mc375: intro to robotics

learning.

adaptation vs learning.

- learning produces changes within an organism that, over time, enables it to perform more effectively within its environment
- adaptation in learning through making adjustments in order to be more attuned to its environment.
  - different time scales: acclimatization (slow) vs homeostatis (rapid)
  - different levels: genotypic vs phenotypic

types of adaptation.

- behavioral -- behaviors are adjusted relative to each other
- evolutionary -- descendents are based on ancestor's performance over long time scales
- sensory -- sensors become more attuned to the environment
- learning as adaptation -- anything else that results in a more ecologically fit agent

importance of learning.

- learning is more than just adaptation
- learning, the ability to improve one's performance over time, is considered the main hallmark of intelligence, and the greatest challenge of AI
- learning is particularly difficult to achieve in physical robots, for all the reasons that make intelligent behavior in the physical world difficult

what learning enables.

- introducing new knowledge (facts, behaviors, rules) into the system
- generalizing concepts
- specializing concepts
- reorganizing information
- creating or discovering new concepts
- creating explanations
- reusing past experiences

why learn in robots?

- providing the robot with the ability to adapt to changes in its task and/or environment
- providing the robot with the ability to improve performance
- providing the robot designer with a way of automating the design and/or programming of the robot

what can be done?

- automatically design the robot's body
- automatically design a physical network for its processor
- automatically generate its behaviors
- automatically store and re-use its previous executed plans
- automatically improve the way its layers interact
- automatically tune the parameters within the behaviors and many more ...

what has been done?

- parts of robot bodies, brains (i.e., processors), and programs have been automatically generated (i.e., learned)
- robots given initial programs have used experience and trial & error to improve the programs (from parameter tuning to switching entire behaviors)
- robots programmed for a task have adapted to changes in the environment (e.g., new obstacles, new maps, heavier loads, new goals)

challenges in robots

- situatedness in the world: noise, occlusion, dynamics, hard to model, etc.
- real time constraints: slow learners die easily in the real world
- simultaneous and multi-modal: multiple goals & tasks, need to walk and talk (not just either or)

challenges in learning.

- credit assignment: who is to credit/blame for outcome?
- saliency: what is relevant right now?
- new term: when should a new concept/representation be created?
- indexing: how to organize the memory?
- utility: what should be forgotten?

levels of learning.

- within a behavior
  - suitable responses
  - suitable stimulus
  - suitable behavioral function mapping
  - magnitude of response (gain)
  - whole new behavior
- within a behavior assemblage
  - suitable set of behaviors
  - relative strengths
  - suitable coordination function

learning methods.

- reinforcement learning
- neural network (connectionist) learning
- evolutionary learning
- co-evolutionary learning
- learning from experience
  - memory-based
  - case-based
- inductive learning
- explanation-based learning
- multistrategy learning

types of learning.

- numeric or symbolic
  - numeric: manipulate numeric functions
  - symbolic: manipulate symbolic representations
- inductive or deductive
  - inductive: generalize from examples
  - deductive: optimize what is known
- continuous or batch
  - continuous: during interaction with the world
  - batch: after interaction, all at once

some terminology.

- reward/punishment
  - positive/negative feedback
- cost function/performance metric
  - scalar (usually) goodness measure
- induction
  - generating a function (a hypothesis) that approximates the observed examples
- teacher/critic
  - provides feedback

more terminology.

- plant/model
  - system/agent that we want to train
- convergence
  - reaching a desired (or steady) state
- credit assignment problem
  - who should get the credit/blame?
  - hard to tell over time
  - hard to tell in multi-robot systems

reinforcement learning (RL).

- currently the most popular approach to learning on mobile robots
- inspired by conditioning in psychology
- Law of effect: applying a reward immediately after the response increases its probability of reoccurring while providing punishment after the response will decrease the probability [Thorndike 1911]
- translated to robotics: some combinations of stimuli (i.e., sensor readings and/or state) and responses (i.e., actions/behaviors) are coupled with subsequent reward in order to increase their probability of future use

reinforcement learning, cont.

- desirable responses of outcomes are positively reinforced (rewarded) and thus strengthened, while undesirable ones are negatively reinforced (punished) and thus weakened
- this very general notion can be translated into a variety of specific reinforcement learning algorithms

challenges in RL.

- learning from delayed rewards: the problem is difficult if the feedback is not immediate
- credit assignment problem: when something good or bad happens, what exact state/condition-action/behavior should be rewarded or punished?
- common approach: use the expected value of exponentially weighted past/future reinforcement

RL algorithms.

- proscribe exact mathematical functions that associate states/situations, actions/behaviors, reinforcement and various associated parameters
- some of these algorithms (Q-learning, TD learning, etc.) have well-understood convergence properties; they are guaranteed to make the robot learn the optimal solution

optimality in learning.

- optimality depends on strong assumptions: the robot must be given infinitely many trials of each state/action combination, must know what state it is in and what action it has executed, and the world must not change too quickly
- but: there is not enough time for infinite trials, outcomes of actions are uncertain, the world can change
- thus: optimality is impractical

observability.

- the world to a robot is partially observable, the robot does not know exactly what state/situation it is in at all times
- learning algorithms have been developed for partially observable environments, and while they are more realistic, they require even more time to converge
- in general, managing uncertainty in learning is very hard

unsupervised learning.

- RL is a form of unsupervised learning
- RL allows a robot to learn on its own, using its own experiences (and some built-in notions of desirable and undesirable situations, associated with reward and punishment)
- the designer can also provide reinforcement (reward/punishment) directly, to influence the robot
- the robot is never told what to do

RL function and critic.

- RL systems contain a reinforcement function, which determines when and how much positive or negative reinforcement to deliver to the robot
- the critic is the part of the RL system which provides the reinforcement, i.e., executes the reinforcement function

types of critic.

- the critic can be
  - external: if the user provides the reinforcement
  - internal: if the system itself provides the reinforcement
- in both cases the approach is unsupervised in that the answer is never given explicitly by the critic

what can be RL learned?

- policy: the function that maps the states/situations to actions/behaviors
- utility: the function that gives a value to each state
- both of the above are learned relative to a specific goal
- if the goal changes, so must the policy and/or the utility function

adaptive heuristic critic.

- example of an RL critic [Barto & Sutton]
- the process of learning what action to take in what state (the policy) is separate from learning the value of each state (the utility function)
- both are based on trying different actions in different states and observing the outcomes over time

Q learning.

- the most popular RL algorithm [Watkins 1980's]
- a single utility Q-function is learned in order to evaluate both actions and states
- shown to be superior to AHC
- Q values are stored in a table, usually
- updated at each step, using an update rule

  $Q(x,a) \leftarrow Q(x,a) + b(r + \lambda E(y) - Q(x,a))$

  x = state r = reward

  a = action λ = discount factor

  b = learning rate E(y) = utility of the state y, computed as max(Q(y,a)) for all actions a
- guaranteed to converge to optimal, given infinite trials

supervised learning.

- requires the user to give the exact solution to the robot in the form of the error direction and magnitude
- thus, the user must know the exact behavior for each situation
- this approach can take a very long time and requires user/designer supervision, which is not always desirable

neural networks.

- hebbian learning (increase synaptic strength along pathways associated with stimulus and correct response)
- perceptron learning (delta rule or back-propagation)

NN's are RL.

- in all NN's, the goal is to minimize the error between the network output and the desired output
- achieved by adjusting the weights on the network connections
- NN's perform supervised RL with immediate error feedback

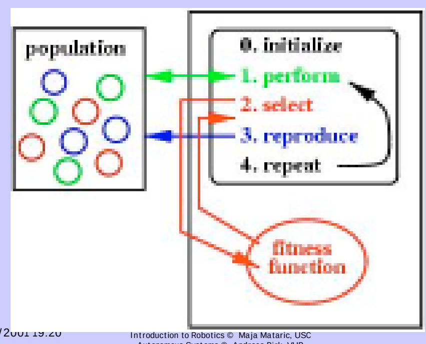
associative learning.

- learning new behaviors by associating sensors and actions into rules
- example:
  - 6-legged walking (Edinburgh)
  - whisker sensors first, IR and light later
  - 3 actions: left, right, ahead
  - user provided feedback -- *shaping*
  - learned avoidance, pushing, wall-following, light seeking

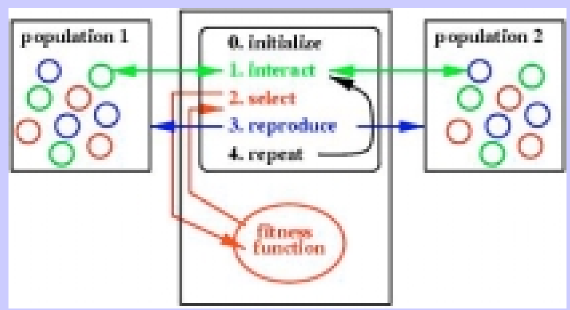
neural network examples.

- robot motion planning, articulation/manipulation
- control of complex plants: robots, aircraft
- control and coordination of multiple vehicles
- some domains and tasks lend themselves very well to supervised NN learning
- the answer to any given situation is well known and can be trained
- best example is robot motion planning for articulation/manipulation; NN's widely used for learning inverse kinematics

evolution.



co-evolution.



evolutionary methods.

- genetic/evolutionary approaches are based on the evolutionary search metaphor
- states/situations and actions/behaviors are represented as "genes"
- different combinations are tried in different "individuals" in a "population"
- individuals with the highest "fitness", that perform the best, are kept as "survivors"; the others are discarded -- this is the *selection* process

evolutionary methods, cont.

- the survivors' genes are mutated, crossed-over, etc., and new individuals are so formed, which are then tested and scored
- in effect, the evolutionary process is searching through the space of solutions to find the one with the highest fitness
- often used for solving optimization problems
- trick is proper definition of representation, operators and fitness function

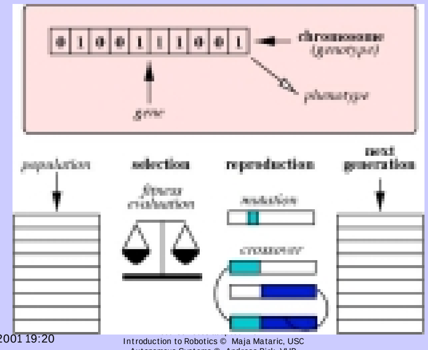
evolutionary methods:

applications.

- tuning parameters (such as gains in a control system)
- developing controllers (policies) for individual robots
- developing group strategies for multi-robot systems (by testing groups as populations)

genetic algorithms.

[Holland, 1975]



GA's vs GP's.

- GA = genetic algorithm
  - representation is strings of genes
- GP = genetic programming
  - representation is pieces of executable programs
- GP operates at a higher level of abstraction than GA's
- examples:
  - classifier systems
  - evolving structure/behavior [Sims]

fuzzy logic.

- fuzzy control produces actions using a set of fuzzy rules based on fuzzy logic
- this includes:
  - fuzzifying: mapping sensor readings into a set of fuzzy inputs
  - fuzzy rule base: a set of IF-THEN rules
  - fuzzy inference: maps fuzzy sets onto other fuzzy sets using membership functions
  - defuzzifying: mapping a set of fuzzy outputs onto a set of crisp output commands

fuzzy control.

- fuzzy logic allows for specifying behaviors as fuzzy rules
- such behaviors can be smoothly blended together
- fuzzy rules can be learned

memory-based learning (MBL).

- involves storing the exact parameters of a situation/action
- important for systems with a great deal of parameters, where blind search through parameter space would take too long
- takes a lot of space for data storage
- a trade-off against computation time
- assumed it is best to use space to remember rather than time to compute

symbolic learning.

- induction on decision trees
- learning new rules given a set of predefined rules
- e.g., by inference or other logical techniques
- example: RoboSOAR
  - logical reasoning system
  - production system based on implications or rules
  - uses chunking
  - conflict resolution mechanism when different rules apply
  - continuous conversion of deliberative information to efficient representations

multi-strategy learning.

- some learning methods are effectively combined
- NN's and RL is common
  - use NN's to generalize states
  - then apply RL in a reduced state space
- NN's and MBL
  - use NN's to interpolate between instances
- NN's and fuzzy is also popular
- NN's and GA's is also useful

learning to play games.

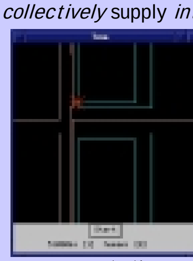
- learning from humans
  - Checkers [Samuels, 1959]
  - too many games are needed
  - humans are noisy
  - humans are learning
- learning from computers
  - Backgammon [Tesauro, 1992]
  - lack of generalization
  - deceptive landscape
  - premature convergence

Tron.

- learning from humans on the Internet

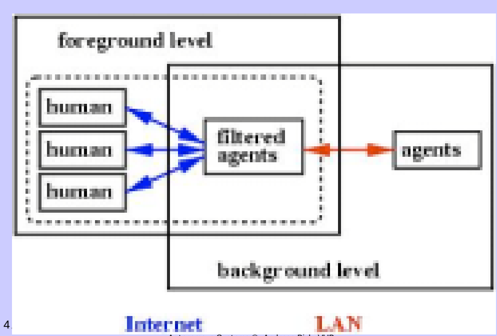
[Funes, Sklar, Juillé & Pollack, 1998]

- human users *collectively* supply *fitness function*
- software agents *collectively* supply *intelligent opponent*

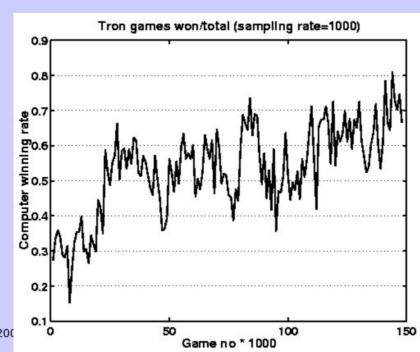


http://www.demo.cs.brandeis.edu/tron

Tron architecture: two levels of co-evolution.



Tron results: computer learning.



state of the art.

- there are many learning techniques
  - use dependant on application
- robotics is a particularly challenging domain for learning
  - very few successful examples
  - active area of research
- research journals
  - Machine Learning, Artificial Intelligence, Fuzzy Computing, Neural Networks, Adaptive Computing

state of the art, cont.

- supervised learning has been used to learn controllers for complex architectures, e.g., many DOF, as well as for navigating specific paths and performing articulated skills (e.g., juggling, pole-balancing)
- MBL has been used to learn controllers for very similar manipulator tasks (as above)
- RL has been used to learn controllers for individual navigating robots, groups of foraging robots, map-learning and maze-learning robots
- evolutionary learning has been used to learn controllers for individual navigating robots, maze-solving robots, herding robots and foraging robots

reading.

- Reward Functions for Accelerated Learning [Mataric]
  (in course pack II)
- Issues in Evolutionary Robotics [Harvey, Husbands and Cliff]
  (in course pack II)