

## CSCE 478/878 Lecture 0: Administrivia

Stephen D. Scott

August 22, 2006

1

### Welcome to 478/878!

- 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 2 handouts:
  1. Syllabus
  2. Copies of slides
- In addition, check these out on the web (mandatory!):
  1. Homework 0
  2. Prerequisite Test

2

## CSCE 478/878 Lecture 1: Introduction

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

August 22, 2006

3

### Outline

- What is Machine Learning?
- 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?

4

### What is Machine Learning?

- Developing algorithms to learn from experience
- In particular, to generalize from what has already been seen
- Algorithm sees training data in the form of attribute-label pairs, e.g.
  - Attribute 1, called "Humidity," can have a value of "Normal" or "High"
  - Attribute 2, called "Air temperature," is real-valued
  - ... and so on ...
  - The label in this case can then be "Yes" or "No," indicating whether I would want to play tennis on that day
- The learning algorithm gets several such labeled examples and infers a hypothesis, which is a function that takes new (unlabeled) examples and predicts a label for them

5

### Why Machine Learning?

- (Somewhat) new kind of capability for computers
  - Data mining
    - \* E.g. extracting new information from medical records, maintenance records, etc.
  - Self-customizing programs
    - \* A spam filter that learns what is spam and what is not, specific to your user profile
  - Applications we can't program by hand
    - \* E.g. speech recognition, autonomous driving
- Understanding human learning and teaching
  - Mature mathematical models might lend insight into biological ones
- The time is right
  - Significant progress in algorithms and theory
  - Enormous amounts of data and applications
  - Substantial computational power
  - Growing industry

6

### Typical Datamining Task

Data:

Patient103 time=1	Patient103 time=2	Patient103 time=n
Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C-Section: ?	Elective C-Section: no	Elective C-Section: no
Emergency C-Section: ?	Emergency C-Section: ?	<b>Emergency C-Section: Yes</b>

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

7

### Datamining Result

Data:

Patient103 time=1	Patient103 time=2	Patient103 time=n
Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C-Section: ?	Elective C-Section: no	Elective C-Section: no
Emergency C-Section: ?	Emergency C-Section: ?	<b>Emergency C-Section: Yes</b>

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

Accuracy over training data:  $26/41 = .63$ ,  
Accuracy over independent test data:  
 $12/20 = .60$

8

### Credit Risk Analysis

Data:

Customer103: (time=0)	Customer103: (time=t1)	Customer103: (time=tn)
Years of credit: 9	Years of credit: 9	Years of credit: 9
Loan balance: \$2,400	Loan balance: \$3,250	Loan balance: \$4,500
Income: \$52k	Income: ?	Income: ?
Own House: Yes	Own House: Yes	Own House: Yes
Other delinquent accts: 2	Other delinquent accts: 2	Other delinquent accts: 3
Max billing cycles late: 3	Max billing cycles late: 4	Max billing cycles late: 6
Profitable customer?: ?	Profitable customer?: ?	<b>Profitable customer?: No</b>

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]

9

### Software that Customizes to User

- Spam filtering has asymmetric costs
  - Important to filter out spam, but more important to avoid filtering out "ham" (legitimate email)
    - \* E.g. don't want mass email from Chancellor Perlman filtered out
  - Thus general-purpose spam detection rules can be applied, but have to be cautious on how aggressive they are
    - \* E.g. What if the email with "Subject: Viagra" was from your pharmacist?
- Since there is no single set of rules that work universally well, a spam filter needs to **learn** what is spam and what is not, specific to your user profile
  - Adaptive filtering also helps keep up with spammers' new techniques

```
cse> cat /etc/motd
... If you receive spam mail on your account
please bounce a copy to "spam@cse-mail.unl.e
to help train our Bayesian spam filters. ...
```

- Can also use learning to beat the spam filters :-(

10

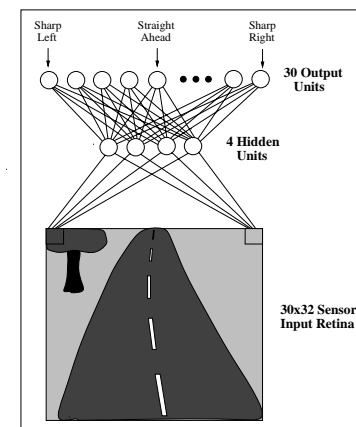
### Software that Customizes to User (cont'd)

- Google News personalized to your tastes ("Don't show me items like this.")
- Trainable voice-recognition systems

11

### Problems Too Difficult to Program by Hand

ALVINN [Pomerleau] drives 70 mph on highways



12

### Relevant Disciplines

- AI: Learning as a search problem, using prior knowledge and data to guide learning
- Bayesian methods (probability theory): Computing probabilities of hypotheses (functions) 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

13

### 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

14

### 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?

15

### 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?

16

### What Exactly Should be Learned?

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

17

### 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 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  $V$ , aka hypothesis

18

**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

$bp$ , etc. are features;  $w_0$ , etc. represent hypothesis

**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  $b$  (desired value of  $\hat{V}(b)$ )
- For  $b$  such that game over, this is easy, e.g.  $\langle \langle bp = 3, rp = 0, bk = 1, rk = 0, bt = 0, rt = 0 \rangle, +100 \rangle$
- 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:

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

- 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  $f_i \in \{bp, rp, \dots, rt\}$ , update weight  $w_i$ :

$$w_i \leftarrow w_i + c \cdot f_i \cdot error(b)$$

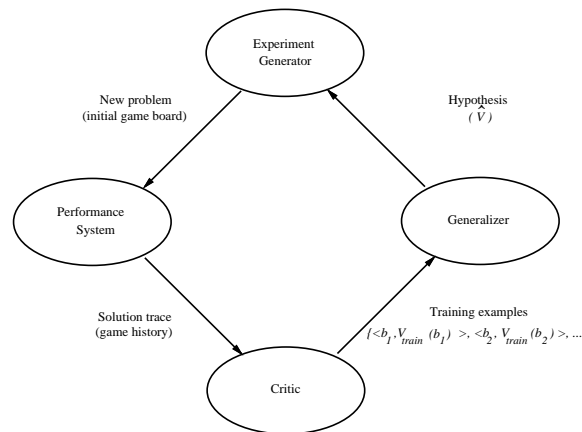
- $c$  is some small constant, say 0.1, to moderate the rate of learning
- Intuition: Larger  $|error| \Rightarrow$  greater change, larger  $f_i \Rightarrow$  greater contribution to error

**Putting it Together**

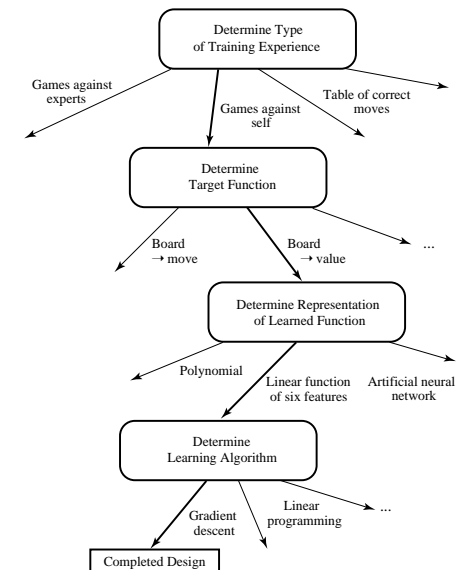
- Initialize  $\hat{V}$ 's weights
- Use  $\hat{V}$  to play games against self, output sequence of board states for each game
- Label each  $b$  with  $\hat{V}(Successor(b))$ :
 

state	$b_1$	$\rightarrow$	$b_2$	$\rightarrow$	$\dots$	$\rightarrow$	$b_{m-1}$	$\rightarrow$	$b_m$
value	$\hat{V}(b_2)$		$\hat{V}(b_3)$				$\hat{V}(b_m)$		+100/-100
- 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  $\Rightarrow$  may overfit, insufficiently complex  $\Rightarrow$  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]

25

### Some Issues in Machine Learning (cont'd)

- How can prior knowledge of learner help? [Choose set of candidate hypotheses, e.g. appropriate form of  $\mathcal{V}$ ; 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]

26