Slide 1

COM3250 / COM6170

Introduction to Machine Learning

Instructor	Rob Gaizauskas	
	Email:	`r.gaizauskas@dcs.shef.ac.uk`
	Office:	Room G28b @ Computer Science
Classes	Lecture 1:	Monday 10:00 am, SG–LT08 (LT8, St. George's Complex)
	Tutorial/Lab:	Monday 11:00 pm, SG—LT08 (LT8, St. George's Complex)
	Lecture 2:	Tuesday 10:00 am, SG–LT08 (LT11, St. George's Complex)
Homepage	`www.dcs.shef.ac.uk/~robertg/campus_only/com3250/`	
Assessment	Coursework: 30%	
	Examination: 70 %	
Reading Weeks	Weeks 6 and 12	
Office Hours	Email requests for appointment	

Slide 2

Course Aims and Objectives

Aims

- to describe the main approaches to automated concept learning
- to discuss the relationship between natural and artificial forms of learning
- to develop students' skills in designing and building serious artificial intelligence programs

Objectives

By the end of this course the students should:

- understand the main approaches that are used for representing concepts and learning them automatically;
- understand the relationships between natural learning processes and machine learning techniques;
- be able to develop effective software to implement common forms of knowledge representation;
- be able to develop effective software that will apply representative automated learning techniques using such representations.

Slide 3

Course Presuppositions

- knowledge of basic artificial intelligence techniques, from COM1080
- Advisable: some knowledge of machine reasoning techniques, from COM3290 (formerly COM2100).

Slide 4

Course Structure

- Introduction to Machine Learning (Lecture 1)
- Concept Learning (Lecture 2 and 3)
- Decision Tree Learning (Lecture 4 and 5)
- Evaluating Hypotheses (Lecture 6 and 7)
- Bayesian Learning, Bayesian Classifiers, Bayesian Belief Networks (Lecture 8 – 10)
- Instance-based Learning (Lecture 11 and 12)
- Rule Set Learning, Induction and Inductive Logic Programming (Lecture 13 – 16)
- Computational Learning Theory (Lecture 17 and 18)

Slide 5

References

Primary Textbook

T. Mitchell. *Machine Learning*. WCB/McGraw-Hill, Boston, 1997.

`http://www.cs.cmu.edu/~tom/mlbook.html`

Also Recommended

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, San Francisco, 2000.

Slide 6

Introduction to Machine Learning

Lecture Outline:

- What is Machine Learning?
- Why Study Machine Learning?
- Applications of Machine Learning
- Designing a Learning System: Overview
- WEKA: Software for Machine Learning

Reading:

Chapter 1 of Mitchell

Chapter 1 of Witten & Frank

Slide 7

What is Machine Learning ?

- A possible definition:

  The study of how to design computer programs whose performance at some task improves through experience.

  or more precisely (Mitchell):

  Definition: A computer program is said to **learn**
  - from experience E
  - with respect to some class of tasks T and
  - performance measure P

  if its performance at tasks in T as measured by P improves with experience E

- Important issue:

  Are we only interested in *performance* of learning program?

  Or are we also interested in discovering *human-comprehensible descriptions of patterns* in data? (*knowledge discovery*)

Slide 8

What is Machine Learning ? (cont)

- Disciplines Contributing to Machine Learning

Statistics	*Biology*
Artificial Intelligence	*Cognitive Science*
Philosophy	*Computational Complexity*
Information Theory	*Control Theory*

Slide 9

Why Study Machine Learning ?

- *technological or engineering motivation*
  to build computer systems that can improve their performance at tasks with experience (data)
  - massive growth in on-line data
- *computer science motivation*
  to understand better properties of various algorithms for function approximation
  - how the data must be represented
  - how much data they require
  - how accurate they can be
  - how to choose optimal data for training
- *cognitive science motivation*
  to understand better how humans learn by modelling the learning process

Slide 10

Areas of Application of Machine Learning

- Data mining: using historical data to improve decisions – increasingly important given explosion of electronic data. E.g.:
  - *Medicine:* medical records → medical knowledge
    * selecting best embyros from *in vitro* fertilisation based on 60 features of embryos and historical data on viability
  - *Business:* customer records → better business decisions
    * assessing credit-worthiness of loan applicants based on features of former borrowers and repayment outcomes
    * improving customer retention based on discovering patterns of features amongst loyal vs. defecting customers
  - *Agriculture* herd/crop records → better farming decisions
    * improving cull selection from dairy herds by data mining over database of 700 attributes of millions of cows
  - ...

Slide 11

Areas of Application of Machine Learning (cont)

- Software applications we can't program by hand
  - autonomous driving
  - speech recognition
- Self customizing programs
  - Newsreader that learns user interests
- Computer Games
  - Heuristic evaluation functions for combinatorily explosive board games (chess, checkers, Othello)

Slide 12

Designing a Learning System: Overview

Task Design a checkers (draughts) program to play in world tournament.
(cf. Samuels, 1959; `http://www.cs.ualberta.ca/~chinook`)

Performance measure Percentage of games won in world tournament.

Principal design issues

- Choosing the training experience
- Choosing the target function
- Choosing a representation for the target function
- Choosing a function approximation algorithm

Slide 13

Designing a Learning System: Choosing the Training Experience

Choice of type of training experience has significant impact on success or failure of learning system

- Must decide between
  - *direct* training examples – e.g. individual checkers board states and correct moves for each; or
  - *indirect* training examples – e.g. move sequences and final outcomes of games.
    Here learner must additionally decide which moves in the sequence are good/bad (the *credit assignment*) problem.

  Direct training examples easier to learn from but harder/more expensive to obtain
- Must decide how much learner controls training examples
  - teacher suggests board states and correct moves
  - learner asks teacher about novel/confusing board states
  - learner plays itself (no teacher)
  - random process outside learner provides examples
  - learner autonomously explores environment to collect training examples

Slide 14

Designing a Learning System: Choosing the Training Experience (cont)

- Must decide how well training examples represent distribution of examples over test space
  - ideally want training/test distributions to be identical, but not always possible in practice
- Returning to checkers problem:

  Task T : playing checkers

  Performance measure P : percentage of games won in world tournament

  Training experience E : games played against itself (i.e. *indirect* training examples)

Slide 15

Designing a Learning System: Choosing the Target Function

- Must decide *what* will be learned and *how* this will be used by the *performance program*
- For the checkers program, assume program can generate legal moves from any given board state.

  Want the program the learn the *best* move.
- Natural to think of this as learning a function

$$\mathit{ChooseMove} : B \rightarrow M$$

  from set of legal board positions B to set of legal moves M

  In machine learning such a function is called a *target function*.

  Choosing the function to learn is a key design choice.
- Given the indirect training examples available, *ChooseMove* is very difficult to learn.

  Better choice is a function

$$V : B \rightarrow \mathcal{R}$$

  which assigns a *numerical value* to each board state – higher values = better board states
- Our system generates the successor board state for each legal move from the current state, then uses V to evaluate these states and picks the best

Slide 16

Designing a Learning System: Choosing the Target Function (cont)

- Continuing checkers example, choose a specific target function V

  For any board state b:

  1. $V(b) = 100$, if b is a final board state that is won
  2. $V(b) = -100$, if b is a final board state that is lost
  3. $V(b) = 0$, if b is a final board state that is drawn
  4. $V(b) = V(b')$, if b is not a final board state, where b' is the best final board state that can be achieved by playing optimally from b, assuming opponent plays optimally too.
- This recursive definition specifies V for every $b \in B$.

  However, V so defined is not efficiently computable, since for case 4 it requires searching every possible line of play till the end of the game.

  Such a definition is called *non-operational*; we need an *operational* definition – one that can be used to evaluate states/select moves in realistic times.
- Perfectly learning an operational form of V usually very difficult – in general only possible to learn an *approximation* of V.

  Call the function our system actually learns $\hat{V}$ to distinguish it from the ideal target function V

Slide 17

Designing a Learning System: Choosing a Representation for the Target Function

- Many possible representations for target function $\hat{V}$
  - lookup table for every board state
  - rules that match against features of board states
  - neural network trained on features of board states
  - ...
- Tradeoff between
  - expressivity of representation – allows closer approximation to V; and
  - volume of training data required – more expressive representations require more training data
- As a simple example, identify features:

  x_1: number of black pieces on board
  x_2: number of red pieces on board
  x_3: number of black kings on board
  x_4: number of red kings on board
  x_5: number of black pieces threatened by red
  x_6: number of red pieces threatened by black

  and let $\hat{V}$ be defined as a linear function of these features:

  $$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

  where $w_0 \dots w_6$ are weights to be learned by the learning algorithm.

Slide 18

Designing a Learning System: Choosing a Representation for the Target Function (cont)

Design of checkers learning program so far can be summarised as:

Task T : playing checkers

Performance measure P : percentage of games won in world tournament

Training experience E : games played against itself

Target function : $V : Board \rightarrow \mathcal{R}$

Target function representation :

$$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

Slide 19

Designing a Learning System:
Choosing a function approximation algorithm

- To learn $\hat{V}$ need a set of training examples of the form

$$\langle b, V_{train}(b) \rangle$$

where b is a description of a board state in terms of the features $x_1 \ldots x_6$, and $V_{train}(b)$ is the training value we wish to associate with that board state.
E.g.

$$\langle \langle x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 4, x_5 = 0, x_6 = 0 \rangle, +100 \rangle$$

- To estimate training values for intermediate board states, use current estimate of $\hat{V}$ and set

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

Note that $\hat{V}$ is being used to estimate training values that will be used to refine $\hat{V}$
  - the current estimate for $Successor(b)$ is being used to estimate the values for b
  - OK, since estimates will be quite accurate near end of game and accuracy will be iteratively propagated back to earlier board states by the algorithm.

Slide 20

Designing a Learning System:
Choosing a function approximation algorithm

- Use the training examples to update the weights in $\hat{V}$
  - initialise weights to random values or set all equal
- Common approach is to modify weights by minimising the squared error E between the training values and those predicted by the hypothesis $\hat{V}$.

$$E \equiv \sum_{\langle b, V_{train}(b) \rangle \in training\ examples} (V_{train}(b) - \hat{V}(b))^2$$

- One algorithm for this is Least Mean Square (LMS) algorithm.

LMS weight update rule

For each training example $\langle b, V_{train}(b) \rangle$
  - Use current weights to calculate $\hat{V}(b)$
  - For each weight w_i, update as

$$w_i \leftarrow w_i + \eta(V_{train}(b) - \hat{V}(b))x_i$$

where η is a small constant (e.g. 0.1) that controls size of weight update

Slide 21

Designing a Learning System: Choosing a function approximation algorithm (cont)

- By choosing the weight update formula

$$w_i \leftarrow w_i + \eta(V_{train}(b) - \hat{V}(b))x_i$$

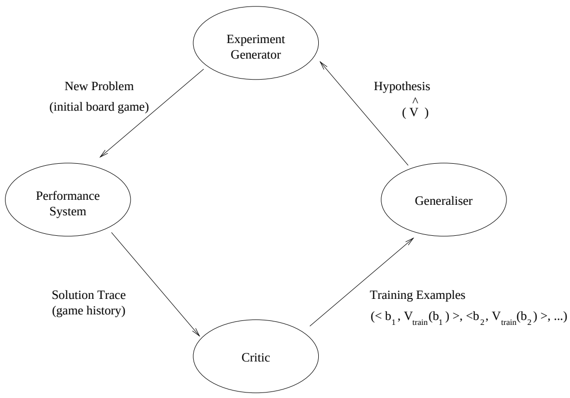
note:

- when $(V_{train}(b) - \hat{V}(b)) = 0$ no weight is changed
- when $(V_{train}(b) - \hat{V}(b)) > 0$, i.e. $\hat{V}(b)$ is too low, then w_i is increased proportionally to x_i – increases $\hat{V}(b)$, decreases E
- when $x_i = 0$ no change occurs to w_i, so weights are only updated for features which occur in training examples

In certain settings LMS can be proved to converge to least squared error approximation to V_{train} values.

Slide 22

Overall Design of Learning System



Performance System Performs the task (e.g. checkers) using learned target function ($\hat{V}$). I.e takes a new problem instance (new game) and proposes a solution (game history).

Critic Generates new training examples from solution trace, e.g. using $V_{train}(b) \leftarrow \hat{V}(Successor(b))$ estimates.

Generaliser Hypothesises target function ($\hat{V}$) from training examples, e.g. using LMS algorithm.

Experiment Generator Generates a new problem instance for the performance system, given the current hypothesis.

Slide 23

Summary

- Machine learning is about creating programs which improve their performance on a given task with experience
- Aside from performance alone, we may also be interested in discovering patterns in our data which we can understand
- ML is increasingly important in applications in all walks of life given the exponential increase in electronic data
- ML also of interest to theoretical computer scientists and to cognitive scientists interested in modelling human learning

Slide 24

Summary (cont)

- The general ML setting involves:
  - identifying the task, performance measure and experience (e.g. checkers, % games won, past games)
  - determining whether the training experience will be direct/indirect, the role of the teacher in training, the representativeness of the training data (e.g. games played against self)
  - choosing the target function to be learned, and if it is not practicably learnable, choosing an approximation of the ideal target function instead (e.g. $V : B \rightarrow \mathcal{R}$ instead of $ChooseMove : B \rightarrow M$)
  - choosing a representation of the target function which looks promising (e.g. some set of attributes describing an instance of interest – # red pieces, black pieces, kings, etc.)
  - choosing an algorithm which approximates the target function (e.g. LMS)

Slide 25

WEKA: Software for Machine Learning

- For labs and assignments we will be using Weka – a software platform for experimenting with and learning about machine learning algorithms (complements Witten & Frank book)
- Weka 3 is available from the University of Waikato in New Zealand at: `http://www.cs.waikato.ac.nz/ml/weka`
- If you have your own laptop you are advised to download and install a copy.
- Weka-3.4.4 is already available on the DCS Linux Desktop, under the directory `/usr/local/pkg/weka-3.4.4`
- Have a play! (see sample data files in data directory that comes with it)