

# Machine Learning DD2431

Örjan Ekeberg

Oct–Dec, 2008

- 1 What is Machine Learning?
  - Definition of Learning
  - Applications
- 2 About the Course
  - Registration
  - Examination
  - Course Contents
  - Labs
- 3 Example
  - A Hypothetical Project

- 1 What is Machine Learning?
  - Definition of Learning
  - Applications
- 2 About the Course
  - Registration
  - Examination
  - Course Contents
  - Labs
- 3 Example
  - A Hypothetical Project

## Specifying the Learning Task

Central components:

- What to do; the *Task*,  $T$
- Improves according to a measurable *Performance*,  $P$
- Improvement based on *Experience*,  $E$

### Playing Chess

- $T$ : select next move, following the rules of chess
- $P$ : percentage of matches won
- $E$ : train against a copy of yourself

### Reading Handwritten Text

- $T$ : interpret words given as bitmap images
- $P$ : fraction correctly interpreted words
- $E$ : database with pre-interpreted text

## Applications

### Sample Applications

- Speech recognition
- Autonomous driving
- Games: Backgammon
- Autonomous robots
- Spam-filter for e-mail

### Role of Learning

**Data mining** Transform data into knowledge

**Vaguely specified tasks** Robotics, speech, vision, games

**Adaptive programs** User adaptable programs/devices

### 1 What is Machine Learning?

- Definition of Learning
- Applications

### 2 About the Course

- Registration
- Examination
- Course Contents
- Labs

### 3 Example

- A Hypothetical Project

### Course Registration

- Centrally: via "Mina Sidor"
- Locally: [res checkin mi08](#)

### Course Information

<http://www.csc.kth.se/DD2431/mi08>

## Examination

Obligatory parts of the course

- Written exam (**tentamen**)
- Four labs

### Bonus Points

- Each lab finished (successfully examined) before its deadline gives one bonus point.
- Max bonus (=4) raises the final grade one step.
- Bonus can not save you from F (failed).
- Bonus points can not be saved to next year.

# Course Contents

- Concept Learning
  - Decision Trees
  - Artificial Neural Networks
  - Statistical Methods
  - Reinforcement Learning
  - Genetic Algorithms
  - Learning Theory
  - Support Vector Machines
  - Rule-based Learning
- Begreppsinläring
  - Beslutsträd
  - Artificiella neuronnät
  - Statistiska tekniker
  - Belöningsbaserad inläring
  - Genetiska algoritmer
  - Lärbarhetsteori
  - Strukturell riskminimering
  - Inläring av regler

# Labs

- 1 Concept Learning & Decision Trees
- 2 Bayes Classifier & Boosting
- 3 Reinforcement Learning
- 4 Support Vector Machines

Note: Labs are not in the schedule.

Online booking of lab examination time-slots.

Examination:

- It is **your** task to convince the examiner that you have done the assignment and understood the results.
- 10 minutes
- No computer

- 1 What is Machine Learning?
  - Definition of Learning
  - Applications
- 2 About the Course
  - Registration
  - Examination
  - Course Contents
  - Labs
- 3 Example
  - A Hypothetical Project

## A Hypothetical Project:

A program that learns to play Reversi (*Othello*)

We must specify

- The task  $T$   
Choose moves following the rules
- How to measure performance  $P$   
Fraction of won matches against a good player
- How to get experience  $E$   
Play against oneself and others

- How do we get enough training data?
- What should the program do, in more detail?
- What internal representation should we use?
- What learning technique should we use?

## Training Data

- Teacher or raw samples?
- Direct or Indirect?  
Will anybody tell you what is a good move?
- Credit assignment  
Good individual moves or a long-term strategy?
- Are we using representative training data?

What do we want the program to learn?

$$P : b \mapsto d$$

$b$ : current board state

$d$ : our next move

Hard!

$$V : b \mapsto \mathcal{R}$$

$b$ : board state after our move

$\mathcal{R}$ : "value" of the state

"Value" can be defined recursively

$V(b) = 100$  for any winning final state

$V(b) = -100$  for any losing final state

$V(b) = V(b')$  where  $b'$  is the final state reached after playing optimally from  $b$ .

Can't be computed!

Dramatic approximation:

$$V(b) \approx V(\text{succ}(b))$$

$\text{succ}(b)$ : the state reached after  $b$

More approximations:

$$V_{\text{train}}(b) \leftarrow \hat{V}(\text{succ}(b))$$

$\hat{V}(\cdot)$ : Our estimate of  $V(\cdot)$

$V_{\text{train}}(b)$ : New training sample for estimating  $V(b)$

Given the right conditions,  $\hat{V}$  will converge to  $V$ .

How do we represent  $\hat{V}$ ?

- Set of Rules?
- A Neural Network?
- Selected Features?

Possible suggestion: *Weighted Features*

$$\hat{V}(b) = w_0 + w_1 a_1 + w_2 a_2 + w_3 a_3 + \dots$$

$a_1$ : Number of white tiles

$a_2$ : Number of black tiles

$a_3$ : N:o white corner tiles

$a_4$ : N:o black corner tiles

etc.

Learning  $\equiv$  choose  $w_i, \forall i$

How can we compute  $w_i$  from the training samples?

LMS-rule (*Least Mean Square*)

- 1 Estimate the error from a single sample

$$\delta(b) = V_{\text{train}}(b) - \hat{V}(b)$$

- 2 Update the weights to reduce this error

$$w_i \leftarrow w_i + \eta \delta(b) \cdot a_i$$

Will this work in practice?

Probably, but with very limited performance.

What should be improved?

Primarily the representation of the value function.