CSCI 5582
Artificial Intelligence

Lecture 21
Jim Martin

Today 11/14

• Review
• Hypothesis Learning
  - Version Spaces
• Break
• Relational Learning
  - ILP
Review

• Supervised machine learning
  - Naïve Bayes
  - Decision trees
  - Decision lists
  - Ensembles

Classifiers

• These all provide a way to separate objects into classes based on intrinsic features of the object (encoded as sets of feature/value pairs).
• They don’t necessarily provide a definition for the concept learned
• They can’t deal with relational data.
Classifiers

• Uncle?

Concept Learning

• In concept learning we’d like to learn something akin to a definition: necessary and sufficient conditions for membership in a category
  - Rules out all non-members
  - Includes all members
• And we’d like to be able to deal with relational data
• Assume we’re given positive and negative examples of the concept to be learned.
Data Mining

• The field of data mining is concerned with the extraction of possibly useful rules or patterns from large amounts of data.

Concept Learning

• And most importantly the concept to be learned is expressed in terms of predicates/propositions that are already known (that is we have a domain theory of some kind).
Basics

• In the context of concept learning, a hypothesis is just a theory of the concept that
  - Includes all members of the category
  - Excludes all non-members
• A false negative is...
• A false positive is...

Concept Learning: Search

• Again it's just search. We're searching through the space of possible hypotheses to find one (all?) that do exactly what we want: cover all and only the concepts we're trying to learn.
Current Best Hypothesis

Maintain a single hypothesis at a time.
Perform surgery on it on demand.
• If I give it a positive example and it covers it...
• If I give it a negative example and it rejects it...

Current Best Hypothesis

• If I give a positive example and it rejects it...
  - False negative.
    • Adjust the theory so that
      1. It...
      2. And it...
Current Best Hypothesis

- If I give it a negative example and it accepts it
  - False positive
    - Adjust the theory so that it
      1. ?
      2. ?

CBH

- How?
  - Depends on the language being used. But the critical notion to exploit is generalization/specialization
CBH

• If you need to cover a falsely rejected positive example...
  - You need to generalize your hypothesis
• If you need to reject a false accepted negative example...
  - You need to specialize your hypothesis

How...

• Depends on the language... typically
  - Dropping/adding conditions for membership
  - Adding/removing disjuncts from a definition
So...

• Search is just specializing/generalizing a single hypothesis in response to each successive training example
• Until you cover all and only.
• But....

But

• The backtracking inherent in CBH is pretty horrible.
• It turns out it isn’t really required
• CBH is making commitments early on that it really doesn’t have to make
  - Exploit the hierarchy inherent in the logical structure of the hypotheses
Version Space Learning

• You can represent the space of hypothesis by representing certain boundaries (without representing the hypotheses) themselves.
• In response to false positives and false negatives you simply adjust the boundaries.
• At the end the space of hypotheses within the boundaries are all consistent with the training data.

Version Space Learning

• I give you a single positive training example...
  - What’s the most general theory you can come up with?
  - What’s the most specific theory you can come up with?
Version Space Learning

• Termination....
  - When I run out of training examples
  - When the VS collapses
    • There are no theories left in the space
Break

• Look at
  - holmes.txt and tarzan.txt in
  - www.cs.colorado.edu/~martin/Csci5582

Break

• I'll go over the quiz topics Thursday
Relational Learning and Inductive Logic Programming

• Fixed feature vectors are a very limited representation of objects.
• Examples or target concept may require relational representation that includes multiple entities with relationships among them.
• First-order predicate logic is a more powerful representation for handling such relational descriptions.

ILP Example

• Learn definitions of family relationships given data for primitive types and relations.
  - brother(A,C), parent(C,B) → uncle(A,B)
  - husband(A,C), sister(C,D), parent(D,B) → uncle(A,B)
• Given the relevant predicates and a database populated with positive and negative examples
• By database I mean sets of tuples for each of the relevant relations
FOIL
First-Order Inductive Logic

• Top-down sequential covering algorithm to learn first order theories.
• Background knowledge provided extensionally (i.e. a model).
• Start with the most general rule possible. (T --> P(x))
• Specialize it on demand...
• Specializations of a clause include adding all possible literals one at a time to the antecedent...
  - A --> P
  - B --> P
  - C --> P...
  Where A, B and C are predicates already in the domain theory.

We’re working top-down from the most general hypothesis so what’s driving things?

---

FOIL

• At a high level.
  - Start with the most general H
  - Repeatedly constructs clauses that cover a subset of the positive examples and none of the negative examples.
  - Then remove the covered positive examples
  - Constructs another clause
  - Repeat until all the positive examples are covered.
FOIL Training Data

- Background knowledge consists of complete set of tuples for each background predicate for this universe.
- Example: Consider learning a definition for the target predicate path for finding a path in a directed acyclic graph.

\[
\text{path}(X,Y) :- \text{edge}(X,Y).
\]
\[
\text{path}(X,Y) :- \text{edge}(X,Z), \text{path}(Z,Y).
\]

edge: \{<1,2>,<1,3>,<3,6>,<4,2>,<4,6>,<6,5>\}
path: \{<1,2>,<1,3>,<1,6>,<1,5>,<3,6>,<3,5>,
<4,2>,<4,6>,<4,5>,<6,5>\}

FOIL Negative Training Data

- Negative examples of target predicate can be provided directly, or generated indirectly by making a closed world assumption.
  - Every pair of constants \(X,Y\) not in positive tuples for path predicate.

Negative path tuples:
\{<1,1>,<1,4>,<2,1>,<2,2>,<2,3>,<2,4>,<2,5>,<2,6>,
<3,1>,<3,2>,<3,3>,<3,4>,<4,1>,<4,3>,<4,4>,<5,1>,
<5,2>,<5,3>,<5,4>,<5,5>,<5,6>,<6,1>,<6,2>,<6,3>,
<6,4>,<6,6>\}
Sample FOIL Induction

Pos:  \{<1, 2>, <1, 3>, <1, 6>, <1, 5>, <3, 6>, <3, 5>,
       <4, 2>, <4, 6>, <4, 5>, <6, 5>\}

Neg:  \{<1, 1>, <1, 4>, <2, 1>, <2, 3>, <2, 4>, <2, 5>, <2, 6>,
       <3, 1>, <3, 2>, <3, 3>, <3, 4>, <4, 1>, <4, 3>, <4, 4>, <5, 1>,
       <5, 2>, <5, 3>, <5, 4>, <5, 5>, <6, 1>, <6, 2>, <6, 3>,
       <6, 4>, <6, 6>\}

Start with clause:
path(X, Y).:-.
Possible literals to add:
edge(X, X), edge(Y, Y), edge(X, Y), edge(Y, X), edge(X, Z),
edge(Y, Z), edge(Z, X), edge(Z, Y), path(X, X), path(Y, X),
path(X, Y), path(Y, X), path(X, Z), path(Y, Z), path(Z, X),
path(Z, Y), X=Y,
plus negations of all of these.

Test:
path(X, Y) :- edge(X, X).
Covers 0 positive examples
Covers 6 negative examples
Not a good literal to try.
Sample FOIL Induction

Pos: {<1,2>,<1,3>,<1,6>,<1,5>,<3,6>,<3,5>,
     <4,2>,<4,6>,<4,5>,<6,5>}

Neg: {<1,1>,<1,4>,<2,1>,<2,2>,<2,3>,<2,4>,<2,5>,<2,6>,
      <3,1>,<3,2>,<3,3>,<3,4>,<4,1>,<4,3>,<4,4>,<5,1>,
      <5,2>,<5,3>,<5,4>,<5,5>,<5,6>,<5,5>,<6,1>,<6,2>,<6,3>,
      <6,4>,<6,6>}

test:
path(X,Y) :- edge(X,Y).
Covers 6 positive examples
Covers 0 negative examples
Chosen as best literal. Result is base clause.

Remove covered positive tuples.
Sample FOIL Induction

Pos: \{<1, 6>,<1, 5>,<3, 5>,<4, 5>\}

Neg: \{<1, 1>,<1, 4>,<2, 1>,<2, 2>,<2, 3>,<2, 4>,<2, 5>,<2, 6>,<3, 1>,<3, 2>,<3, 3>,<3, 4>,<4, 1>,<4, 3>,<4, 4>,<5, 1>,<5, 2>,<5, 3>,<5, 4>,<5, 5>,<5, 6>,<6, 1>,<6, 2>,<6, 3>,<6, 4>,<6, 6>\}

Start new clause

path(X,Y):-.

CSCI 5582 Fall 2006
Sample FOIL Induction

Pos: \{<1,6>,<1,5>,<3,5>,<4,5>\}

Neg: \{<1,1>,<1,4>,<2,1>,<2,2>,<2,3>,<2,4>,<2,5>,<2,6>,
<3,1>,<3,2>,<3,3>,<3,4>,<3,5>,<3,6>,<4,1>,<4,2>,
<4,3>,<4,4>,<4,5>,<5,1>,<5,2>,<5,3>,<5,4>,<5,5>,<5,6>,<6,1>,<6,2>,<6,3>,
<6,4>,<6,6>\}

Test:
\texttt{path(X,Y):- edge(X,Z).}

Covers all 4 positive examples
Covers 14 of 26 negative examples
Eventually chosen as best possible literal

CSCI 5582 Fall 2006
Sample FOIL Induction

Pos: \{<1,6,2>,<1,6,3>,<1,5>,<3,5>,
       <4,5>\}

Neg: \{<1,1>,<1,4>,
       <3,1>,<3,2>,<3,3>,<3,4>,<4,1>,<4,3>,<4,4>,
       <6,1>,<6,2>,<6,3>,
       <6,4>,<6,6>\}

Test:
\text{path}(X,Y) :- \text{edge}(X,Z).

Covers all 4 positive examples
Covers 15 of 26 negative examples
Eventually chosen as best possible literal
Negatives still covered, remove uncovered examples.
Expand tuples to account for possible Z values.
Sample FOIL Induction

Pos: \{<1,6,2>,<1,6,3>,<1,5,2>,<1,5,3>,<3,5,6>,<4,5>\}

Neg: \{<1,1>,<1,4>,<3,1>,<3,2>,<3,3>,<3,4>,<4,1>,<4,3>,<4,4>,<6,1>,<6,2>,<6,3>,<6,4>,<6,6>\}

Test:

\text{path}(X,Y) : - \text{edge}(X,Z).

Covers all 4 positive examples
Covers 15 of 26 negative examples
Eventually chosen as best possible literal
Negatives still covered, remove uncovered examples.
Expand tuples to account for possible \(Z\) values.

Sample FOIL Induction

Pos: \{<1,6,2>,<1,6,3>,<1,5,2>,<1,5,3>,<3,5,6>,<4,5,6>\}

Neg: \{<1,1>,<1,4>,<3,1>,<3,2>,<3,3>,<3,4>,<4,1>,<4,3>,<4,4>,<6,1>,<6,2>,<6,3>,<6,4>,<6,6>\}

Test:

\text{path}(X,Y) : - \text{edge}(X,Z).

Covers all 4 positive examples
Covers 15 of 26 negative examples
Eventually chosen as best possible literal
Negatives still covered, remove uncovered examples.
Expand tuples to account for possible \(Z\) values.
**Sample FOIL Induction**

Pos: \{<1, 6, 2>, <1, 6, 3>, <1, 5, 2>, <1, 5, 3>, <3, 5, 6>, <4, 5, 6>\}

Neg: \{<1, 1, 2>, <1, 1, 3>, <1, 4, 2>, <1, 4, 3>, <3, 1, 6>, <3, 2, 6>, <3, 3, 6>, <3, 4, 6>, <4, 1, 2>, <4, 1, 6>, <4, 3, 2>, <4, 3, 6>, <4, 4, 2>, <4, 4, 6>, <6, 1, 5>, <6, 2, 5>, <6, 3, 5>, <6, 4, 5>, <6, 6, 5>\}

Test:

```
path(X, Y) :- edge(X, Z).
```

Covers all 4 positive examples
Covers 15 of 26 negative examples
Eventually chosen as best possible literal
Negatives still covered, remove uncovered examples.
Expand tuples to account for possible Z values.

---

**Sample FOIL Induction**

Pos: \{<1, 6, 2>, <1, 6, 3>, <1, 5, 2>, <1, 5, 3>, <3, 5, 6>, <4, 5, 6>\}

Neg: \{<1, 1, 2>, <1, 1, 3>, <1, 4, 2>, <1, 4, 3>, <3, 1, 6>, <3, 2, 6>, <3, 3, 6>, <3, 4, 6>, <4, 1, 2>, <4, 1, 6>, <4, 3, 2>, <4, 3, 6>, <4, 4, 2>, <4, 4, 6>, <6, 1, 5>, <6, 2, 5>, <6, 3, 5>, <6, 4, 5>, <6, 6, 5>\}

Continue specializing clause:

```
path(X, Y) :- edge(X, Z).
```
Sample FOIL Induction

Pos: \{<1, 6, 2>, <1, 6, 3>, <1, 5, 2>, <1, 5, 3>, <3, 5, 6>, <4, 5, 6>\}

Neg: \{<1, 1, 2>, <1, 1, 3>, <1, 4, 2>, <1, 4, 3>, <3, 1, 6>, <3, 2, 6>, <3, 3, 6>, <3, 4, 6>, <4, 1, 2>, <4, 1, 6>, <4, 3, 2>, <4, 3, 6> <4, 4, 2>, <4, 4, 6>, <6, 1, 5>, <6, 2, 5>, <6, 3, 5>, <6, 4, 5>, <6, 6, 5>\}

Test:
\[
\text{path}(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, Y).
\]

Covers 3 positive examples
Covers 0 negative examples

CSCI 5582 Fall 2006

---

Sample FOIL Induction

Pos: \{<1, 6, 2>, <1, 6, 3>, <1, 5, 2>, <1, 5, 3>, <3, 5, 6>, <4, 5, 6>\}

Neg: \{<1, 1, 2>, <1, 1, 3>, <1, 4, 2>, <1, 4, 3>, <3, 1, 6>, <3, 2, 6>, <3, 3, 6>, <3, 4, 6>, <4, 1, 2>, <4, 1, 6>, <4, 3, 2>, <4, 3, 6> <4, 4, 2>, <4, 4, 6>, <6, 1, 5>, <6, 2, 5>, <6, 3, 5>, <6, 4, 5>, <6, 6, 5>\}

Test:
\[
\text{path}(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, Y).
\]

Covers 4 positive examples    Covers 0 negative examples
Eventually chosen as best literal; completes clause.
Definition complete, since all original \(<X, Y>\) tuples are covered
(by way of covering some \(<X, Y, Z>\) tuples.)
More Realistic Applications

- Classifying chemical compounds as mutagenic (cancer causing) based on their graphical molecular structure and chemical background knowledge.
- Classifying web documents based on both the content of the page and its links to and from other pages with particular content.
  - A web page is a university faculty home page if:
    - It contains the words "Professor" and "University", and
    - It is pointed to by a page with the word "faculty", and
    - It points to a page with the words "course" and "exam"

Rule Learning and ILP Summary

- There are effective methods for learning symbolic rules from data using greedy sequential covering and top-down or bottom-up search.
- These methods have been extended to first-order logic to learn relational rules and recursive Prolog programs.
- Knowledge represented by rules is generally more interpretable by people, allowing human insight into what is learned and possible human approval and correction of learned knowledge.