Intelligent Agents

Boi Faltings

Laboratoire d'Intelligence Artificielle
boi.faltings@epfl.ch
http://moodle.epfl.ch/

Fall 2019

Administrative Information

- Taught by: Boi Faltings, AI Laboratory
- Course site & materials:
  `http://moodle.epfl.ch`
  `-> Informatique -> Master -> Intelligent Agents`
- Moodle Enrolment key: `intelagent77`
- Assistants:
  Panayiotis Danasis, Mi Fei, Adam Richardson

Bibliography

- Michael Wooldridge : An Introduction to Multi-Agent Systems (2nd edition), John Wiley & Sons, 2009.
- Stuart Russell and Peter Norvig: Artificial Intelligence: A Modern Approach (2nd/3rd Edition), Prentice Hall Series in Artificial Intelligence, 2003/2009.

Further background reading: see Moodle site

Prerequisites

- Exercises require significant Java programming.
- Extraordinarily large student numbers mean less individual supervision.

$\Rightarrow$ If you have difficulties with the first exercise (this week), this course may not be for you.

- No previous AI course: easier to make up with textbooks.

Data Science vs. Artificial Intelligence

- Data Science = gaining *insight* from data
- Artificial Intelligence = making intelligent *decisions*
- Insight is a condition for decisions, but does not have to be perfect!

Computers as autonomous agents

Software agents automate individual decisions:

- autonomous devices (home automation, cars, etc.): decisions needed for operation.
- autonomous robots (lawnmower $\Rightarrow$ vacuum cleaner $\Rightarrow$?): actions to achieve and maintain a goal.
- automated stock trading: buy/sell at good moments.
- game playing (Alpha Go): moves to win against an oponent.
- movie characters: behavior that looks like people.
- autonomic computing: manage computing infrastructure.

Computers as mediators

Multi-agent systems find coherent decisions for multiple agents:

- group theater ticket reservations: tradeoff preferences $\Leftrightarrow$ availability.
- autonomous cars: avoid accidents.
- infrastructure (smart grid): balance resource usage.
- traffic management: distribute traffic to avoid jams.
- procurement auctions: optimize supply chain.

What is a software agent?

Defining characteristics:

- autonomy
- reactivity/embeddedness
- proactiveness

Autonomy

Programs are controlled by user interaction.
Agents take action without user control:

- monitor:
  Is current energy price low enough to run wash?
- answer requests:
  Can you reduce power consumption?
- negotiate:
  Buy 1KW for next two hours at $0.10.

Techniques for Autonomy

Procedures are replaced by *behaviors*:

map situation $\Rightarrow$ *action*

Programming agents = defining

- behaviors
- control architecture

Why is this important?

- Fight information overload: user does not have to initiate every action.
- React even when user is not available.
- Implement complex protocols, e.g. for negotiation or bidding in auctions.

Reactivity/Embeddedness

Programs have well-defined input and outputs
Agents

- react to changes in their environment
- change their environment by their actions
- need to act in real time

Techniques for embedded agents

Behaviors rather than procedures

Challenge: real-time response

- table-lookup for instantaneous reaction
- anytime algorithms: solution quality improves with time

Why is this important?

Agents need to act before environment changes:

- place auction bid before close
- turn on freezer before temperature gets too high
- ...

Proactiveness

Agents need to plan for the future:

- to take advantage of opportunities in the environment.
- to avoid future problems, deadlocks, etc.
- to propagate changes or distribute information.

Techniques for proactive agents

- Agents must have explicit *goals*
  Run washing machine before 18:00.
- Goals are linked to *plans* for achieving them.
  Negotiate energy for time slot within budget.
- Plans are continously reevaluated;
  new opportunities lead to replanning
  break laundry cycle in two to reduce cost.
  $\Rightarrow$ deliberative agents

Why is this important?

Goals allow acting on user's behalf.
Actions have preconditions:

- be in the right position to cut grass/vaccum the floor/etc.
- collect information about action effects to make decisions.
- obtain agreement of others for joint actions.

Intelligent Agents

Intelligence = adaptive behavior:

- rationality: choose best action given current situation
- learning: learn to improve choices from past experience

Rational Agents

Programs/Algorithms =

always do the same *thing*

e.g. run washing machine every day.

Rational agents =

do the right *thing*

e.g. run washing machine when enough load and energy is cheap.

Rationality $\Leftarrow$ *goals*

Energy manager:

- satisfy user's wish.
- make sure freezer doesn't go over -8 degrees.
- ...
- make sure laundry gets washed within 2 days.

$\Rightarrow$ action = satisfy the goals!

Extend from plans to actions.

Techniques for implementing rationality

Simple action space (e.g. video games, stock trading):

- optimization
- learning

$\Rightarrow$ policies, neural nets

Complex action sequences (e.g. go, autonomous cars):

- symbolic reasoning
- constraint satisfaction
- planning

$\Rightarrow$ knowledge systems

Why is this important?

- Implement proactiveness in a principled way
- Optimize user's return
- Predictable behavior as a basis for coordination and negotiation

Adaptation ⇒ Learning

- Rationality = adaptation in real time.
- Repeated tasks (driving a car, etc.): learn to act by repeating observed or derived actions.
- Examples: self-driving cars (Tesla), videogame players (Google DeepMind)
- Does not fundamentally change problem.

Reinforcement learning

Agents may need to learn effects of actions:

- satisfaction of its goals.
- changes to the world.
- transitions to other agent states.

Need to balance *exploration* and *exploitation*.

Multi-agent systems

Modern business is handled through software.
Example: supply chains

- each company has multiple suppliers.
- each supplier deals with multiple customers.

$\Rightarrow$ cannot integrate software into single platform.
Companies prefer loosely coupled information systems rather than software integration.
Appropriate model: multiple interacting/communicating agents.

Agent-oriented Software Engineering

Model complex software as collection of agents:

- air traffic control: one agent per aircraft.
- simulation in social sciences/business.

Advantage: more flexible than homogeneous design.
Same techniques as in distributed multi-agent systems.

Smart Infrastructure

Example: smart grid

- use electricity when it is available.
- agents reponsible for households, producers, grid operators.
- coordinate their behavior to achieve overall balance.

Similar: allocate parking spaces, charging stations, air space, etc.

Populations of autonomous agents

- Self-driving cars should not get into each others' way.
- IoT devices need to share wireless spectrum.
- Trading agents influence each others' behavior.

Agents may be *self-interested*: own goals more important than collective goals.

Issues in multi-agent systems

- Coordination of agent actions.
- Making self-interested agents cooperate.
- Distributed implementation.

Coordination

Different agents need to act in a coordinated fashion:

- deliver parts when they are needed
- only 1 truck per delivery
- ...

Difficulties:

- distributedness
- confidentiality
- planning with uncertainty

Agent cooperation

Exploit synergies between agents' goals:

- optimize sharing of resources.
- different agents adress complementary goals.
- one action serves multiple agents' plans.

Techniques for coordination

- Multi-objective optimization techniques:
  - centralized (for tightly coupled problems)
  - distributed (for loosely coupled problems)
- Reinforcement learning to learn coordinated strategies.
- Social choice mechanisms such as voting.

Why is this important?

- Avoid conflicts between multiple intelligent agents (e.g. autonomous vehicles).
- Self-configuring, self-adapting smart infrastructure.
- Cover a joint set of goals by multiple agents.

Self-interest

- Each agent represents a different entity.

$\Rightarrow$ goals may be in conflict.

$\Rightarrow$ agents may not want to cooperate with coordination protocol.

- Most often: several agents want to access the same resource.

Price of Anarchy

Uncoordinated actions have a cost:

- collisions in wifi transmissions.
- too many cars take the same road: traffic jam.
- multiple taxis picking up the same passenger.

Price of anarchy:

$$\frac{\sum Cost(uncoordinatedagents)}{\sum cost(coordinatedagents)}$$

Techniques for dealing with self-interest

- Attach values to achieving goals $\Rightarrow$ *game theory* allows to predict rational agent actions.
- For multiple agents: optimum $\Rightarrow$ *equilibrium*: actions are best responses to each other.
- There can be many equilibria, but (under some conditions) there always is at least one.
- Negotiation: protocols for finding a mutual agreement on a conflicting issue.
- Mechanisms: trusted third party that provides *incentives* for agents to cooperate. Example: auctions.

Why is this important?

- Price of anarchy is often very high: want to avoid uncoordinated behavior.
- Agents may not reveal their true preferences: centralized coordination may not make sense.
- Protect against manipulation by malicious agents.

Distributedness

Programs have common data structures and algorithms
Multi-agent systems model *distributed* systems; agents are independent entities and may:

- be programmed by different people.
- function according to different principles.
- be added and removed during operation.
- be unknown to other agents in the system.

Techniques for distributed agent systems

Agents run on *platforms*:

- runtime environment/interfaces.
- communication languages.
- support for mobility.

Simpler form: web services

Why is this important?

Agent system reflects structure of the real system:

- controlled by their owners.
- local decision making with local information.
- fault tolerant: no central authority.

Agents in practice

Examples of existing applications:

- Infrastructure management (smart grid, etc.).
- Auction agents (eBay), electronic marketplaces.
- Stock trading.
- Personal Agents (Siri, Alexa, etc.) .
- Autonomous robots.
- Large-scale simulation (traffic, social systems).
- Integrating legacy systems.
- Shop floor optimization.

Personal Agents

Best known example: Siri
Main challenges:

- recognizing user intent.
- planning to gather the right information.
- resolving ambiguities in user interaction.
- selecting the right actions to continue the dialogue.

Uses techniques we see in this course + natural language processing.

Agents in Optimization

Optimize use of machine tools (Daimler-Benz):

- assign machines to most important tasks using auctions.
- achieves 99.7 % of theoretical optimum.

Optimize use of trucks (DHL):

- coordination among local centers to share trucks.
- saves 5 % of trucking costs (= double the profit).

In both cases: centralized coordination hard to implement. because of unstructured environment.

Challenges for agent technology

- Difficult to predict behavior: arises from optimization.
- Ethical decisions: when agents are placed in extreme situations, e.g. accidents in self-driving cars.
- Preference elicitation: communicating users' goals and preferences.
- User acceptance: people may not accept computers acting on their behalf.
- Legal issues: agents are not considered legal entities.

Are Intelligent Agents Dangerous?

- Leaving decisions to computers can be dangerous: humans could loose control (Hawking, Gates, Musk, ...).
- Rational agents allow incorporating ethical constraints.

$\Rightarrow$ part of the solution, not the problem!

Outline of the course

Topics to be treated:

- individual agents: reactive/deliberative.
- agent coordination and cooperation.
- self-interested agents: game theory.
- platforms and models for heterogeneous systems.

Practical exercises

- Theoretical material in courses complemented with practical exercises.
- Some paper exercises, but mostly programming in Java using an agent platform.
- Some of the exercises will be graded and count 40% towards the final grade of the course.

Final Exam

- Final exam will test all aspects of the course.
- Generally 2-3 major questions that require solving a problem using the techniques developed in the course.
- A few simple questions about other aspects of the course.
- Allowed: one A4 sheet of notes
  - if machine-written: one-sided
  - if hand-written: two-sided (or 2 one-sided sheets)
- Time: Thursday, December 19th, 13:00-16:00
- Reserve the date now!

Summary

Software agents:

autonomous and proactive software

Multi-agent systems:

Model of heterogeneous, federated software systems

Intelligence = adaptivity is an important element for both