Outline

- Why Machine Learning?
- What is a well-defined learning problem?
- An example: learning to play checkers
- What questions should we ask about Machine Learning?

Why Machine Learning

- Recent progress in algorithms and theory
- Growing flood of online data
- Computational power is available
- Budding industry

Three niches for machine learning:

- Data mining: using historical data to improve decisions
  - medical records → medical knowledge
- Software applications we can’t program by hand
  - autonomous driving
  - speech recognition
- Self customizing programs
  - Newsreader that learns user interests

Typical Datamining Task

Data:

<table>
<thead>
<tr>
<th>Patient101 time=1</th>
<th>Patient102 time=2</th>
<th>Patient103 time=n</th>
</tr>
</thead>
</table>

Given:

- 9714 patient records, each describing a pregnancy and birth
- Each patient record contains 215 features

Learn to predict:

- Classes of future patients at high risk for Emergency Cesarean Section

Datamining Result

Data:

<table>
<thead>
<tr>
<th>Patient101 time=1</th>
<th>Patient102 time=2</th>
<th>Patient103 time=n</th>
</tr>
</thead>
</table>

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:**

<table>
<thead>
<tr>
<th>Customer103: (time=t0)</th>
<th>Customer103: (time=t1)</th>
<th>Customer103: (time=tn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years of credit: 9</td>
<td>Years of credit: 9</td>
<td>Years of credit: 9</td>
</tr>
<tr>
<td>Loan balance: $2,400</td>
<td>Loan balance: $3,250</td>
<td>Loan balance: $4,500</td>
</tr>
<tr>
<td>Income: $52k</td>
<td>Income: $75k</td>
<td>Income: $75k</td>
</tr>
<tr>
<td>Own House: Yes</td>
<td>Own House: Yes</td>
<td>Own House: Yes</td>
</tr>
<tr>
<td>Other delinquent accts: 2</td>
<td>Other delinquent accts: 3</td>
<td>Other delinquent accts: 3</td>
</tr>
<tr>
<td>Max billing cycles late: 3</td>
<td>Max billing cycles late: 4</td>
<td>Max billing cycles late: 6</td>
</tr>
<tr>
<td>Profitable customer?: ?</td>
<td>Profitable customer?: ?</td>
<td>Profitable customer?: No</td>
</tr>
</tbody>
</table>

**Rules learned from synthesized data:**

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]

### Other Prediction Problems

**Customer purchase behavior:**

<table>
<thead>
<tr>
<th>Customer103: (time=t0)</th>
<th>Customer103: (time=t1)</th>
<th>Customer103: (time=tn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex: M</td>
<td>Sex: M</td>
<td>Sex: M</td>
</tr>
<tr>
<td>Age: 53</td>
<td>Age: 53</td>
<td>Age: 53</td>
</tr>
<tr>
<td>Income: $50k</td>
<td>Income: $50k</td>
<td>Income: $50k</td>
</tr>
<tr>
<td>Own House: Yes</td>
<td>Own House: Yes</td>
<td>Own House: Yes</td>
</tr>
<tr>
<td>MS Products: Word</td>
<td>MS Products: Word</td>
<td>MS Products: Word</td>
</tr>
<tr>
<td>Computer: Pentium</td>
<td>Computer: Pentium</td>
<td>Computer: Pentium</td>
</tr>
<tr>
<td>Purchase Excel?: ?</td>
<td>Purchase Excel?: ?</td>
<td>Purchase Excel?: Yes</td>
</tr>
</tbody>
</table>

**Customer retention:**

<table>
<thead>
<tr>
<th>Customer103: (time=t0)</th>
<th>Customer103: (time=t1)</th>
<th>Customer103: (time=tn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex: M</td>
<td>Sex: M</td>
<td>Sex: M</td>
</tr>
<tr>
<td>Age: 53</td>
<td>Age: 53</td>
<td>Age: 53</td>
</tr>
<tr>
<td>Income: $50k</td>
<td>Income: $50k</td>
<td>Income: $50k</td>
</tr>
<tr>
<td>Own House: Yes</td>
<td>Own House: Yes</td>
<td>Own House: Yes</td>
</tr>
<tr>
<td>Checking: $0</td>
<td>Checking: $0</td>
<td>Checking: $0</td>
</tr>
<tr>
<td>Savings: $0</td>
<td>Savings: $0</td>
<td>Savings: $0</td>
</tr>
<tr>
<td>Current-customer?: yes</td>
<td>Current-customer?: yes</td>
<td>Current-customer?: No</td>
</tr>
</tbody>
</table>

**Process optimization:**

<table>
<thead>
<tr>
<th>Product172: (time=t0)</th>
<th>Product172: (time=t1)</th>
<th>Product172: (time=tn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage: mix</td>
<td>Stage: mix</td>
<td>Stage: mix</td>
</tr>
<tr>
<td>Mixing speed: 60rpm</td>
<td>Mixing speed: 49rpm</td>
<td>Mixing speed: 50rpm</td>
</tr>
<tr>
<td>Viscosity: 1.3</td>
<td>Viscosity: 1.0</td>
<td>Viscosity: 1.2</td>
</tr>
<tr>
<td>Fat content: 15%</td>
<td>Fat content: 12%</td>
<td>Fat content: 12%</td>
</tr>
<tr>
<td>Density: 2.8</td>
<td>Density: 1.0</td>
<td>Density: 1.1</td>
</tr>
<tr>
<td>Spectral peak: 2800</td>
<td>Spectral peak: 2800</td>
<td>Spectral peak: 2800</td>
</tr>
<tr>
<td>Product underweight?: ?</td>
<td>Product underweight?: ?</td>
<td>Product underweight?: Yes</td>
</tr>
</tbody>
</table>

Problems Too Difficult to Program by Hand

ALVINN [Pomerleau] drives 70 mph on highways

### Software that Customizes to User

**Welcome to the Windy City**

Wine Spectator. A wine country magazine bringing you the latest news from the world's leading wine regions. Whether you're looking for the latest reviews, travel tips, or are just simply curious about the world of fine wines, Wine Spectator is the place to be. From the vineyards to the tasting rooms, we cover it all. And now, thanks to our new wine glasses, you can enjoy the best of both worlds. So sit back, relax, and let the good times roll.

http://www.wisewire.com
Where Is this Headed?

Today: tip of the iceberg

- First-generation algorithms: neural nets, decision trees, regression ...
- Applied to well-formatted database
- Budding industry

Opportunity for tomorrow: enormous impact

- Learn across full mixed-media data
- Learn across multiple internal databases, plus the web and newsfeeds
- Learn by active experimentation
- Learn decisions rather than predictions
- Cumulative, lifelong learning
- Programming languages with learning embedded?

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?
A problem: is training experience representative of performance goal?

Choose the Target Function

• \( \text{ChooseMove} : \text{Board} \to \text{Move} \) ??
• \( V : \text{Board} \to \mathbb{R} \) ??
• ...

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 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

Choose Representation for Target Function

• collection of rules?
• neural network ?
• polynomial function of board features?
• ...

A Representation for Learned Function

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

- \( bp(b) \): 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:

- \( V_{\text{train}}(b) \leftarrow V(\text{Successor}(b)) \)

Choose Weight Tuning Rule

LMS Weight update rule:

Do repeatedly:
- Select a training example \( b \) at random
  1. Compute error \( (b) \):
     \[ \text{error}(b) = V_{\text{train}}(b) - \hat{V}(b) \]
  2. For each board feature \( f_i \), update weight \( w_i \):
     \[ w_i \leftarrow w_i + c \cdot f_i \cdot \text{error}(b) \]

\( c \) is some small constant, say 0.1, to moderate the rate of learning

Design Choices

Determine Type of Training Experience
- Games against expert
- 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
- Linear programming
- Gradient descent
- Linear programming

Completed Design
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?