Machine Learning (COMP-652)

Instructor: Doina Precup

Email: dprecup@cs.mcgill.ca

Grader: Pavel Dimitrov

Class web page:

http://www.cs.mcgill.ca/∼dprecup/courses/ml.html

Outline

- Administrative issues
- What is machine learning?
- Why study machine learning?
- Formulating machine learning problems
- Machine learning questions

Administrative issues

- Class materials:
  - Tom Mitchell, Machine Learning (main text)
  - Additional readings: distributed in class and/or posted on the web page
  - Class notes: posted on the web page
- Prerequisites:
  - Knowledge of a programming language (e.g. C, C++, Java, LISP, Matlab)
  - Some knowledge of probabilities and statistics
  - Some AI background is recommended *but not required*

Evaluation

- Seven homework assignments (35%)
- Two in-class written examinations (20%)
- Project (30%)
  - Reading research papers on a chosen topic
  - Implementing and/or experimenting with algorithms related to the topic
  - a written report on your findings
  - Possibly a class presentation
- Reading assignments (15%)
- Participation to class discussions (up to 2% extra credit)

What is learning?

- H.Simon: Any process by which a system improves its performance
- M.Minsky: Learning is making useful changes in our minds
- Michalsky: Learning is constructing or modifying representations of what is being experienced
- Valiant: Learning is the process of knowledge acquisition in the absence of explicit programming

Why study machine learning?

- Easier to build a learning system than to hand-code a working program! E.g.:
  - Robot that learns a map of the environment by wandering around it
  - Programs that learn to play games by playing against themselves
- Improving on existing programs, e.g
  - Instruction scheduling and register allocation in compilers
  - Combinatorial optimization problems
- Discover knowledge and patterns in highly dimensional, complex data

Why study machine learning?

- Solving tasks that require a system to be adaptive, e.g.
  - Speech and handwriting recognition
  - “Intelligent” user interfaces
- Understanding animal and human learning
  - How do we learn language?
  - How do we recognize faces?
- Creating real AI!

  “If an expert system–brilliantly designed, engineered and implemented–cannot learn not to repeat its mistakes, it is not as intelligent as a worm or a sea anemone or a kitten.” (Oliver Selfridge).

What are good machine learning tasks?

- There is no human expert

  E.g., DNA analysis

- Humans can perform the task but cannot explain how

  E.g., character recognition

- Desired function changes frequently

  E.g., predicting stock prices based on recent trading data

- Each user needs a customized function

  E.g., news filtering

Very brief history

- Studied ever since computers were invented (e.g. Samuel's checkers player)
- Coined as "machine learning" in late 70s - early 80s
- Very active research field, several yearly conferences (e.g., ICML, NIPS), major journals (e.g., Machine Learning, Journal of Machine Learning Research)
- The time is right to start studying in the field!
  - Recent progress in algorithms and theory
  - Growing flood of on-line data to be analyzed
  - Computational power is available
  - Growing demand for industrial applications

Related disciplines

- Artificial intelligence
- Probability theory and statistics
- Computational complexity theory
- Control theory
- Information theory
- Philosophy
- Psychology and neurobiology

Three niches for machine learning

- Data mining
  - Using historical data to improve decisions

    E.g., medical records $\rightarrow$ medical knowledge
  - Finding patterns in data

    E.g., finding new star clusters
- Software applications we cannot program by hand

  E.g., autonomous driving, speech recognition
- Self customizing programs

  E.g., newsreader that learns user interests

Typical datamining task

Data:

Patient103 time=1	*Patient103* time=2	*Patient103* time=n
Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C–Section: ?	Elective C–Section: no	Elective C–Section: no
Emergency C–Section: ?	Emergency C–Section: ?	**Emergency C–Section: Yes**
...	...	...

Given:

- 9714 patient records, each describing a pregnancy and birth
- Each patient record contains 215 features

Learn to predict: classes of future patients at high risk for Emergency Cesarean Section

Datamining result

Patient103 time=1

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: no
PreviousPrematureBirth: no
Ultrasound: ?
Elective C–Section: ?
Emergency C–Section: ?
...

Patient103 time=2

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: YES
PreviousPrematureBirth: no
Ultrasound: abnormal
Elective C–Section: no
Emergency C–Section: ?
...

...

Patient103 time=n

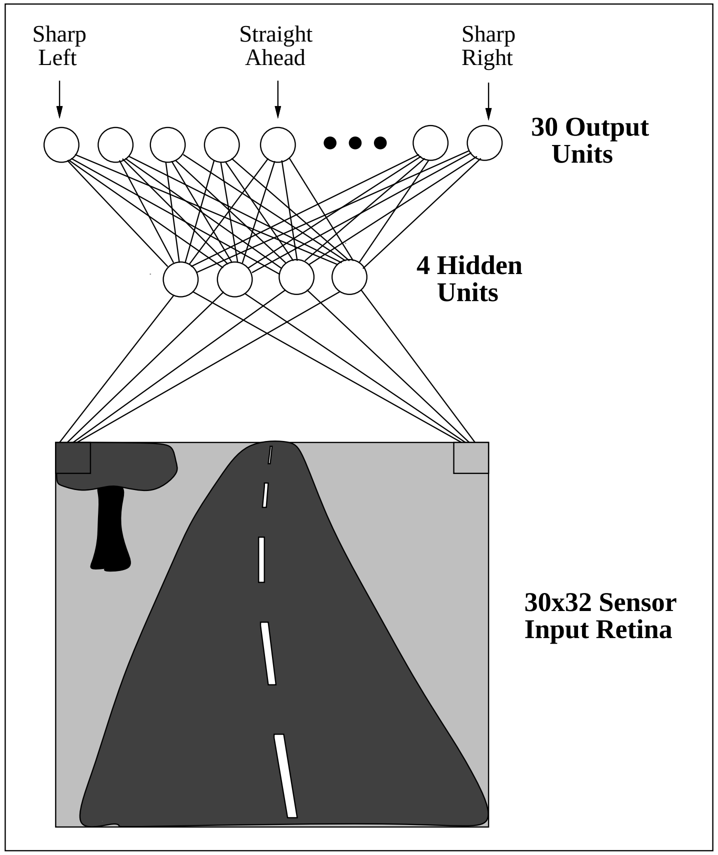
Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: no
PreviousPrematureBirth: no
Ultrasound: ?
Elective C–Section: no
Emergency C–Section: Yes
...

One of 18 learned rules:

```
If    No previous vaginal delivery, and
      Abnormal 2nd Trimester Ultrasound, and
      Malpresentation at admission
Then Probability of Emergency C-Section is 0.6
 Over training data: 26/41 = .63,
 Over test data: 12/20 = .60
```

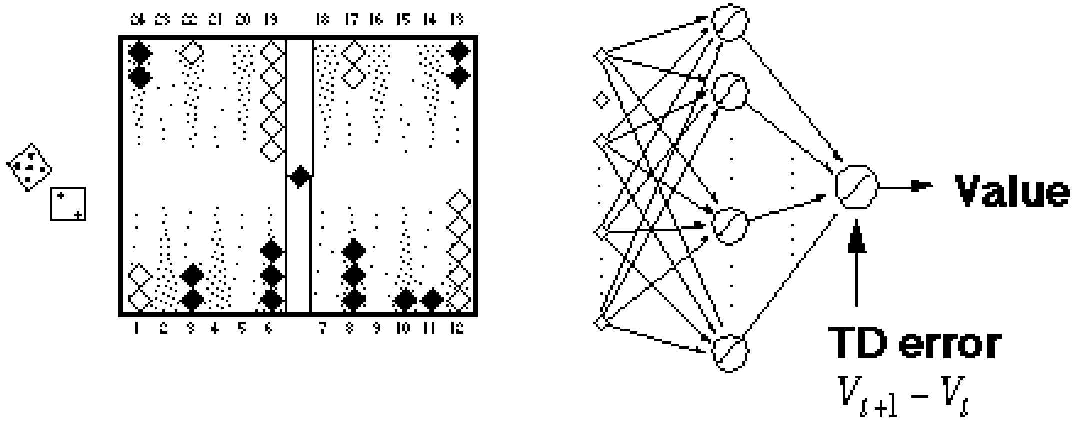
Problems too difficult to program by hand

ALVINN [Pomerleau] drives 70 mph on highways



Tesauro, 1992–1995

TD-Gammon



Action selection by 2–3 ply search

Start with a random network

Play millions of games against self

Learn a value function from this simulated experience

This produces arguably the best player in the world

Software that customizes itself

- Interactive software is everywhere (text editors, web browsers, spreadsheets, ...)
- Most programs can be customized, by setting “preferences”
- But this is a tedious, manual process, not accessible to all users
- Better solution: watch what the user is doing, and try to model its interests/goals, then use the model to get better interaction
- E.g., web pages that re-organize the links

Important application areas

- Bioinformatics: sequence alignment, analyzing micro-array data,
- Computer vision: object recognition, tracking, segmentation, active vision, ...
- Robotics: state estimation, map building, decision making
- Graphics: building realistic simulations
- Speech: recognition, speaker identification
- Financial analysis: option pricing, portfolio allocation
- E-commerce: automated trading agents, data mining, spam, ...
- Medicine: diagnosis, treatment, drug design,...
- Computer games: building adaptive opponents
- Multimedia: retrieval across diverse data bases

What is the future?

Today: tip of the iceberg

- First-generation algorithms: neural nets, decision trees, regression ...
- Applied to well-formated database
- Budding industry

Opportunity for tomorrow: enormous impact

- Learn across multiple databases, plus the web and news wires
- Learn by active experimentation
- Learn decisions rather than predictions
- Cumulative, lifelong learning
- Programming languages with learning embedded?

What is a learning problem?

Learning = Improving with experience at some task

More precisely:

- Improve over task T,
- with respect to performance measure P,
- based on experience E.

E.g. Learn to play checkers

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

Posing learning problems

	Task Definition	Performance Measure	Training Experience
Speech recognition			
Robot driving			
Language learning			

Type of training experience

- Direct or indirect?
- Teacher or not?

A problem: is training experience representative of performance goal?

Choose the target function

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

Possible definition for target function V

- if b is a final board state that is won, then $V(b) = 100$
- if b is a final board state that is lost, then $V(b) = -100$
- if b is a final board state that is drawn, then $V(b) = 0$
- if b is a not a final state in the game, then $V(b) = V(b')$, where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

This gives correct values, but is not operational

Choose representation for target function

- Collection of rules?
- Neural network ?
- Polynomial function of board features?
- ...

A representation for learned function

$$w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

- $bp(b)$: number of black pieces on board b
- $rp(b)$: number of red pieces on b
- $bk(b)$: number of black kings on b
- $rk(b)$: number of red kings on b
- $bt(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $rt(b)$: number of black pieces threatened by red

Obtaining training examples

One rule for estimating training values:

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

where:

- $V(b)$: the true target function
- $\hat{V}(b)$: the learned function
- $V_{train}(b)$: the training value

Choose weight tuning rule

LMS Weight update rule:

Do repeatedly:

1. Select a training example b at random
2. Compute $error(b)$:

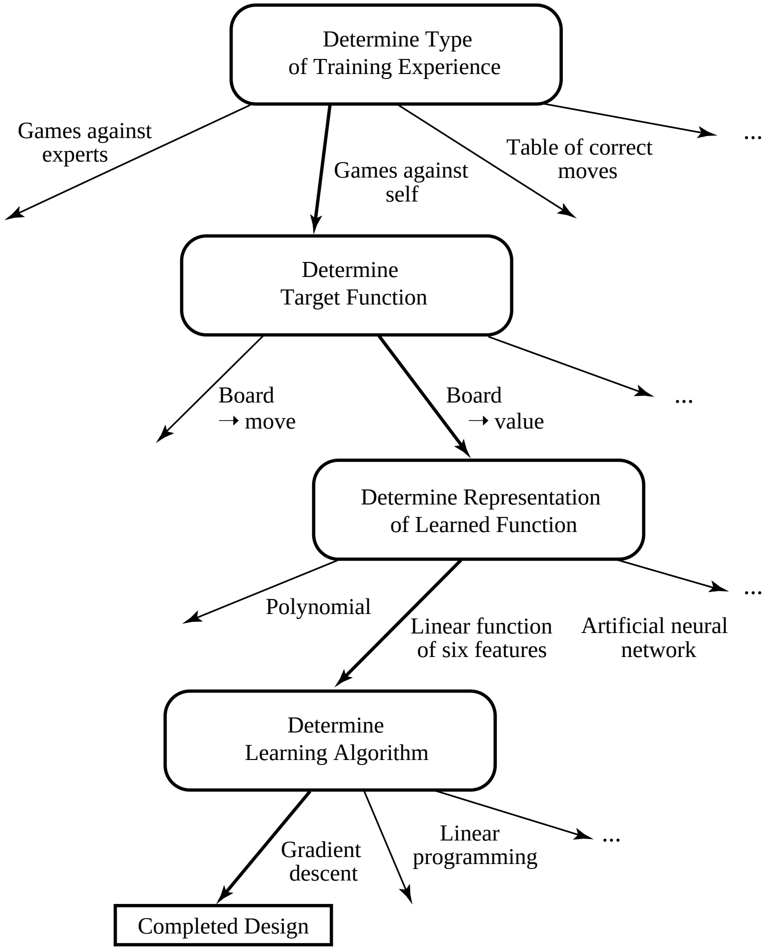
$$error(b) = V_{train}(b) - \hat{V}(b)$$

3. For each board feature f_i, update weight w_i:

$$w_i \leftarrow w_i + c \cdot f_i \cdot error(b)$$

c is some small constant, say 0.1, to moderate the rate of learning

Design choices



Some issues in machine learning

- What algorithms can approximate functions well (and when)?
- How does number of training examples influence accuracy?
- How does complexity of hypothesis representation impact it?
- How does noisy data influence accuracy?
- What are the theoretical limits of learnability?
- How can prior knowledge of learner help?
- What clues can we get from biological learning systems?
- How can systems alter their own representations?

Kinds of learning

Based on the information available:

- Supervised learning
- Reinforcement learning
- Unsupervised learning

Based on the role of the learner

- Passive learning
- Active learning

Supervised learning

- Training experience: a set of **labeled examples** of the form $v_1\, v_2\, \ldots v_n, o$, where v_i are values for *input variables* and o is the *output*
- This implies the existence of a "teacher" who knows the right answers
- What to learn: A **function** $f : V_1 \times V_2 \times \cdots \times V_n \rightarrow O$, which maps the input variables into the output domain
- Performance measure: minimize the error (loss function) on the training examples

Reinforcement learning

- Training experience: interaction with an environment; the agent receives a numerical reward signal
- E.g., a trading agent in a market; the reward signal is the profit
- What to learn: a way of behaving that leads to a lot of reward in the long run
- Performance measure: maximize the long-term reward

Unsupervised learning

- Training experience: unlabeled data
- What to learn: interesting associations in the data
- E.g., image segmentation, clustering
- Performance measure: ?
- Often there is no single correct answer

Passive and active learning

- Traditionally, learning algorithms have been *passive learners*

  Data $\rightarrow$ Learner $\rightarrow$ Model

- *Active learners* are instead allowed to query the environment
  - Ask questions
  - Perform experiments
- Open issues: how query the environment optimally? how to account for the cost of queries?