Introduction to Machine Learning

CS 586
Machine Learning

Prepared by Jugal Kalita

With help from Alpaydin's *Introduction to Machine Learning* and Mitchell's *Machine Learning*

Machine Learning: Definition

- *Mitchell 1997:* A computer program R 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.
- *Alpaydin 2010:* Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Examples of Machine Learning Techniques and Applications

- Learning Association
- Learning how to classify
- Regression
- Unsupervised Learning
- Reinforcement Learning

Learning Association

- This is also called *(Market) Basket Analysis*.

- If people who buy X typically also buy Y, and if there is a customer who buys X and does not buy Y, he or she is a potential customer for Y.

- Learn *Association Rules*: Learn a conditional probability of the form $P(Y \mid X)$ where Y is the product we would like to condition on X, which is a product or a set of products the customer has already purchased.

Algorithms for Learning Association

- The challenge is how to find good associations fast when we have millions or even billions of records.
- Researchers have come up with many algorithms such as
  - Apriori: The best-known algorithm using breadth-first search strategy along with a strategy to generate candidates.
  - Eclat: Uses depth-first search and set intersection.
  - FP-growth: uses an extended prefix-tree to store the database in a compressed form. Uses a divide-and-conquer approach to decompose both the mining tasks and the databases.

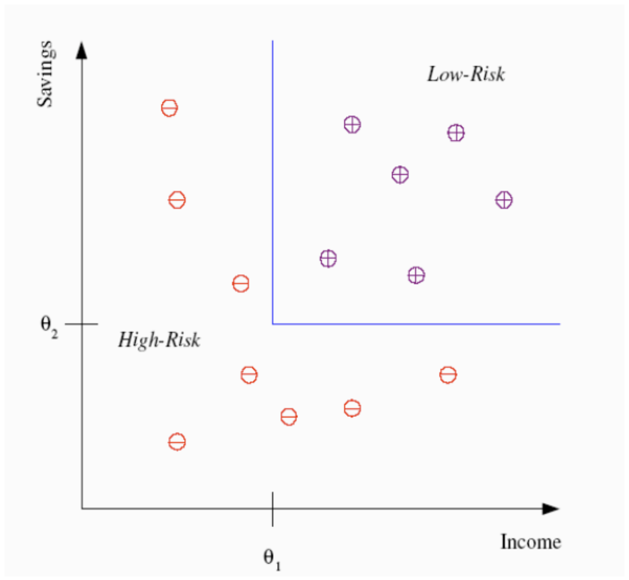
Applications of Association Rule Mining

- Market-basket analysis helps in cross-selling products in a sales environment.
- Web usage mining.
- Intrusion detection.
- Bioinformatics.

Algorithms for Classification

- Given a set of labeled data, learn how to classify unseen data into two or more classes.
- Many different algorithms have been used for classification. Here are some examples:
  - Decision Trees
  - Artificial Neural Networks
  - K-nearest Neighbor Algorithm
  - Kernel Methods such as Support Vector Machines
  - Bayesian classifiers

Example Training Dataset of Classification

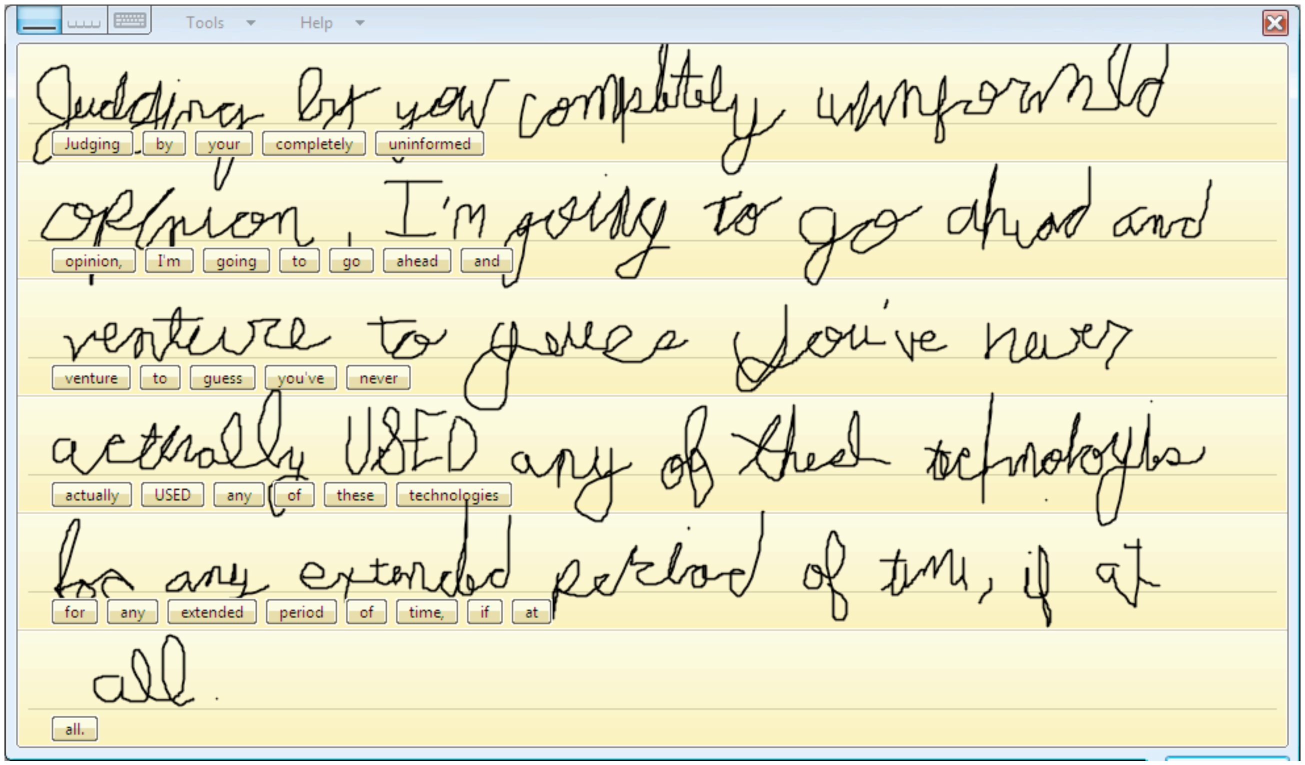


- Taken from Alpaydin 2010, Introduction to Machine Learning, page 6.
- We need to find the boundary (here two lines) between the data representing classes.

Applications of Classification

- Credit Scoring: Classify customers into high-risk and low-risk classes given amount of credit and customer information.
- Handwriting Recognition: Different handwriting styles, different writing instruments. 26 *2 = 52 classes for simple Roman alphabet. determining car license plates for violation. Language models may be necessary.
- Printed Character Recognition: OCR, determining car license plates for driving violations. Issues are fonts, spots or smudges, figures for OCR, weather conditions, occlusions, etc. Language models may be necessary.

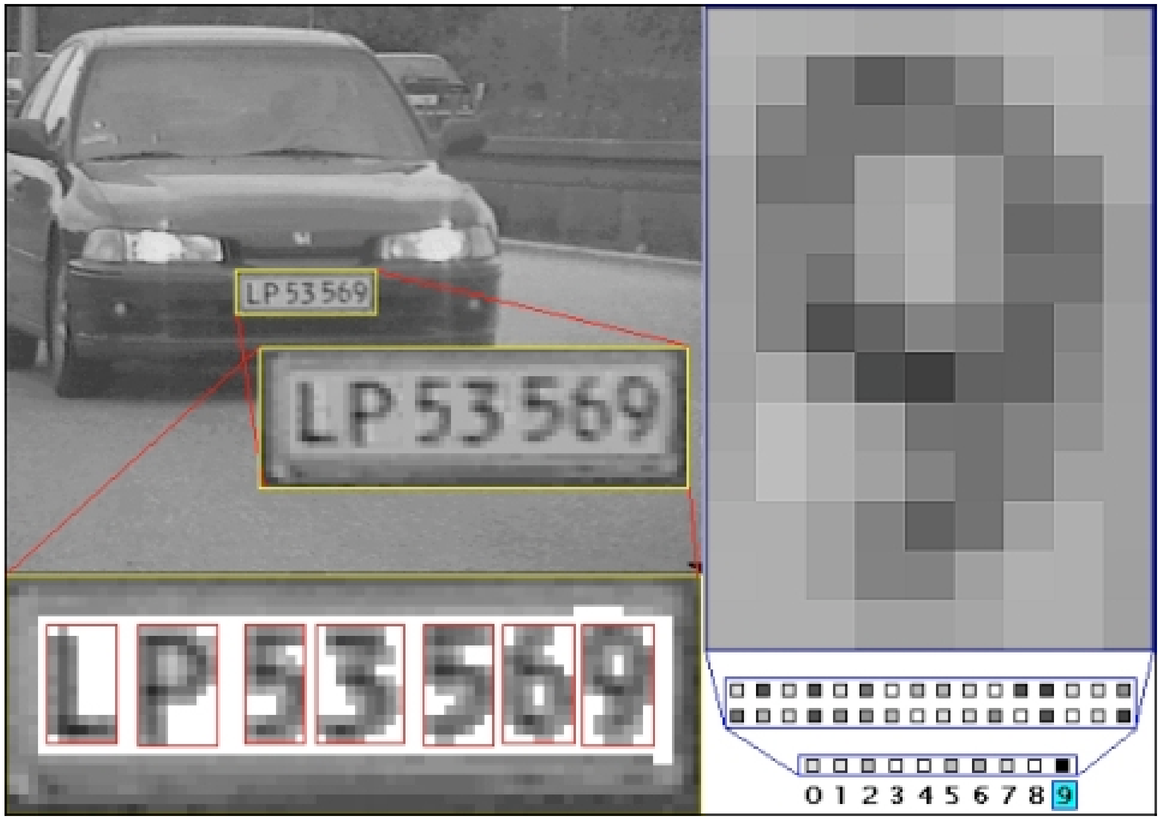
Handwriting Recognition on a PDA


Tools
Help
Judging by your completely uninformed
opinion, I'm going to go ahead and
venture to guess you've never
actually USED any of these technologies
for any extended period of time, if at
all.

Taken from

http://www.gottabemobile.com/forum/uploads/322/recognition.png.

License plate Recognition

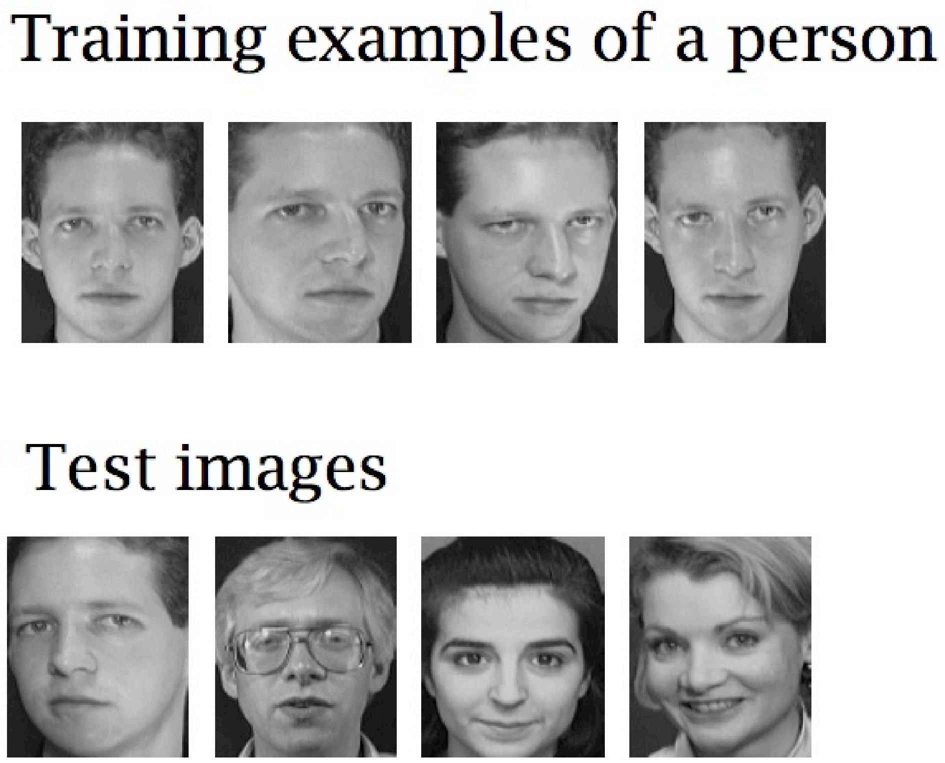


Taken from http://www.platerecognition.info/. This may be an image when a car enters a parking garage.

Applications of Classification (Continued)

- Face Recognition: Given an image of an individual, classify it into one of the people known. Each person is a class. Issues iclude poses, lighting conditions, occlusions, occlusion with glasses, makeup, beards, etc.
- Medical Diagnosis: The inputs are relevant information about the patient and the classes are the illnesses. Features include patient's personal information, medical history, results of tests, etc.
- Speech Recognition: The input consists of sound waves and the classes are the words that can be spoken. Issues include accents, age, gender, etc. Language models may be necessary in addition to the acoustic input.

Face Recognition

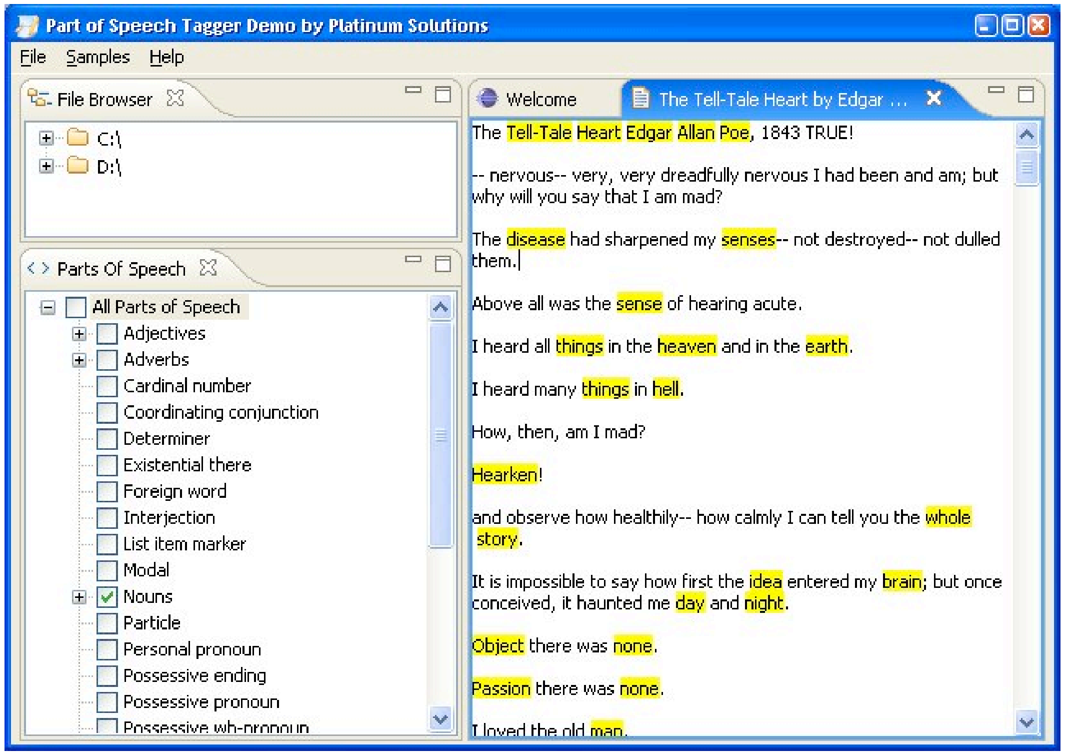


Taken from http://www.uk.research.att.com.

Applications of Classification (Continued)

- Natural Language Processing: Parts-of-speech tagging, parsing, machine translation, spam filtering, named entity recognition.
- Biometrics: Recognition or authentication of people using their physical and/or behavioral characteristics. Examples of characteristics: Images of face, iris and palm; signature, voice, gait, etc. Machine learning has been used for each of the modalities as well as to integrate information from different modalities.

POS tagging


Part of Speech Tagger Demo by Platinum Solutions
File
Samples
Help
File Browser
C:\
D:\
Parts Of Speech
Welcome
The Tell-Tale Heart by Edgar ...

- [ ] All Parts of Speech
  - [ ] Adjectives
  - [ ] Adverbs
  - [ ] Cardinal number
  - [ ] Coordinating conjunction
  - [ ] Determiner
  - [ ] Existential there
  - [ ] Foreign word
  - [ ] Interjection
  - [ ] List item marker
  - [ ] Modal
  - [✓] Nouns
  - [ ] Particle
  - [ ] Personal pronoun
  - [ ] Possessive ending
  - [ ] Possessive pronoun
  - [ ] Possessive wh-pronoun

The Tell-Tale Heart Edgar Allan Poe, 1843 TRUE!

-- nervous-- very, very dreadfully nervous I had been and am; but why will you say that I am mad?

The disease had sharpened my senses-- not destroyed-- not dulled them.

Above all was the sense of hearing acute.

I heard all things in the heaven and in the earth.

I heard many things in hell.

How, then, am I mad?

Hearken!

and observe how healthily-- how calmly I can tell you the whole story.

It is impossible to say how first the idea entered my brain; but once conceived, it haunted me day and night.

Object there was none.

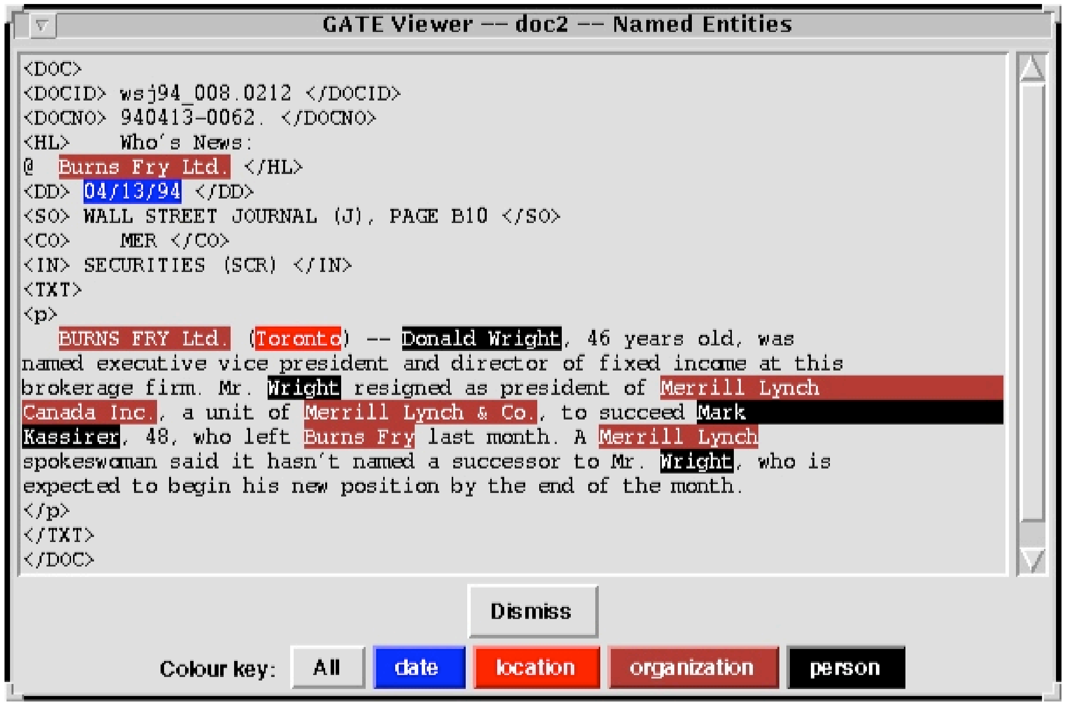
Passion there was none.

I loved the old man.

Taken from

http://blog.platinumsolutions.com/files/pos-tagger-screenshot.jpg.

Named Entity Recognition



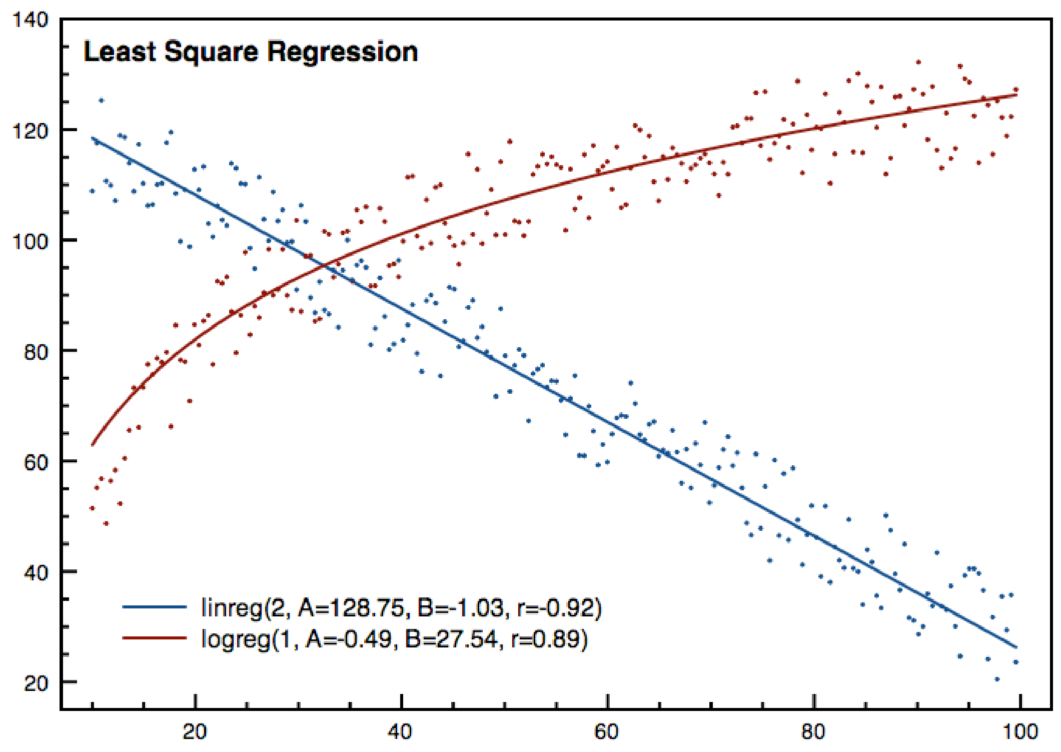
```
<DOC>
<DOCID> wsj94_008.0212 </DOCID>
<DOCNO> 940413-0062. </DOCNO>
<HL>    Who's News:
@  Burns Fry Ltd. </HL>
<DD> 04/13/94 </DD>
<SO> WALL STREET JOURNAL (J), PAGE B10 </SO>
<CO>    MER </CO>
<IN> SECURITIES (SCR) </IN>
<TXT>
<p>
   BURNS FRY Ltd. (Toronto) -- Donald Wright, 46 years old, was
named executive vice president and director of fixed income at this
brokerage firm. Mr. Wright resigned as president of Merrill Lynch
Canada Inc., a unit of Merrill Lynch & Co., to succeed Mark
Kassirer, 48, who left Burns Fry last month. A Merrill Lynch
spokeswoman said it hasn't named a successor to Mr. Wright, who is
expected to begin his new position by the end of the month.
</p>
</TXT>
</DOC>
```

Taken from http://www.dcs.shef.ac.uk/ hamish/IE/userguide/ne.jpg.

Regression

- Regression analysis includes any techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. Regression analysis helps us understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed.
- Regression can be linear, quadratic, higher polynomial, log-based and exponential functions, etc.

Regression



Taken from http://plot.micw.eu/uploads/Main/regression.png.

Applications of Regression

- Navigation of a mobile robot, an autonomous car: The output is the angle by which the steering wheel should be turned each time to advance without hitting obstacles and deviating from the root. The inputs are obtained from sensors on the car: video camera, GPS, etc. Training data is collected by monitoring the actions of a human driver.

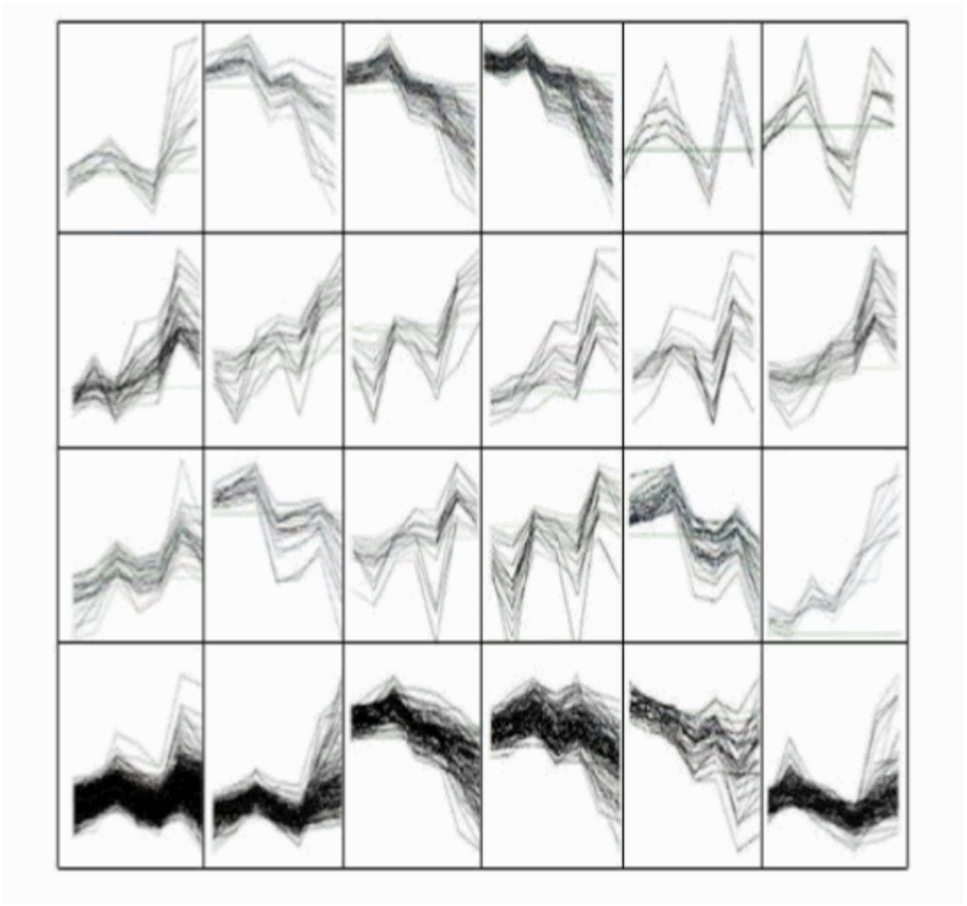
Unsupervised Learning

- In supervised learning, we learn a mapping from input to output by analyzing examples for which correct values are given by a supervisor or a teacher or a human being.
- Examples of supervised learning: Classification, regression.
- In unsupervised learning, there is no supervisor. We have the input data only. The aim is to find regularities in the input.

Unsupervised Learning: Clustering

- Bioinformatics: Clustering genes according to gene array expression data.
- Finance: Clustering stocks or mutual based on characteristics of company or companies involved.
- Document clustering: Cluster documents based on the words that are contained in them.
- Customer segmentation: Cluster customers based on demographic information, buying habits, credit information, etc. Companies advertise differently to different customer segments. Outliers may form niche markets.

Clustering



Taken from a paper by Das, Bhattacharyya and Kalita, 2009.

Applications of Clustering (continued)

- Image Compression using color clustering: Pixels in the image are represented as RGB values. A clustering program groups pixels with similar colors in the same group; such groups correspond to colors occurring frequently. Colors in a cluster are represented by a single average color. We can decide how many clusters we want to obtain the level of compression we want.
- High level image compression: Find clusters in higher level objects such as textures, object shapes,whole object colors, etc.

Reinforcement Learning

- In some applications, the output of the system is a sequence of *actions*.
- A single action is not important alone.
- What is important is the *policy* or the sequence of correct actions to reach the goal.
- In reinforcement learning, reward or punishment comes usually at the very end or infrequent intervals.
- The machine learning program should be able to assess the goodness of "policies"; learn from past good action sequences to generate a "policy".

Applications of Reinforcement Learning

- Game Playing: Games usually have simple rules and environments although the game space is usually very large. A single move is not of paramount importance; a sequence of good moves is needed. We need to learn good game playing policy.
- Example: Playing world class backgammon or checkers.

Applications of Reinforcement Learning (continued)

- Robot navigating in an environment: A robot is looking for a goal location to charge, or to pick up trash, to pour a liquid, to hold a container or object. At any time, the robot can move in many in one of a number of directions, or perform one of several actions. After a number of trial runs, it should learn the correct sequence of actions to reach the goal state from an initial state, and do it efficiently. Or it should learn what sequence of actions causes it to pick up most amount of trash.

Relevant Disciplines

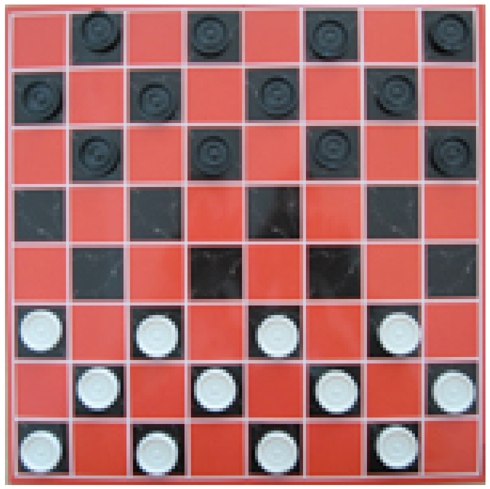
- Artificial intelligence
- Computational complexity theory
- Control theory
- Information theory
- Philosophy
- Psychology and neurobiology
- Statistics
- Bayesian methods
- $\cdots$

What is the Learning Problem?: A Specific Example in Details

- Reiterating our definition: Learning = Improving with experience at some task
  - Improve at task T
  - with respect to performance measure P
  - based on experience E.

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

Checkers



Taken from http://www.learnplaywin.net/checkers/checkers-rules.htm.

- 64 squares on board. 12 checkers for each player.
- Flip a coin to determine black or white.
- Use only black squares.
- Move forward one space diagonally and forward. No landing on an occupied square.

Checkers: Standard rules

- Players alternate turns, making one move per turn.
- A checker reaching last row of board is "crowned". A king moves the same way as a regular checker, except he can move forward or backward.
- One must jump if its possible. Jumping over opponent's checker removes it from board. Continue jumping if possible as part of the same turn.
- You can jump and capture a king the same way as you jump and capture a regular checker.
- A player wins the game when all of opponent's checkers are captured, or when opponent is completely blocked.

Steps in Designing a Learning System

- Choosing the Training Experience
- Choosing the Target Function: What should be learned?
- Choosing a Representation for the Target Function
- Choosing a Learning Algorithm

Type of Training Experience

- Direct or Indirect?
  - *Direct*: Individual board states and correct move for each board state are given.
  - *Indirect*: Move sequences for a game and the final result (win, loss or draw) are given for a number of games. How to assign credit or blame to individual moves is the *credit assignment* problem.

Type of Training Experience (continued)

- Teacher or Not?
  - Supervised: Teacher provides examples of board states and correct move for each.
  - Unsupervised: Learner generates random games and plays against itself with no teacher involvement.
  - Semi-supervised: Learner generates game states and asks the teacher for help in finding the correct move if the board state is difficult or confusing.

Type of Training Experience (continued)

- Is the training experience good?
  - Do the training examples represent the distribution of examples over which the final system performance will be measured?
  - Performance is best when training examples and test examples are from the same/a similar distribution.
  - Our checker player learns by playing against oneself. Its experience is indirect. It may not encounter moves that are common in human expert play.

Choose the Target Function

- Assume that we have written a program that can generate all legal moves from a board position.
- We need to find a target function $ChooseMove$ that will help us choose the best move among alternatives. This is the learning task.
- $ChooseMove : Board \rightarrow Move$. Given a board position, find the best move. Such a function is difficult to learn from indirect experience.
- Alternatively, we want to learn $V : Board \rightarrow \Re$. Given a board position, learn a numeric score for it such that higher score means a better board position. Our goal is to learn V.

A Possible (Ideal) Definition for Target Function

- 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 because it is not efficiently computable since it requires searching till the end of the game. We need an *operational* definition of V.

Choose Representation for Target Function

We need to choose a way to represent the ideal target function in a program.

- A table specifying values for each possible board state?
- collection of rules?
- neural network ?
- polynomial function of board features?
- ...

We use $\widehat{V}$ to represent the actual function our program will learn. We distinguish $\widehat{V}$ from the ideal target function V. $\widehat{V}$ is a *function approximation* for V.

A Representation for Learned Function

$$\begin{aligned}\hat{V} \;=\;& w_0 + w_1 \cdot x_1(b) + w_2 \cdot x_2(b) + w_3 \cdot x_3(b) + w_4 \cdot x_4(b)\\ & + w_5 \cdot x_5(b) + w_6 \cdot x_6(b)\end{aligned}$$

- $x_1(b)$: number of black pieces on board b
- $x_2(b)$: number of red pieces on b
- $x_3(b)$: number of black kings on b
- $x_4(b)$: number of red kings on b
- $x_5(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $x_6(b)$: number of black pieces threatened by red

It is a simple equation. Note a more complex representation requires more training experience to learn.

Specification of the Machine Learning Problem at this time

- Task T: Play checkers
- Performance Measure P: % of games won in world tournament
- Training Experience E: opportunity to play against self
- Target Function: $V : Board \rightarrow \Re$
- Target Function Representation:

$$\begin{aligned}\hat{V} &= w_0 + w_1 \cdot x_1(b) + w_2 \cdot x_2(b) + w_3 \cdot x_3(b) + w_4 \cdot x_4(b) \\ &\quad + w_5 \cdot x_5(b) + w_6 \cdot x_6(b)\end{aligned}$$

The last two items are design choices regarding how to implement the learning program. The first three specify the learning problem.

Generating Training Data

- To train our learning program, we need a set of training data, each describing a specific board state b and the training value $V_{train}(b)$ for b.
- Each training example is an ordered pair $\langle b, V_{train}(b) \rangle$.
- For example, a training example may be

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

  This is an example where black has won the game since $x_2 = 0$ or red has no remaining pieces.
- However, such clean values of $V_{train}(b)$ can be obtained only for board value b that are clear win, loss or draw.
- For other board values, we have to estimate the value of $V_{train}(b)$.

Generating Training Data (continued)

- According to our set-up, the player learns indirectly by playing against itself and getting a result at the very end of a game: win, loss or draw.
- Board values at the end of the game can be assigned values. How do we assign values to the numerous intermediate board states before the game ends?
- A win or loss at the end does not mean that every board state along the path of the game is necessarily good or bad.
- However, a very simple formulation for assigning values to board states works under certain situations.

Generating Training Data (continued)

- The approach is to assign the training value $V_{train}(b)$ for any intermediate board state b to be $\widehat{V}(Successor(b))$ where $\widehat{V}$ is the learner's current approximation to V (i.e., it uses the current weights w_i) and $Successor(b)$ is the next board state for which it's again the program's turn to move.

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

- It may look a bit strange that we use the current version of $\widehat{V}$ to estimate training values to refine the very same function, but note that we use the value of $Successor(b)$ to estimate the value of b.

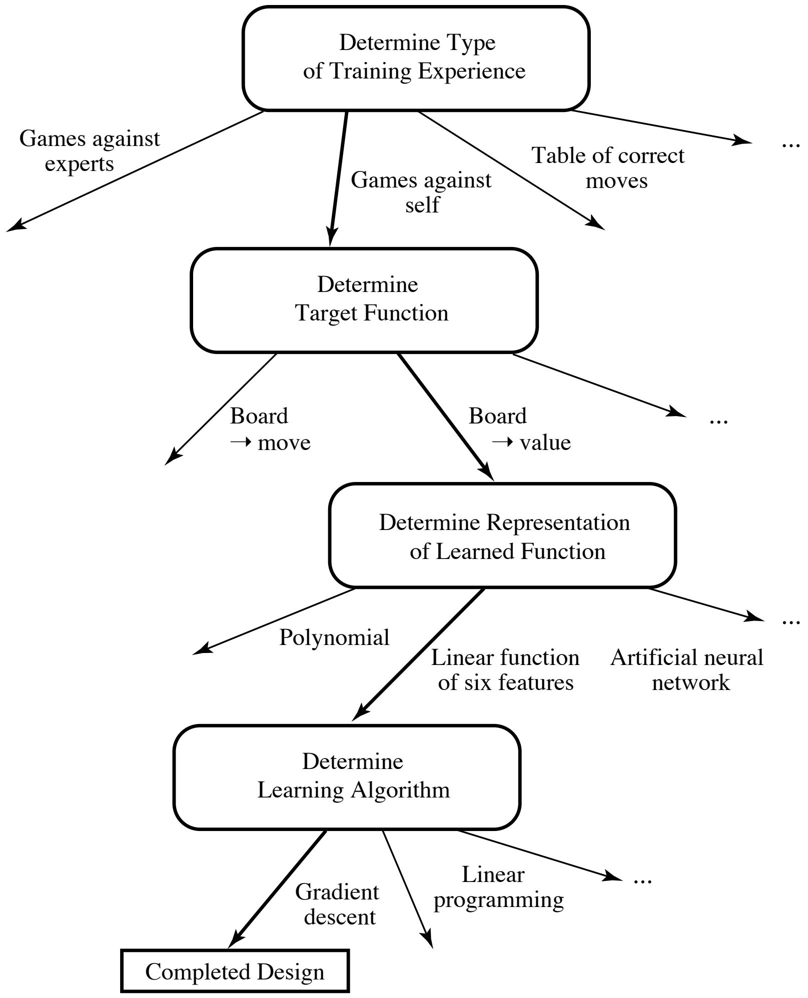
Training the Learner: Choose Weight Training Rule

LMS Weight update rule:

```
For each training example $b$ do
    Compute $error(b)$:
            $error(b) = V_{train}(b) - \widehat{V}(b)$
    For each board feature $x_i$, update weight $w_i$:
            $w_i \leftarrow w_i + \eta \cdot x_i \cdot error(b)$
    Endfor
Endfor
```

Here η is a small constant, say 0.1, to moderate the rate of learning.

Checkers 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

Taken from Page 13, Machine Learning by Tom Mitchell, 1997.

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?