Learning

A set of
fundamental ideas

- # Types of Adaptation (McFarland)
    - **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
- # Learning Methods
    - Reinforcement learning
    - Neural network (connectionist) learning
    - Evolutionary learning
    - Learning from experience
        - memory-based
        - case-based
    - Inductive learning
    - Explanation-based learning
    - Multistrategy learning

Types of Learning

- **Numeric or symbolic**
  - **numeric:** manipulated numeric functions
  - **symbolic:** manipulate symbolic representations
- **Inductive or deductive**
  - **inductive:** generalize from examples
  - **deductive:** optimize what is known
- **Continuous or batch**
  - **continuous:** during interaction w/ 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
- 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

- # Unsupervised Learning
  - **Reinforcement Learning (RL)** allows a robot to learn on its own, using its own experiences as reinforcement (with some built-in notion of desirable and undesirable situations, associated with reward and punishment)
  - the designer can also <u>provide reinforcement</u> (reward/punishment) directly, to influence the robot
  - However, the robot is <u>never told what to do</u>
- # Q Learning Algorithm
  - **Q(x,a) <-- Q(x,a) + b (r + lambda E(y) - Q(x,a))**
    - **x** is state, **a** is action
    - **b** is learning rate
    - **r** is reward
    - **lambda** is discount factor (0,1)
    - **E(y)** is the utility of the state **y**, computed as **E(y) = max(Q(y,a))** for all actions **a**
  - Guaranteed to <u>converge</u> to optimal, given <u>infinite trials</u>

- Supervised Learning
    - 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)
    - Algorithm.

- NNs are RL
  - In all NNs, the goal is to minimize the error between the network output and the desired output
  - This is achieved by adjusting the weights on the network connections
  - **Note:** NNs are a form of reinforcement learning
  - NNs perform supervised RL with immediate error feedback
- **Classical Conditioning**
  - Classical conditioning comes from psychology (Pavlov 1927)
  - Assumes that **U**nconditioned **S**timuli (e.g., food) cause **U**nconditioned **R**esponses (e.g., salivation); **US** => **UR**
  - A **C**onditioned **S**timulus is, over time, associated with an unconditioned response (**CS** => UR)
  - E.g., CS (bell ringing) => UR (salivation)
  - Instead of encoding SR rules, conditioning can be used to form the associations automatically
  - Can be encoded in NNs

- **Connectionist Adaptive Heuristic Conditioning (AHC)**
    - Learn set of gain multipliers for exploration (Gachet et al)
- **Associative Learning**
  - Learning new behaviors by associating sensors and actions into rules
  - **E.g.,:** 6-legged walking (Edinburgh U.)
    - Whisker sensors ***first***, IR and light later
    - **3 actions:** left,right, ahead
    - User provided feedback (shaping)
    - Learned
      - avoidance,
      - pushing,
      - wall following,
      - light seeking

- **2-Layer Perceptron Learning**
  - Edinburgh R2
  - Whisker sensors first, IR and light later
  - **3 actions:** left,right, ahead
  - Experimenter 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

• More NN Examples

- Some domains and tasks lend themselves very well to supervised NN learning
- The best example is *robot motion planning* for **articulation/manipulation**
- The answer to any given situation is well known, and can be trained
- E.g., NNs are widely used for learning inverse kinematics

• Evolutionary Methods

- Genetic/evolutionary approaches are based on the evolutionary search metaphor
- in them, the states/situations and actions/behaviors are represented as "genes”
- different combinations are tried by various "individuals" in "populations".
- individuals with the highest "fitness" perform the best, are kept as survivors,

Evolutionary Methods (cont)

- and the others are discarded.
  - This is the selection process.
  - The survivors' "genes" are mutated, crossed-over, 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.
  - Solving optimization problems using fitness function
    - operators
    - Represent agent by a string (of genes)
  - Select ‘best’ individuals for reproduction and apply
    - Cross over, mutation

- **Summary of Evolution**
  - Evolutionary methods solve search and optimization problems using
    - a fitness function
    - operators
  - They represent the solution as a genetic encoding (string)
  - They select ‘best’ individuals for reproduction and apply: Cross over, mutation
  - They operate on populations
- **Levels of Application**
  - 1) for tuning parameters (such as gains in a control system)
  - 2) for developing controllers (policies) for individual robots
  - 3) for developing group strategies for multi-robot systems (by testing groups as populations)

Genetic Algorithm vs Genetic Programming

- **GAs v. GPs**
  - When applied to strings of genes, the approaches are classified as genetic algorithms (GA)
  - When applied to pieces of executable programs, they approaches are classified as genetic programming (GP)
  - GP operates at a higher level of abstraction than GA
- **Classifier Systems**
  - Use GAs to learn **rulesets**
  - ALECSYS - **Autonomouse**
  - Learn behaviors and coordination

Questions and Problems

- Propose how to use classical conditioning for a walking robot. Write a Lisp code for it. You want to teach robot various behaviors, not only walking.
- How to use Q algorithm to teach robot various gaits?
- Think how to adopt the decision diagrams, inductive learning and other ideas shown earlier to teach a hexapod various gaits.

Sources

- Maja Mataric