LECTURE 7

SUMMARY

1. Machine Learning 1

2. Learning in agent-based systems ([1], Chapter 6) 6

1. Machine Learning

- Learning problems represent an important research direction in AI, **machine learning** (ML)
- **ML**= the study of system models, that based on a set of data (training data) improve their **performance** (on a particular **task**) by **experiences** and by learning some specific domain knowledge.
- Three main directions for ML
    - **Data mining** – extract knowledge from data (use historical data to improve decisions, predictions, etc).
    - **Software applications we cannot program by hand** – autonomous driving, speech recognition, handwriting recognition, game playing, etc.
    - **Self customizing programs** – programs that adapt to changing conditions
        - Learn the users interests

- The attempt of modeling the human reasoning leads to the notion of *intelligent reasoning*.
- Most of the AI systems are *deductive* ones, capable to draw conclusions (make inferences) based on their initial (or supplied) knowledge, without having the capability to generate new knowledge
- The intelligence of an agent is incorporated in its *program part* (at a given moment, the agent will choose the best action, in the way it was programmed to act)
- In situations in which the agent has incomplete information (knowledge) about the environment in which the agent acts, LEARNING is the only way that the agent could get the needed knowledge.
    - The **learning** assures the **autonomy** of the agent
- Learning problems – learn to represent a function (function approximation).
    - f: Inputs->Outputs – *target* function
    - $h \approx f$ hypothesis
    - Search for the *hypothesis* that best fits the (training) data

1.1 Types of learning

1. **Supplied learning (SL)**
    - predictive models
    - **applications**
        - intrusion detection, data rectification for process control, image (pattern) recognition, predictions
    - completely labeled training data

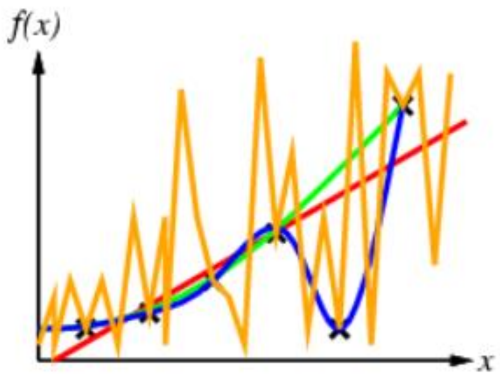
        - A trainer submits the input/output exemplary patterns and the learner has to adjust the parameters of the system autonomously, so that it can yield the correct output patterns when faced with a new input pattern.
        - A set of training examples (x_i, $f(x_i)$), where f is the target function to be learned, is provided to the learner and the aim is to determine (an approximation of) f by some adaptive algorithm.
    - issues in supervised learning
        - **<u>overfitting</u>** (learn by heart)
            - The model learns very well the training data (it has high performance on the training data set), but it does not generalize well on unseen data (low performance on a testing data set)
        - **<u>underfitting</u>**
            - the model is too simple, it cannot capture the structure of the data
        - Noisy training data (errors in data, outliers, irrelevant data)
    - 2 important types of supervised learning
        - **Inductive learning**
            - determine a hypothesis h such that $h(x_i) \approx f(x_i)$
            - how to compare two hypotheses approximately close to f?
                - *inductive bias*
        - **Analogical learning**
            - Identifying analogies between an experienced problem instance and a new problem
            - E.g. case-based reasoning (CBR)

2. **Unsupervised learning (UL)**
    - descriptive models
    - completely unlabeled training data
    - in absence of trainers, the desired output for a given input instance is not known, and consequently, the learner has to adapt its parameters autonomously.
3. **Reinforcement learning (RL)**
    - learning by interaction with the environment
    - an autonomous agent learns to perform an optimal sequence of actions to reach a goal
    - **<u>applications</u>**
        - game playing, robotics and control
4. **Semi-supervised learning**
    - falls between SL and UL
    - the learner is provided with a small amount of labeled data and a large amount of unlabeled data
    - graph-based methods, kernel-based methods, aso.

<u>Inductive logic programming</u> (ILP)
- subfield of ML which uses first-order logic to represent hypotheses and data
- specifically targets problems involving structured data and background knowledge
- tackles a wide variety of problems in machine learning, including classification, regression, clustering, and reinforcement learning.
- **applications**
    - in bio- and chemo-informatics, natural language processing, and web mining.

1.2 Supervised learning

- example: Inductive learning
- learn a function from examples
- Occam’s razor
    - prefer the simplest hypothesis consistent with data
    - complex models tend not to generalize well



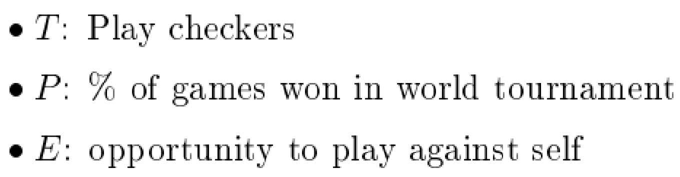
[2]

- **Classification** vs. **regression**
    1. learning a relationship between an input (vector *x*) and an output (*y*) from data
    2. classification estimates the discrete output *y* (known as the class)
        - binary/multi-class classification
        - predicting tumor cells as benign or malignant
        - categorizing news according to their domain (finance, sports, weather, etc)
        - classifying secondary structures of proteins
        - classifying credit card transactions
        - ….
    3. regression estimates the function **f** such that *y*=**f**(*x*) with some confidence measure
        - predicting the exchange rate
        - predicting the price of houses
        - predicting the age of a person
        - …..
- **Offline vs. online** learning
    1. **Offline (batch) learning**
        - generates the best predictor by learning on the entire data set at once
    2. **Online learning**
        - data becomes available in a sequential order and is used to update the best predictor for future data at each step
        - learning incrementally (one instance at time)
        - useful where is computationally unfeasible to train over the entire data set (e.g. stock price prediction)
        - time dependent

- **Eager** vs. **lazy** learning
    1. **Eager learning**
        - an example of offline learning
        - the system tries to construct a general, input independent target function during the training of the system
            1. a global model (hypothesis) is built during the training step
            2. slow training, fast evaluation
        - builds a global estimate of the target function
        - deal much better with noise
        - generally unable to provide good local approximations of the target function
        - e.g. ANNs, SVMs, DTs, NBC, etc
    2. **Lazy learning**
        - little or no offline processing of the training data
        - generalization beyond the training set is delayed until a query is made to the system
        - simply stores the training examples
            - fast training, slow evaluation
        - e.g. instance-based learning methods (kNN, LWR, CBR)
        - provide a local approximation for the target function, for each new query instance

1.3 The learning problem

- specification of the learning task
    - *T*, *P*, *E*
        - improve over task *T*
        - with respect to performance measure *P*
        - based on experience *E*
    - e.g, learning to play a board game [1]



- T: Play checkers
- P: % of games won in world tournament
- E: opportunity to play against self

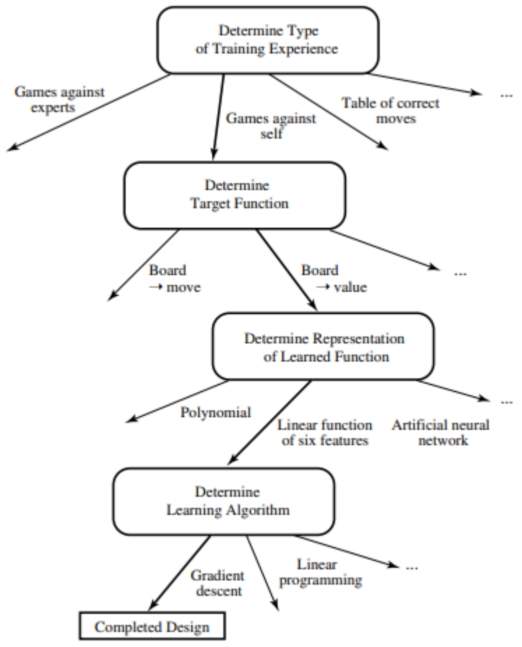
        - what experience?
            - direct or <u>indirect</u>?
            - teacher or <u>not</u>?
            - is training experience representative to the performance goal?
        - what exactly should be learned?
        - how shall it be represented?
        - what specific algorithm to learn it?

Choose the Target Function

- $ChooseMove : Board \rightarrow Move$??
- $V : Board \rightarrow \Re$??
- ...

- components of a learning system
  - the **performance system**
    - responsible with providing the output, using the learned target function
  - the **critic**
    - responsible with providing a feedback to the learner (e.g. training examples in case of SL)
  - the **generalizer**
    - responsible with producing an output hypothesis that is the estimate of the target function
  - the **experiment generator**
    - mainly in RL scenarios
    - takes as input the current hypothesis (currently learned function) and outputs a new problem for the performance system to explore

Design Choices


Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board → move
Board → value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

[4]

2. Learning in agent-based systems ([1], Chapter 6)

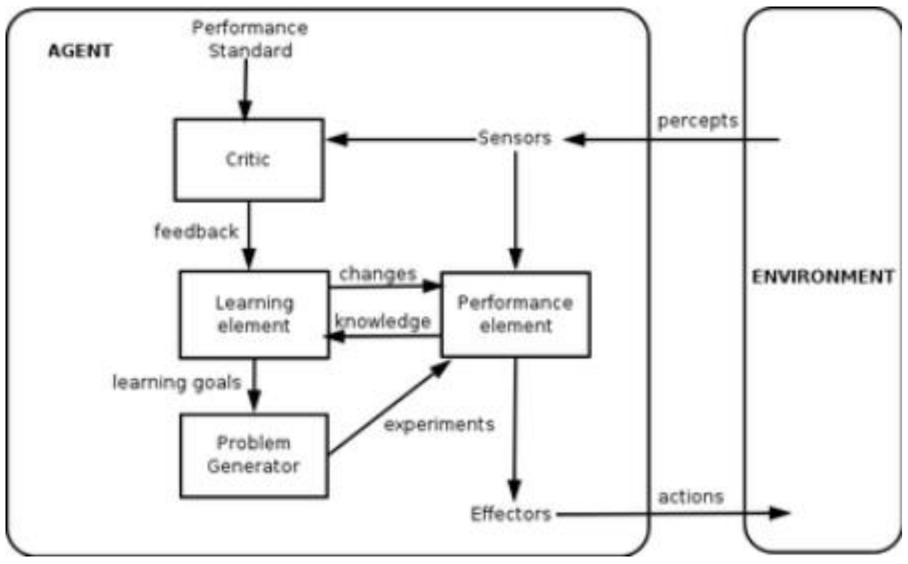
2.1 The general model of learning agents

A learning agent can be divided into four conceptual components

- **Learning element** – which is responsible for making improvements
- **Performance element** – which is responsible for selecting external actions (as for non-learning agents)
- The **critic** is designed to tell the learning element how well the agent is doing and how the performance element should be modified to do better in the future. The critic employs a fixed standard of performance (the performance standard is a fixed measure that is conceptually outside the agent).
- The **problem generator** – is responsible for suggesting actions that will lead the agent to new and informative experiences. If the agent is willing to explore the environment and do some (perhaps) suboptimal actions in the short run, it might discover much better actions for the long run. The problem generator's job is to suggest these exploratory actions.
    - Not all learning agents will need a problem generator – a teacher, or the agent's normal mode of operation, may provide sufficient feedback for learning

Available feedback

- *Supervised* learning – some (input, output) examples are provided by a teacher (supervisor) – labelled examples
- *Reinforcement* learning – learn by interaction with the environment (receiving *rewards*)
- *Unsupervised* learning – no feedback is given to the leaner - unlabelled examples
- *Semisupervised* learning - labelled and unlabelled examples



2.2 Learning in MAS

- The first attempt in ML – to treat learning as a centralized and isolated process that takes place in intelligent stand alone systems.
- Nowadays, using ML in MAS is a large development area. There are two main reasons
    1. there is a great need to provide MAS with learning capabilities
        - MAS must need to react in complex, large, open, dynamic and unpredictable environments => the only way is learning
    2. a general view on ML that contains not only monoagent learning, but multiagent learning also, can lead to a better understanding of the general and fundamental principles of learning in natural and computational systems.

Cooperative learning in MAS

- There is a special interest for descentralized approaches in solving real world problems.
- Learning in MAS = applying ML in problems that contain several agents
- 2 basic aspects
    - Because several agents are implied => the search space can be very large.
    - Learning can imply several learning entities, each of which learn and adapt to others.
- 2 approaches
    1. **Team learning** – a single learning entity that learns to improve the performance of the whole team of agents
    2. **Concurrent learning** – individual learning processes for individual agents, considering in the same time the quality of the team of agents

The learning is

- **Centralized** – if the learning process is executed in all its aspects by a single agent, and no interaction with other agents is needed (the agent that learns acts as it is alone).
- **Descentralized** – if several agents are implied in the same learning process. This means that the activities that constitute the learning process are executed by different agents.

2.2.1 Team learning

- It is the simplest approach for learning in MAS – we can use standard learning techniques (for a single agent), there is a single entity that learns.
- Another **advantage** is that the agents tend to action in order to maximize the reward of the while team, instead of maximizing their own rewards.
- **Disadvantages**
    1. Combinatorial explosion of states during the learning process
        - Ex. if agent A can be in one of 100 states and B can be in one of 100 states => the team can be in one of 10.000 states
    2. The centralization of the learning algorithm – all the resources must be available in a single place, where all the computations are made. This can be a disadvantage in domains in which data are distributed.
- 3 types

    - Homogeneous learning – all the learning entities develop a single behaviour that is used by each member of the team
        - cellular automata
    - Heterogeneous learning – unique behaviors for each agent in the team
    - Hybrid – combination of the first two – divide the team into subteams (the agents from a subteam are homogeneous).
- **Evolutionary computation** – a good approach for team learning

2.2.2 Concurrent learning

- Alternative to cooperative multiagent learning in which each agent from the team learns independently how to improve their performance and the whole team performance, also.
- A main advantage is that CL divides the joint search space in smaller individual search spaces
    - If the problem can be decomposed such that the individual behaviors are relatively disjoint => a substantial reduction of the search space and of the computational complexity
- Ex. concurrent **Q-learning**

Applications

- Robotic agents
    - Football – RoboCup
    - Maze searching
- Distributed Vehicle Monitoring
    - optimize the traffic in a network of intersections (each agent controls a semaphore in an intersection
- Routing problems in networks and network management
- Air traffic control
- Electricity distribution management
- Distributed health-care
    - assist in medical diagnosis
    - suitability to MAS
        - data and resources are descentralized
        - data dynamicity
- Industrial production
    - planning a sequence of operations in order to achieve a certain goal (product)

- Distributed ANN, DT, HMM, SVM, kNN, RL, clustering, SOM, etc
- Ant algorithms
    - social insects
        - bees, wasps, termites, ants
    - swarm intelligence
        - intelligence of a group
    - relation to RL and MAS
    - applications
        - TSP
        - Routing in a Communication Network

- Vehicle Routing
- Graph coloring
- …

More about Learning in MAS may be found in [1], Chapter 6.

Bibliography

[1] Weiss, G. (Ed.): Multiagent Systems: *A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999 (available at www.cs.ubbcluj.ro/~gabis/weiss/Weiss.zip) [Ch. 4, 6]
[2] Șerban Gabriela, *Sisteme Multiagent în Inteligenta Artificiala Distribuita. Arhitecturi si Aplicatii*, Editura RisoPrint, Cluj-Napoca, 2006
[3] Czibula Gabriela, *Sisteme inteligente. Instruire automată*, Editura RisoPrint, Cluj-Napoca, 2008
[4] Mitchell, T., *Machine Learning*, McGraw Hill, 1997 (available at www.cs.ubbcluj.ro/~gabis/ml/ml-books)