CSCI444/944

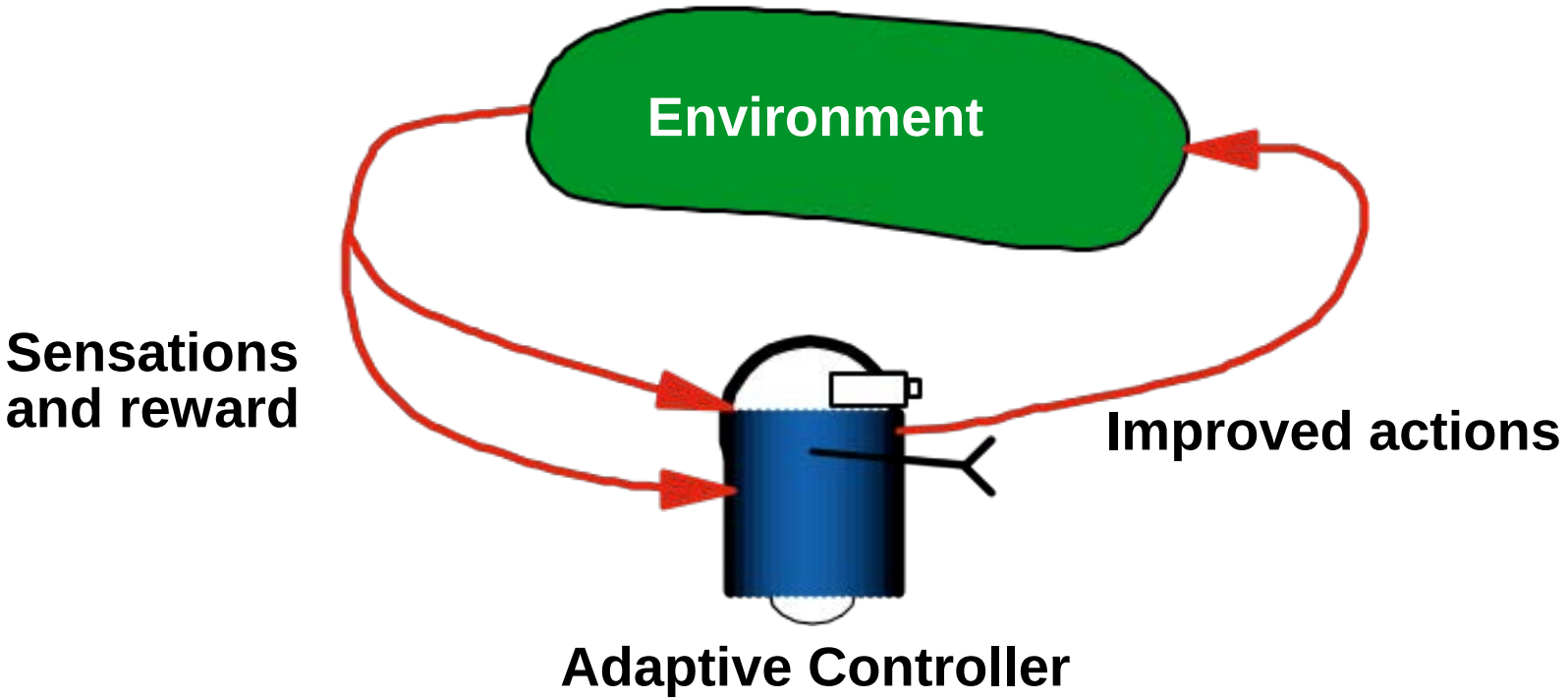
Perception and Planning

Week 8
Perception & Behaviour Learning

Learning & Adaptive Behaviours

- **Learning**: Produce changes within an agent that over time enable it to perform more effectively within its environment.
- **Adaptation**: Learning by making adjustments in order to be more attuned to its environment
  - Phenotypic (within an individual agent) or genotypic (evolutionary)
  - Acclimatisation (slow) or homeostasis (rapid)

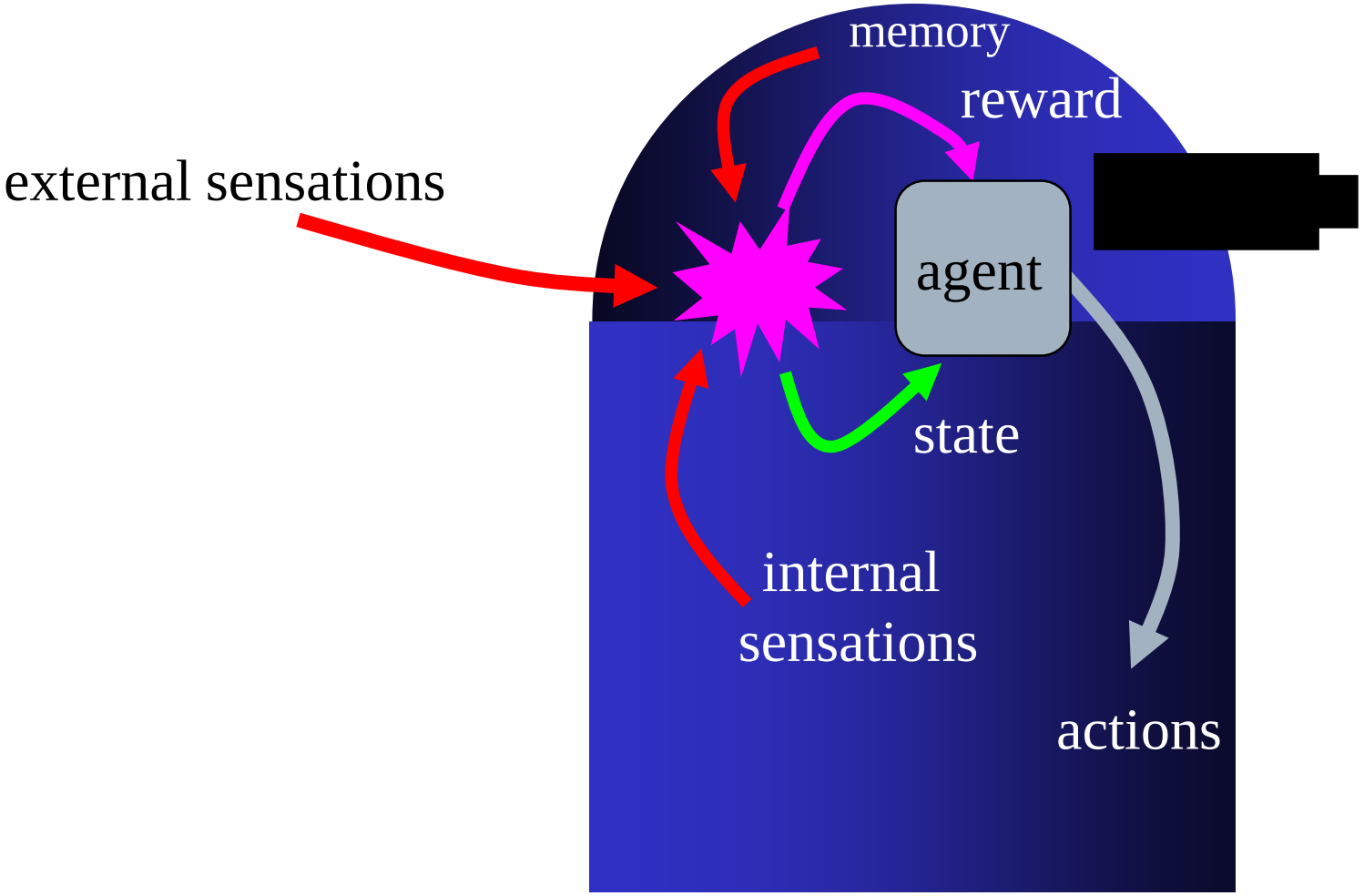
Learning & Adaptive Behaviours



"Perception-learning-action loop"

Learning & Adaptive Behaviours

Zooming in..



Learning & Adaptive Behaviours

- Learning can improve performance in many ways:
  - Introduce new knowledge (facts, behaviours, rules)
  - Generalise concepts
  - Specialise concepts for specific situations
  - Reorganise information
  - Create or discover and adapt to new concepts
  - Create explanations
  - Reuse past experiences
  - Incorporate experiences from other agents
  - …

Learning & Adaptive Behaviours

- Learning to interpret sensor information
  - Recognizing objects in the environment is difficult
  - Sensors provide prohibitively large amounts of data
  - Programming of all required objects is generally not possible
- Learning new strategies and tasks
  - New tasks have to be learned on-line in the home
  - Different inhabitants require new strategies even for existing tasks
- Adaptation of existing control policies
  - User preferences can change dynamically
  - Changes in the environment have to be reflected

Learning & Adaptive Behaviours

- Supervised learning by teaching:
  - Robots can learn from direct feedback from a teacher (i.e. a user or an external knowledge DB) that indicates the correct strategy.
    - The robot learns from examples provided by a teacher.
    - Requires the availability of relevant knowledge.
- Learning from demonstration (Imitation):
  - Robots learn by observing an agent (human or other robot) to perform the required task.
    - The robot has to be able to “understand” what it observes and map it onto its own capabilities.
- Learning by exploration:
  - Robots learns autonomously by trying different actions and observing their results.
    - Unsupervised learning.
    - The robot learns a strategy that optimizes reward.

Learning Methods

- Reinforcement learning
- Artificial Neural network (connectionist) learning
  - i.e. Multi-layer Perceptron networks
- Evolutionary learning
  - i.e. Genetic Algorithms
- Learning from demonstration
- Inductive learning
- Other
  - Trajectory Velocity Learning
  - Automatically deriving behaviours
  - …

Reinforcement Learning

- Motivated by psychology (the Law of Effect, Thorndike 1991):
- Applying a reward immediately after the occurrence of a response increases its probability of reoccurring, while providing punishment after the response will decrease the probability.
  - Learn from success and failure “by doing”.
- RL is one of the most widely used methods for adaptation in agent systems such as in robotics.

Reinforcement Learning

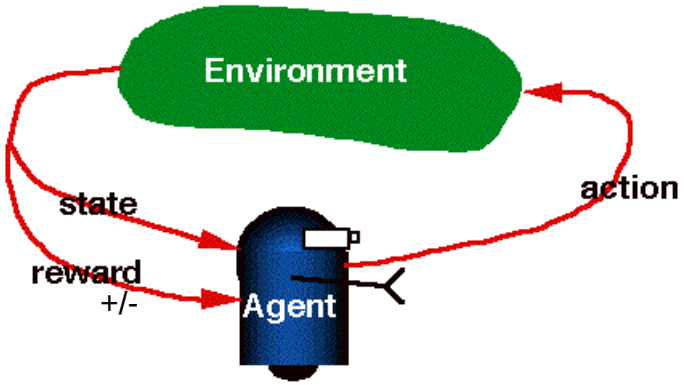
- The reward is provided either by the user or the environment
  - Intermittent user feedback
  - Generic rewards indicating unsafe or inconvenient actions or occurrences
- The robot has to explore its actions to determine what their effects are
  - Actions change the state of the environment
  - Actions achieve different amounts of reward
- During learning the robot has to maintain a level of safety.

Reinforcement Learning

- Reinforcement learning (RL) is a form of unsupervised learning. i.e. acting in the environment which returns a reward signal.
  - Learning what to do.
  - The reward defines the problem.
- In RL, the purpose of the robot or the agent is to learn the best policy, i. e. sequence of actions to reach a given goal.
  - Learn how to maximise the reward.
  - Learn optimal strategy by exploration (trial and error).

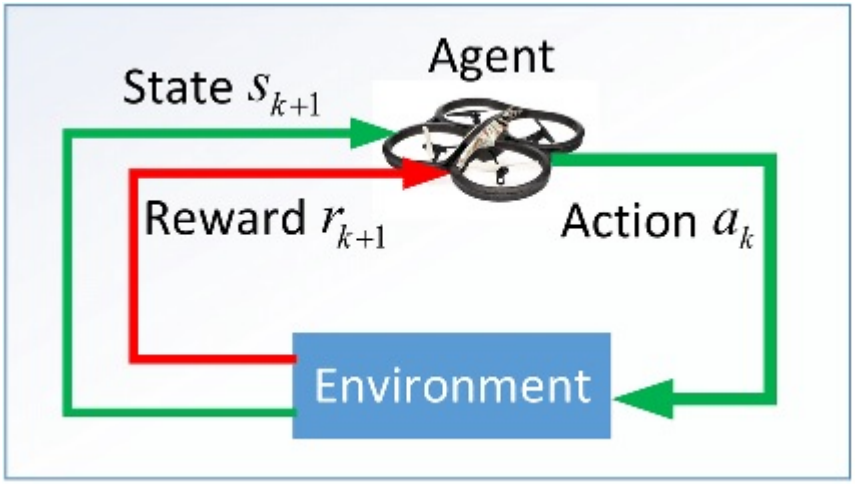
Reinforcement Learning

- **Goal:** learn an optimal policy that chooses the best action for every set of possible inputs.
- **Policy:** state/action mapping that determines which actions to take.
- Desirable outcomes are strengthened and undesirable outcomes are weakened.
- **Critic:** evaluates the system's response and applies reinforcement.
  - **external**: the user provides the reinforcement
  - **internal**: the system itself provides the reinforcement (reward function)



Reinforcement Learning

- In order to learn via RL the robot perceives environment states in a set *S* and is able to perform actions from set A.
- The robot chooses the action associated with the input state that has the most reward.
- The robot then measures the consequence of its action (positive or negative) and updates the level of reward associated with the perception/action pair.

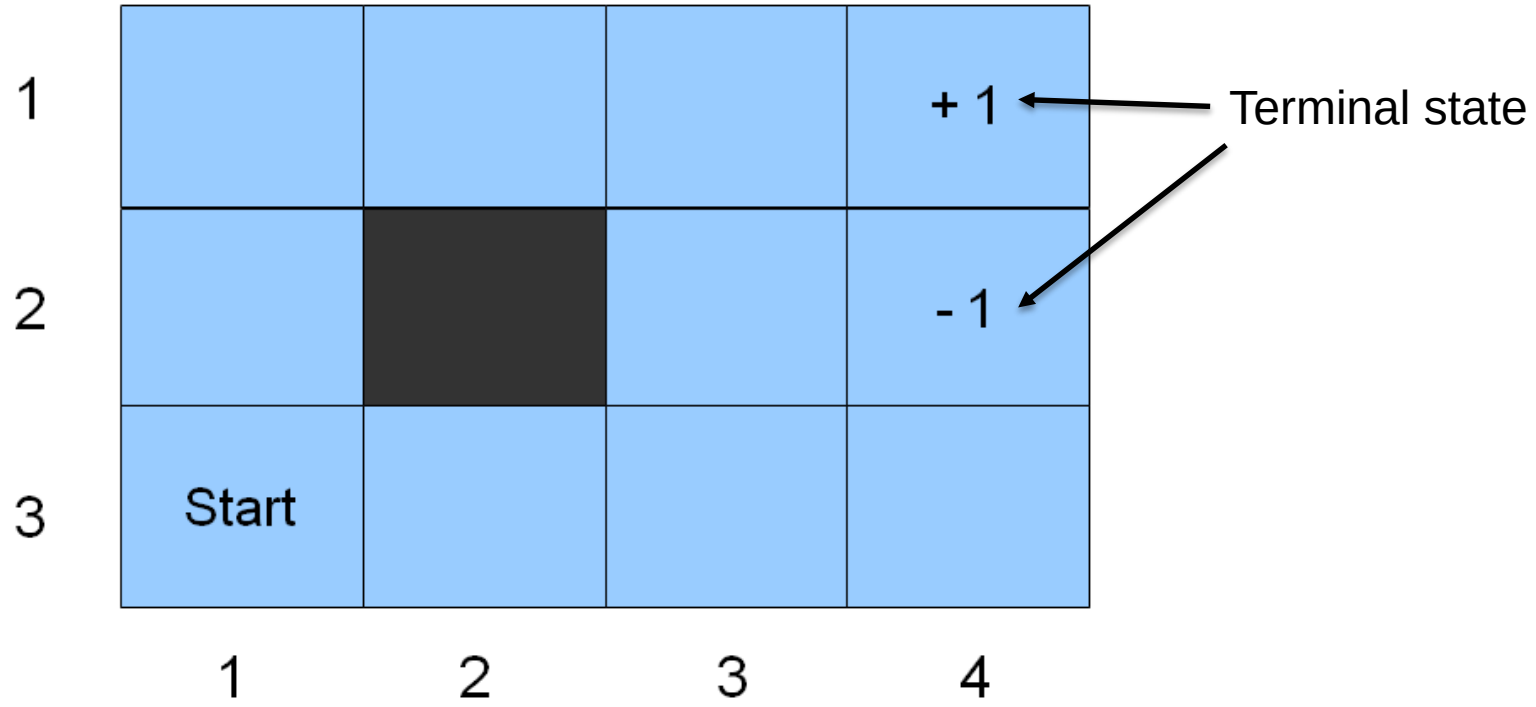


Reinforcement Learning

- This gives transition probabilities from a state to the next upon performing a given action: P(s(t+1) = s′ | s(t), a(t)).
- The robot attempts to maximise a value function V expressing the expected cumulated rewards *r(t)* over time t.
- The basic equation expressing the value function and enabling learning of the optimal policy $\pi*$ is the Bellman equation:

$$V^{\pi^*}(s) = \max_a \left[r(s, a) + \gamma \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \right]$$

Example: Reinforcement Learning



What is the optimal path when assuming r = -0.02 for all non-terminal states and A=(up,down,left,right)?

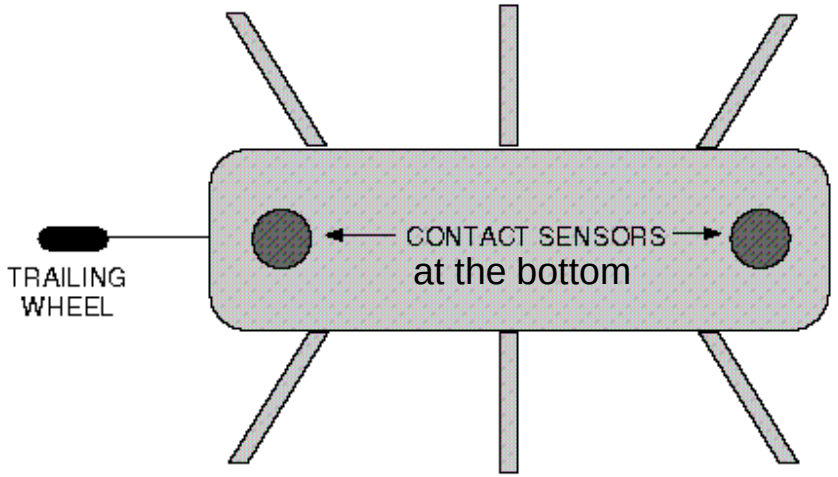
Example taken from CSCI424 (Reasoning and Learning).

- Study CSCI424 for relevant RL algorithms.

Learning to Walk with RL

Genghis: hexapod robot

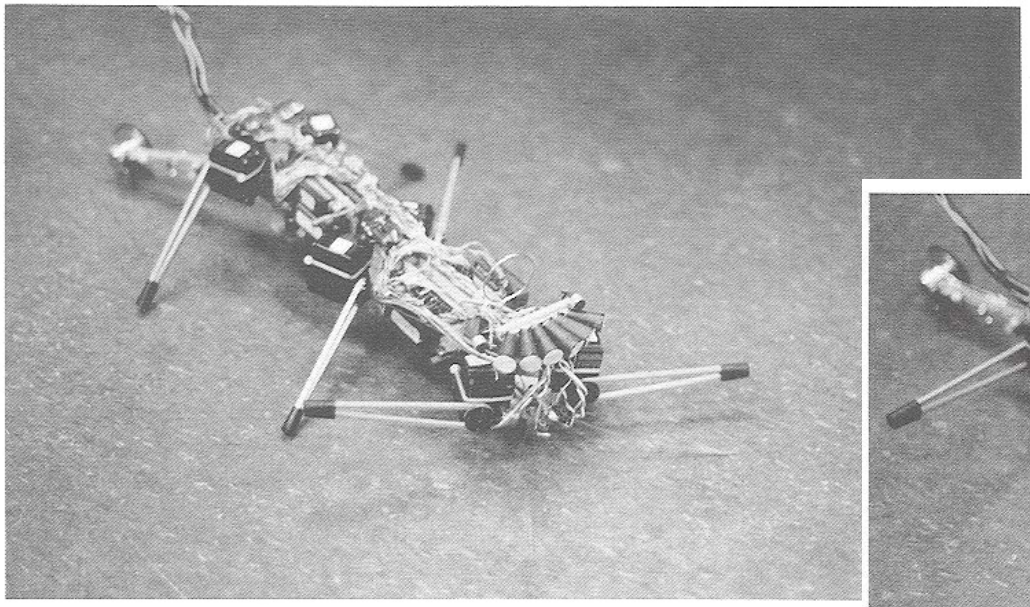
Learned stable tripod stance and tripod gait



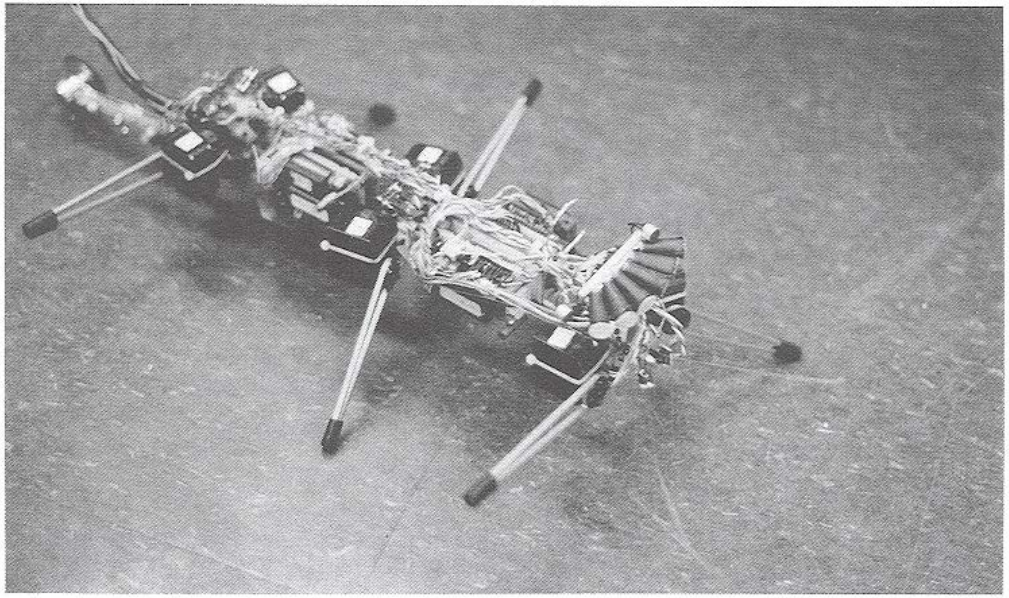
Genhis Maes, Brooks (1990)

- Rule-based subsumption controller
- Two sensor modalities for feedback:
- Two touch sensors to detect hitting the floor: negative reward
- Trailing wheel to measure progress: positive reward

Learning to Walk with RL



Initially, the behaviors do not know yet how to coordinate the legs. Two front leg “swing forward” behaviors are being activated at the same time, Genghis falls down and receives negative feedback.



Gradually, a more coherent “walk” emerges. Behavior converges towards a tripod gait: two groups of three legs are being swung forward alternately.

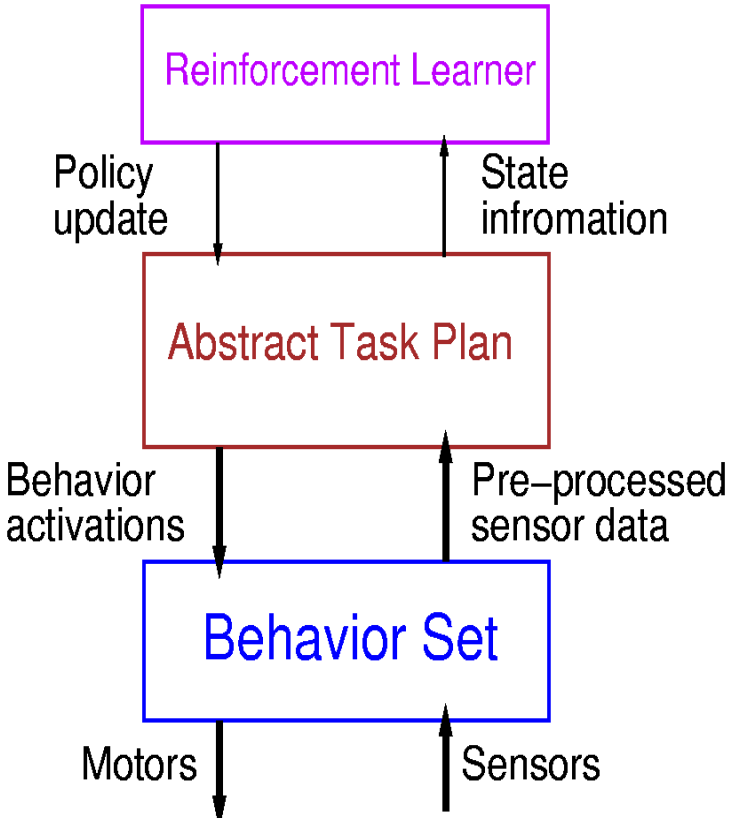
Genhis Maes, Brooks (1990)

Scaling Up: Learning Complex Tasks from Simpler Tasks

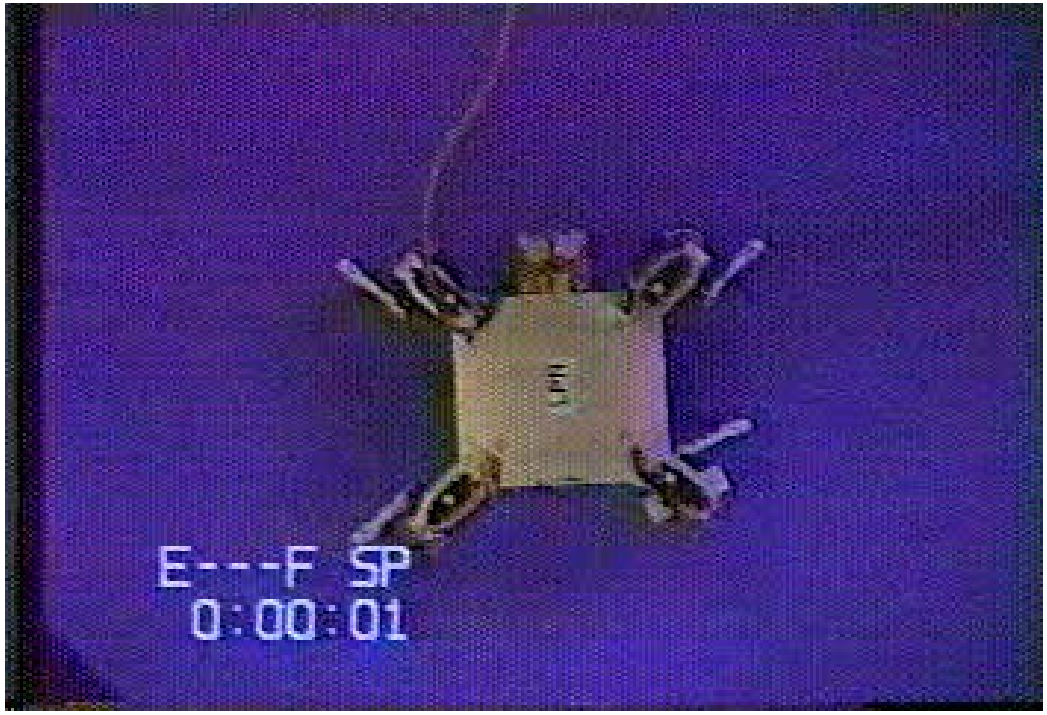
- RL can be computationally very expensive.
  - I.e. $O(N^3)$, where N is the number of states.
  - Complex tasks are hard to learn since they involve long sequences of actions that have to be correct in order for reward to be obtained.
- Complex tasks can be learned as shorter sequences of simpler tasks
  - Use control strategies expressed as subgoals that are more compact and simpler.
  - Fewer conditions have to be considered if simpler tasks that are already solved.
  - New tasks can be learned faster.
  - Hierarchical Reinforcement Learning.
    - Learning with abstract actions.
    - Acquisition of abstract task knowledge.

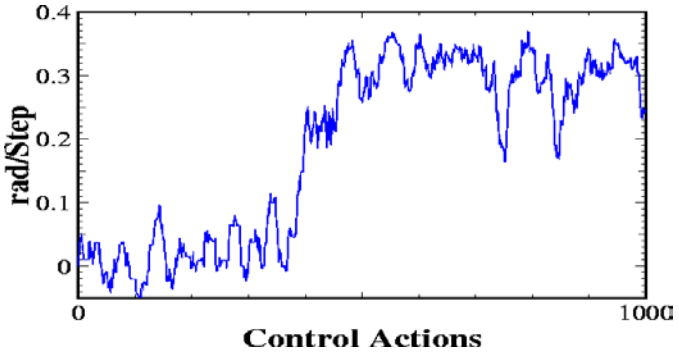
Example: Reinforcement Learning in a Hybrid Architecture

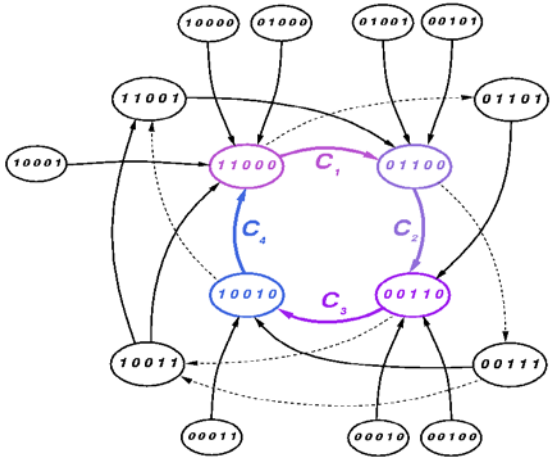
- Policy Acquisition Layer
  - Learning tasks without supervision
- Abstract Plan Layer
  - Learning a system model
  - Basic state space compression
- Reactive Behavior Layer
  - Initial competence and reactivity



Example Task: Learning to Walk

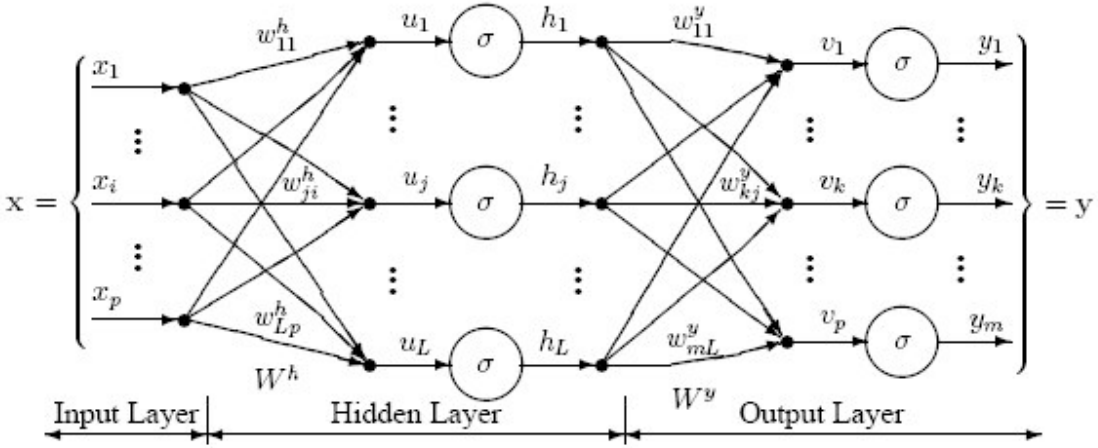






Neural Networks

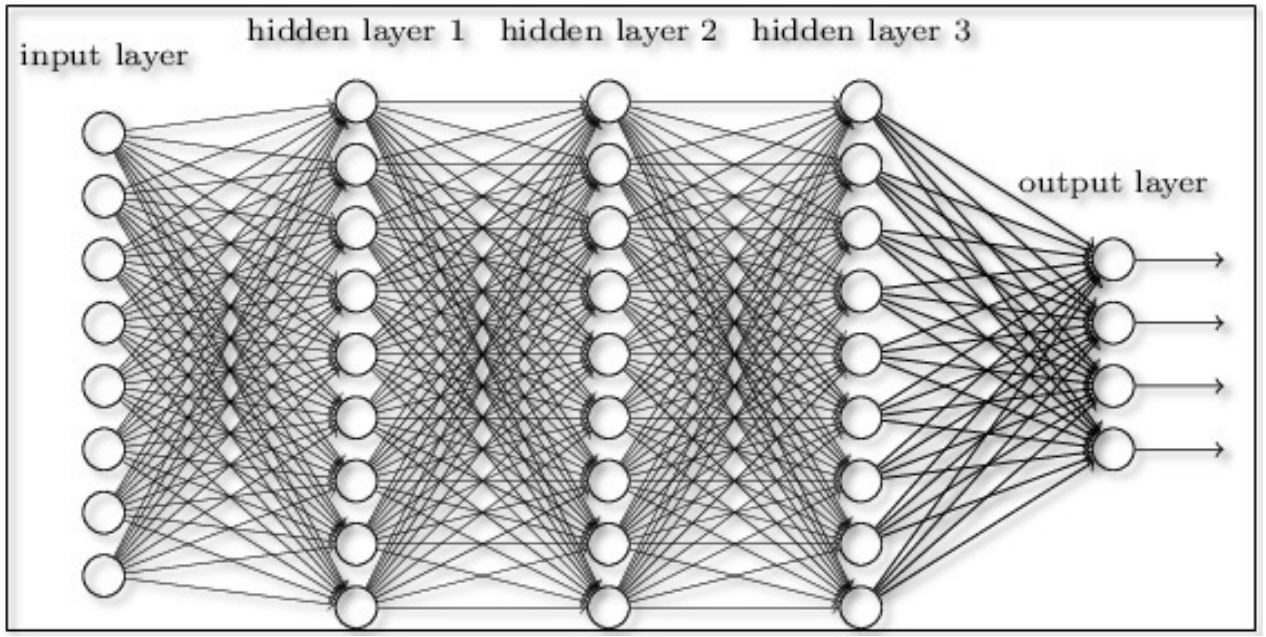
- One of the most commonly supervised learning methods.
- Used for approximating real-valued and vector-valued target functions.
- Inspired from biology: learning systems are built from complex networks of interconnecting neurons.
- The goal is to minimize, for a given input, the error between the network output and the desired output
  - This is achieved by adjusting the weights on the network connections.



The multi-layer preceptron

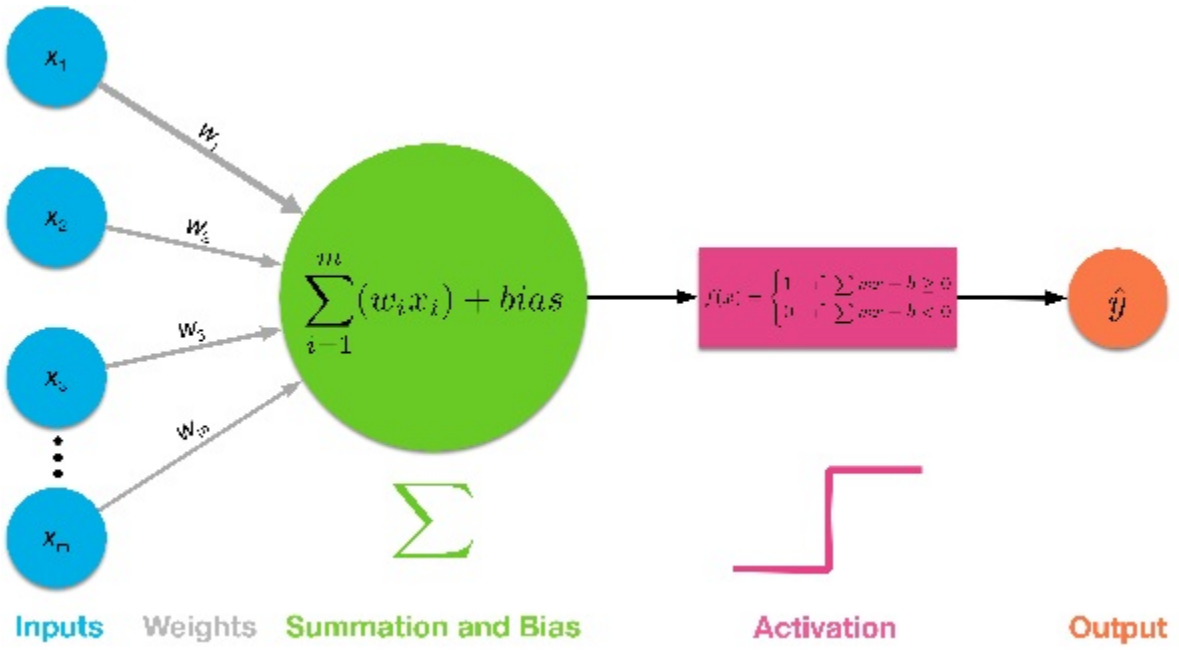
Neural Networks

- Neural networks (NN) are composed of elementary computational units (neurons) linked by weighted connections.
  - Neurons are commonly organized in layers.
  - There can be arbitrary many layers and each layer can have an arbitrary number of neurons.
  - Layers are fully connected via weighted links. A neuron receives the sum of outputs from all neurons in the previous layer.



Neural Networks

- Each neuron computes the weighted sum of its inputs and activates (produces an output) according to an activation function.
  - Threshold, linear, and sigmoid functions are most commonly used.



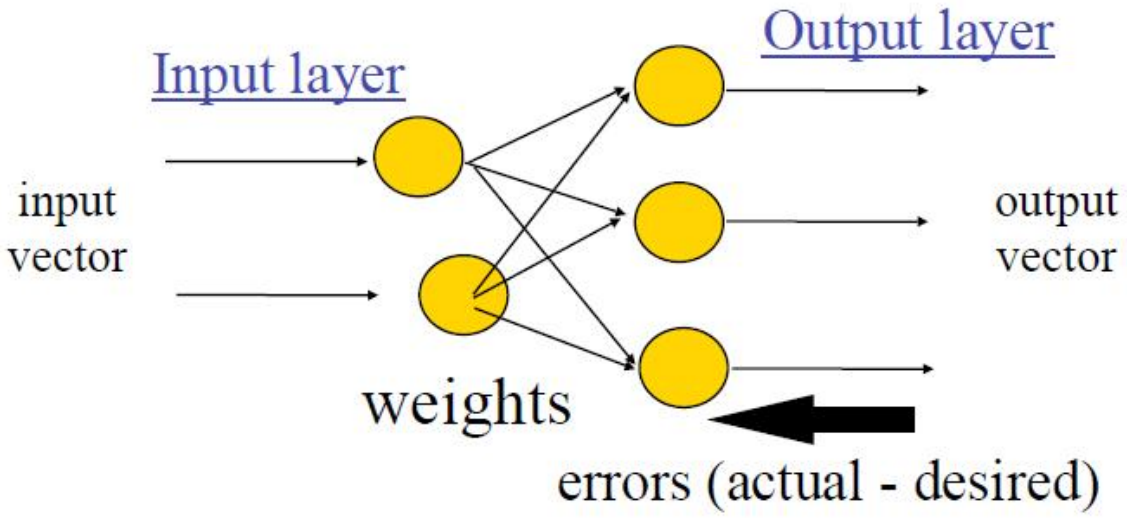
Example of a neuron using threshold activation.

Neural Networks

- The neurons and the connections define the architecture of a NN. The basic structure of a feed forward network, *the most commonly used*, includes an input layer, one or more hidden layers, and an output layer.
  - Called Multi-layer Perceptron (MLP) networks.
- Through the connection weights, the ***output*** of a MLP is a function of its inputs.
- The training of the NN consists of adjusting the weight to match a desired output (supervised learning). Unlike Bayesian learning, MLP give no insight into the classifier but they are most suited for real time work.
  - MLPs are “black box” models.

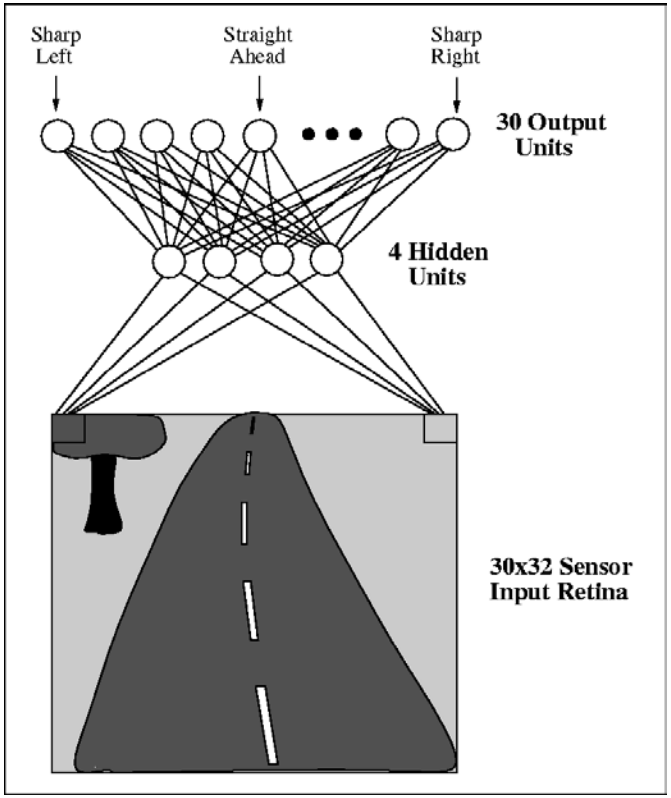
Neural Network Training

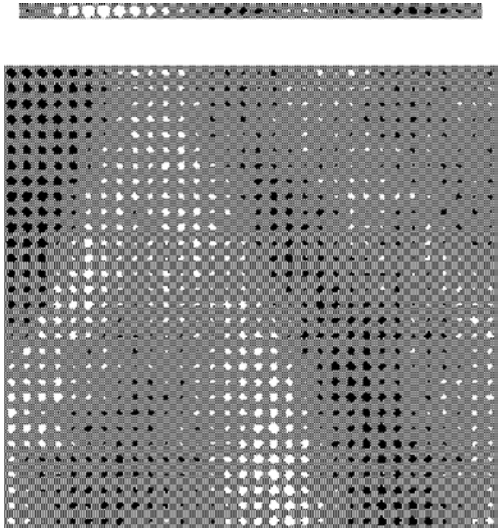
- Present training patterns to net (inputs, outputs).
- Determine errors (Actual – Desired).
- Change the weight of each connection so that the network produces a better approximation of the desired output.



ALVINN

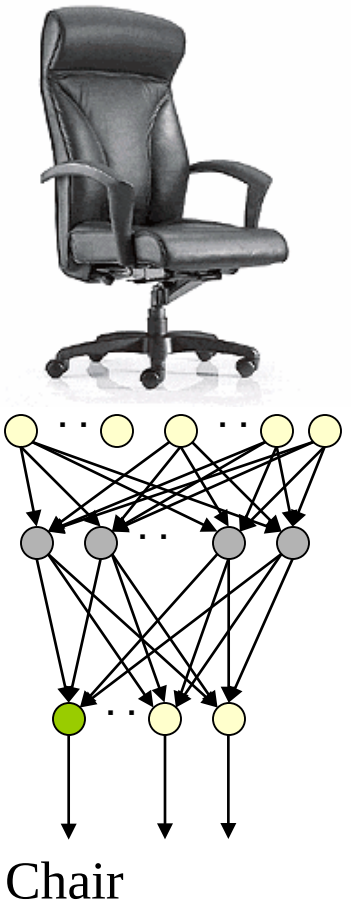
- ALVINN (Autonomous Land Vehicle using an MLP), Dean Pomerleau (1991)
  - Pittsburg to San Diego: 98.2% autonomous.
  - https://www.youtube.com/watch?v=2KMAAmkz9go
- Created a foundation.
- All modern approaches to ALV use NN.





NN for Learning Sensory Patterns

- Learning to Identify Objects
  - How can a particular object be recognized ?
    - Programming recognition strategies is difficult because we do not fully understand how we perform recognition.
    - Learning techniques permit the robot system to form its own recognition strategy.
  - Supervised learning can be used by giving the robot a set of pictures and the corresponding classification.
    - Neural networks
      - Convolutional Deep Learning Letworks
    - Decision trees



Evolutionary Learning

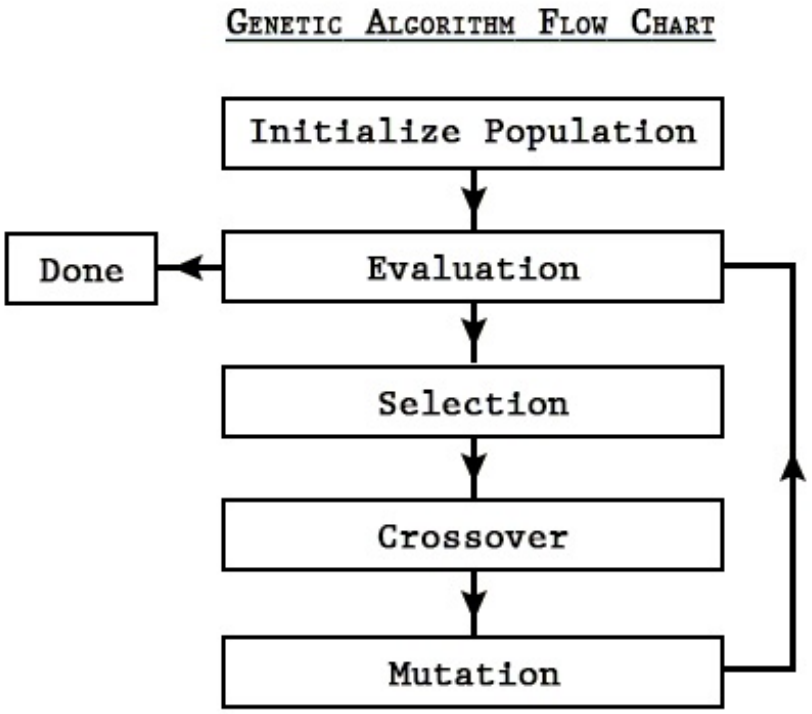
- Inspired from evolutionary biology.
- Individuals in populations have a particular fitness with respect to a task.
- Individuals with the highest fitness are kept as survivors.
- Individuals with poor performance are discarded: the process of natural selection.
- Evolutionary process: search through the space of solutions to find the one with the highest fitness.

Genetic Algorithms

- GAs simulate the survival of the fittest among individuals over consecutive generation for solving a problem.
  - The basic techniques of GAs are designed to simulate processes in natural systems necessary for evolution, especially those that follow the principles first laid down by Charles Darwin of "survival of the fittest".
  - In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones.
- Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA.
- Each individual represents a point in a search space and a possible solution.

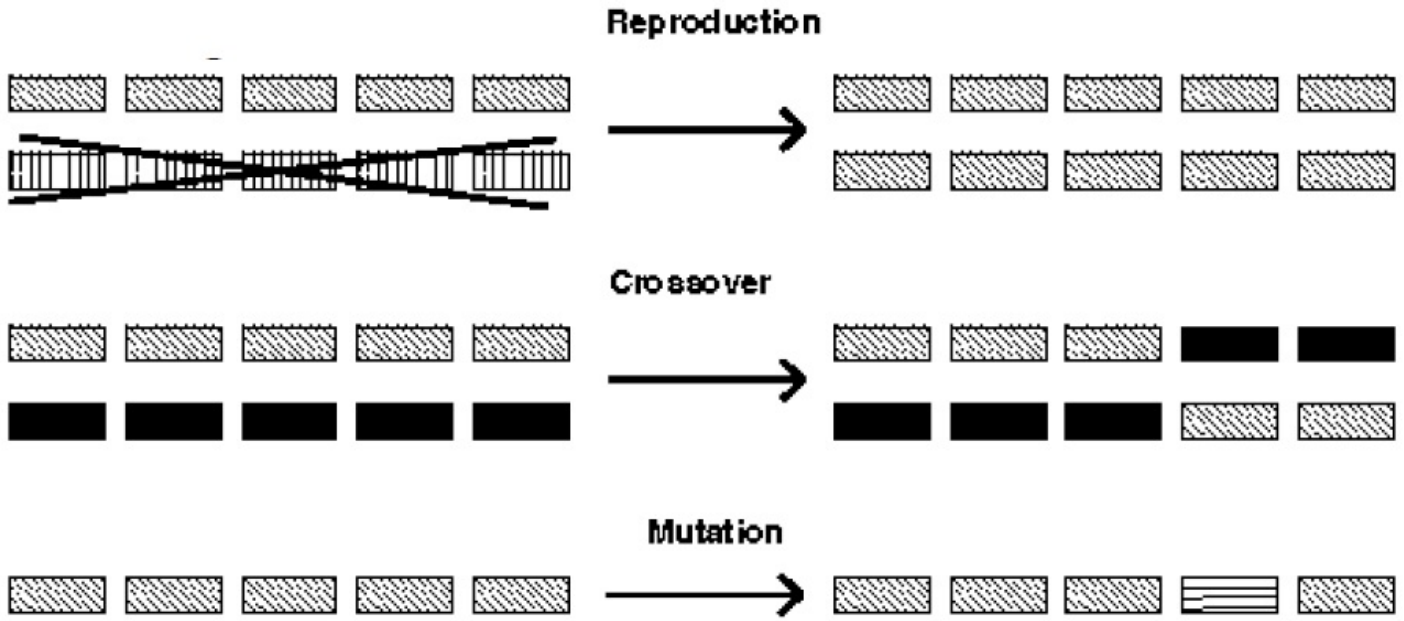
Genetic Algorithms

- The individuals in the population are then made to go through a process of evolution.

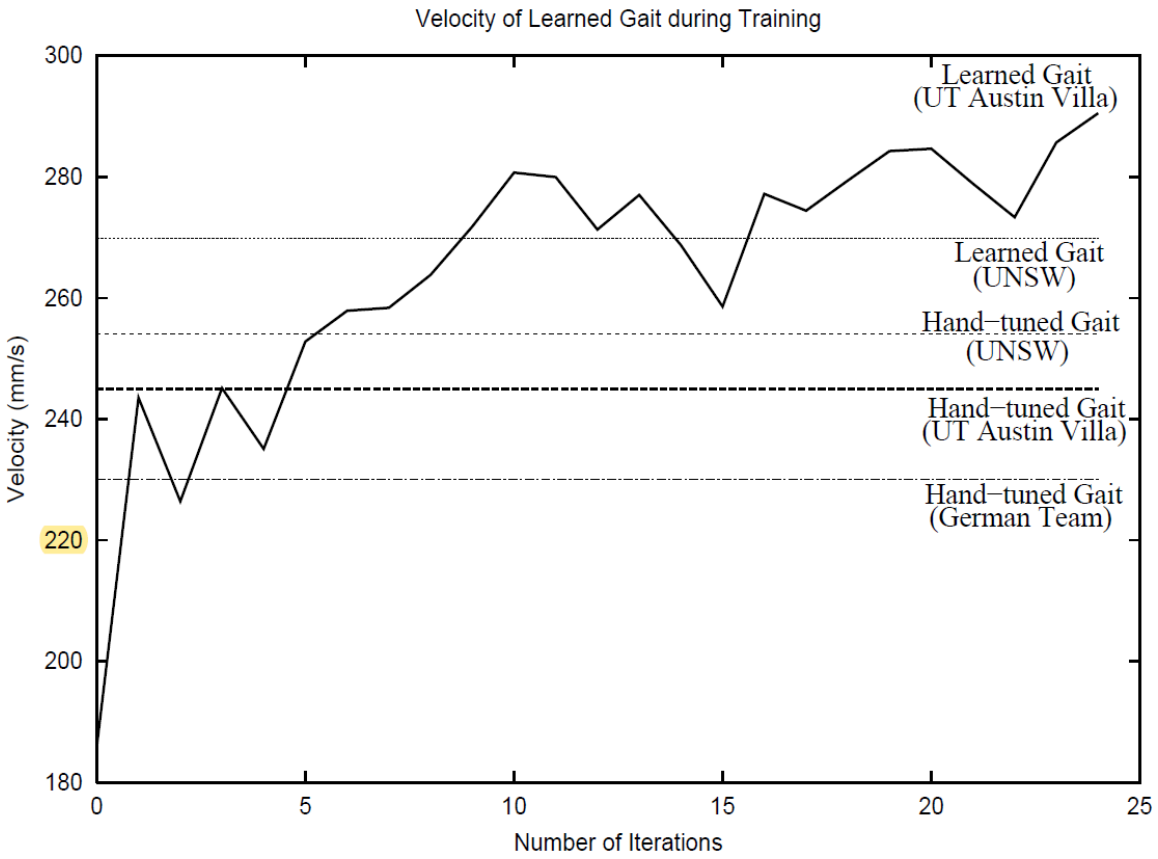


Genetic Algorithms

- Knowledge is encoded as bit strings: chromosome
- Each bit represents a “gene”
- Biologically inspired operators are applied to yield better generations.

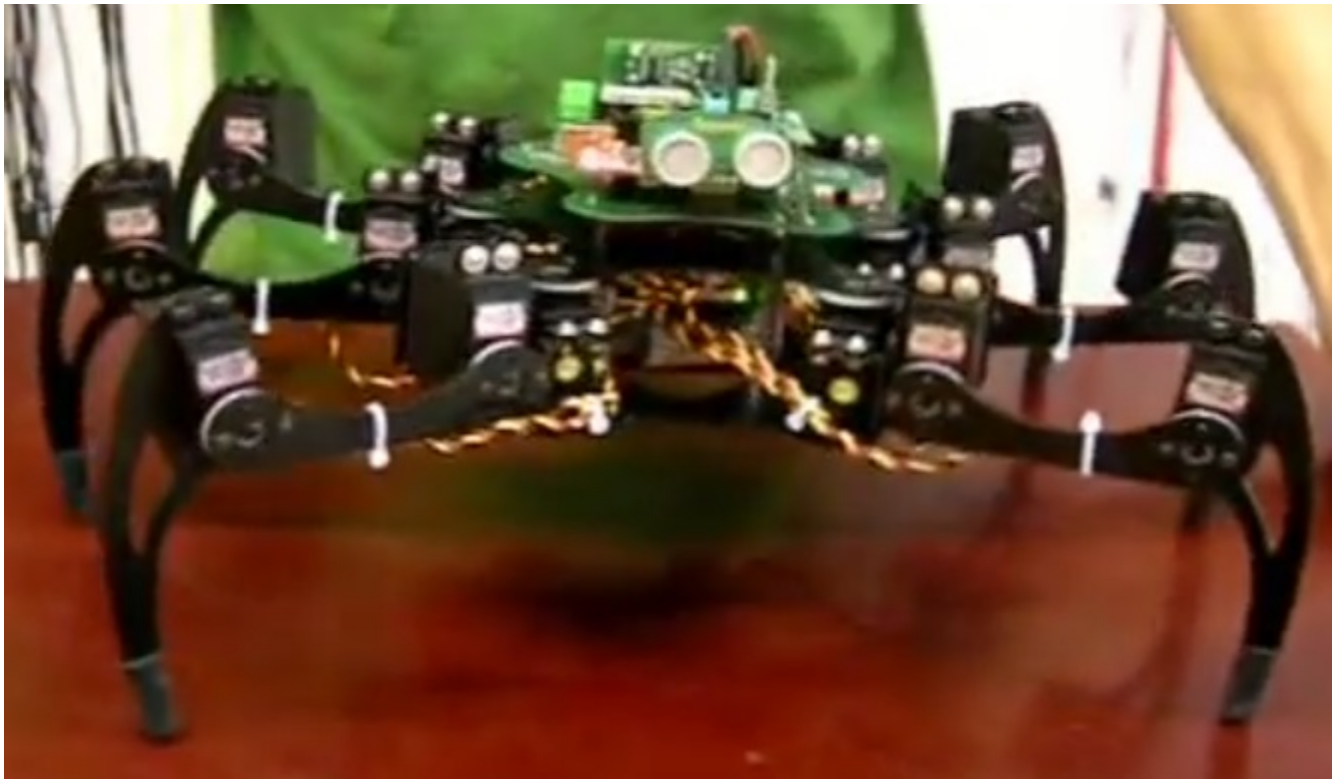


Example: Learning to Walk



Robo Cup (Sony Aibo), Nate Kohl & Peter Stone (2004)

Example: Learning to Walk



Auto adaptable walking robot with genetic algorithm.

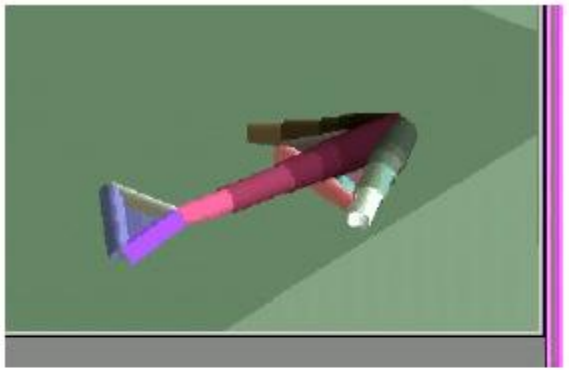
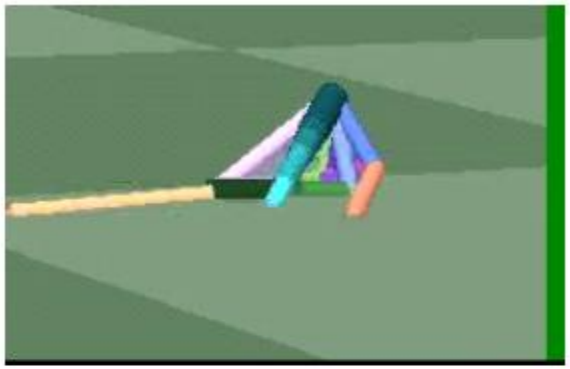
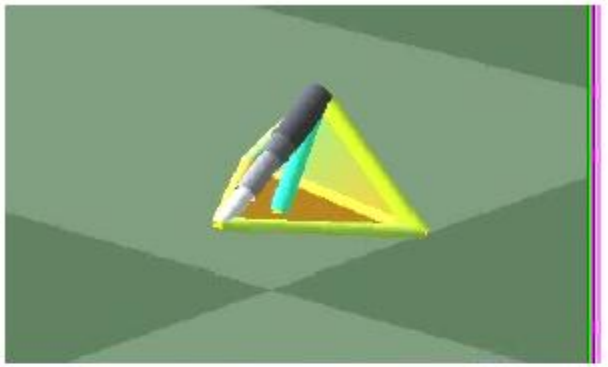
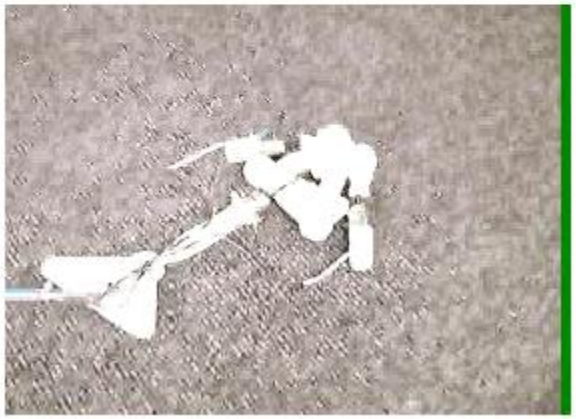
https://www.youtube.com/watch?v=KHV7fWvnn_0

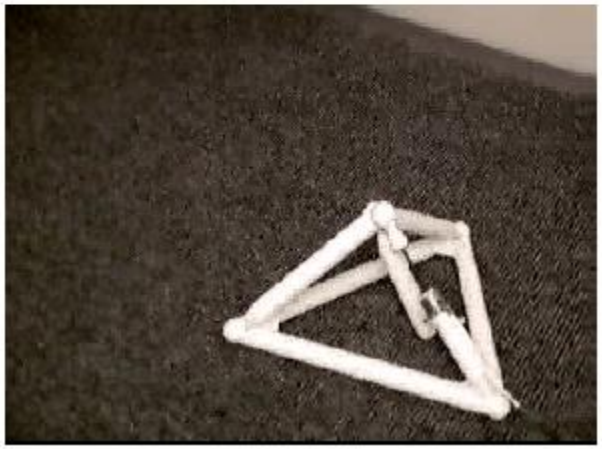
Evolving Structure and Control

- Karl Sims 1994 developed morphology and control for virtual creatures performing swimming, walking, jumping, and following.
- Genotypes encoded as directed graphs are used to produce 3D kinematic structures.
- Genotype encode points of attachment
- Sensors used: contact, joint angle and photosensors

Evolving Structure and Control

- Jordan Pollak - Real structures



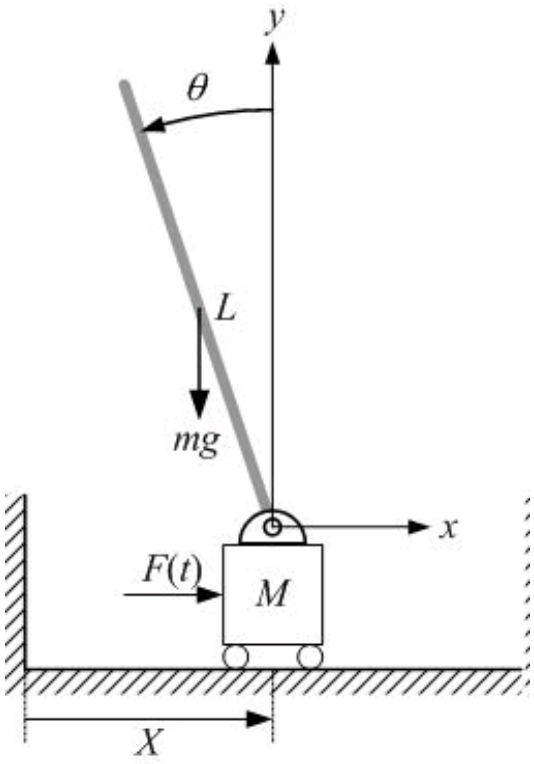
https://www.youtube.com/watch?v=bBt0imn77Zg

Learning from Demonstration

- Requires the user to give the exact solution to the robot in the form of the error direction and magnitude.
- The user must know the exact desired behaviour for each situation.
- Supervised learning involves training, which can be very slow; the user must supervise the system with numerous examples.

Learning from Demonstration

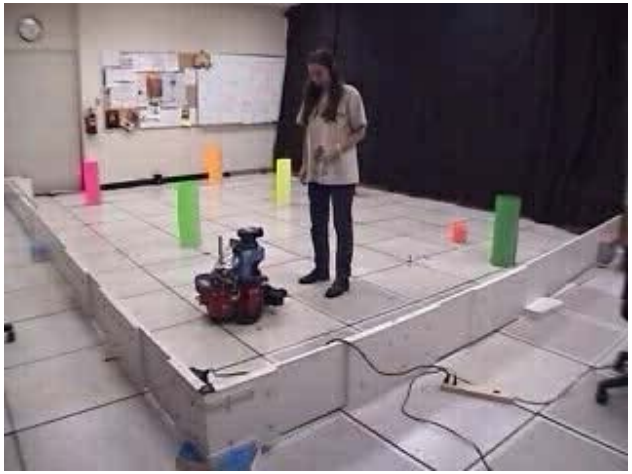
One of the first approaches aimed at solving the pole balancing problem, pendulum-swing-up, via demonstration. [S. Schaal (1997)]



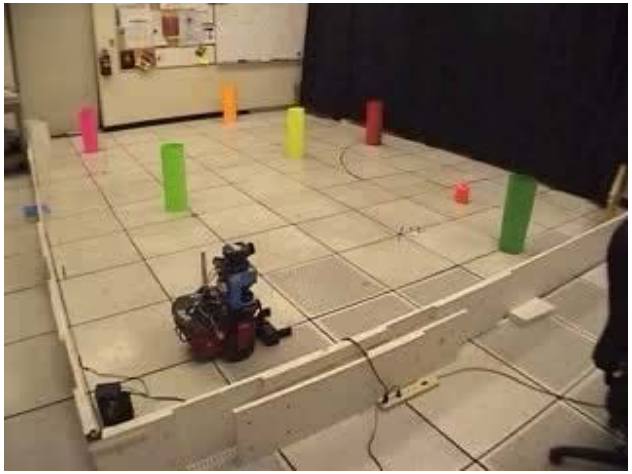
Learning from Demonstration

- Inspiration:
  - Human-like teaching by demonstration.

Human demonstration



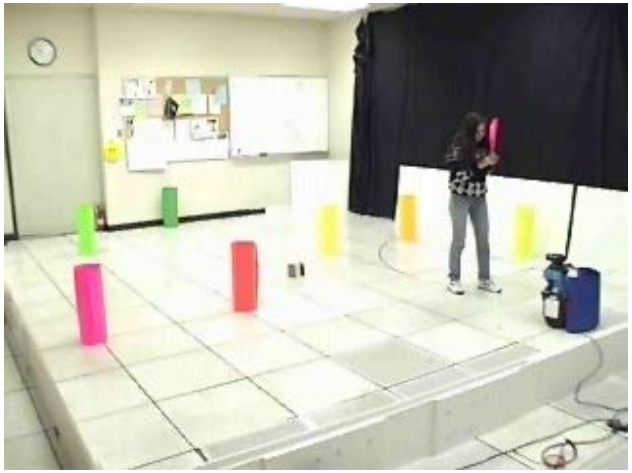
Robot Performance



Learning from Demonstration

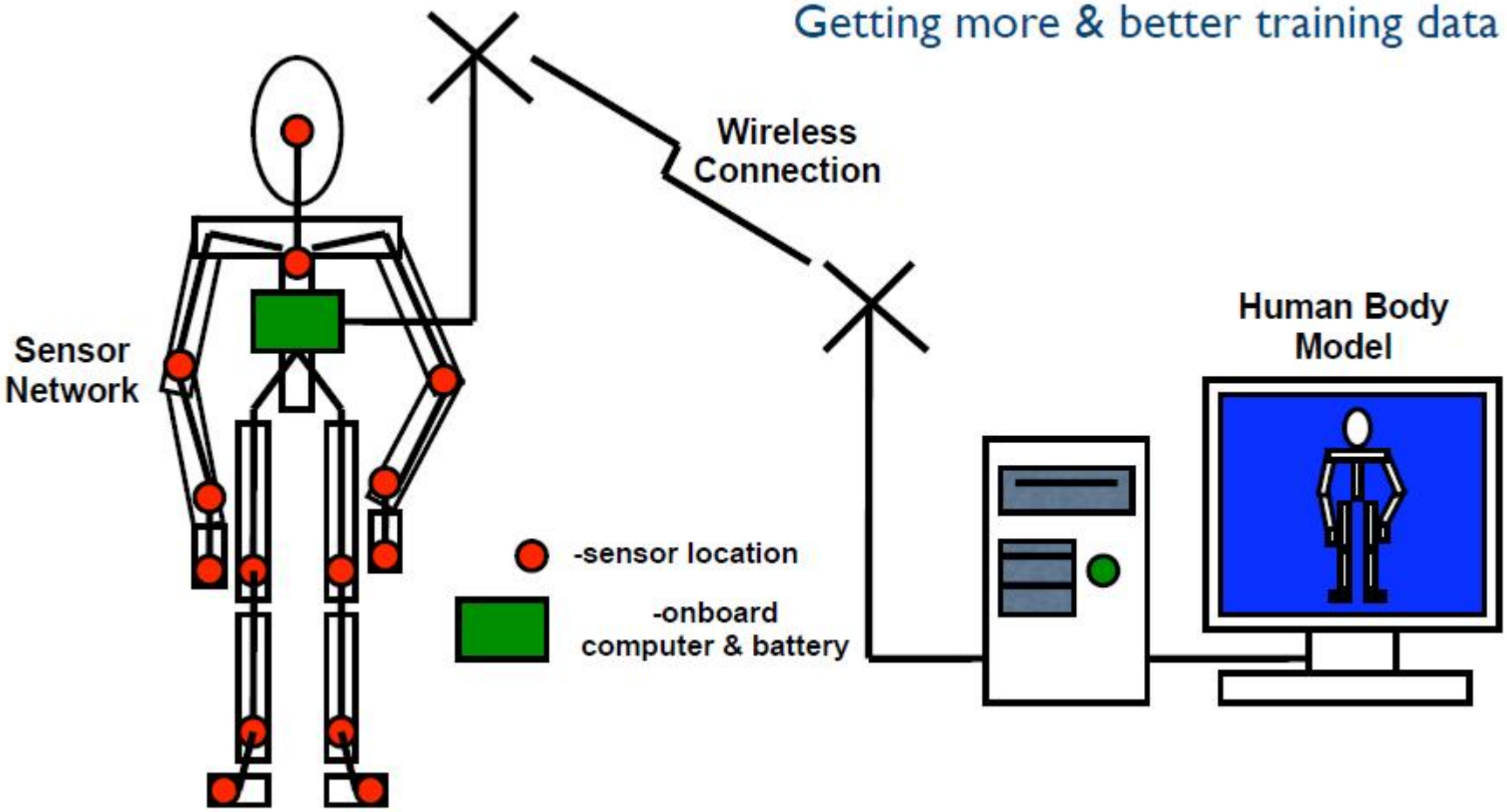
- Transfer of task knowledge from humans to robots.
  - Navigate and perform tasks.
  - Requires to track the teacher and all objects.
  - Very complex.

Human demonstration

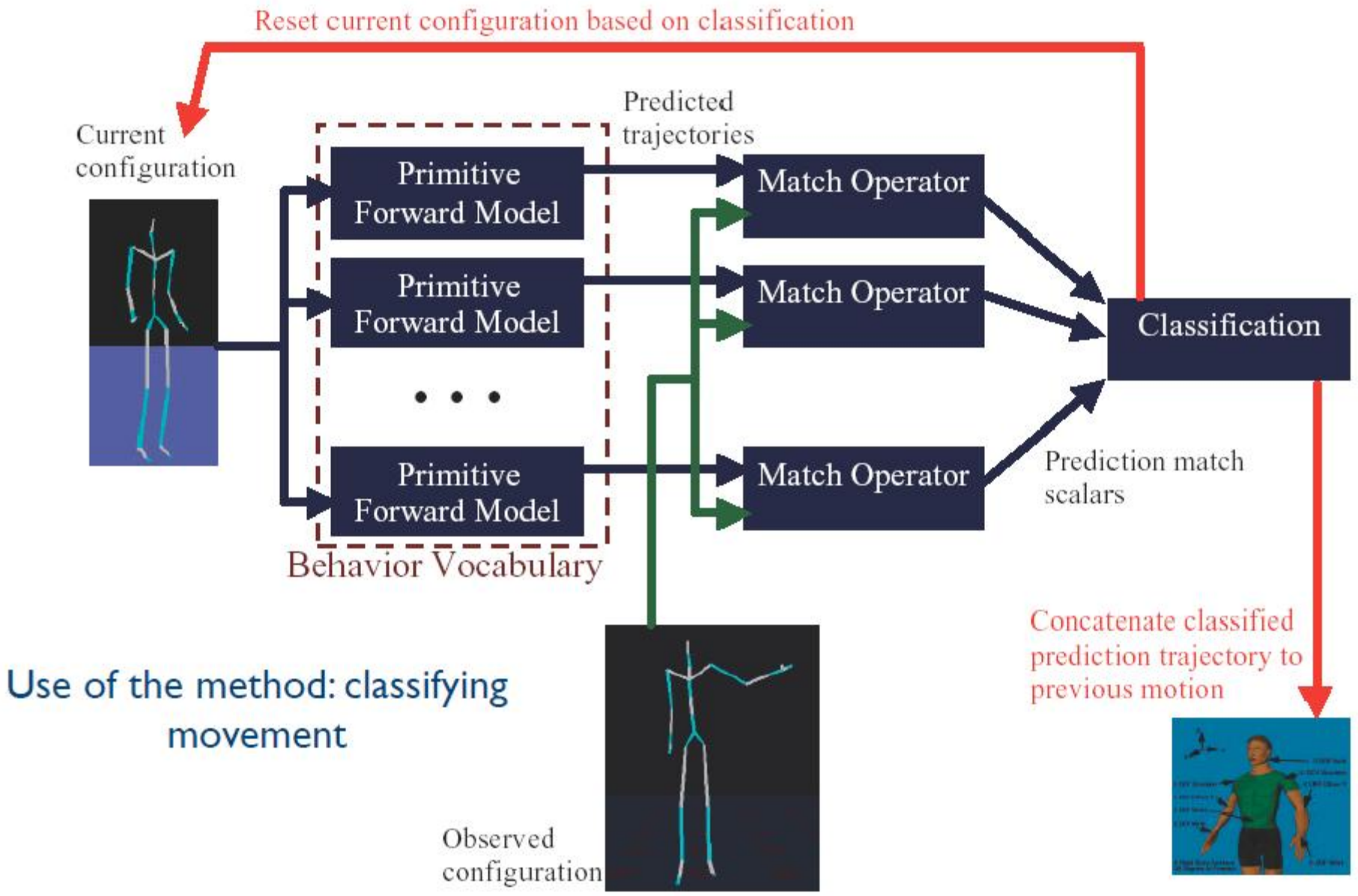


Robot Performance

IMU Motion Capture Suit



Automatically Deriving Behaviours



Use of the method: classifying movement

Inductive Logic Learning

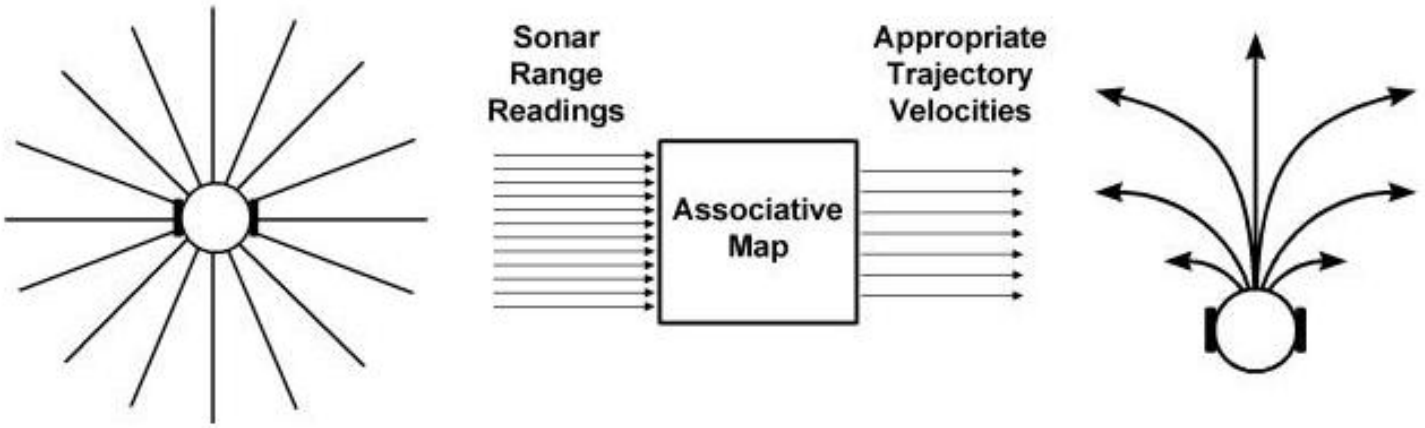
- One of the main supervised symbolic learning schemes is inductive logic programming (ILP).
  - A form of discovery learning.
  - Aims at learning general rules
- Approach is based on logic representations and first-order logic inference, and uses contextual knowledge, i.e. already acquired knowledge.
- The purpose of ILP is to synthesise (or learn) logic programs given a set of variables for which they return true or false values, and so-called background knowledge.
- The programs form an explanation of the observations.

Inductive Logic Learning

- These programs are of course not unique. Therefore the difficulty is to find a minimal (necessary and sufficient) program that is compatible with the data, i. e., that returns true for the positive variables and false for the negative ones.
- This program must also be able to generalise, i. e. to produce the correct answer, when applied to new instances.
- In general, **ILP is not tractable**.

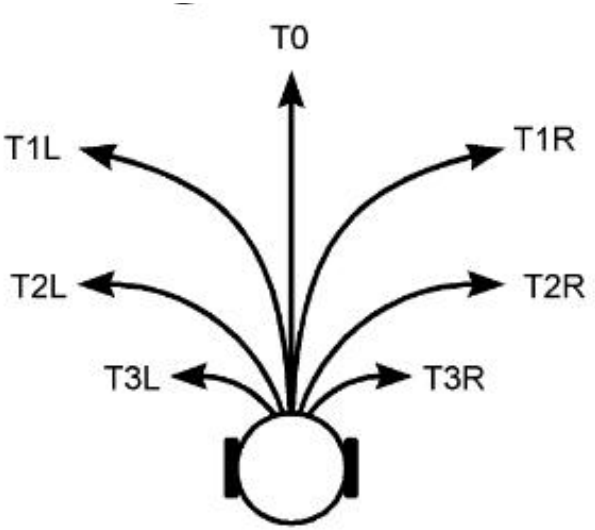
Trajectory Velocity Learning

- Trajectory Velocity Learning involves learning a map between the robot's input sensors and appropriate velocities for negotiating the robot's predefined output trajectory commands.



Trajectory Velocity Learning

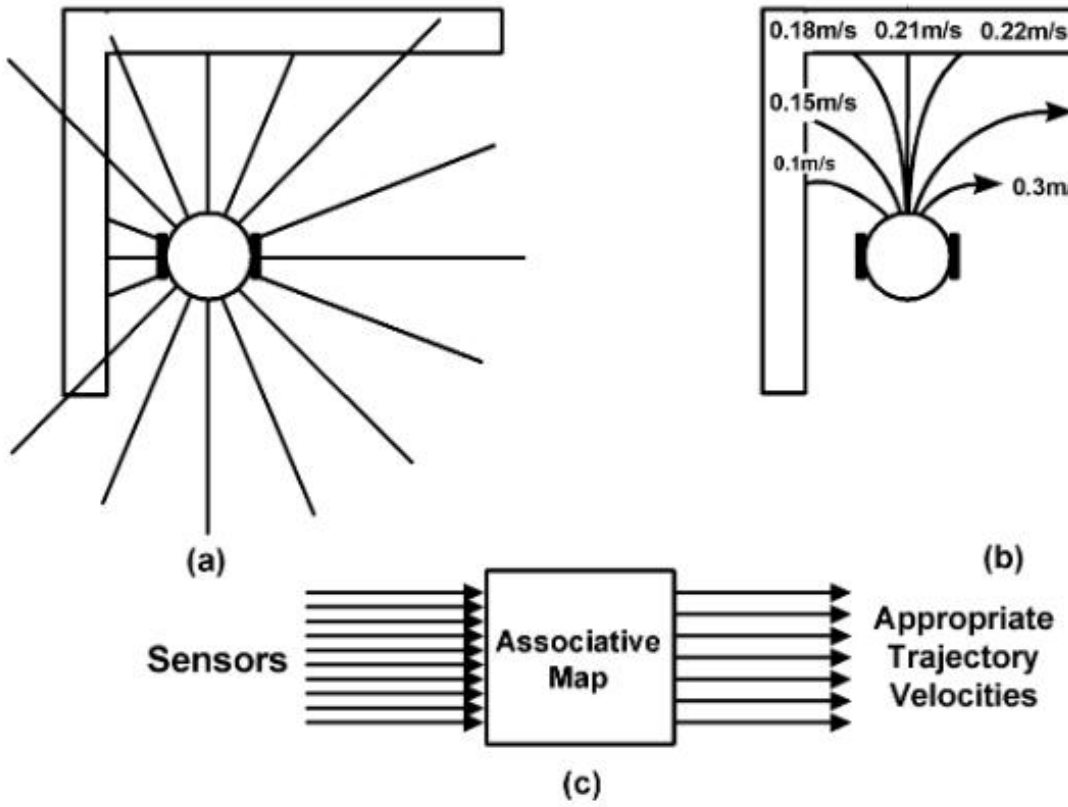
- Each output trajectory command has a predefined radius and maximum velocity for moving in free space.



Trajectory	Radius m	Max Velocity m/s
T0	N/A	0.6
T1L, T1R	2.0	0.5
T2L, T2R	1.0	0.4
T3L, T3R	0.3	0.3

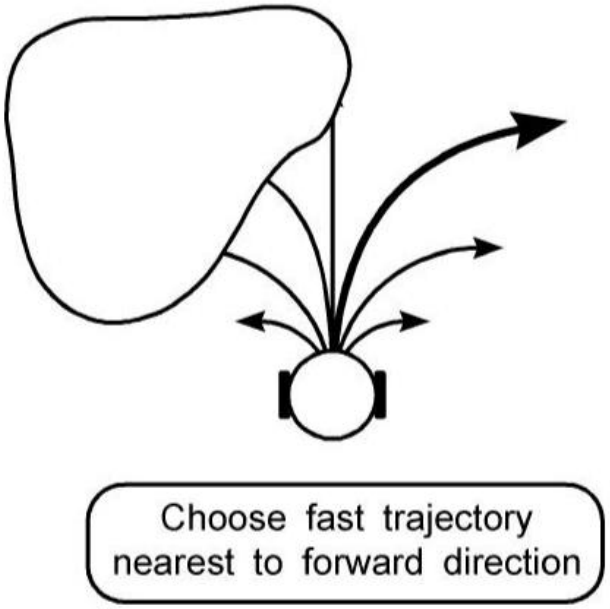
Trajectory Velocity Learning

- Perceiving Appropriate Trajectory Velocities from Sensor Data.



Trajectory Velocity Learning

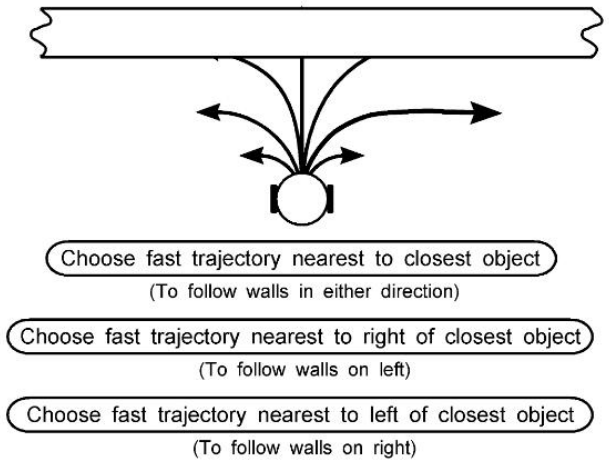
- By being able to perceive appropriate trajectory velocities directly from sensor data, obstacle avoidance behaviour can be performed by giving the robot a single instruction to follow fast 1 trajectories nearest to the forward direction.



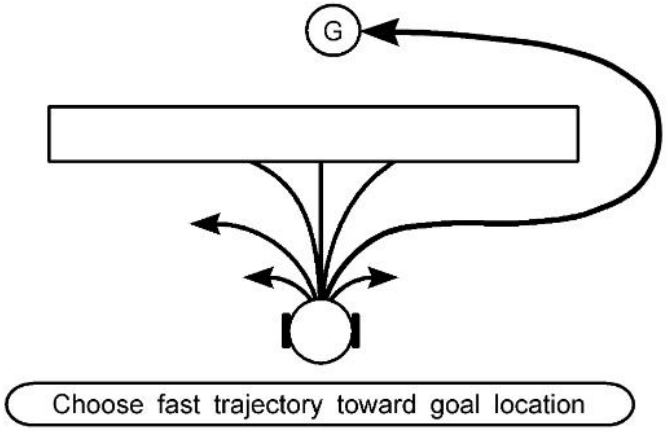
Object avoidance behaviour

Trajectory Velocity Learning

- Similarly, wall following or goal seeking behaviour can also be produced by providing the robot with a single instruction to follow fast trajectories nearest to the closest object or the goal location.



Wall following behaviour



Goal seeking behaviour

Automatically Deriving Behaviours

- Learn a vocabular of motion modules (behaviours).
- Learn context of behaviours from examples.
- Method:
  - Input: kinematic motion; time series of joint angles
  - Motion segmentation
    - Partition input motion into conceptually indivisible motion segments.
  - Grouping of behaviour exemplars
    - Spatio-temporal Isomap dimension reduction and clustering.
  - Generalising behaviours into forward models
    - Interpolation of a dense sampling for each behavior.
  - Meta-level exemplar grouping
    - Additional embedding iterations for higher level behaviours.

O. C. Jenkins, M. J Matarić, **"Automated Derivation of Behaviour Vocabularies for Autonomous Humanoid Motion"**, *Autonomous Agents and Multiagent Systems,* Melbourne, Australia, July 14-16, 2003.

Learning

- When to learn and when to program?
- Programming:
  - When a well defined behaviour is required.
  - Maintain full control over behaviour.
  - No time or no computational resources available for learning.
- Learning:
  - When too difficult, too complex, or too expensive to program.
  - Required adaptability to new unforeseeable situations.
  - Increased robustness and ability to evolve w/o manual intervention is required.

Further Reading

- For a more complete coverage of the methods described in these sildes you can refer to AI textbooks, e.g. Russell and Norvig **Artificial Intelligence: A Modern Approach (3rd Edition)**