# CSCE274 Robotic Applications and Design
# Fall 2020
# Learning for robots

Ioannis Rekleitis
Computer Science and Engineering
University of South Carolina
yiannisr@cse.sc.edu

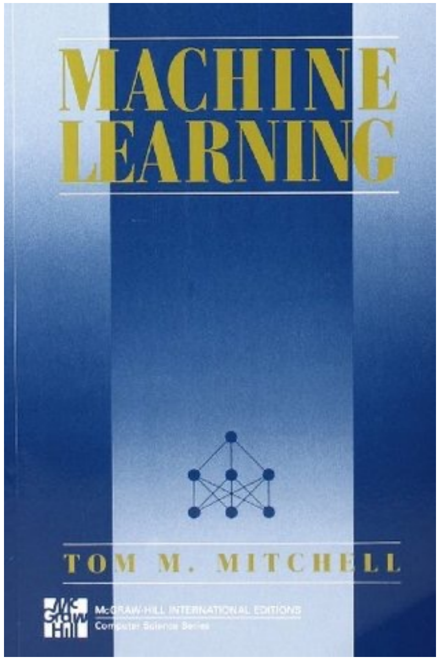
# From MultiRobot Systems

# From Navigation

# Machine learning

- *Machine Learning* is a field that studies computer algorithms that improve automatically through experience

- Such techniques can be used in a robot so that it can learn about itself




Source: cs.cmu.edu/~tom/mlbook.html

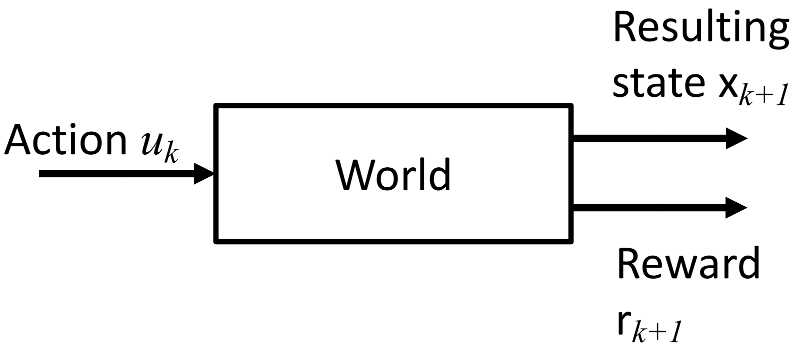
# Machine learning techniques

- Unsupervised learning: no external supervisor tells the robot what to do

- Supervised learning: external supervisor who provides a training dataset

# Reinforcement learning

- *Reinforcement learning* is an example of unsupervised learning
- A robot interacts with the world, performing an action, which causes a change in the world and accordingly the robot receives a reward



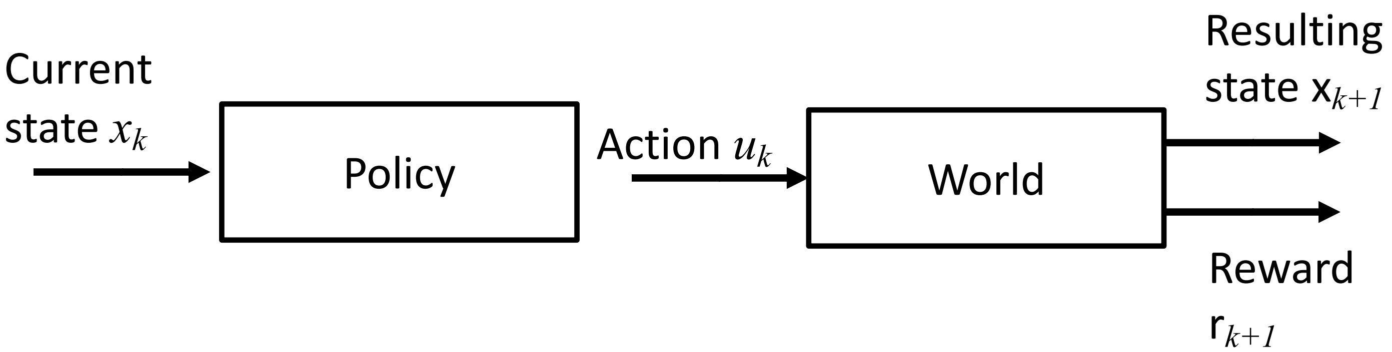
# Reinforcement learning

- Model similar to planning formulation
  - States $X$
  - Actions $U$
  - Transitions: probability distribution $P(x_{k+1} \mid x_k, u_k)$
  - Rewards $P(r_{k+1} \mid x_k, u_k)$

- Note that
  - Reinforcement learning typically considers an *infinite horizon* instead of a specific goal region
  - Transitions are probabilistic because they are *unpredictable*
  - Rewards are the dual of costs

# Reinforcement learning

- A Reinforcement learning algorithm provides a policy, namely a function that maps from states to actions

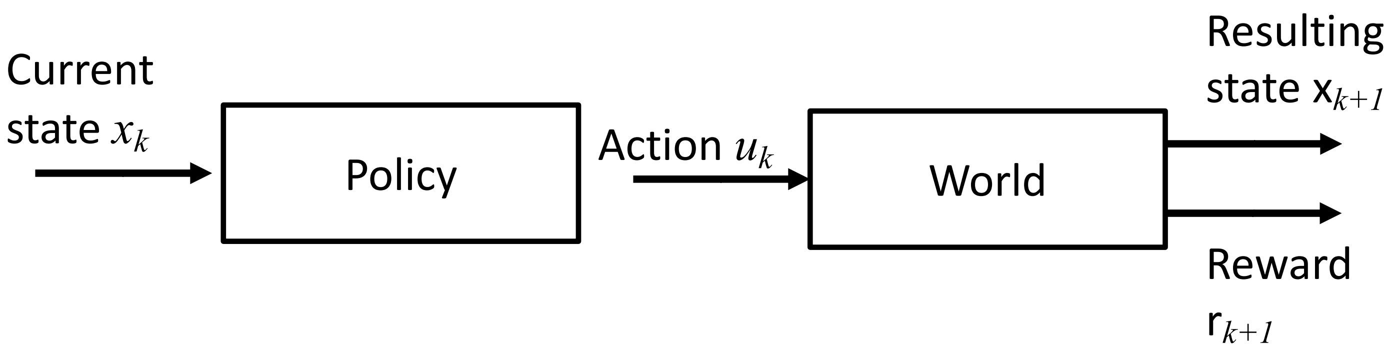


# Reinforcement learning

- The policy is chosen in such a way that the robot's discounted future reward is maximized

$$R = \sum_{i=k}^{\inf} \gamma^{k-i} r_i$$

where
$\gamma \in (0,1)$ is the discount factor that weighs immediate rewards compared to future rewards



# $Q$-learning

- $Q$-learning is a reinforcement learning technique that can be used to find an optimal policy
- The idea is to use a "quality function" that maps state-action pairs to discounted future rewards assuming
  - The robot at a current state $x_k$
  - It selects $u_k$
  - It chooses optimal actions in all of the future stages

# $Q$-learning

- The $Q$-learning algorithm learns $Q$ values as the robot selects new actions and observes the world
- The algorithm is

1. Initialize the $Q$ table
2. Execute an action $u_k$
3. Observe the new state $x_{k+1}$ and receive the reward $r_k$
4. Update the $Q$ table

   $$Q(x_k, u_k) \leftarrow (1-\alpha)Q(x_k, u_k) + \alpha(r_k + \gamma[\max_{u \in U} Q(x_{k+1}, u)])$$

   where $\alpha$ is a learning rate parameter which controls how rapidly $Q$ is changed
5. Repeat from Step 2.

# Reinforcement learning

- Example of $Q$-table for a robot

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 8 |

# Reinforcement learning

- Assume
  - Robot in state $b$ and executes $z$
  - The new state is $a$ and reward received is $11$
  - $\alpha=0.1$, $\gamma = 0.9$
- What is the new $Q$-table?

# Reinforcement learning

- Update $Q$-table by applying the formula

$$Q(x_k, u_k) \leftarrow (1-\alpha)Q(x_k, u_k) + \alpha(r_k + \gamma[\max_{u \in U} Q(x_{k+1}, u)])$$

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 18 |

# Reinforcement learning

– Update $Q$-table by applying the formula

- Robot in state *b* and executes *z*
- The new state is *a* and reward received is *11*
- *α=0.1, γ = 0.9*

$$Q(x_k, u_k) \leftarrow (1-\alpha)Q(x_k, u_k) + \alpha(r_k + \gamma[\max_{u \in U} Q(x_{k+1}, u)])$$

*Q(b,z) <-(1-0.1) Q(b,z) +0.1(11+0.9(max Q(a,u)))*

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 8 |

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 18 |

# Reinforcement learning

– Update $Q$-table by applying the formula

- Robot in state *b* and executes *z*
- The new state is *a* and reward received is *11*
- *α=0.1, γ = 0.9*

$$Q(x_k, u_k) \leftarrow (1-\alpha)Q(x_k, u_k) + \alpha(r_k + \gamma[\max_{u \in U} Q(x_{k+1}, u)])$$

*Q(b,z) <-(0.9) 8 +0.1(11+0.9(100))*

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 8 |

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 18 |

# Reinforcement learning

– Update $Q$-table by applying the formula

- Robot in state *b* and executes *z*
- The new state is *a* and reward received is *11*
- *α=0.1, γ = 0.9*

$$Q(x_k, u_k) \leftarrow (1-\alpha)Q(x_k, u_k) + \alpha(r_k + \gamma[\max_{u \in U} Q(x_{k+1}, u)])$$

*Q(b,z) <- (0.9) 8 +0.1(101)=7.2+10.1=17.3 round up to 18*

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 8 |

| State | Action | $Q$ |
|---|---|---|
| $a$ | $x$ | 90 |
| $a$ | $y$ | 100 |
| $a$ | $z$ | 80 |
| $b$ | $x$ | 7 |
| $b$ | $y$ | 10 |
| $b$ | $z$ | 18 |

# Exploration vs. exploitation

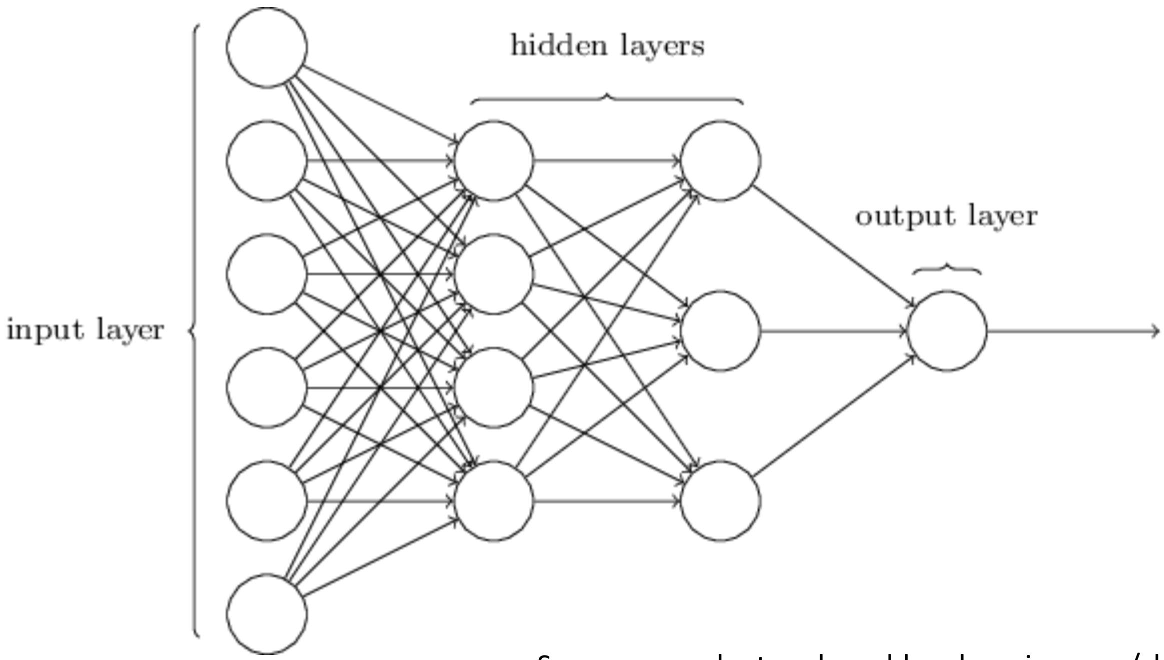
- What actions should the robot choose and execute?
  - Exploration: choose an action that the robot doesn't know about yet
  - Exploitation: choose an action with the largest Q value from the current state

$$\pi(x) = \arg\max_{u \in U} Q(x, u)$$

- Trade-off between exploration and exploitation

# Neural network

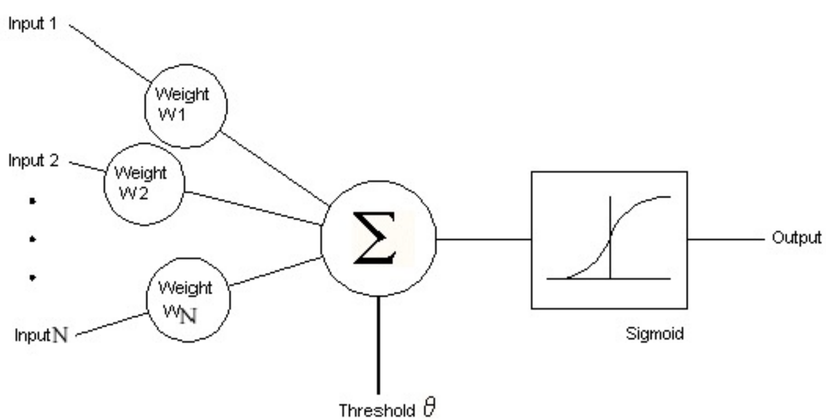
- Neural network learning is an example of supervised machine learning




Source: neuralnetworksanddeeplearning.com/chap1.html

# Neural network

- Neural network components
  - Perceptron: main processing unit that includes a summation function that takes weighted inputs and an activation function (e.g., sigmoid)

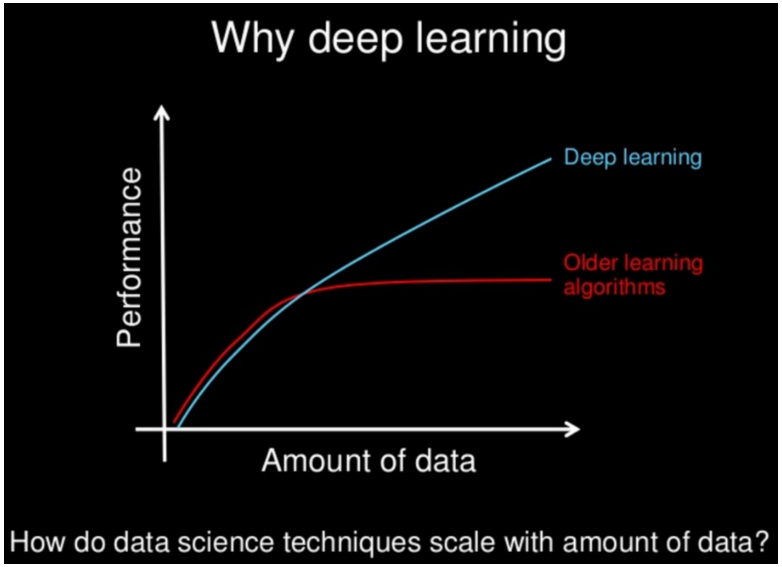



Source: homepages.gold.ac.uk/nikolaev/311perc.htm

  - Training with labeled data to compute *error* and adjust weights

# Deep learning

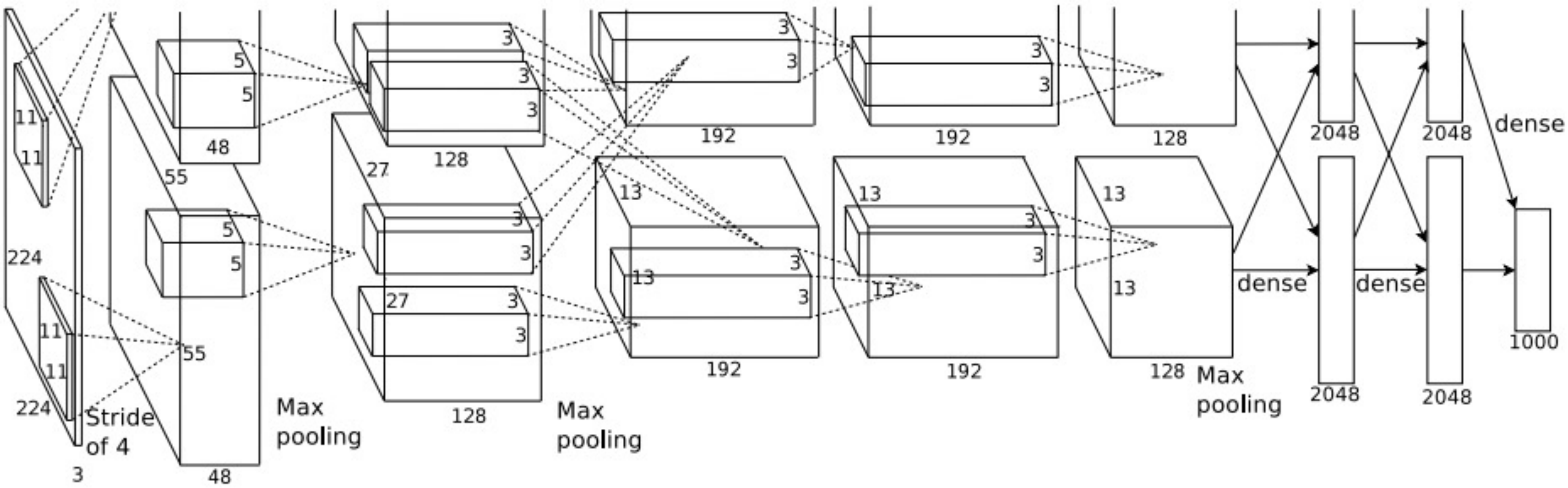
- Deep neural network is becoming an important technology also in robotics
- Driven by increase in computational power (e.g., GPU)




Source: http://www.slideshare.net/ExtractConf (Andrew Ng)

# Deep learning

- Deep neural network
  - E.g., AlexNet




Source: [Krizhevsky et al., 2012, NIPS]

# Learning by demonstration

- Learning by demonstration is to train the robot by showing how to do tasks

# Long-term learning

- Forgetting is also an important part to discard outdated previously learned information
  - Necessary because finite memory space and old information might not be correct
- Long-term learning however is one of the aims