Lecture 7: Intro to Machine Learning

Rachel Greenstadt
November 10, 2010
Reminders

- Machine Learning exercise out today
- We’ll go over it
- Due 11/24
Machine Learning

- Definition: the study of computer algorithms that improve automatically through experience

- Formally:
  - Improve at task T
  - with respect to performance measure P
  - based on experience E

- Example: Learning to play Backgammon
  - T: play backgammon
  - P: number of games won
  - E: data about previously played games
Where is ML useful?

• Self-customizing software
  • spam filters, learning user preferences
• Data mining
  • medical records, credit fraud
• Software that can’t be written by hand
  • speech recognition, autonomous driving
• Other examples?
Learning agents
Learning element

• Design of a learning element is affected by
  – Which components of the performance element are to be learned
  – What feedback is available to learn these components
  – What representation is used for the components

• Type of feedback:
  – Supervised learning: correct answers for each example
  – Unsupervised learning: correct answers not given
  – Reinforcement learning: occasional rewards
Inductive Learning

• Supervised
  • “Teacher” provides labeled examples
  • Opposite of unsupervised, e.g. clustering

• Inductive Inference
  • Given: samples of an unknown function $f$
    • $(x, f(x))$ pairs
  • Goal: find a function $h$ that approximates $f$
Machine Learning terms

• Supervised examples

• Training set : set of examples used to improve the algorithm

• Test set : set of examples set aside to evaluate the algorithm based on performance measure
Inductive Learning

• Construct/adjust \( h \) to agree with \( f \) on training set (the supervised examples)

• (\( h \) is consistent if it agrees with \( f \) on all examples)

• E.g. curve-fitting
Inductive Learning

- Construct/adjust $h$ to agree with $f$ on training set (the supervised examples)
- ($h$ is consistent if it agrees with $f$ on all examples)
- E.g. curve-fitting
Inductive Learning

- Construct/adjust \( h \) to agree with \( f \) on training set (the supervised examples)
- \( h \) is consistent if it agrees with \( f \) on all examples
- E.g. curve-fitting
Inductive Learning

- Construct/adjust $h$ to agree with $f$ on training set (the supervised examples)
- ($h$ is consistent if it agrees with $f$ on all examples)
- E.g. curve-fitting
Inductive Learning

• Construct/adjust $h$ to agree with $f$ on training set (the supervised examples)

• ($h$ is consistent if it agrees with $f$ on all examples)

• E.g. curve-fitting

Occam’s razor: prefer the simplest hypothesis consistent with the data
Induction Task Example

- Given: 9714 patient records, each describing a pregnancy and a birth
- Each record contains 215 features
- Learn to predict which future patients are at high risk of C-section
Induction Task Example

Data:

Patient103  time=1 → Patient103  time=2 → ... → Patient103  time=n

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: no
PreviousPrematureBirth: no
Ultrasound: ?
Elective C-Section: ?
Emergency C-Section: ?

...  

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: YES
PreviousPrematureBirth: no
Ultrasound: abnormal
Elective C-Section: no
Emergency C-Section: ?

...  

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: no
PreviousPrematureBirth: no
Ultrasound: ?
Elective C-Section: no
Emergency C-Section: Yes

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
Decision Trees: PlayTennis
Decision Trees

- **Representation:**
  - Each internal node tests an attribute
  - Each branch is an attribute value
  - Each leaf assigns a classification
Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless $f$ nondeterministic in $x$) but it probably won't generalize to new examples

- Prefer to find more compact decision trees
Hypothesis spaces

How many distinct decision trees with \( n \) Boolean attributes?
= number of Boolean functions
= number of distinct truth tables with \( 2^n \) rows = \( 2^{2^n} \)

• E.g., with 6 Boolean attributes, there are \( 18,446,744,073,709,551,616 \) trees

How many purely conjunctive hypotheses (e.g., \( \text{Hungry} \land \neg \text{Rain} \))?
• Each attribute can be in (positive), in (negative), or out
  \( \Rightarrow 3^n \) distinct conjunctive hypotheses
• More expressive hypothesis space
  – increases chance that target function can be expressed
  – increases number of hypotheses consistent with training set
  \( \Rightarrow \) may get worse predictions
Decision tree learning

• Aim: find a small tree consistent with the training examples
• Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```plaintext
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value $v_i$ of best do
            examples$_i$ ← {elements of examples with best = $v_i$}
            subtree ← DTL(examples$_i$, attributes − best, MODE(examples))
            add a branch to tree with label $v_i$ and subtree subtree
        return tree
```
Choosing an attribute

• Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

• *Patrons*? is a better choice
Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)
Attribute-based representations

- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_5$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_6$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_7$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_8$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_9$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

Exercise: Build a consistent decision tree from the examples
Decision trees

- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:
Entropy

• $I(P(v_1),...,P(v_n)) = \sum_{i=1}^{n} (-P(v_i)\log_2 P(v_i))$

• Tossing a fair coin
  – $I(1/2,1/2) - \frac{1}{2} \log_2 1/2 - \frac{1}{2}\log_2 1/2 = 1$ bit
  – If we weight the coin, information gain by tossing is reduced
Information gain

- A chosen attribute $A$ divides the training set $E$ into subsets $E_1, \ldots, E_v$ according to their values for $A$, where $A$ has $v$ distinct values.

$$\text{remainder}(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Information Gain (IG) or reduction in entropy from the attribute test:

$$IG(A) = I\left(\frac{p}{p + n}, \frac{n}{p + n}\right) - \text{remainder}(A)$$

- $\text{remainder}(A)$ is the remaining uncertainty after splitting on the attribute

- Choose the attribute with the largest IG
Information gain

For the training set, $p = n = 6, I(6/12, 6/12) = 1$ bit

Consider the attributes *Patrons* and *Type* (and others too):

\[
IG(Patrons) = 1 - \left[ \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6}) \right] = 0.0541 \text{ bits}
\]

\[
IG(Type) = 1 - \left[ \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) \right] = 0 \text{ bits}
\]

*Patrons* has the highest IG of all attributes and so is chosen by the DTL algorithm as the root
Example contd.

- Decision tree learned from the 12 examples:

- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data
Build a decision tree

• Whether or not to move forward at an intersection, given the light has just turned green.
A decision tree for deciding whether to move forward at a traffic intersection, given a green light.
Performance measurement

• How do we know that $h \approx f$?
  1. Use theorems of computational/statistical learning theory
  2. Try $h$ on a new test set of examples
     (use same distribution over example space as training set)

**Learning curve** = % correct on test set as a function of training set size
Question

• Suppose we generate a training set from a decision tree and then apply decision tree learning to that training set. Is it the case that the learning algorithm will eventually return the correct tree as the training set goes to infinity? Why or why not?
Summary

• Learning needed for unknown environments, lazy designers
• Learning agent = performance element + learning element
• For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
• Decision tree learning using information gain
• Learning performance = prediction accuracy measured on test set
Bayesian Learning

• Conditional probability
• Bayesian classifiers
• Assignment
• Information Retrieval performance measures
Bayes’ Theorem

• If \( P(h) > 0 \), then
  • \( P(h|d) = \frac{[P(d|h)P(h)]}{P(d)} \)

• Can be derived from conditional probability
  • \( P(A|B) = \frac{P(A \cap B)}{P(B)} \)
  • \( P(d|h) \) is \textit{likelihood}, \( P(h|d) \) is \textit{posterior}, \( P(h) \) is \textit{prior}, \( P(d) \) is \( \alpha \)
Bayesian Exercises

• \( P(h|d) = \frac{[P(d|h)P(h)]}{P(d)} \)

• A patient takes a lab test and the result comes back positive. The test has a false negative rate of 2% and false positive rate of 2%. Furthermore, 0.5% of the entire population have this cancer. What is the probability of cancer if we know the test result is positive?

• A math teacher gave her class two tests. 25% of the class passed both tests and 42% of the class passed the first test. What is the probability that a student who passed the first test also passed the second test?
Bayesian Learning Example

Suppose there are five kinds of bags of candies:

10% are $h_1$: 100% cherry candies
20% are $h_2$: 75% cherry candies + 25% lime candies
40% are $h_3$: 50% cherry candies + 50% lime candies
20% are $h_4$: 25% cherry candies + 75% lime candies
10% are $h_5$: 100% lime candies

Then we observe candies drawn from some bag: ⬤⬤⬤⬤⬤⬤⬤⬤⬤⬤⬤⬤

What kind of bag is it? What flavour will the next candy be?
Bayesian Learning
Curve for Example

Posterior probability of hypotheses

Number of samples in \( d \)

Posterior probability of hypothesis
Bayesian Learning

Example

Prediction probability

\[ P(\text{next candy is lime} \mid d) \]

Number of samples in d
Naive Bayesian classifier

- Why naive? Makes independence assumptions which may not hold...

- Goal: given a target, determine which class it belongs to, based on the prior probabilities of features in the target

- In our case we’re looking to classify movie reviews as positive, negative, or neutral

- Shouldn’t be too hard <200 lines of code
Classifying reviews: Changeling

- Eastwood is a brilliant filmmaker who leaves nothing to chance. The details of the era are sensational.

- Though gripping at times, the pace of the movie plods along with all deliberate speed that might prompt occasional glances at the wristwatch.
How to Classify?

• Need to pick a set of features to use to distinguish positive and negative reviews

• Starting feature: words, consider the frequency of words in positive and negative reviews

• You will use this feature, and come up with one of your own
How to classify

- Compute product of all conditional probabilities of each of the features of the document, multiplied by the prior probability of any document being a member of the class

- Consider \( f \) a set of \( n \) features of document \( d \)

\[ P(\text{positive} \mid f) = P(\text{positive}) \prod_{i=1}^{n} P(f_i \mid \text{positive}) \]

Here is an expanded version of this calculation:

\[ P(\text{positive} \mid f) = P(\text{positive}) \times P(\text{I} \mid \text{positive}) \times P(\text{love} \mid \text{positive}) \times P(\text{school} \mid \text{positive}) \]
Evaluating Your Classifier

• *Precision* - fraction of documents properly classified or *positive predictive value*
  - True positives/(true positives + false positives)

• *Recall* - the percentage of true positives (*sensitivity*) correctly identified
  - True positives/(true positives + false negatives)

• *f-measure* - the weighted harmonic mean of precision and recall
  - \( F_1 = \frac{(2 \times \text{precision} \times \text{recall})}{\text{precision} + \text{recall}} \)
  - More generally
    - \( F_\beta = \frac{[(1 + \beta^2) \times (\text{precision} \times \text{recall})]}{(\beta^2 \times \text{precision} + \text{recall})} \)
Readings

- Nilesh Dalvi, Pedro Domingos, Mausam Sumit, Sanghai Deepak Verma. 
  Adversarial Classification. in Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining (KDD), 2004.