CSCE 496/896 Lecture 0: Administrivia

Stephen D. Scott

August 23, 1999

Welcome to 496/896!

- This is section 2, Machine Learning
- Please check off your name on the roster, or write your name if you're not listed
  - If you write your name, indicate if you plan to register
- You should have 5 handouts:
  1. Course Information
  2. List of Topics
  3. Homework 0
  4. Prerequisite Test
  5. Copies of slides

CSCE 496/896 Lecture 1: Introduction

Stephen D. Scott
(Adapted from Tom Mitchell's slides)

August 23, 1999

Outline

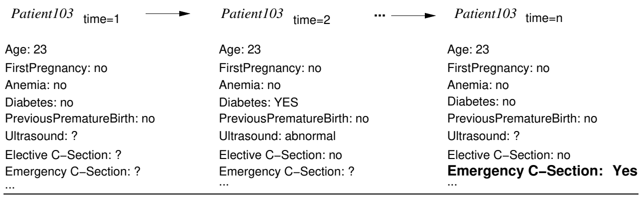
- Why Machine Learning?
- Relevant disciplines
- What is a well-defined learning problem?
- An example: learning to play checkers
- What issues arise in machine learning problems?

Why Machine Learning?

- New kind of capability for computers
  - Data mining
    - E.g. extracting new information from medical records, maintenance records, etc.
  - Self-customizing programs
    - A learning newsreader/browser that learns what you like and seeks it out
  - Applications we can't program by hand
    - E.g. speech recognition, autonomous driving
- Understanding human learning and teaching
  - Mature mathematical models might lend insight
- The time is right
  - Recent progress in algorithms and theory
  - Enormous amounts of data and applications
  - Substantial computational power
  - Budding industry

Typical Datamining Task

Data:



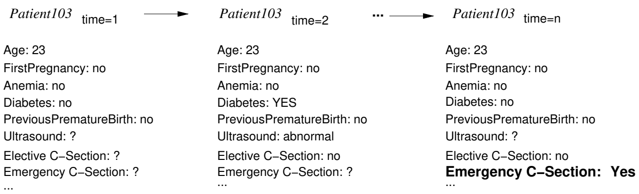
Given:

- 9714 patient records, each describing a pregnancy and birth
- Each patient record contains 215 features, some unspecified (we have little control over data)

Learn to predict classes of future patients at high risk for emergency cesarean section

Datamining Result

Data:

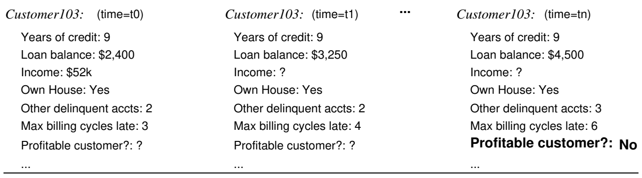


One of 18 learned rules:

```
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
```


Credit Risk Analysis

Data:



Rules learned:

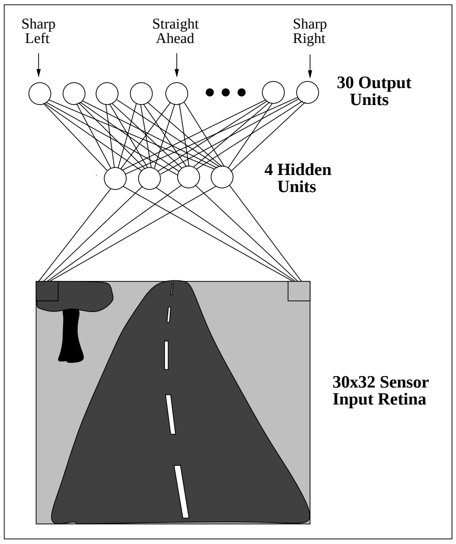
```
If   Other-Delinquent-Accounts > 2, and
     Number-Delinquent-Billing-Cycles > 1
Then Profitable-Customer? = No
     [Deny Credit Card application]

If   Other-Delinquent-Accounts = 0, and
     (Income > $30k)  OR  (Years-of-Credit > 3)
Then Profitable-Customer? = Yes
     [Accept Credit Card application]
```

Software that Customizes to User

E.g. http://www.wisewire.com

Problems Too Difficult to Program by Hand

ALVINN [Pomerleau] drives 70 mph on highways



Relevant Disciplines

- AI: Learning as a search problem, using prior knowledge and data to guide learning
- Bayesian methods (probability theory): Computing probabilities of hypotheses and labels
- Computational complexity theory: Bounds on inherent complexity of learning, e.g. time, data
- Control theory: Learning to control processes to optimize performance measures
- Information theory: Measuring information entropy and content
- Philosophy: Occam's razor (simplest explanation is best)
- Psychology and neurobiology: Practice improves performance, biological justification for artificial neural networks
- Statistics: Estimating generalization performance

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

Designing a Learning System

E.g. Learning to Play Checkers

Part of problem specification:

- T: Play checkers
- P: Percent of games won in world tournament

Within our control:

- 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?
  - Given board states + best move for that state
  - Given move sequences + outcome of game: Infer "goodness" of move X by whether game won or lost and X's contribution
- Teacher or not?
  - Teacher gives learner board states + best move
  - Learner asks teacher for best move for particular state
  - Learner plays against itself and has no teacher, only feedback from environment
  - If teacher exists, how nice is it?
- Is training experience representative of performance goal?
  - Will the data (games) seen in training reflect those seen in practice?

What Exactly Should be Learned?

- Going to learn a target function (target concept), but of what type, i.e. what is the concept class?
  - $ChooseMove : Board \rightarrow Move$
    (might work with direct info, difficult with indirect)
  - $V : Board \rightarrow \Re$
    (better choice for indirect info: learn value of board states)
  - Other possibilities

Possible Defn 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.

Gives correct results, but is not operational, i.e. not efficiently computable

Typically limit ourselves to an approximation $\hat{V}$ of V

Representing $\hat{V}$

I.e. the Hypothesis Class

- Could use collection of rules, neural network, polynomial function of board features, etc.

- Expressiveness of type of function must be chosen carefully [goodness of approx. vs. data requirements: e.g. interpolation]

Example 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 Learner's Training Examples

- For learner to approximate V with $\hat{V}$, need to give learner examples $\langle b, V_{train}(b)\rangle$, where $V_{train}(b)$ is the label of b

- For b s.t. game over, this is easy, e.g.

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

- What about intermediate board positions? One rule for estimating training values:

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

- Where do the examples come from?

Choose a Weight Training Rule (Learning Algorithm)

- Common goal: Minimize squared error:

$$(1) \quad \sum_{\text{training exs}} \left(V_{train}(b) - \hat{V}(b)\right)^2$$

- Popular algorithm: **LMS Weight update rule**
  - Initialize weights
  - 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 $f_i \in \{x_1, \ldots, x_6\}$, 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

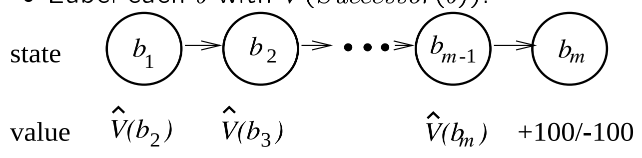
- Intuition: Larger $|error| \Rightarrow$ greater change, larger $f_i \Rightarrow$ greater contribution to error

Putting it Together

- Initialize $\hat{V}$'s weights

- Use $\hat{V}$ to play games against self, output sequence of board states for each game

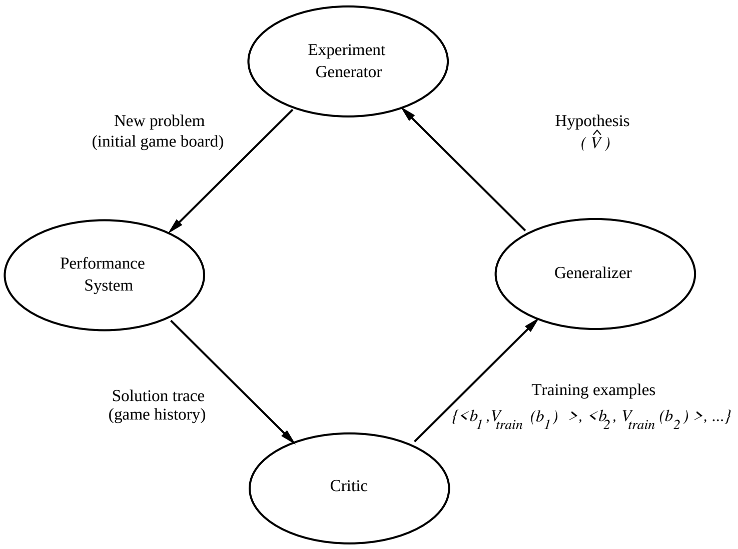
- Label each b with $\hat{V}(Successor(b))$:



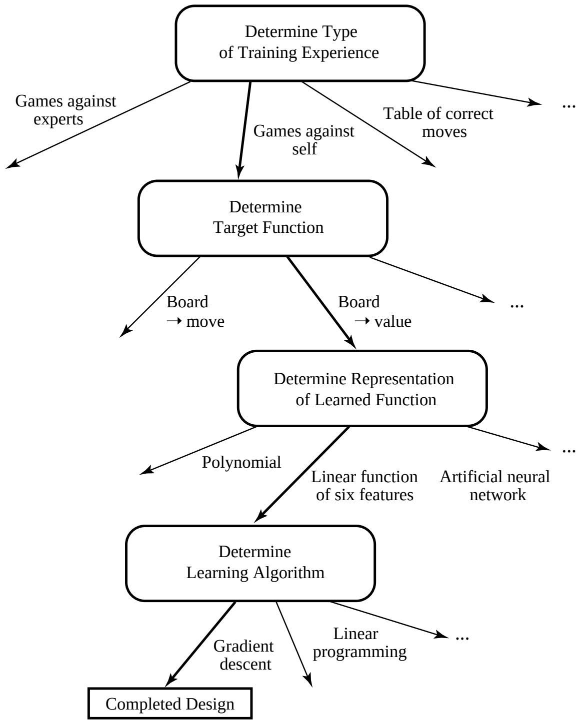
- Learn new weights, yielding new $\hat{V}$

- Start new set of games

Putting it Together
(cont'd)



Design Choices



Some Issues in Machine Learning

- What algorithms can approximate functions well (and when)? [E.g. LMS minimizes Eq. 1 in some situations]
- How does number of training examples influence accuracy? [How well will it generalize given a training set of particular size?]
- How does complexity of hypothesis representation impact it? [Too complex ⇒ may overfit, insufficiently complex ⇒ poor approximator; also influences data requirements; e.g. interpolation]
- How does noisy data [noise in examples and labels] influence accuracy?
- What are the theoretical limits of learnability? [Worst-case data and time requirements, hardness results]

Some Issues in Machine Learning (cont'd)

- How can prior knowledge of learner help? [Choose set of candidate hypotheses, e.g. appropriate form of $\hat{V}$; choose search procedure]
- What clues can we get from biological learning systems? [E.g. artificial neural networks, evolutionary algorithms]
- How can systems alter their own representations? [E.g. choosing from among different types of classifiers, learning to learn]