Real-world Multi-agent Systems

Revision lecture

Akin Delibasi
Lecturer in Distributed Systems
Computer Science
University College London, UK

Overview of lectures

1. Introduction of multi-agent systems
2. Sensing and Perception
3. Control of agents
4. Motion planning
5. Advanced optimization
6. Multi-agent planning
7. Machine learning and AI
8. Multi-agent learning
9. Guest lecture, real-world case studies
10. Revision lecture & QA

Lecture 1: Introduction of multi-agent systems

- Definition of multi-agent systems
- General concept of MAS in the context of IoT
- Core concepts of multi-agent systems
  - Concepts of (Intelligent) Agents
  - Characteristics of an Agent
  - Types of Agents (mind vs physical body)
  - Agent-oriented solution (vs object-oriented programming)
- Reasoning and decision-making
  - Rule based explicit reasoning
  - Common sense reasoning (LLM based agents; see similar examples/content in Lecture 9: real-world case studies)

Lecture 2: sensing and perception

- Part 1: sensing and signal processing
  - ❑ Fundamental of signal processing: Sampling, ADC and DAC (resolution), frequency domain (understand signals are composed of different frequency components)
  - ❑ Core elements in digital signal processing: Sampling & quantization
  - ❑ Exercise the code example of digital filters
  - ❑ Understand key parameters for processing of noisy sensory measurements
  - ❑ Understand the difference between offline and online signal processing
- Part 2: Perception. understand what tools are out there:
  - Object recognition (eg, Yolo library)
  - Segmentation (eg, Meta's SAM segment-anything.com)
  - Pose estimation (eg, ArUco markers, being able to explain how ArUco markers are used in lab practical)
  - Text recognition (OCR)

Lecture 3: Control of agents

- Control of agents
  - ❑ Concept feedback control, why closed loop? Corrective actions, following expected targets
  - ❑ Understanding PID control, meaning of each gain, what does P, D, I means (same physical interpretation in both continuous and discrete forms).
  - ❑ Understand PD controller in an analogy to a spring-damper system, the resonance frequency of this virtual spring-damper determines the bandwidth of the PD controller.
  - ❑ How to write pseudo code of discrete PID controller.
- Numerical simulation of physical systems
  - ❑ Understand numerical integration (a → v → s)
  - ❑ Combining simulation + PID control, see matlab code example, ***PID_Solution.m***.
- Able to debug or complete the code, see ***PID_Quiz.m*** (download from Moodle)

Lecture 4: Motion Planning of Agents

- Motion Planning
  - **Definition**: Motion Planning is a computational problem to find ***a sequence of valid configurations*** to move an object/agent from a *start* to a *goal* position.
- Continuous planning: Potential Fields (PF) method:
  - Attractive and Repulsive forces
  - Potential field matlab code: see how to add attractive and repulsive forces together; ability to understand and correct pseudo code.
  - Limitations of PF method: local minima
- Sampling-based planning (pseudo code):
  - Rapidly exploring Random Tree (RRT): principle, pros and cons.
  - Probabilistic Roadmap (PRM) (Apply A* on PRM)
  - Pros and cons: RRT vs PRM

Lecture 5: Optimization

Least square

$$\underbrace{\begin{bmatrix} 1, x_1^{-1}, x_1^{-2}, x_1^{-3}, x_1^{-4} \\ 1, x_2^{-1}, x_2^{-2}, x_2^{-3}, x_2^{-4} \\ \vdots \\ 1, x_k^{-1}, x_k^{-2}, x_k^{-3}, x_k^{-4} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}}_{b}$$

Coefficients are the unknowns here

Map our formulation to the LS equation. Solution of pseudo inverse

$$x = A^{\dagger} b$$

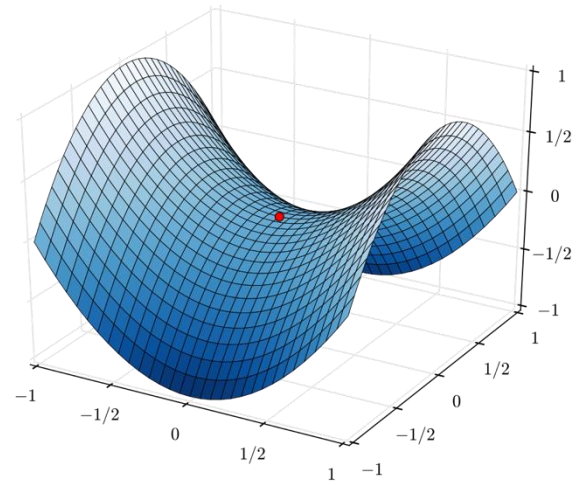
Lecture 5: Optimization

- Optimization: gradient ascent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\alpha} \cdot \bigtriangledown \boldsymbol{f}(\boldsymbol{x}_k)$$

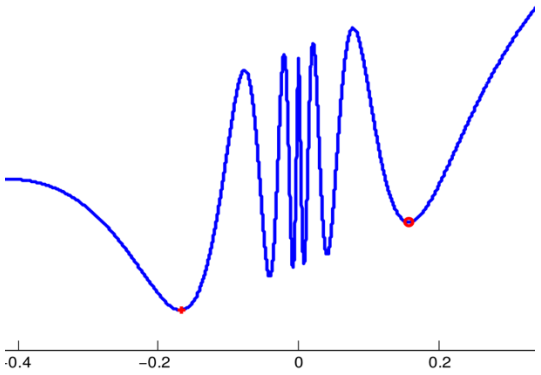
- Gradient descent

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{\alpha} \cdot \bigtriangledown \boldsymbol{f}(\boldsymbol{x}_k)$$



- **Saddle point**:
- gradient = 0 does not necessarily mean an extrema.

- Local minima problem



Lecture 5: Cost function *f*(*x*)

What is a cost function f(x)?

Design of cost function f(x): minimization of error terms (targeted quantities to minimize)

$$\boldsymbol{f(q) = ||F(q) - y^d||^2_{Q_P} + ||O(q) - vec^d||^2_{Q_r} + ||q - q_0||^2_{Q_j}}$$

$$\boldsymbol{q_{min} \leq q \leq q_{max}}$$

q: joint configuration → decision variables

$\mathbf{q_0}$, $\mathbf{q_{min}}$, $\mathbf{q_{max}}$: initial, minimum, maximum joint configurations

F(q), O(q): forward kinematics, returns end effector position and orientation

$\mathbf{y^d}$, $\mathbf{vec^d}$: desired task space or workspace position, orientation

$\mathbf{Q_p}$, $\mathbf{Q_r}$, $\mathbf{Q_j}$ are the weighting matrices for workspace position, rotation, and joint/configuration space respectively.

Lecture 6: Multi-agent path planning

- What is MAPP?
- Learn the algorithms for MAPP
  - ❑ Priority approach
  - ❑ Conflict Based Approach
- Drawbacks of CBS
- Meta Agent CBS

Artificial Intelligence 219 (2015) 40–66

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Conflict-based search for optimal multi-agent pathfinding

Guni Sharon [a,*], Roni Stern [a], Ariel Felner [a], Nathan R. Sturtevant [b]

[a] Information Systems Engineering, Ben Gurion University, Be'er Sheba, 85104, Israel
[b] Department of Computer Science, University of Denver, Denver, CO, USA

ARTICLE INFO

Article history:
Received 16 September 2013
Received in revised form 23 November 2014
Accepted 25 November 2014
Available online 2 December 2014

Keywords:
Heuristic search
Multi-agent
Pathfinding

ABSTRACT

In the *multi-agent pathfinding* problem (MAPF) we are given a set of agents each with respective start and goal positions. The task is to find paths for all agents while avoiding collisions. Most previous work on solving this problem optimally has treated the individual agents as a single 'joint agent' and then applied single-agent search variants of the A* algorithm.

In this paper we present the Conflict Based Search (CBS) a new optimal multi-agent pathfinding algorithm. CBS is a two-level algorithm that does not convert the problem into the single 'joint agent' model. At the high level, a search is performed on a *Conflict Tree* (CT) which is a tree based on conflicts between individual agents. Each node in the CT represents a set of constraints on the motion of the agents. At the low level, fast single-agent searches are performed to satisfy the constraints imposed by the high level CT node. In many cases this two-level formulation enables CBS to examine fewer states than A* while still maintaining optimality. We analyze CBS and show its benefits and drawbacks.

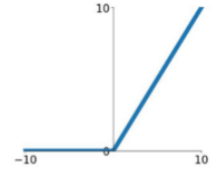
Additionally we present the *Meta-Agent CBS* (MA-CBS) algorithm. MA-CBS is a generalization of CBS. Unlike basic CBS, MA-CBS is not restricted to single-agent searches at the low level. Instead, MA-CBS allows agents to be merged into small groups of joint agents. This mitigates some of the drawbacks of basic CBS and further improves performance. In fact, MA-CBS is a framework that can be built on top of any optimal and complete MAPF solver in order to enhance its performance. Experimental results on various problems show a speedup of up to an order of magnitude over previous approaches.

Lecture 7: Machine learning and AI

- Turing Test
- Machine learning techniques:
  - ❑ Supervised Learning
  - ❑ Unsupervised Learning
  - ❑ Reinforcement Learning
- Fundamentals of neural networks:
  - ❑ Activation functions: where is suitable to put ReLu, Sigmoid, Tanh
  - ❑ Concept of Backpropagation
  - ❑ Reward design: minimization problem as in optimization → maximization

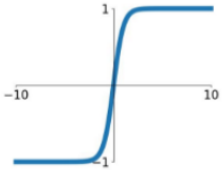
ReLU

$\max(0, x)$



tanh

$\tanh(x)$



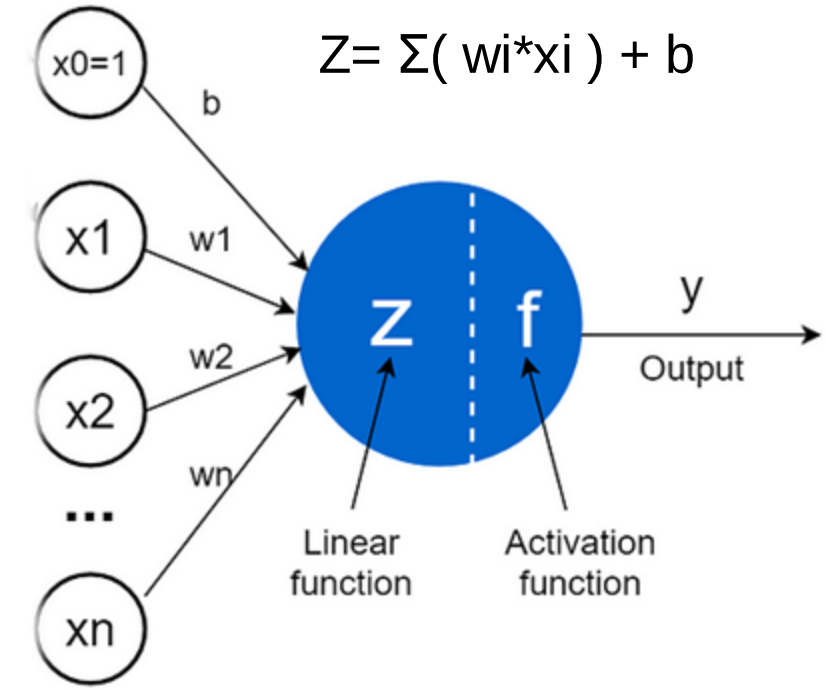
Updating weights/biases by backpropagation

- Gradients are computed using backpropagation, the weights in the neural network can be updated using **gradient descent** (refer the lecture: advanced optimization): adjust the weights in the direction that reduces the error.
- Rule to update the weight by gradient descent:

$$w = w - \alpha \cdot \frac{\partial E}{\partial w}$$

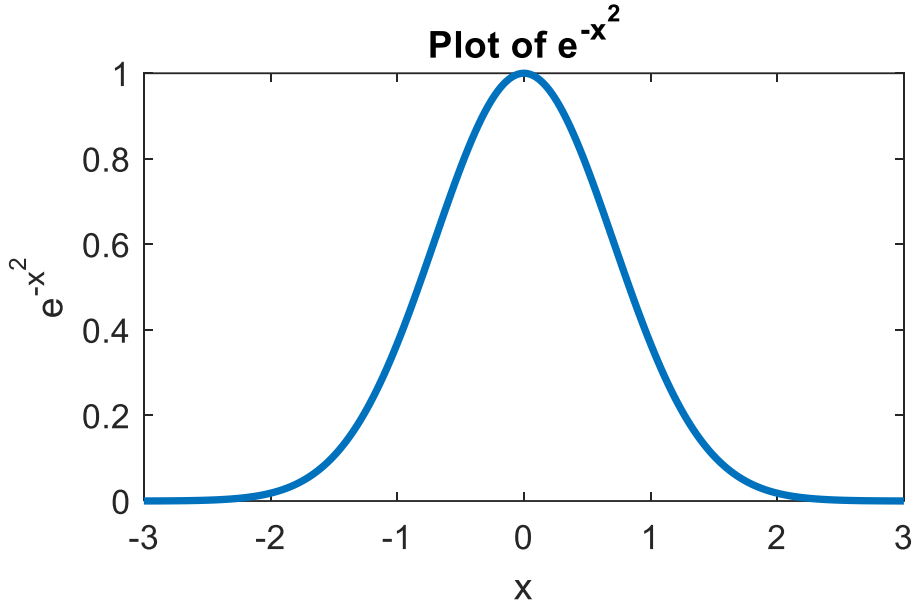
- Similarly, the bias can be updated as:

$$b = b - \alpha \cdot \frac{\partial E}{\partial b}$$



Reward design

- Turn the minimization problem in ***optimization*** into a maximization problem



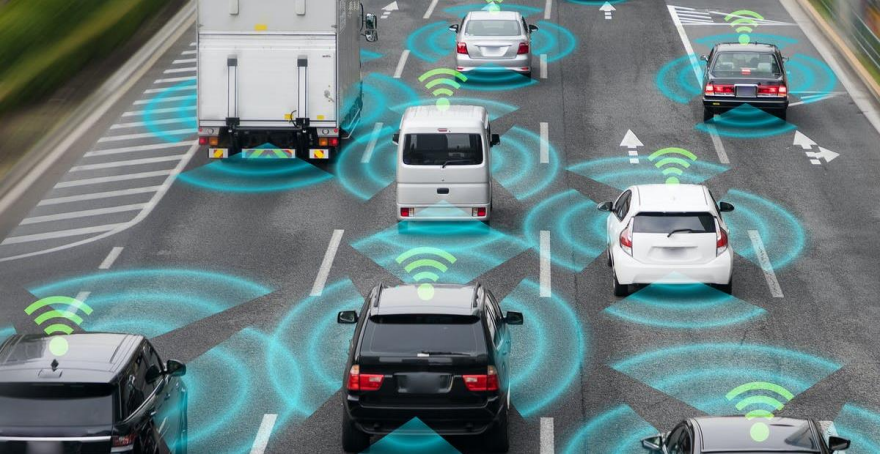
- **Objective Function** for optimization

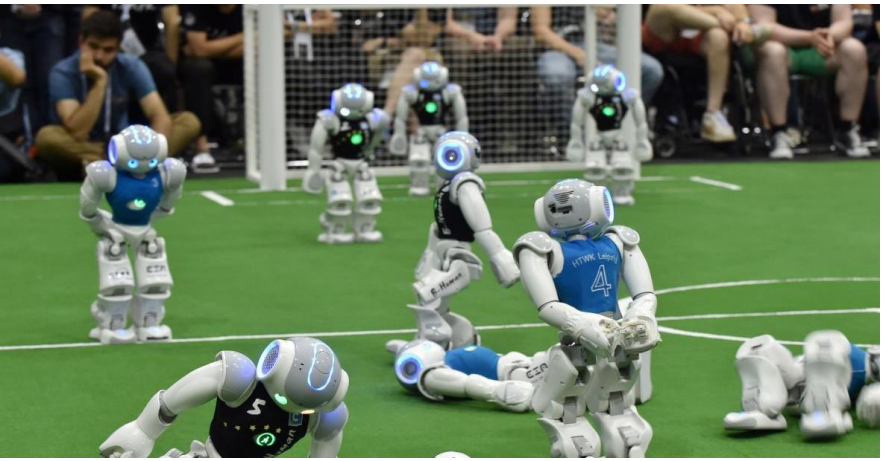
$$\boldsymbol{f(q) = ||F(q) - y^d||^2_{QP} + ||O(q) - vec^d||^2_{Qr} + ||q - q_0||^2_{Qj}}$$

$$\boldsymbol{R} = Q_p e^{-(F(q)-y^d)^2} + Q_r e^{-(O(q)-\mathrm{vec}^d)^2} + Q_j e^{-(q-q_0)^2}$$

Lecture 8: Multi-agent learning

- What is Reinforcement Learning?
- Markov Decision Process
- Q-Learning
- What is MARL?
- Deep Q-Network
- Policy Gradient
- Advantage Actor Critic
- Centralised Critic
- Value Decomposition





Lecture 9: Real-World Case Studies: Research and Applications

- What real world applications require Multi-Agent Reinforcement Learning?
  - ❑ Warehouse robots
  - ❑ Autonomous vehicles (fleet)
- Concepts and principles of Parameter Sharing among the same type of agents.
- Open questions: how to use LLMs as RL agents?

Application-driven and inspired questions

- How to apply concepts, knowledge, algorithms learned from lectures, plus lab practical, and use them to solve a presented case?

Exemplar question: Consider a scenario where a team of autonomous robots are deployed in a warehouse to perform tasks such as picking up, moving, and sorting packages. The warehouse is dynamically changing with staff and objects, packages arriving and departing, and the robots need to coordinate their actions to perform tasks efficiently.

Propose:

1. What sensors for perception to equip the robots with? Justify your choice.
2. What process is needed for pre-processing the sensing before used for perception? Why?
3. Propose how they can communicate with each other to share information.
4. Propose solutions for robots to coordinate and perform their tasks.
5. Suggest what perception techniques can be used for robots to avoid collision?