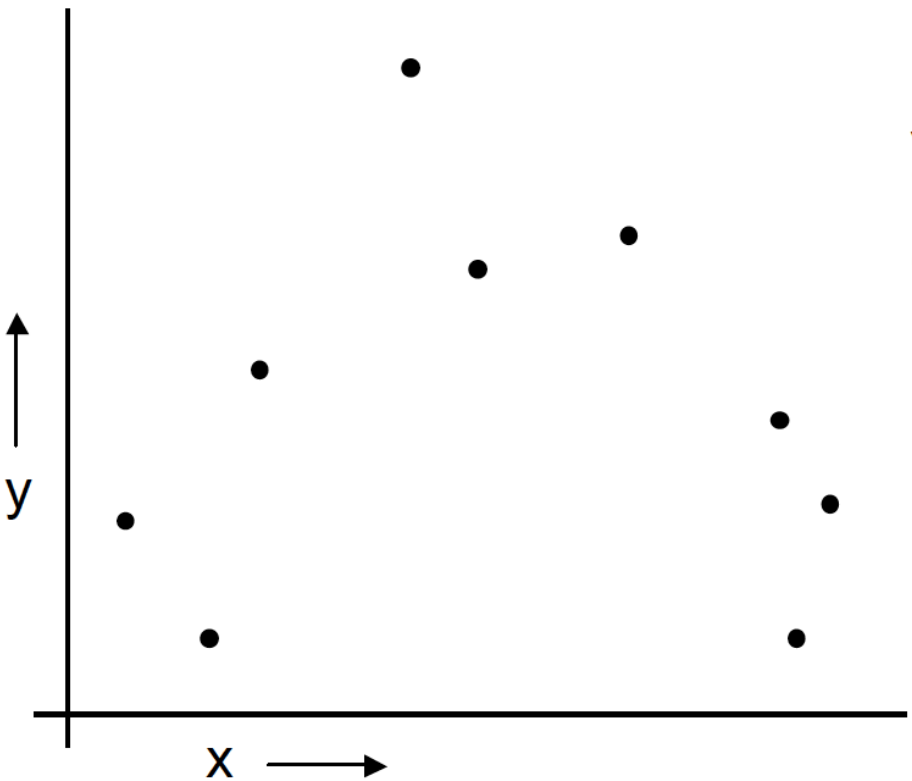
- Allow agent to build or modify model
  - Example: Robot map its environment
    - Robot must generate some representation of its environment
    - Able to query this representation
- Models may not be intuitive to programmer
  - Mapping of inputs to solutions

RUTGERS

Decision Trees Revisited

- Used tables of data to build decision trees
- Tree we built could be seen as a learned model
  - Learned from data in table
  - Example of data mining
- Could we adapt decision tree algorithm to build tree dynamically?
  - Build tree as we classify data
  - Adapt tree based on data being classified

A Regression Problem

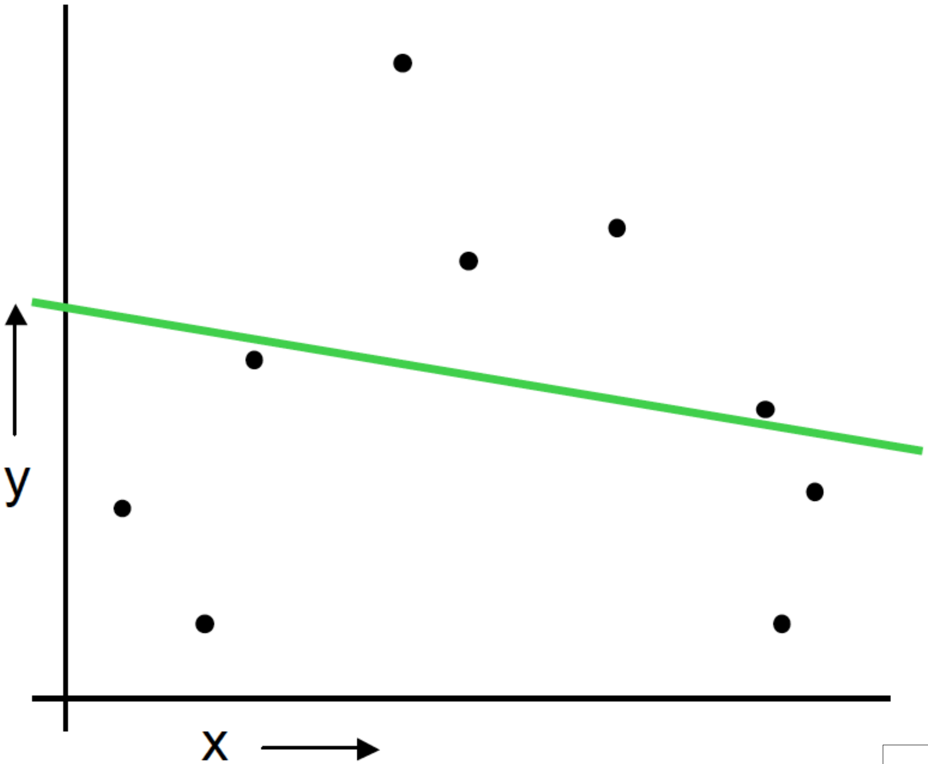


y = f(x) + noise

Can we learn f from this data?

Let's consider three methods…

Linear Regression



Objective: Minimize the

Sum of Squared Errors

i.e. sum of squared differences

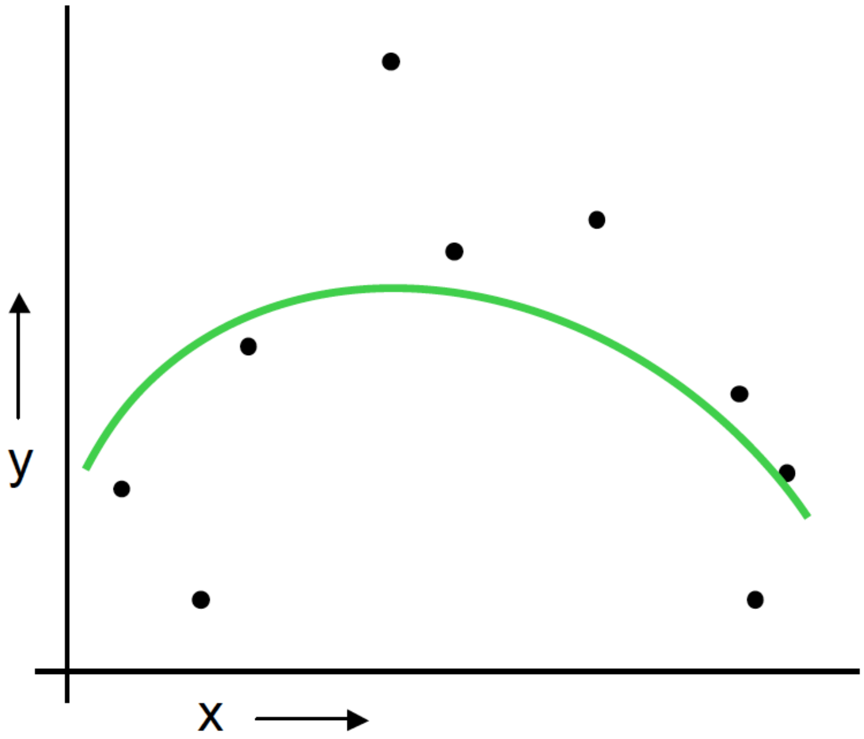
between y values

and the green line

$y = w_0 + w_1 \cdot x$

- ŷi is the prediction of the linear model
- yi the actual value for input xi
- Then minimize: Q = Σi (ŷi – yi)2

Quadratic Regression



Objective: Minimize the

Sum of Squared Errors

i.e. sum of squared differences

between y values

and the green curve

$$y = w_0 + w_1 \cdot x + w_2 \cdot x^2$$