Outline

- Why machine learning
- Some examples
- Relevant disciplines
- What is a well-defined learning problem
- Learning to play checkers
- Machine learning issues
- Best computer checkers player
Why Machine Learning?

- New kind of capability for computers
  - Database mining
    - Medical records → medical knowledge
  - Self customizing programs
    - Learning junk mail filter
  - Applications we can't program by hand
    - Autonomous driving
    - Speech recognition
- Understand human learning and teaching
- Time is right
  - Recent progress in algorithms and theory
  - Growing flood of online data
  - Computational power is available
  - Budding industry
**Example: Rule and Decision Tree Learning**

**Data:**

- **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: ?

- ... until **Patient103** time=n
  - Age: 23
  - FirstPregnancy: no
  - Anemia: no
  - Diabetes: no
  - PreviousPrematureBirth: no
  - Ultrasound: ?
  - Elective C-Section: no
  - Emergency C-Section: **Yes**

**Learned rule:**

If No previous vaginal delivery, and Abnormal 2nd Trimester Ultrasound, and Malpresentation at admission, and No Elective C-Section

Then Probability of Emergency C-Section is 0.6

Over training data: 26/41 = .634
Over test data: 12/20 = .600
Example: Neural Network Learning

- ALVINN (Autonomous Land Vehicle In a Neural Network) drives 70 mph on highways
  - [www.ri.cmu.edu/projects/project_160.html](http://www.ri.cmu.edu/projects/project_160.html)
Relevant Disciplines

- Artificial intelligence
- Bayesian methods
- Computational complexity theory
- Control theory
- Information theory
- Philosophy
- Psychology and neurobiology
- Statistics
What is the Learning Problem?

- Learning = Improving with experience at some task
  - 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
Learning to Play Checkers

- T: Play checkers
- P: Percent of games won in world tournament
- What experience?
- What exactly should be learned?
- How shall it be represented?
- What specific algorithm to learn it?
Type of Training Experience

- Direct or indirect?
- Teacher or not?

Problem
- Is training experience representative of performance goal?
Choose the Target Function

- ChooseMove : Board $\rightarrow$ Move ??
- $V : Board \rightarrow \mathbb{R}$ ??
- ...

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 a *draw*, then $V(b) = 0$
- If $b$ is 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

\[ \hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b) \]

- \( bp(b) \): 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

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

One rule for estimating training values:

- $V_{\text{train}}(b) \leftarrow \hat{V} (\text{Successor}(b))$
Choose Weight Tuning Rule

- LMS Weight update rule:
  - Do repeatedly:
    - Select a training example $b$ at random
      1. Compute $error(b)$:
         $$error(b) = V_{train}(b) - \hat{V}(b)$$
      2. 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 0.5, to moderate the rate of learning
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
Machine Learning Issues

- 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?
Best Computer Checkers Player

- Reigning champion: Chinook (1996)
  - [www.cs.ualberta.ca/~chinook](http://www.cs.ualberta.ca/~chinook)

- Search
  - Parallel alpha-beta

- Evaluation function
  - Linear combination of ~20 weighted features
  - Weights hand-tuned (learning ineffective)

- End-game database
- Opening book database
Checkers is Solved

- Chinook team weakly solves checkers (2007)
  - Ultra-weakly solved
    - Perfect play result is known, but not a strategy for achieving the result
  - Weakly solved
    - Both the result and a strategy for achieving the result from the start of the game are known
  - Strongly solved
    - Result computed for all possible game positions

- Computational proof
  - End-game database for all $\leq 10$ piece boards
  - Provably-correct search from start to $\leq 10$-piece board

- Result: Perfect checkers play results in a draw