CS148 - Building Intelligent Robots
Lecture 6: Learning for Robotics

Instructor: Chad Jenkins (cjenkins)
Administrivia: good news

• No class next Tuesday 10/12
  – you can show up, but I will not be here

Rudy, you are like a robotics teacher out of the country

A robotics teacher out of the country?

Yeah, no class!
Administrivia: bad news

• Someone left the Lego lab open and unattended yesterday!!

• This is a huge problem and can lead to disaster for the class
  – if the kits were to disappear, how would you implement the labs and projects

• This situation must be taken seriously
  – thus, I will deduct a 1% from the final grade of ALL students in the standard track if lab is left open and unattended again
  – next infraction will be 2%, then 4%, 8%, ....
Machine learning (from Wikipedia)

• Machine learning is an area of artificial intelligence involving developing techniques to allow computers to "learn".
  – More specifically, machine learning is a method for creating computer programs by the analysis of data sets, rather than the intuition of engineers.
  – Machine learning overlaps heavily with statistics, since both fields study the analysis of data.
  – Applications: medical diagnosis, detecting credit card fraud, stock market analysis, classifying DNA sequences, speech and handwriting recognition, game playing and robot locomotion.
Machine learning taxonomy

- Machine learning groups into the following categories:
  - supervised learning: an algorithm generates a function that maps inputs to desired outputs
    - given data for $x$ and $y$, find $f(x) = y$
    - classification, regression
  - unsupervised learning: an algorithm generates a model for a set of inputs
    - given $x$, find models underlying $x$
    - feature extraction, density estimation
  - reinforcement learning: an algorithm learns a policy of how to act given an observation of the world
    - find a policy $u$ such that expected outcomes $o = u(x, \text{actions})$
  - learning to learn: an algorithm learns its own inductive bias based on previous experience.
Supervised learning: regression

- **Ask N students:**
  - $x$: # of CS classes taken
  - $y$: typical Mountain Dew consumption

- **Supervised problem:**
  - function of MD consump. w.r.t. CS background
    - $f(x) = y$
Supervised learning: regression

- **Ask N students:**
  - $x$: # of CS classes taken
  - $y$: typical Mountain Dew consumption

- **Supervised problem:**
  - function of MD consump. w.r.t. CS background
    - $f(x) = y$

- **Linear regression**
  - fit a line: $f(x) = ax + b = y$
Unsupervised learning: dimension reduction

• Ask N students:
  – x1: # of CS classes taken
  – x2: typical Mountain Dew consumption

• Unsupervised problem:
  – find underlying coordinate system

• Principal Components Analysis
  – find linear system that best expresses data
Examples for robotics

• **Inverse dynamics**
  – \( f(\text{desired states}) = \text{control commands} \)
  – collect control commands and states from robot teleoperation

• **Inverse kinematics**
  – \( f(\text{endeffector position}) = \text{joint angles} \)
Unsupervised learning: clustering

- **Ask N CS students:**
  - $x_1$: # of systems classes taken
  - $x_2$: # of AI classes taken
  - $x_3$: # of theory classes taken

- **Unsupervised problem:**
  - find categories of students
    - sets of students $C_1$, $C_2$, etc.
Unsupervised learning: clustering

• Ask N CS students:
  – $x_1$: # of systems classes taken
  – $x_2$: # of AI classes taken
  – $x_3$: # of theory classes taken
  – 3 dimensional data

• Unsupervised problem:
  – find categories of students
    • sets of students $C_1$, $C_2$, etc.

• Clustering
  – estimates cluster associations

• K-means clustering
  – assume $K$ clusters with initial locations
  – find cluster nearest to each point
  – move cluster to centroid
Supervised learning: classification

- From clustering we know:
  - \( x \): classes taken
  - \( y \): category (AI, systems, ...)

![Graph showing classification in AI, Systems, and Theory categories.](image-url)
Supervised learning: classification

• From clustering we know:
  – $x$: classes taken
  – $y$: category (AI, systems, ...)

• Find $f(x) = y$
  – decision boundaries
 Supervised learning: classification

• From clustering we know:
  – x: classes taken
  – y: category (AI, systems, ...)

• Find $f(x) = y$
  – decision boundaries

• Classify new point $x_{\text{new}}$
Supervised learning: classification

• From clustering we know:
  – $x$: classes taken
  – $y$: category (AI, systems, ...)

• Find $f(x) = y$
  – decision boundaries

• Classify new point $x_{\text{new}}$
  – using decision boundaries
Examples for robotics

• Behavior arbitration
  – \( f(\text{sensor readings}) = \text{behavior selection} \)

• Landmarking for robot navigation
  – \( f(\text{sensor readings}) = \text{landmark category} \)

• Neural navigation of mobile robots
  – \( f(\text{brain readings}) = \text{controller states} \)
Reinforcement learning (from Wikipedia)

- A class of problems in machine learning which postulate an agent exploring an environment in which the agent perceives its current state and takes actions.

- The environment, in return, provides a reward (which can be positive or negative).

- Reinforcement learning algorithms attempt to find a policy for maximizing cumulative reward for the agent over the course of the problem.
Reinforcement learning (from Wikipedia)

- RL differs from supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

- RL focuses on on-line performance
  - balance between
    - exploration (of uncharted territory)
    - exploitation (of current knowledge).
Formal RL model

• A RL model consists of
  – a discrete set of $S$ states
    • models describing the robot’s environment
  – a discrete set of $A$ actions
    • actions the robot can take to change state
  – a set of scalar reinforcement signals $R$
    • functions evaluating short-term and long-term reward
  – a robot control policy $P$
    • given state $s$ at time $t$, selects action $a$ to maximize rewards $r$
    • what we are trying to learn
Formal RL model

• A RL model consists of
  – a discrete set of $S$ states
    • models describing the robot’s environment
  – a discrete set of $A$ actions
    • actions the robot can take to change state
  – a set of scalar reinforcement signals $R$
    • functions evaluating short-term and long-term reward
  – a robot control policy $P$
    • given state $s$ at time $t$, selects action $a$ to maximize rewards $r$
    • what we are trying to learn

Does anyone see a problem with this?
Issues for reinforcement learning

• Estimation of states and state transitions

• Partial observability
  – robot observes noisy or incomplete information about the world

• Discretization of states
  – make assumptions or use domain knowledge

• Discretization of actions/behaviors
  – hand coded robot controllers or
  – learn them automatically (this is my research)
Approaches to reinforcement learning

• Find policies as the utility or value of actions with respect to outcomes

• Two general approaches to learning policies
  – Search
    • search over the space of actions to find their utility
    • techniques: breadth-first, depth-first, genetic algorithms
  – Statistical modeling
    • probabilistically model the utility of taking actions
    • use statistical techniques with dynamic programming
    • techniques: Markov Decision Processes
Genetic algorithm procedure

- Randomly generate “DNA” of an initial population $M(0)$
  - an individual has a genotype that encodes a control policy

- Compute and save the fitness $u(m)$ for each individual $m$ in the current population $M(t)$
  - users defines the fitness function

- Define selection probabilities $p(m)$ for each individual $m$ in $M(t)$ so that $p(m)$ is proportional to $u(m)$

- Generate new population $M(t+1)$ by probabilistically selecting individuals from $M(t)$ to produce offspring
  - genetic operators: crossover, mutation, ...

- # Repeat step 2 until satisfying solution is obtained.
Constraint optimization

• Genetic algorithms are related to constraint optimization

• Constraint optimization consists of
  – an objective function to be minimized (fitness function)
  – a set of constraint functions to be maintained
Markov Decision Processes (MDPs)

- a set of states $S$
- a set of actions $A$
- a function of expected reward $R(s,a) \rightarrow \text{real numbers}$
- a state transition function $T(s,a) \rightarrow \Pi(S)$
  - a member of $\Pi(S)$ is a probability distribution over the set $S$
    - $\Pi(S)$ maps states to probabilities
- $T(s,a,s')$ is the probability of making a transition from state $s$ to state $s'$ using action $a$. 
The Markov Property

• A system is Markovian
  – if the state transitions are independent of previous state transitions or agent actions

• The Markov property allows for future states to be estimated using only the current state

• The past and the future are independent given the present
Partially Observable MDPs (POMDPs)

- Robots rarely have complete information

- A robot can only estimate the current state of the environment
  - state estimation for robot belief $b$

- Incorporate into MDP
  - finite set of observations $I$
    - the probability of observing $w$ and ending in state $s'$ after taking action $a$
  - observation probability $O(s', a, w)$
Hidden Markov Models (HMMs)
Petri-nets
State estimation: localization

- Estimate the distribution of probable robot locations
  - Each particle is a hypothesis of a probable robot location
- By navigating the world, impossible hypotheses are eliminated
- Over time, the particle distribution identifies robot location

Fox et al.
Particle filtering

- Condensation

- Distribution as particles
  - particle = hypothesis

- Evaluate distribution through observation on particles
Mapping

• Represent environment as a distribution

• Estimate the probability of a position of the world being occupied

Thrun et al.
From AAAI94
Learning from demonstration

• Humans and the natural world are working models of control and policy learning

• Leverage human tutelage and/or performance to build robot controllers
Probabilistic road maps: learning phase

- Build map of valid configurations
  - start with an initial configuration

Space of valid configurations
Space of invalid configurations

A robot configuration
Boundary of valid configurations

Configuration space
\( C = [\Theta_1, \Theta_2, \ldots \Theta_N] \)

[Kavraki, Svetska, Latombe, Overmars, 95]
Probabilistic road maps: learning phase

- Build map of valid configurations
- Sample neighbors of current config

[Kavraki, Svetska, Latombe, Overmars, 95]
Probabilistic road maps: learning phase

• Build map of valid configurations
• Sample neighbors of current config
• Determine valid neighbors

[Kavraki, Svetska, Latombe, Overmars, 95]
Probabilistic road maps: learning phase

- Build map of valid configurations
- Sample neighbors of current config
- Determine valid neighbors
  - remove invalid
  - place edge transitions between valid neighbors

[Kavraki, Svetska, Latombe,Overmars, 95]
Probabilistic road maps: learning phase

- Build map of valid configurations
- Sample neighbors of current config
- Determine valid neighbors
- Continue exploration from valid neighbors

[Kavraki, Svetska, Latombe, Overmars, 95]
Probabilistic road maps: query phase

• Given learned map

• Find a valid control path between two configurations

• Search on an undirected graph

[Kavraki, Svetska, Latombe, Overmars, 95]
Additional references

- Duda and Hart, “Pattern Classification”
- Bishop, “Neural Networks for Pattern Recognition”
Additional references


• Read my papers (I command you... Muhuwahahaha)