21 Introduction to machine learning

Types of learning phenomena

Gaining knowledge	reading instruction
Discovering patterns	observing experiencing experimenting
Acquiring skills	practice adapting
Development	maturation growing
Evolution	

Machine learning

Building systems that improve with experience

Central question

How to acquire useful performance with reasonable amount of experience?

Answer

Not by magic!

There is a fundamental tradeoff between
- amount of experience
- amount of prior constraints

Strong mathematical limits
- No such thing as "universal" learning

21.1 Mathematical modeling

- Formalize learning environment
  - types of inputs/outputs to system
- Formalize measures of learning performance
  - rate of improvement
  - level of achievement
- Formalize language for expressing constraints and assumptions
- Design provably effective/efficient learning algorithms, or analyze capabilities of learning algorithms

E.g. formalizing learning environment

Data can be represented as a table with some columns representing input attributes, and some columns representing output attributes

E.g. formalizing measures of learning performance

Measure:

- rate of improvement
- level of achievement

Evaluating a model:

- divide data into training and testing part (9:1)
- train model on training part
- test on the testing part
- held-out or validation data, cross-validation

E.g. language for expressing constraints

- learning function: linear, quadratic
- learning Bayesian network
- learning decision tree
- learning rules, version space algorithm
- learning neural networks

21.2 Some successful applications of machine learning

- data mining: using historical data to improve decisions; e.g., medical records, corporate databases
- computer interpretation: applications we cannot program by hand; e.g., speech recognition, automated driving
- self-customizing programs: spam-filter, news reader that learns user interests

1 Checkers player

Arthur Samuel wrote a program that learned how to play checkers in the 50s.

How?

- Learn to evaluate board positions; weighted combination of board features (# black pieces, # red pieces, # black kings, etc.)
- Play against previous version of self
- Use evaluation of next state as target evaluation for current state, and adjust combination weights to make these evaluations more alike

  (E.g., if an optimal minimax path (after searching up to a certain depth) leads from situation x to y, then the values of the evaluation function for x and y should be the same. If they are not the same, we treat the value in y as the correct value (y is further from the root), and update the weights a bit so that the value of x is close to the value of y.)

- Win and loss values at leaves are backed up

Others

- Backgammon: Tesauro 80s 90s
- Chess: Baxter 98 (automatically improved rating from 1650 to 2150 in 3 days, playing 300 games on the web)

2 Zip code reader

In late-80's/early-90's Yann le Cun developed a system that automatically learned read zip codes on envelopes (US Postal service)

How?

- Train on large corpus of manually labeled examples

$$\begin{array}{cc} \text{image}_1 & \text{label}_1 \\ \text{image}_2 & \text{label}_2 \\ \vdots & \vdots \end{array}$$

- Use a neural net model to predict labels
- Adjust neural network parameters to minimize training prediction error

Others

- Cheque readers
- Handwriting recognition
- Face recognition
- Tracking

3 Speeding up automated problem solvers

Learn to speed up performance of a correctly working problem solver (much researched during the 80's, but less lately)

How?

- Learn search control
  - how to rank alternatives
  - how to prune alternatives
- Learn macro steps

Others

- Transaction processing
- Database query processing

4 Robot exploration

Autonomous exploration and mapping of an unknown environment

How?

- Remember landmarks, topology
- Explore unknown regions

5 Industrial plant optimization

Optimize operating parameters of a plant while in operation

- e.g. pulp mill:
  - roller pressure
  - conveyor rate

How?

- Very tricky
- Continuously explore parameter space

21.3 Fundamental types of learning problems

A learning system maps experience to base performance system

Set of base systems = set of hypotheses, or *hypothesis space*

1 Batch vs online

- Batch:
  - separate training then testing phase
  - pure exploration during training
- On-line:
  - no separate training phase
  - learn while doing
  - tradeoff between acting to do well vs. acting to gain information

2 Complete vs. partial vs. point-wise feedback

- Complete:
  - every item of experience evaluates every candidate base system
  - e.g.
    * base system = $f : X \to Y$ prediction rule
    * experience = $\langle x, y \rangle$ observation and label
  - feedback tells what best systems would have done on experience item
- Partial:
  - items of experience only evaluate a *subset* of possible candidates
  - e.g.
    * base system = controller $\pi : S \to A$
    * experience = $\langle s, a, r \rangle$
    * only evaluates controllers π that would have taken action a in state s

  - feeback only tells how well you did, not what best system would have done on experience item

- Pointwise:
  - each item of experience evaluates only one base system
  - e.g.
    * standard optimization

3 Passive vs. active learning

- Passive:
  - experience is given, we cannot chose the inputs; observation
- Active:
  - we can choose the input for which the output will be provided; experimentation

4 Stationary vs. non-stationary environment

- Stationary:
  - evaluation does not change over time
- Non-stationary:
  - evaluation changes over time

5 Acausal vs. causal learning

- Acausal:
  - actions do not affect future experience
- Causal:
  - actions do affect future experience

Readings

Dean, Allen, Aloimonos: Sections 5.1, 5.2
Russell and Norvig: Sections 18.1, 18.2, 18.5
Hastie, Tibshirani, Friedman: Chapter 1