Rule Induction

Instructor: Jesse Davis

Announcements

- We will go over HW 1 next week [will be graded by then]
- HW 2 due next week
- Lecture slides are online

Outline

- Bayes net review
- Propositional rule induction
- First-order rule induction

Joint Distributions

- Joint distribution $P(X_1,\ldots,X_n)$ will tell you about any possible setting of $X_1,\ldots,X_n$
  - $P(X_1=\text{red},X_3=\text{medium},X_{100}=\text{heavy})$
  - $P(X_1=\text{red},\ldots,X_n=\text{heavy})$

- A BN captures a joint distribution

- Even NB allows you to answer any query

Conditional Distribution

P(A | B)

- A = {small, medium, large}
- B = {blue, green, red}

	P(A | B = blue)	P(A | B =green)	P(A | B = red)
small	0.2	0.06	0.5
medium	0.6	0.25	0.4
large	0.2	0.69	0.1

Note: P(A | B=blue) + P(A | B= red) is not meaningful

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer

Goal: Prob(cancer | +)

Bayes rule: $P(c \mid +) = [P(+ \mid c)\, P(c)] / P(+)$

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) =?
  - Prob(+ | cancer) = ?
  - Prob(+ | ¬cancer) = ?
  - Prob(+) = ?
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = ?
  - Prob(+ | ¬cancer) = ?
  - Prob(+) = ?
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | ¬cancer) = ?
  - Prob(+) = ?
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | $\neg$cancer) = 0.03
  - Prob(+) = ?
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | ¬cancer) = 0.03
  - Prob(+) = P(+|c)* P(c) + P(+|¬c) * P(¬c)
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | ¬cancer) = 0.03
  - Prob(+) = 0.98* 0.008 + 0.03 * 0.992
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | ¬cancer) = 0.03
  - Prob(+) = 0.0376
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | ¬cancer) = 0.03
  - Prob(+) = 0.0376
  - Prob(cancer | +) = ?

Bayes Rule Example

- Patient has a lab test with a positive results
  - Returns positive if cancer is present 98% of time
  - Returns negative if cancer is absent 97% of time
  - 0.8% of people have this cancer
- What is the prob of:
  - Prob(cancer) = 0.008
  - Prob(+ | cancer) = 0.98
  - Prob(+ | ¬cancer) = 0.03
  - Prob(+) = 0.0376
  - Prob(cancer | +) = 0.209

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = ?$ $P(C_2|HT) = ?$ $P(C_3|HT) = ?$

$$P(C_1|HT) = \alpha P(HT|C_1)P(C_1) = \alpha P(H|C_1)P(T|C_1)P(C_1)$$

C_1 C_2 C_3

$P(H|C_1) = 0.1$ $P(H|C_2) = 0.5$ $P(H|C_3) = 0.9$

$P(C_1) = 1/3$ $P(C_2) = 1/3$ $P(C_3) = 1/3$

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = ?$ $P(C_2|HT) = ?$ $P(C_3|HT) = ?$

$P(C_1|HT) = P(C_1|H) * P(C_1|T) * P(C_1)$

$P(C_2|HT) = P(C_2|H) * P(C_2|T) * P(C_2)$

$P(C_3|HT) = P(C_3|H) * P(C_3|T) * P(C_3)$

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = ?$ $P(C_2|HT) = ?$ $P(C_3|HT) = ?$

$P(C_1|HT) = a*0.1 * 0.9 * 0.33 = a*0.0297$

$P(C_2|HT) = a*0.5 * 0.5 * 0.33 = a*0.0825$

$P(C_3|HT) = a*0.9 * 0.1 * 0.33 = a*0.0297$

Don't sum to 1!!!!

$P(C_1|HT) + P(C_2|HT) + P(C_3|HT) = 1$

Now normalize!

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = ?$ $P(C_2|HT) = ?$ $P(C_3|HT) = ?$

$P(C_1|HT) = 0.1 * 0.9 * 0.33 = 0.0297$

$P(C_2|HT) = 0.5 * 0.5 * 0.33 = 0.0825$

$P(C_3|HT) = 0.9 * 0.1 * 0.33 = 0.0297$

a = ?

Now normalize!

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = ?$ $P(C_2|HT) = ?$ $P(C_3|HT) = ?$

$P(C_1|HT) = a*0.1 * 0.9 * 0.33 = a*0.0297$

$P(C_2|HT) = a*0.5 * 0.5 * 0.33 = a*0.0825$

$P(C_3|HT) = a*0.9 * 0.1 * 0.33 = a*0.0297$

$a = 1/[0.0297 + 0.0825 + 0.0297] = 1/0.1419$

Now normalize!

Experiment 2: Tails

Which coin <u>did</u> I use?

$P(C_1|HT) = ?$ $P(C_2|HT) = ?$ $P(C_3|HT) = ?$

$P(C_1|HT) = 0.1 * 0.9 * 0.33 = 0.0297 / 0.1419 = 0.21$

$P(C_2|HT) = 0.5 * 0.5 * 0.33 = 0.0825 / 0.1419 = 0.58$

$P(C_3|HT) = 0.9 * 0.1 * 0.33 = 0.0297 / 0.1419 = 0.21$

$a = 1/[0.0297 + 0.0825 + 0.0297] = 1/0.1419$

Now normalize!

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = 0.21$ $P(C_2|HT) = 0.58$ $P(C_3|HT) = 0.21$

$$P(C_1|HT) = \alpha P(HT|C_1)P(C_1) = \alpha P(H|C_1)P(T|C_1)P(C_1)$$

C_1

$P(H|C_1) = 0.1$

$P(C_1) = 1/3$

C_2

$P(H|C_2) = 0.5$

$P(C_2) = 1/3$

C_3

$P(H|C_3) = 0.9$

$P(C_3) = 1/3$

Experiment 2: Tails

Which coin did I use?

$P(C_1|HT) = 0.21$ $P(C_2|HT) = 0.58$ $P(C_3|HT) = 0.21$

C_1

$P(H|C_1) = 0.1$

$P(C_1) = 1/3$

C_2

$P(H|C_2) = 0.5$

$P(C_2) = 1/3$

C_3

$P(H|C_3) = 0.9$

$P(C_3) = 1/3$

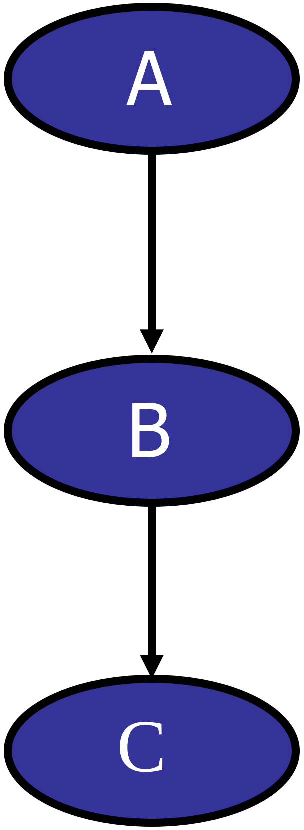
Normalization for NB

$$\text{prob(ex=pos)} = \frac{e^{\log[\text{prob(ex = pos)}]}}{e^{\log[\text{prob(ex = pos)}]} + e^{\log[\text{prob(ex = neg)}]}}$$

$$\log[\text{prob(ex=pos)}] = \log\,[\text{prob(pos)}] + \Sigma_i \log[\text{prob}(F_i \mid \text{pos})]$$

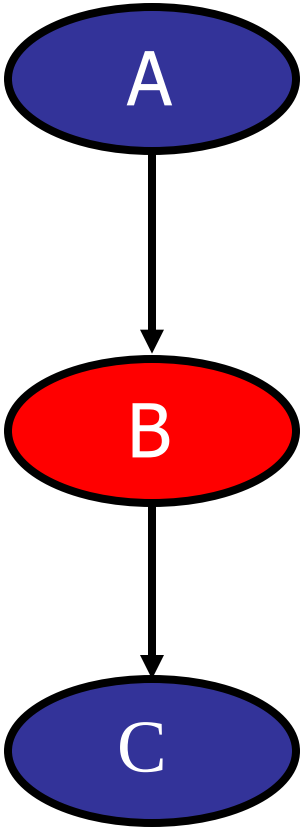
$$\log[\text{prob(ex=neg)}] = \log\,[\text{prob(neg)}] + \Sigma_i \log[\text{prob}(F_i \mid \text{neg})]$$

D-Separation



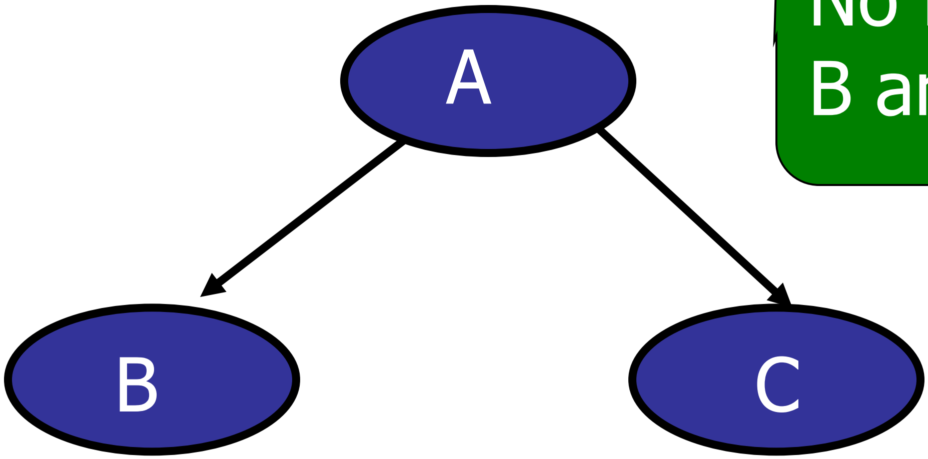
No Evidence:
A and C are dependent

D-Separation



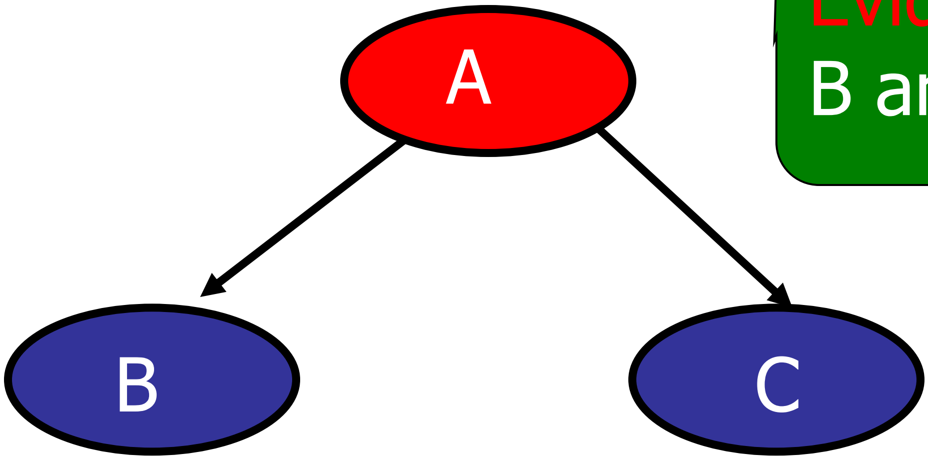
Evidence at B:
A and C are independent

D-Separation



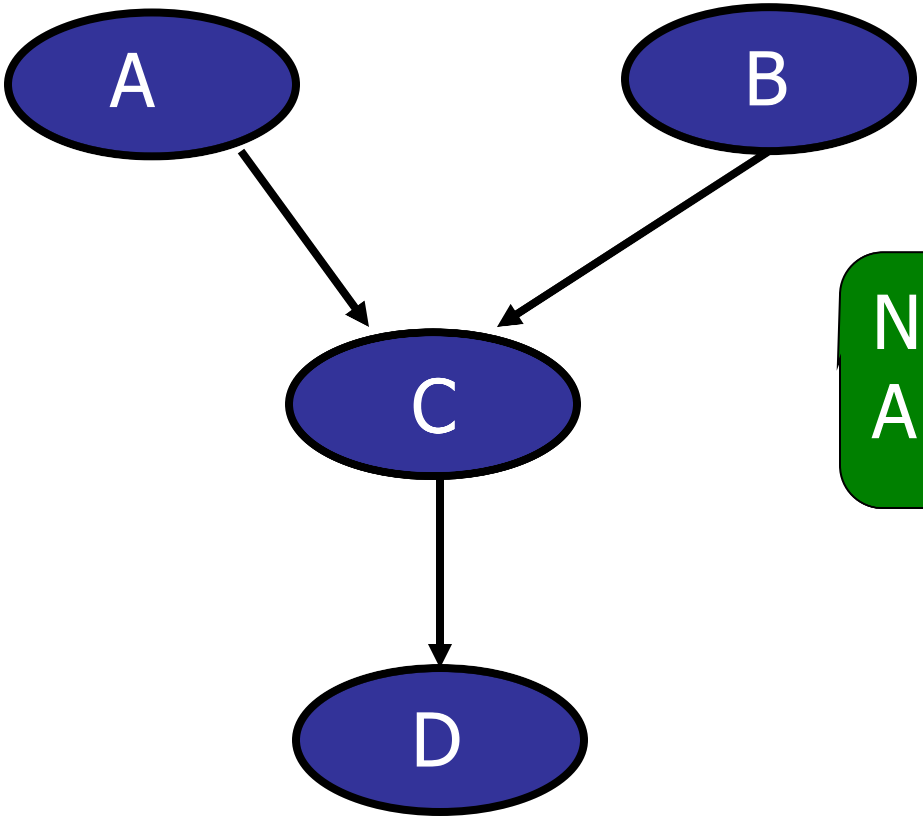
No Evidence:
B and C are dependent

D-Separation



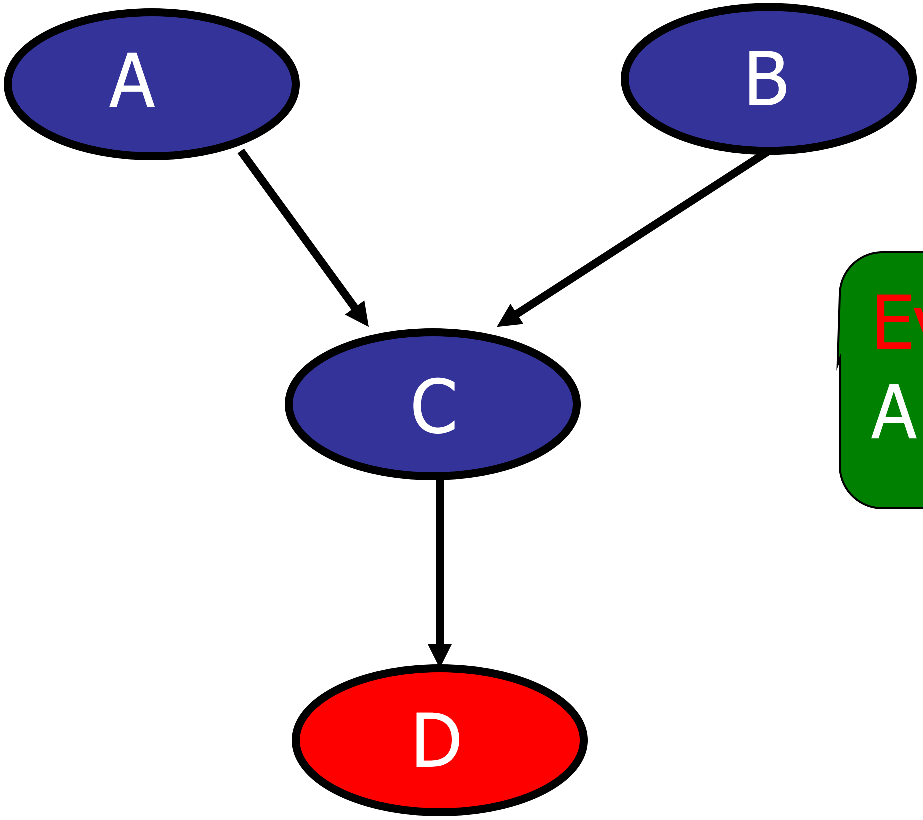
Evidence at A:
B and C are independent

D-Separation



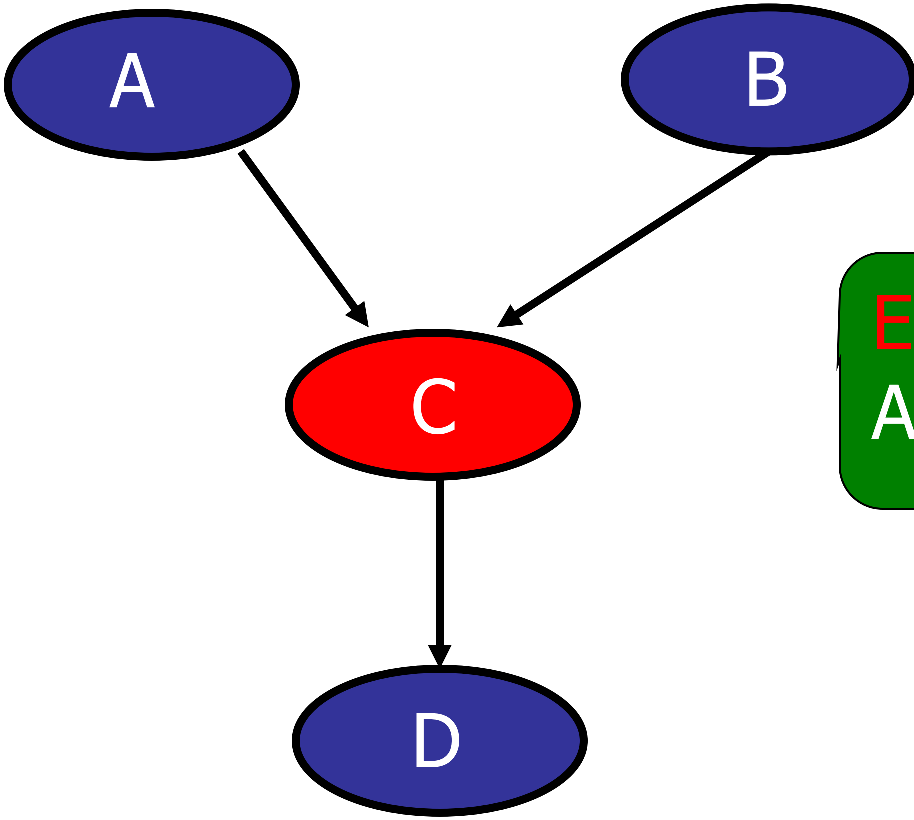
No Evidence:
A and B are independent

D-Separation



Evidence at D:
A and B are dependent

D-Separation



Evidence at C:

A and B are dependent

Outline

- Bayes net review
- Propositional rule induction
- First-order rule induction

Rule Sets

- Effective hypothesis space
  - Variable sized hypotheses
  - Can represent any Boolean function
  - Can represent both discrete and continuous features

- Classify learning algorithm as follows:
  - Constructive search: Learn by adding rules to rule set. Each rule is built by adding tests
  - Eager
  - Batch

Terminology

- A rule consists of a body and a head

  Body $\Rightarrow$ Head
- The body consists of a conjunction of tests
  - Attribute = value
  - Attribute > value
  - Attribute < value
  - Where ‘Attribute’ is one of the features and ‘value’ appears in the training data
- The head contains a class label

Relationship to Propositional Logic

- A rule is also called a Horn (or definite) clause

  Outlook = Sunny $\wedge$ Humidity = High $\Rightarrow$ Don't play

  Is logically equivalent to

  $\neg$[Outlook = Sunny] $\vee$ $\neg$[Humidity = High] $\vee$ [Don't play]

- Horn clause: Zero or one positive test
- Definite clause: Exactly one positive test
  - E.g.: $\neg$[Outlook = Sunny] $\vee$ $\neg$[Humidity = High] is a Horn clause, but not a definite clause

Terminology

- A rule set is a disjunction of rules

  Outlook = Sunny $\wedge$ Humidity = Normal $\Rightarrow$ Play

  Outlook = Overcast $\Rightarrow$ Play

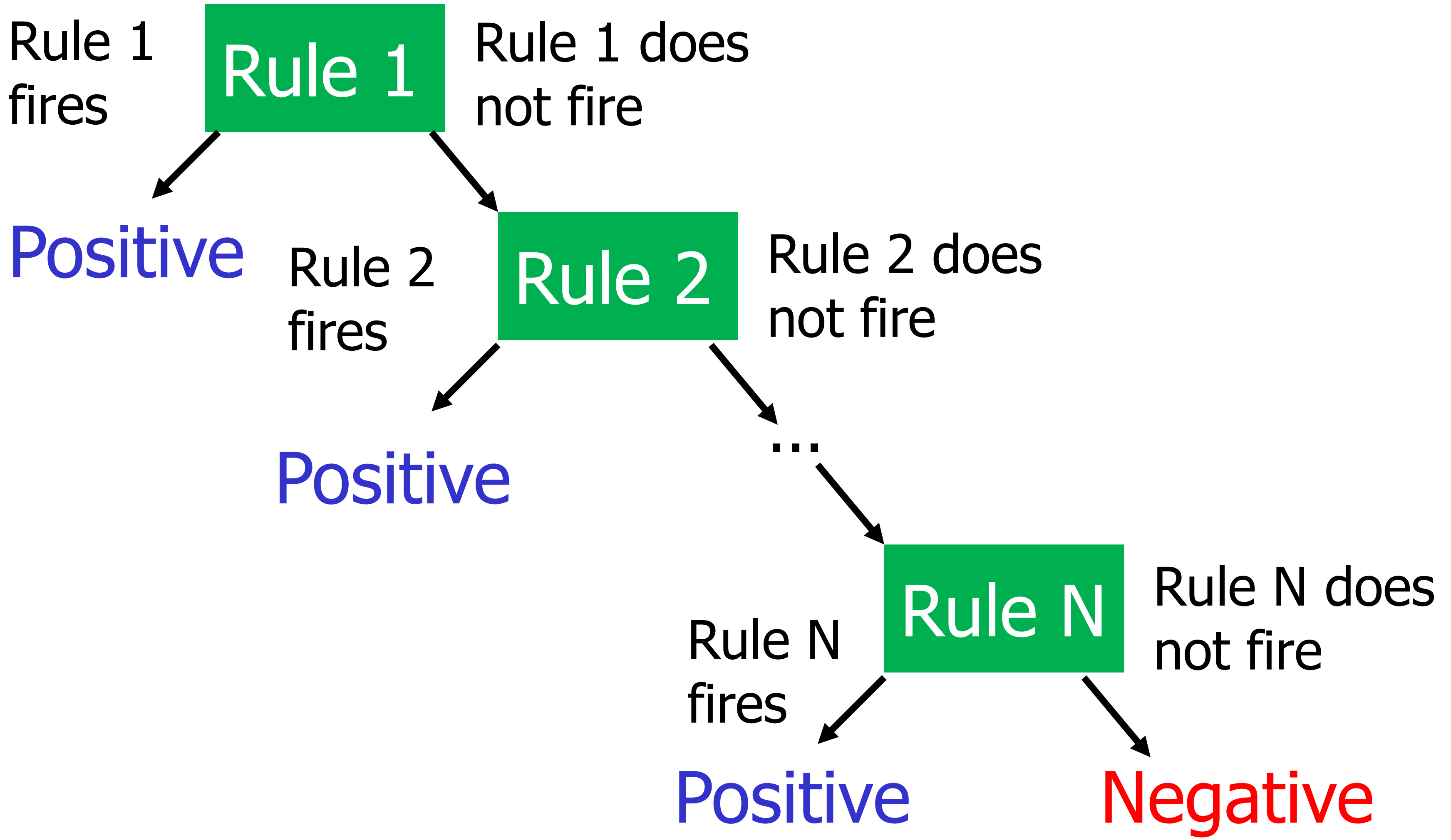
  Outlook = Rain $\wedge$ Wind = Weak $\Rightarrow$ Play
- Rule covers an example: True of the example

  Outlook = Sunny $\wedge$ Humidity = Normal $\Rightarrow$ Play

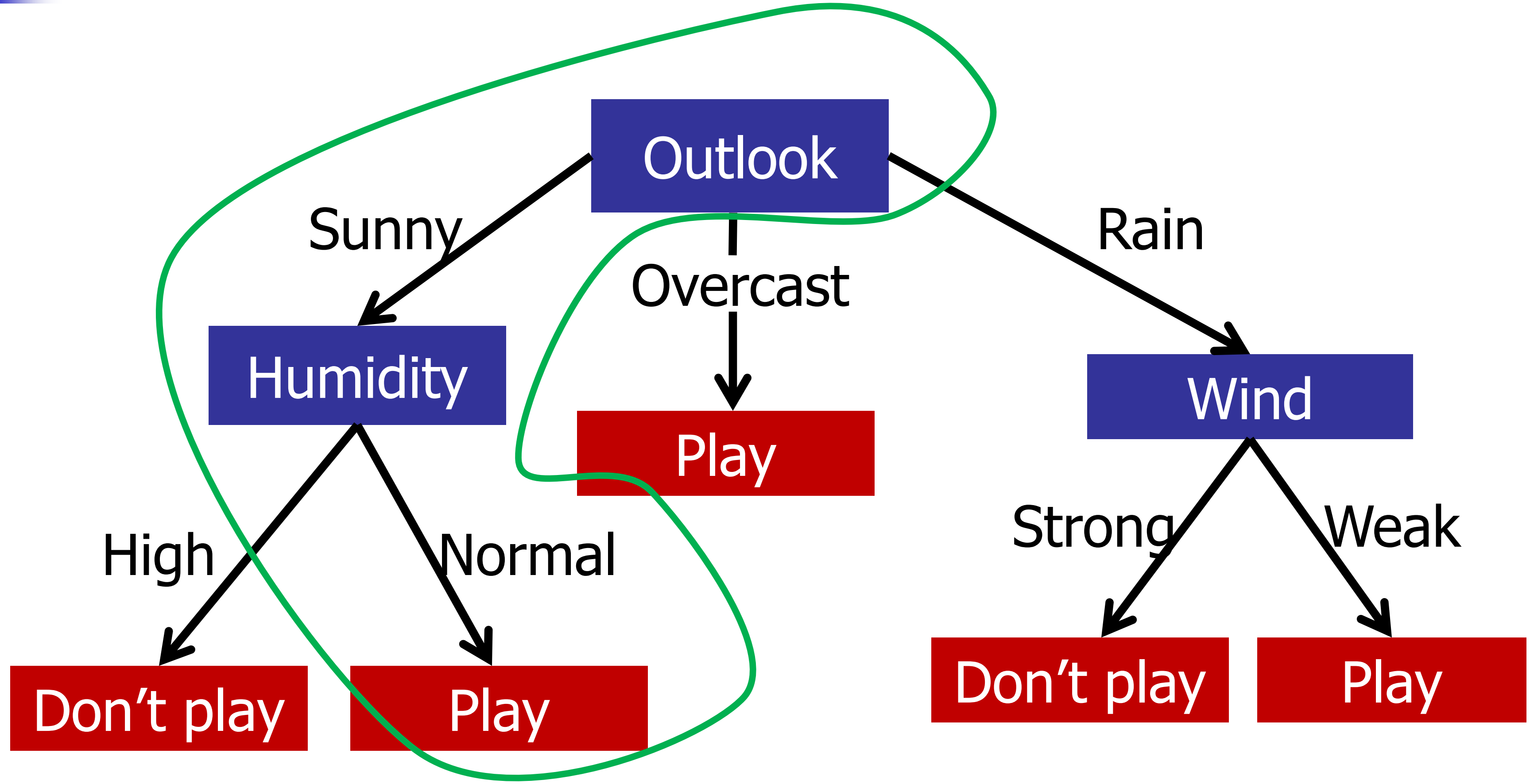
  covers any example

  sunny,?,normal,?,?
- Usually all rules are predictive of one class
- Combine rules through a decision list

Decision List



Relationship to Decision Trees



The previous rule set is equivalent to this decision tree

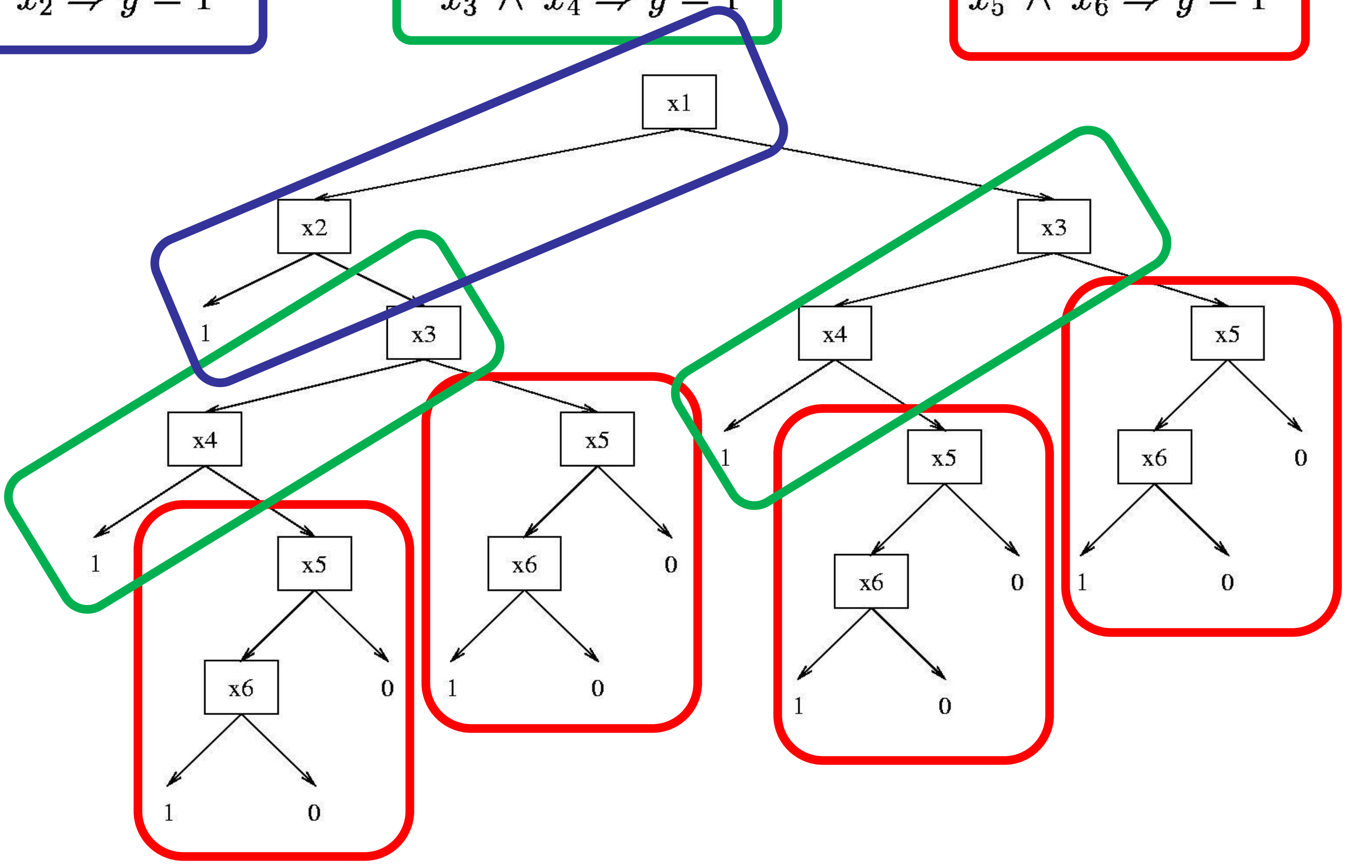
Relationship to Decision Trees

A small set of rules can correspond to a big decision tree, because of the *Replication Problem*.

$x_1 \wedge x_2 \Rightarrow y = 1$

$x_3 \wedge x_4 \Rightarrow y = 1$

$x_5 \wedge x_6 \Rightarrow y = 1$



Learning a Single Rule

GrowRule(P,N)

R = {}

Repeat

choose ‘best test’ $x_i \Theta v$ to add R, $\Theta \in \{=, \neq, \leq, \geq\}$

R = R U $x_i \Theta v$

P = P – all positive examples not satisfied by R

Until P is empty

Return R

Learning a Single Rule

$\Rightarrow$ **Play**

P = 0
N = 0
O = O $\Rightarrow$ **Play**

P = 2
N = 3
O = S $\Rightarrow$ **Play**

P = 3
N = 2
O = R $\Rightarrow$ **Play**

…

P = 3
N = 3
W = S $\Rightarrow$ **Play**

P = 0
N = 0
O = R $\wedge$ H = L $\Rightarrow$ **Play**

…

P = 3
N =0
O = R $\wedge$ W = W $\Rightarrow$ **Play**

P = 0
N = 3
O = R $\wedge$ W = S $\Rightarrow$ **Play**

Called Top-down or general-to-specific induction
Each step called a refinement or specialization

Choosing the Best Test

- Rule R covers P_0 and N_0 and R′ covers P_1 and N_1
- Relative frequency $\frac{P_1}{P_1 + N_1}$
- Coverage: $P_1 - N_1$
- Gain: $P_1 \left[\left[\frac{-P_0}{P_0 + N_0} \log \frac{P_0}{P_0 + N_0} \right] - \left[\frac{-P_1}{P_1 + N_1} \log \frac{P_1}{P_1 + N_1} \right] \right]$

P_1 helps manage the trade off between gain and covering many positive examples

Learning a Set of Rules
(Called Divide-and-conquer or Cover-removal)

GrowRuleSet(P,N)

A = {}

Repeat

    R = GrowRule(P,N)

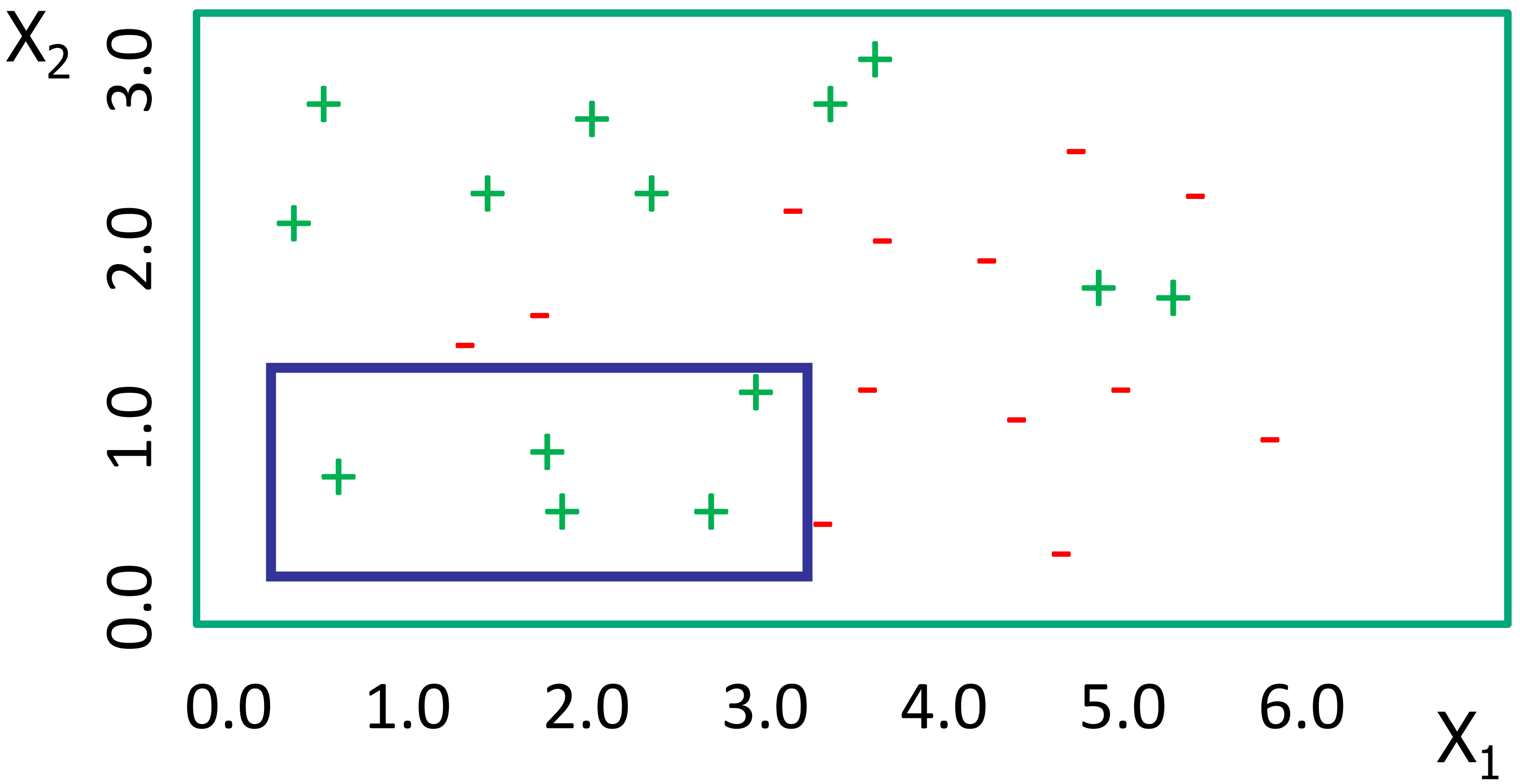
    A = R U A

    P = P – all positive examples that satisfy R

Until P is empty

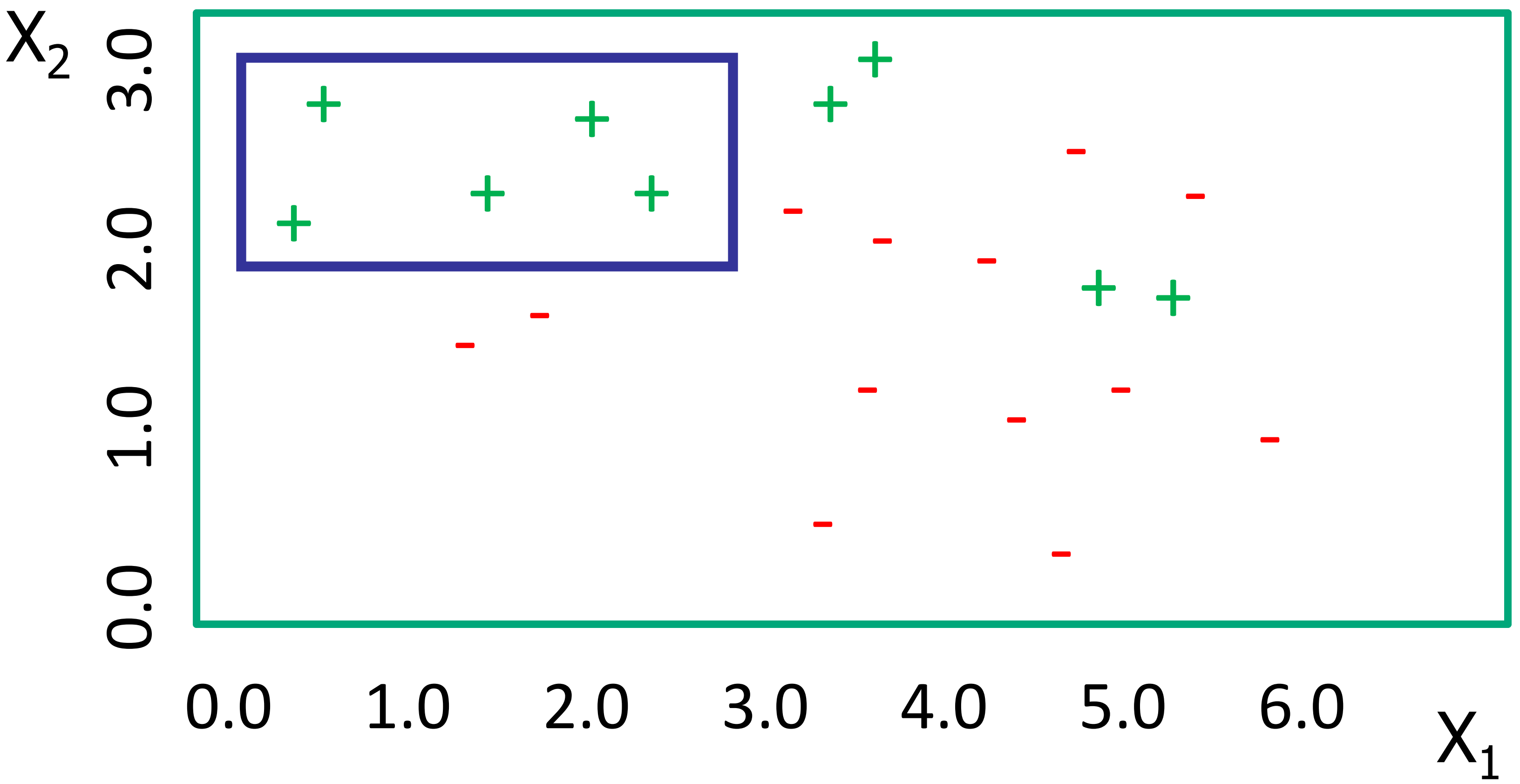
Return A

Cover-Removal Example



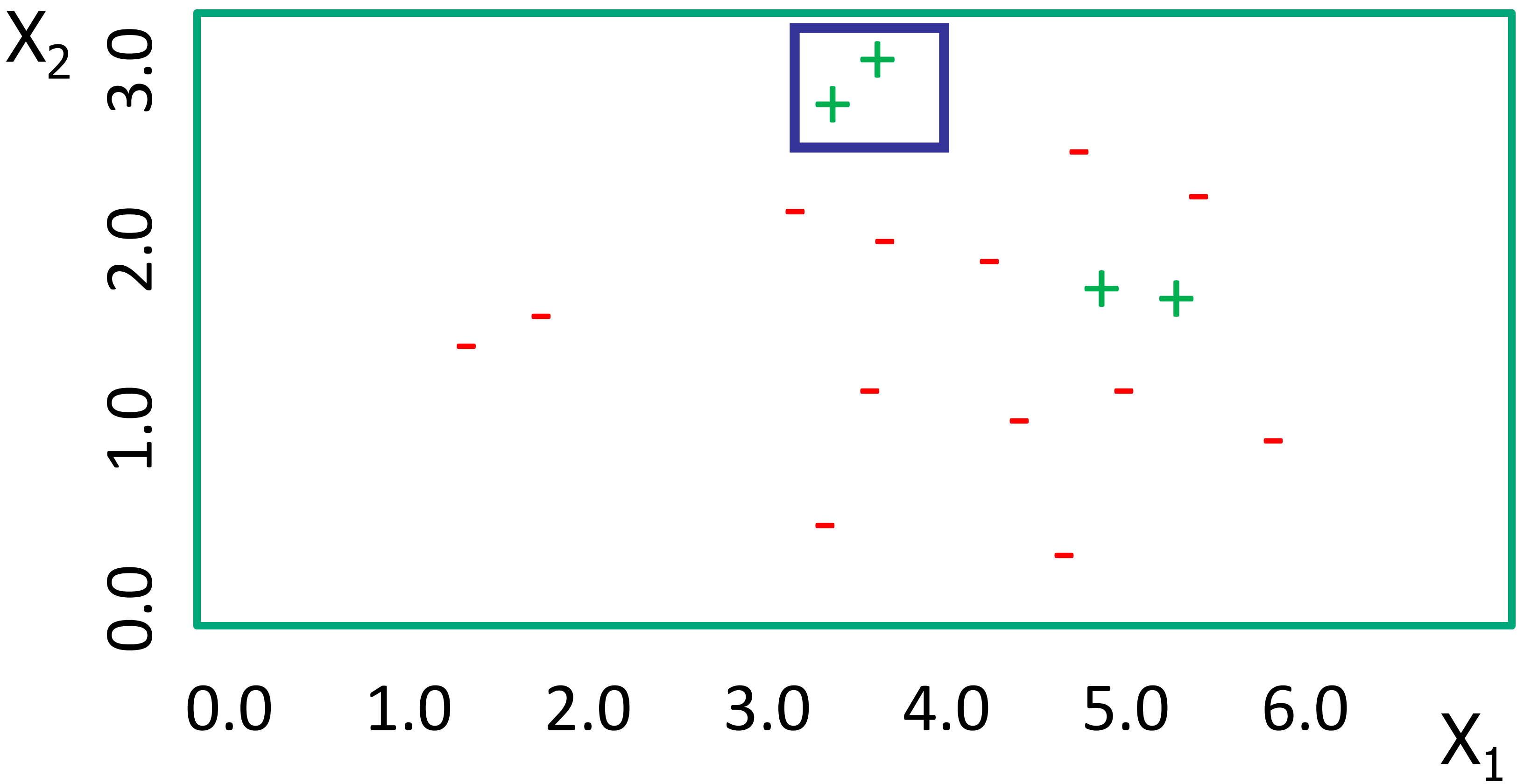
Rule 1: $X_1 < 3 \wedge X_2 < 1.2 \Rightarrow +$

Cover-Removal Example



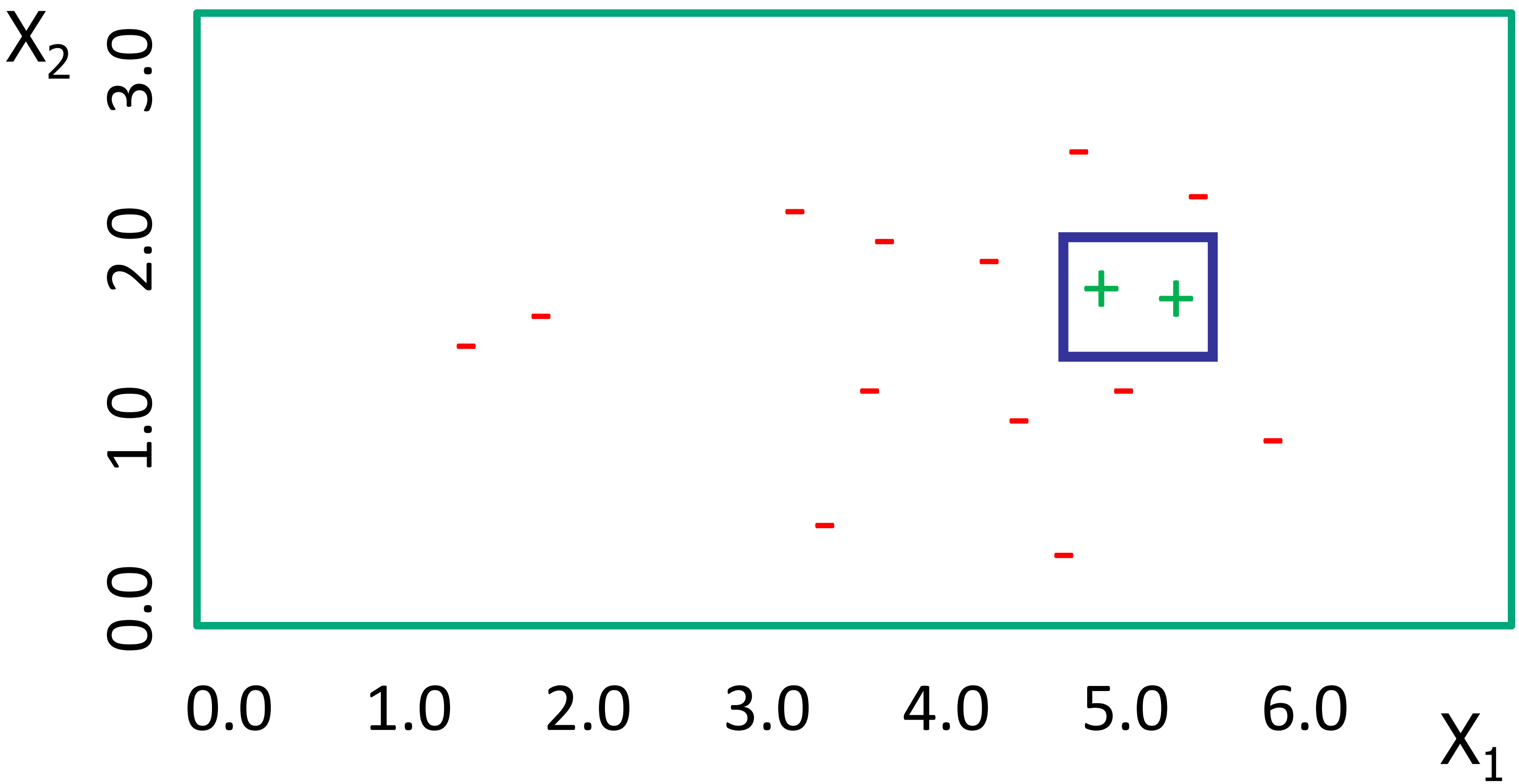
Rule 2: $X_1 < 2.5 \wedge X_2 > 2.0 \Rightarrow +$

Cover-Removal Example



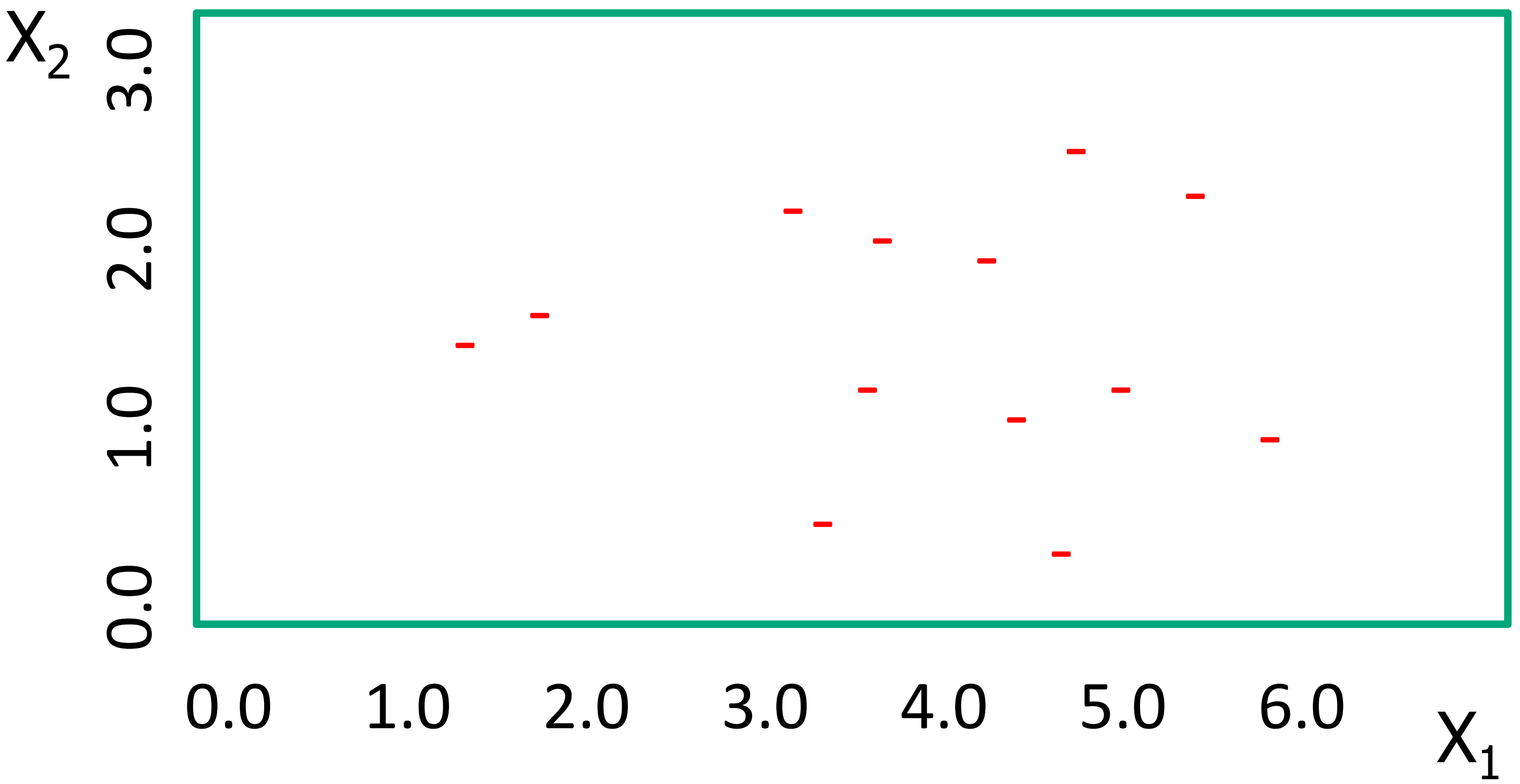
Rule 3: $X_1 > 3 \wedge X_1 < 4.0 \wedge X_2 > 2.5 \Rightarrow +$

Cover-Removal Example

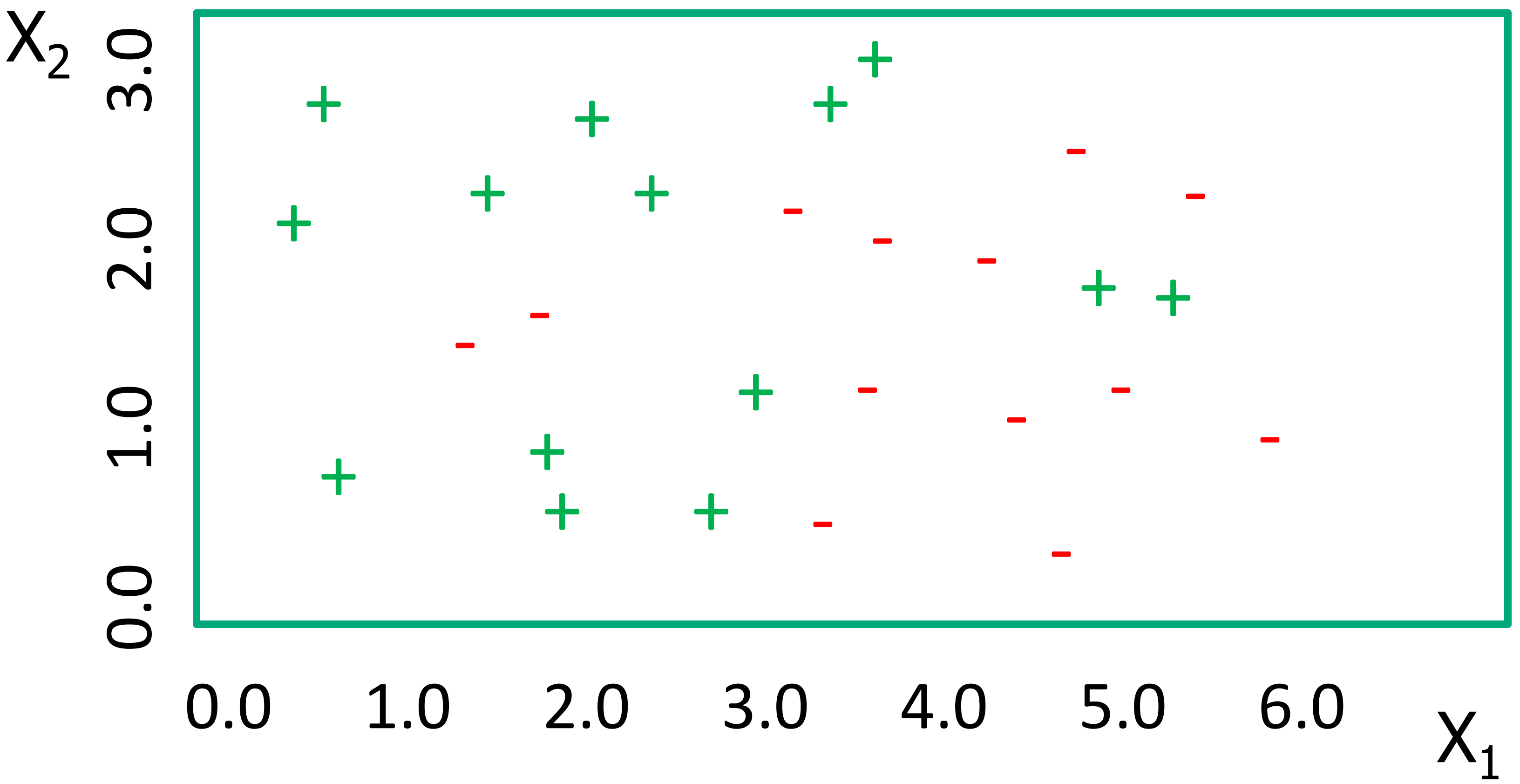


Rule 4: $X_1 > 4.8 \wedge X_1 < 5.5 \wedge X_2 > 1.5 \wedge X_2 < 2.0 \Rightarrow +$

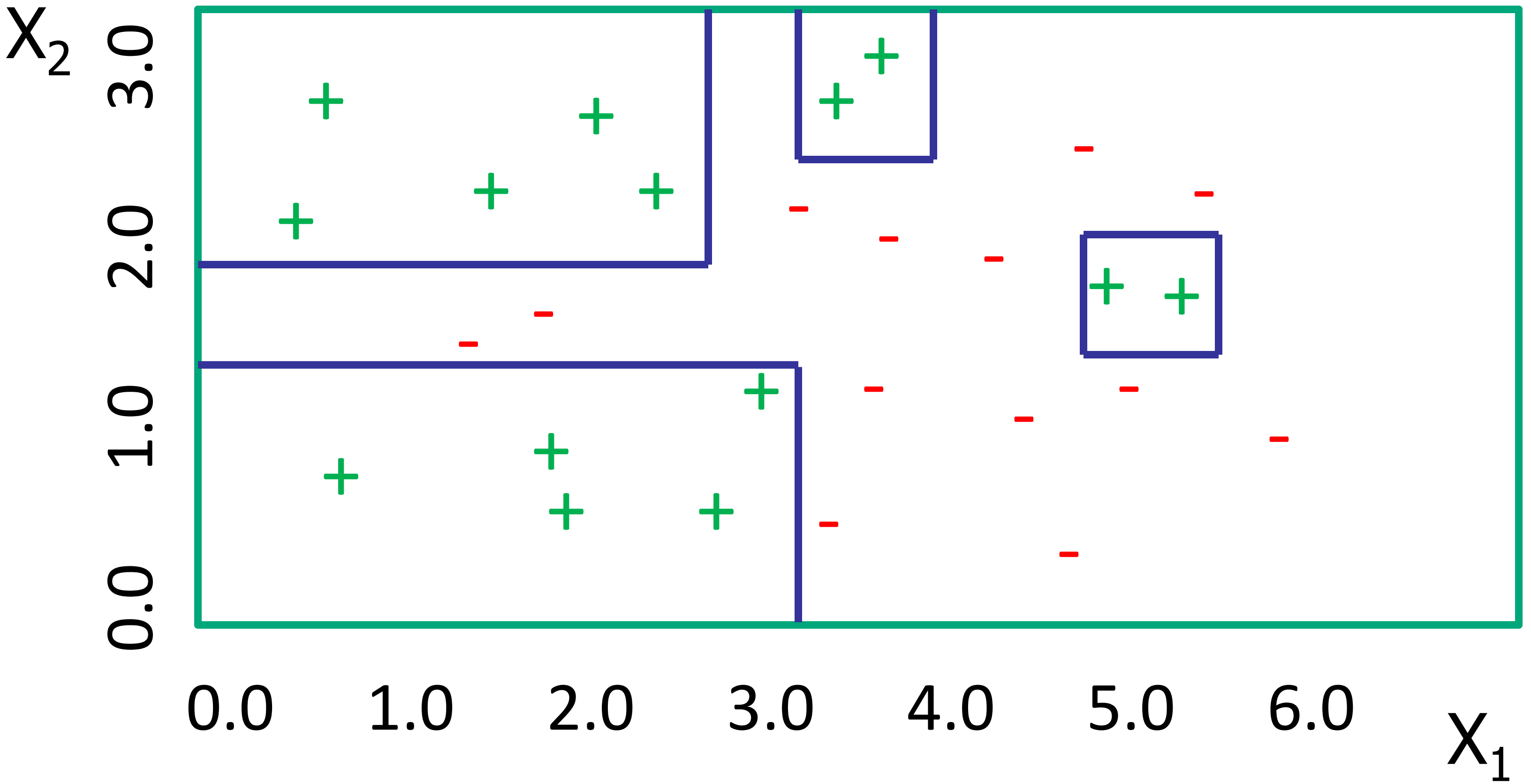
Cover-Removal Example



How Does a Rule Set Partition Feature Space?



How Does a Rule Set Partition Feature Space?



Other Search Procedures

- The presented algorithms all use greedy search
- Finding the smallest set of rules is NP-hard
- Other search procedures often lead to results
  - Round robin replacement
  - Backfitting
  - Beam search
  - Specific-to-general search

Round Robin Replacement

- Build entire rule set
- Delete the first rule learned
- Find all training examples uncovered by any remaining rule
- Learn rule(s) to cover these training examples
- This can be repeated for each original rule
- Allows a later rule to capture the positive examples of a rule learned earlier

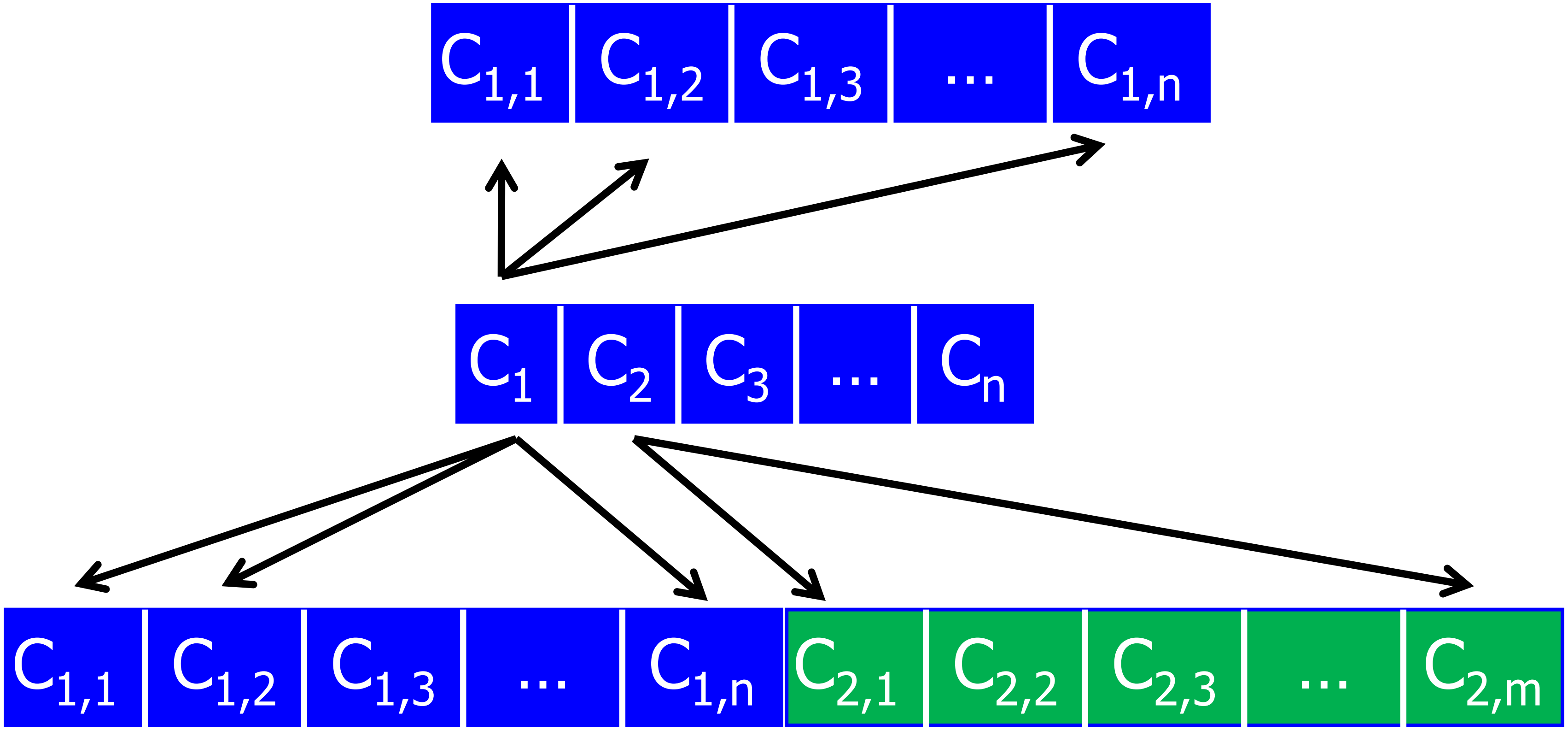
Backfitting

- After adding each rule to the rule set, perform round robin replacement
- Typically, just do several iterations, which converges quickly
- Repeat the process of learning a rule and the performing round robin replacement until all positive examples are covered

Beam Search

- Instead of building just one rule at a time, we build B rules
- At each step, consider all possible refinements for each of the B rules
- Score these refinements and select the top B
- Iterate until it is not possible to add a test
- Add best rule to the rule set

Beam Search Pictorially



Bottom-Up Learning

Day	Outlook	Temp	Humid	Wind	PlayTennis?
d1	s	h	h	w	n
d2	s	h	h	s	n
d3	o	h	h		y
d4	r	m	h		
d5	r	c	n		
d6	r	c	n		
d7	o	c	n		
d8	s	m	h		
d9	s	c	n	w	y
d10	r	m	n	w	y
d11	s	m	n	s	y
d12	o	m	h	s	y
d13	o	h	n	w	y
d14	r	m	h	s	n

Feature combination never occurs for ‘positive’ class – why waste time evaluating it?

One Simple Idea

- Unifies rule induction and k-NN
- Treat each example as a rule
- Generalize rules by dropping conditions
  - Find nearest example of same class
  - Drop conditions such that rule satisfies both examples [and no negative examples]
  - Note: If no accepted generalizations, form of nearest-neighbors!

Finding a Generalization

Day	Outlook	Temp	Humid	Wind	PlayTennis?	Distance
d3	o	h	h	w	y	?
d4	r	m	h	w	y	?
d5	r	c	n	w	y	?
d7	o	c	n	s	y	?
~~d9~~	~~s~~	~~c~~	~~n~~	~~w~~	~~y~~	~~?~~
d10	r	m	n	w	y	?
d11	s	m	n	s	y	?
d12	o	m	h	s	y	?
d13	o	h	n	w	y	?

$\boxed{\text{Outlook=s} \wedge \text{Temperature=c} \wedge \text{Humid=n} \wedge \text{Wind=w} \Rightarrow \text{y}}$

Bottom-Up Learning

Day	Outlook	Temp	Humid	Wind	PlayTennis?	Distance
d3	o	h	h	w	y	3
d4	r	m	h	w	y	3
d5	r	c	n	w	y	1
d7	o	c	n	s	y	2
~~d9~~	~~s~~	~~c~~	~~n~~	~~w~~	~~y~~	~~?~~
d10	r	m	n	w	y	2
d11	s	m	n	s	y	3
d12	o	m	h	s	y	4
d13	o	h	n	w	y	2

$\boxed{\text{Outlook=s} \wedge \text{Temperature=c} \wedge \text{Humid=n} \wedge \text{Wind=w} \Rightarrow \text{y}}$

Example

$\boxed{\text{Covers 4 pos, 0 neg}}$

Humid=n $\wedge$ Wind=w $\Rightarrow$ y

$\boxed{\text{Covers 2 pos, 0 neg}}$

Temp=c $\wedge$ Humid=n $\wedge$ Wind=w $\Rightarrow$ y

Out=s $\wedge$ Temp=c $\wedge$ Humid=n $\wedge$ Wind=w $\Rightarrow$ y

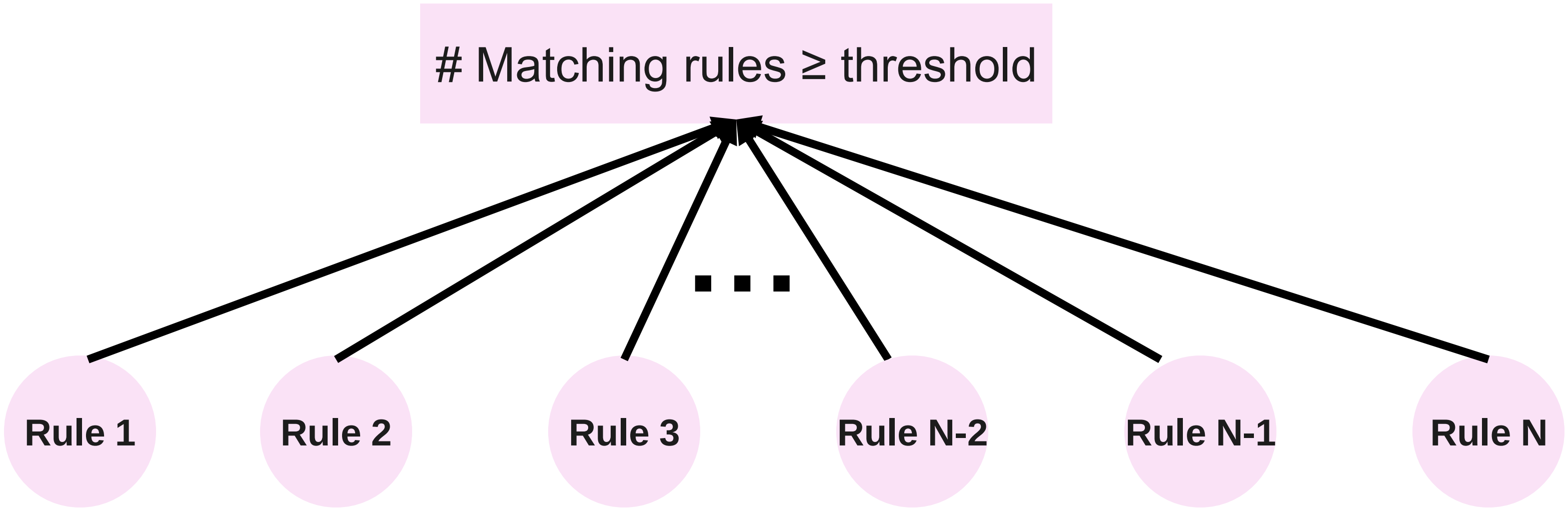
Learning Rules for Multiple Classes

What if rules for more than one class?

Two possibilities:

- Order rules (decision list)
- Weighted vote (e.g., weight = accuracy $\times$ coverage)

Voting Example



Notes on Rule Induction

- When scoring rules, be weary of small sample sizes
  - Use m-estimates as mentioned last week
- Cover-removal is a clever idea for directing the search space
  - Prone to building rules for each noisy example
- Bottom-up learning generally is the most accurate – though can be slower

Outline

- Bayes net review
- Propositional rule induction
- First-order rule induction
  - Motivation and first-order logic
  - FOIL
  - Inverting resolution: Cigol
  - Progol
  - Applications

Learning First-Order Rules

- Why do this?
  - Capture information about related entities
  - Can learn in relational DBs
  - No longer restricted to fixed-length feature vectors

- Can learn rules such as:

  ancestor(X,Y) $\leftarrow$ parent(X,Y)

  ancestor(X,Y) $\leftarrow$ parent(X,Z), ancestor(Z,Y)
- Prolog programs are sets of such rules

First-Order Rule for Classifying Web Pages

[Slattery, 1997]

course(A) $\leftarrow$
  has-word(A, instructor),
  $\neg$ has-word(A, good),
  link-from(A, B),
  has-word(B, assign),
  $\neg$ link-from(B, C)

Train: 31/31, Test: 31/34

First-Order Logic Review

- Four symbols:
  - **Constants:** anna, bob
  - **Variables:** X, Y
  - **Predicates/relations:** friends(X,Y)
  - **Functions:** motherOf(X)

- **Grounding:** Replace all variables by constants
  - friends(X,Y) → friends(anna,bob)

Note: We are using Prolog notation

First-Order Logic Review

- **Substition:** Maps variables to constants
  - $\Theta = \{x \rightarrow anna,\ y \rightarrow bob\}$
  - Θ[mother(X,Y) $\rightarrow$ parent(X,Y)] is mother(anna,bob) $\rightarrow$ parent(anna,bob)
- **Literal:** A predicate or its negation
- **Formula:** Sets of literals with $\leftrightarrow, \leftarrow, \rightarrow, \wedge, \vee$
  - sister(X,Y) $\leftrightarrow$ sibling(X,Y)
  - mother(X,Y) $\rightarrow$ parent(X,Y)

First-Order Logic Review

- **Clause:** A disjunction of literals
- **Horn-clause:** Clause with zero or one positive literals
- **Definite-clause:** clause with EXACTLY one positive literal
  - $\neg mother(X,Y) \vee parent(X,Y)$
- **Theory:** Set of clauses

First-Order = Relational Database

PatientID	Gender	Birthday
P1	M	3/22/63

patient(p1,m, 3/22/63)
…

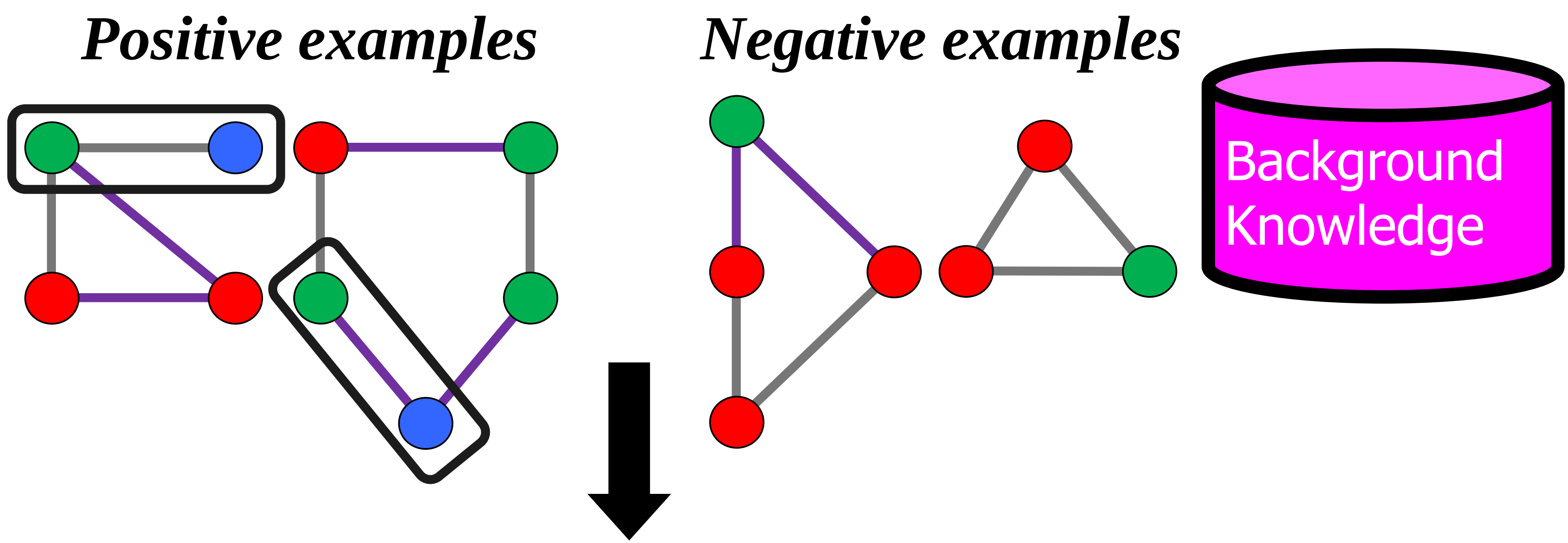
PatientID	Date	Physician	Symptoms	Diagnosis
P1	1/1/01	Smith	palpitations	hypoglycemic
P1	2/1/03	Jones	fever, aches	influenza

diagnosis(p1, 1/1/01, smith, palpitations, hypoglycemic)
diagnosis(p1, 2/1/03, jones, fever, flu)
diagnosis(p1, 2/1/03, jones, aches, flu)
…

Queries in First-Order Logic

- Selection:
  - $millionaire(X) \leftarrow income(X,Y), Y > 1{,}000{,}000$
- Projection:
  - $father(X) \leftarrow father(X,Y)$
- Union:
  - $parent(X,Y) \leftarrow mother(X,Y)$
  - $parent(X,Y) \leftarrow father(X,Y)$
- Can also define cross-product, join, set difference, etc.

Inductive Logic Programming (ILP) [Muggleton & De Raedt, J. Log. Prog.'94]



pos(g) ← edge(G, X, Y), color(X, green), color(Y, blue).

Aleph (Srinivasan), Progol (Muggleton), FOIL (Quinlan), etc.

ILP Problem Definition

Given: Set of positive examples, set of negative examples, background knowledge (BK), all in first-order logic

Do: Learn first-order rule set that when combined with BK, entails the positive examples but not the negative examples

Outline

- Propositional rule induction
- First-order rule induction
  - Motivation and first-order logic
  - FOIL
  - Inverting resolution: Cigol
  - Progol
  - Applications

FOIL: First-Order Inductive Learner

- Same as propositional divide-and-conquer except for:
  - Different candidate specializations
  - Different evaluation function

Specializing Rules in FOIL

Learning rule: $target(x_1,\ldots,x_n) \leftarrow L_1,\ldots,L_m$

Candidate specializations add new literals

- $Q(V_1,\ldots,V_r)$ such that at least one of $V_1,\ldots,V_r$ already appears in the clause
- $Equals(X_i,X_j)$, where X_i, X_j already appear in the clause
- The negation of either of the above literals

Specializing Rules in FOIL

Target: sibling(X,Y) ←

Predicates: brother, sister, father, etc.

Legal refinements

sibling(X,Y) ← sister(X,Y)
sibling(X,Y) ← brother(X,Y)
sibling(X,Y) ← sister(X,Z)
sibling(X,Y) ← sister(Z,Y)
sibling(X,Y) ← equals(X,Y)
sibling(X,Y) ←¬equals(X,Y)
…

Illegal refinements

sibling(X,Y) ← sister(Z,Z)
sibling(X,Y) ← brother(A,B)
sibling(X,Y) ← father(A,B)
…

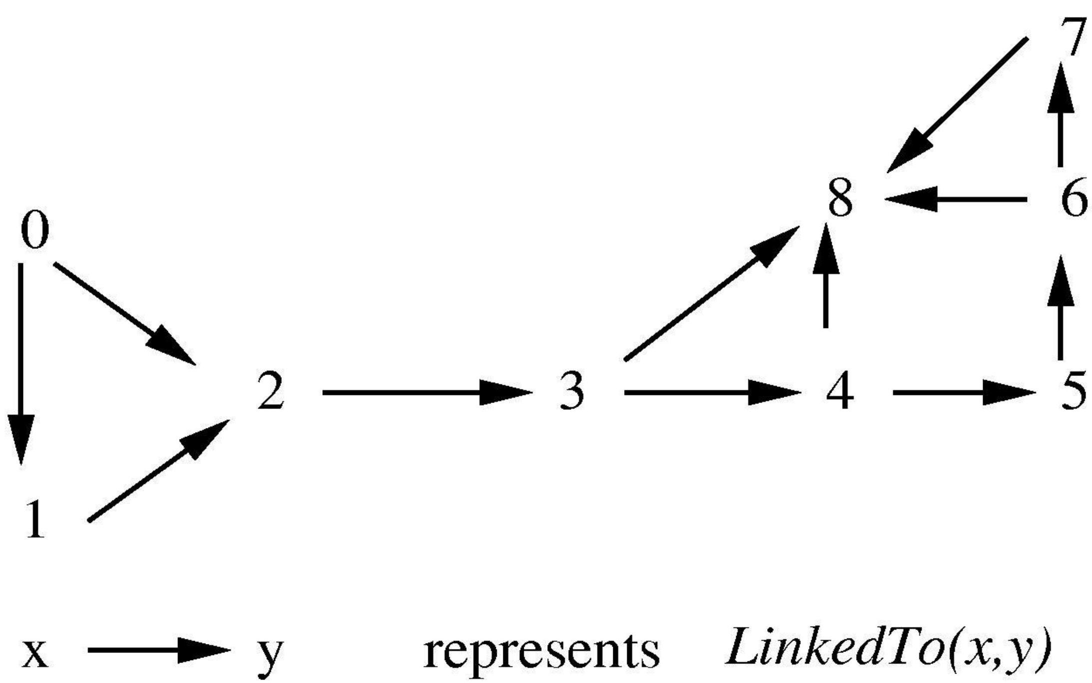
Information Gain in FOIL

$$Foil\_Gain(L,R) \equiv t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

Where

- L is the candidate literal to add to rule R
- $p_0 =$ number of positive bindings of R
- $n_0 =$ number of negative bindings of R
- $p_1 =$ number of positive bindings of $R + L$
- $n_1 =$ number of negative bindings of $R + L$
- $t =$ no. of positive bindings of R also covered by $R + L$

FOIL Example



FOIL Example

Representing negative info:
1. Encode ¬linkedTo(3,1)
2. Absence implies fact is false

Positive Examples	Negative Examples	Background Knowledge
canReach(0,1)	¬canReach(1,0)	linkedTo(0,1)
canReach(0,2)	¬canReach(2,1)	linkedTo(0,2)
canReach(0,3)	¬canReach(2,0)	linkedTo(1,2)
canReach(0,4)	¬canReach(3,2)	linkedTo(2,3)
canReach(0,5)	¬canReach(3,1)	linkedTo(3,4)
…	…	…
canReach(7,8)	¬canReach(8,7)	linkedTo(7,8)

Target function: canReach(X,Y) true iff there is a directed path from X to Y

FOIL Hypothesis Space

Set of clauses using predicates linkedTo and canReach

canReach(X,Y) ←

canReach(X,Y) ← linkedTo(X,Y)

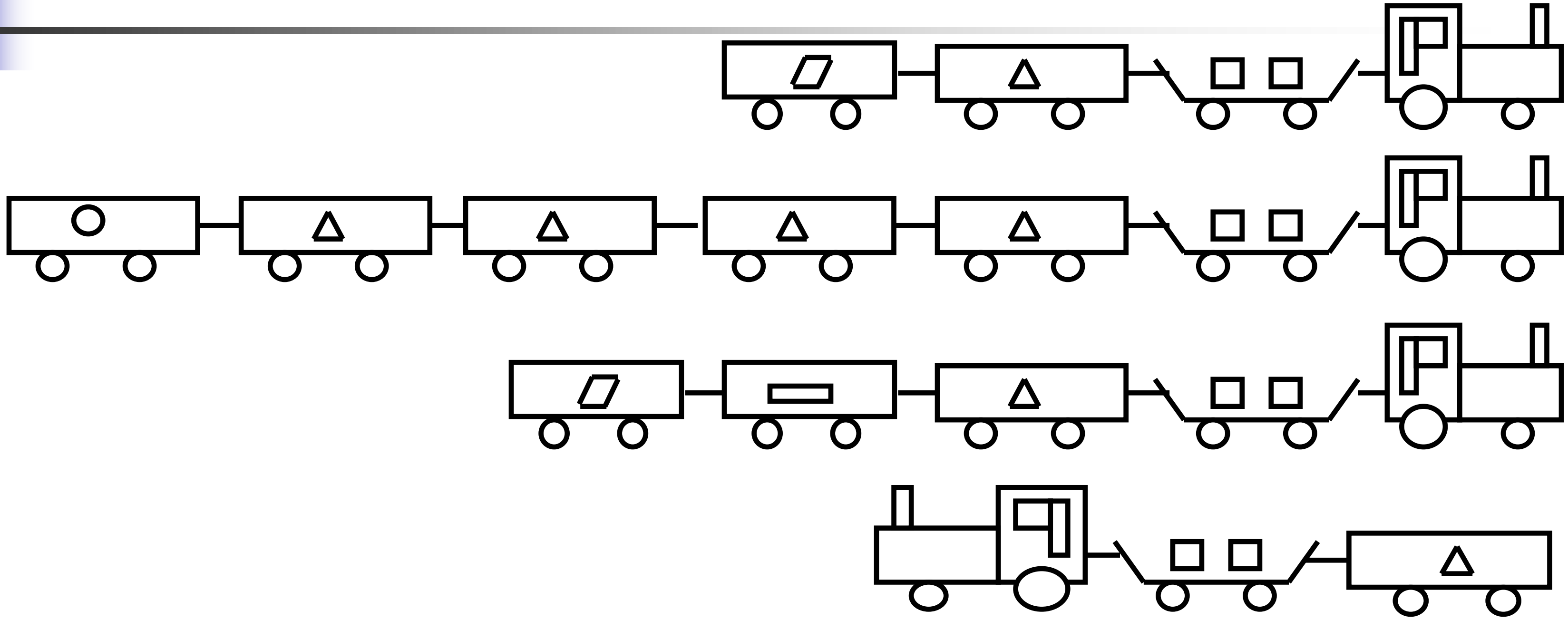
canReach(X,Y) ← linkedTo(Y,X)

…

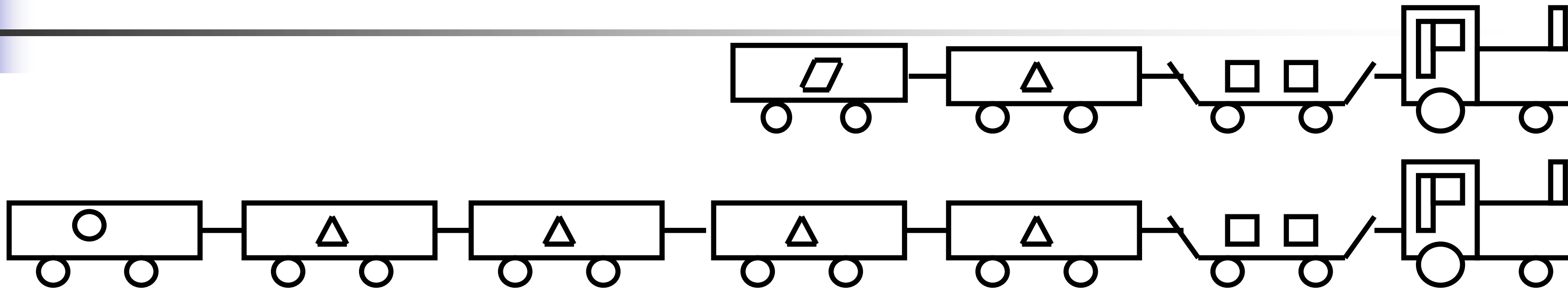
canReach(X,Y) ← linkedTo(X,Z)

Even for small problem the hypothesis space can be large!

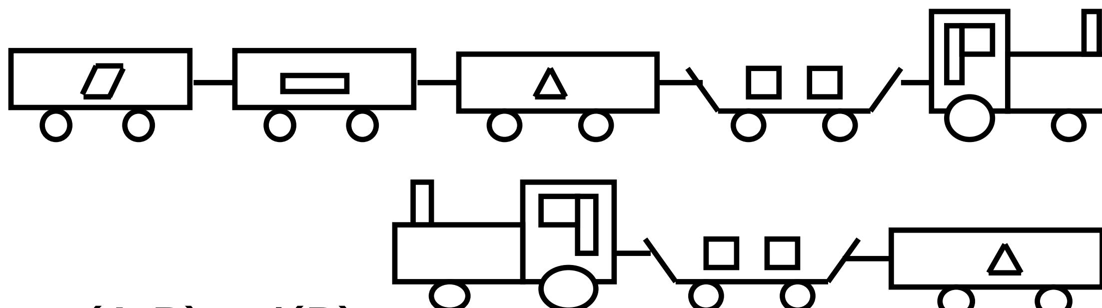
FOIL Gain Subtlety



FOIL Gain Subtlety



right(A) :- car(A,B),
tri(B)



A	B
train1	car2
train2	car2
train2	car3
train2	car4
train2	car5
train3	car2
train4	car2

right(A) :- car(A,B), tri(B),
car(A,C), circle(C)

A	B	C
train2	car2	car6
train2	car3	car6
train2	car4	car6
train2	car5	car6

right(A) :- car(A,B), tri(B),
car(A,C), para(C)

A	B	C
train1	car2	car3
train3	car2	car4

One Problem with FOIL

$\text{happy}(X) \leftarrow \boxed{\text{bankAccount}}(X,Y),\ Y = \text{high}$

This literal is non-discriminating because most people have at least one bank account!

FOIL will have a hard time learning this clause

Overcoming Greedy Search's Myopia

- Solution 1: One-step lookahead
  - Try all pairs of candidate literals
  - Prohibitively slow

- Solution 2: Restrict lookahead to template clauses [i.e., only consider things of a certain form]
  - E.g., R(A,B) ^ S(B,C)

Outline

- Propositional rule induction

- First-order rule induction
  - Motivation and first-order logic
  - FOIL
  - Inverting resolution: Cigol
  - Progol
  - Applications

Types of Logical Reasoning

- Deduction
  - Given: Rule set, antecedents
  - Do: Derive consequents
- Induction:
  - Given: Possible antecedents and conquents
  - Do: Discover rules (map (sets) antecedents to consequents)
- Abduction
  - Given: Rule sets, consequents
  - Do: Infer which antecedents are true

Induction as Inverted Deduction

Induction is finding h such that

$$(\forall \langle x_i, f(x_i)\rangle \in D)\ B \wedge h \wedge x_i \vdash f(x_i)$$

where

- x_i is ith training instance
- $f(x_i)$ is the target function value for x_i
- B is other background knowledge

So let's design inductive algorithm by inverting operators for automated deduction.

Induction as Inverted Deduction

"Pairs of people $\langle u, v \rangle$ such that child of u is v"

$$
\begin{aligned}
f(x_i): &\quad Child(Bob, Sharon) \\
x_i: &\quad Male(Bob), Female(Sharon), Father(Sharon, Bob) \\
B: &\quad Parent(u, v) \leftarrow Father(u, v)
\end{aligned}
$$

What satisfies $(\forall \langle x_i, f(x_i) \rangle \in D)\ B \wedge h \wedge x_i \vdash f(x_i)$?

$$
\begin{aligned}
h_1: &\quad Child(u, v) \leftarrow Father(v, u) \\
h_2: &\quad Child(u, v) \leftarrow Parent(v, u)
\end{aligned}
$$

Induction as Inverted Deduction

We have mechanical *deductive* operators $F(A, B) = C$, where $A \wedge B \vdash C$

Need *inductive* operators

$$O(B, D) = h \text{ where } (\forall \langle x_i, f(x_i) \rangle \in D)\ (B \wedge h \wedge x_i) \vdash f(x_i)$$

Induction as Inverted Deduction

Positives:

- Subsumes earlier idea of finding h that "fits" training data
- Domain theory B helps define meaning of "fit" the data

$$B \wedge h \wedge x_i \vdash f(x_i)$$

- Suggests algorithms that search H guided by B

Induction as Inverted Deduction

Negatives:

- Doesn't allow for noisy data. Consider

$$(\forall \langle x_i, f(x_i) \rangle \in D) \ (B \wedge h \wedge x_i) \vdash f(x_i)$$

- First order logic gives a *huge* hypothesis space H

  $\rightarrow$ Overfitting

  $\rightarrow$ Intractability of calculating all acceptable h's

Deduction: Resolution Rule

- **Step 1:** Given clauses C_1 and C_2 find literal L from C_1 such that $\neg L$ appears in C_2

- **Step 2:** Form resolvent C by including all literals C_1 and C_2 except for L and $\neg L$

- More precisely, the resolvent C is:
  $C = (C_1 - \{L\})\ U\ (C_2 - \{L\})$
  where – is set difference and U is union

Deductive Resolution Example

$$\neg P \vee Q \vee \boxed{\neg S} \vee \neg T$$

$$\neg P \vee Q \vee \boxed{S} \vee \neg T$$

$$\overline{\neg P \vee Q \vee \neg T}$$

Inverted Resolution (Propositional)

- Given clauses C_1 and C, find literal L that occurs in C_1 but not C

- Form clause C_2 by including the following literals
  $C_2 = (C - (C_1 - \{L\})) \cup \{\neg L\}$

Inverting Resolution Example

C_1: PassExam ∨ $\boxed{\text{¬KnowMaterial}}$

C_2: $\boxed{\text{KnowMaterial}}$ ∨ ¬Study

C: PassExam ∨ ¬Study

C_1: PassExam ∨ ¬KnowMaterial

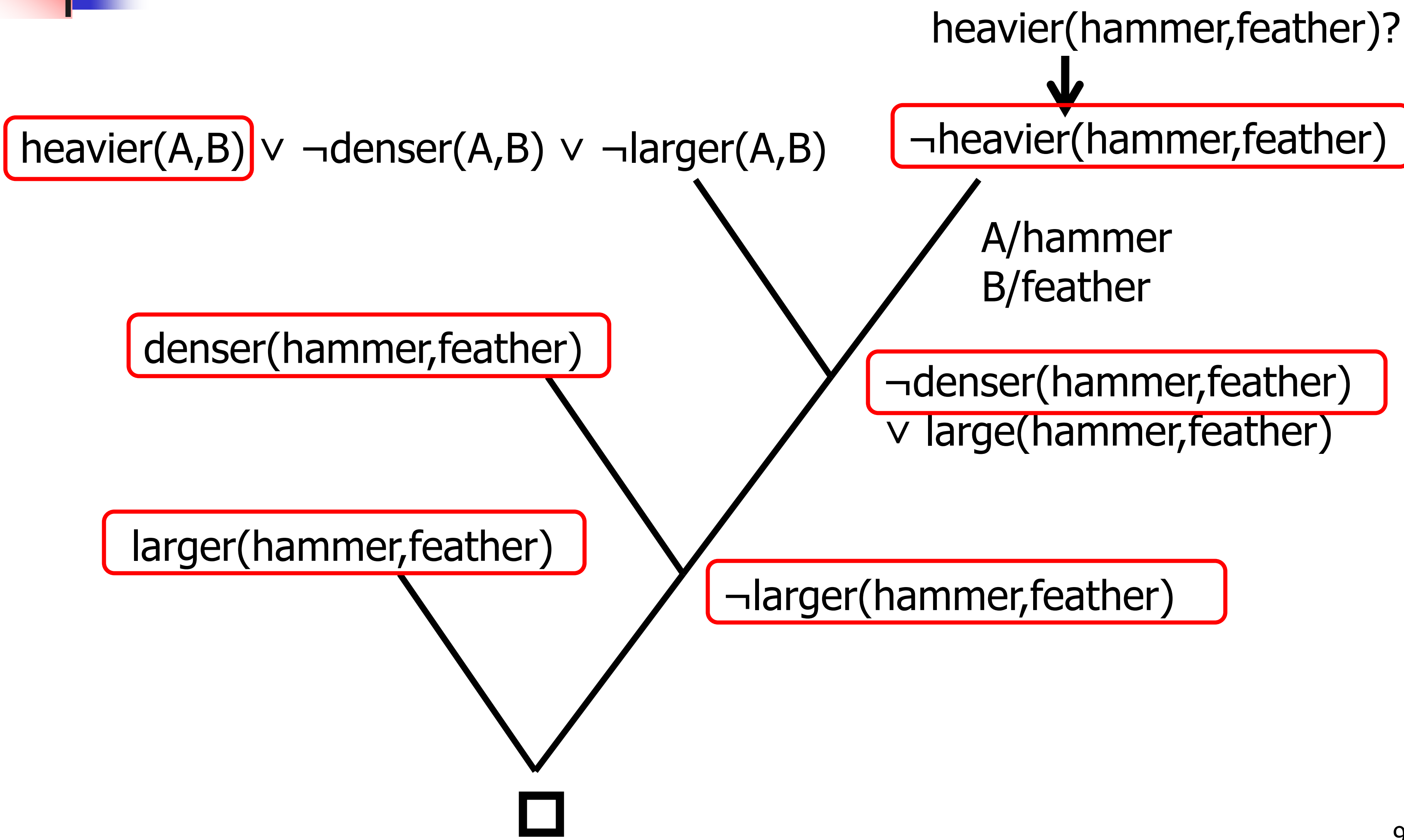
C_2: $\boxed{\text{KnowMaterial}}$ ∨ ¬Study $\boxed{\text{∨ PassExam}}$

C: PassExam ∨ ¬Study

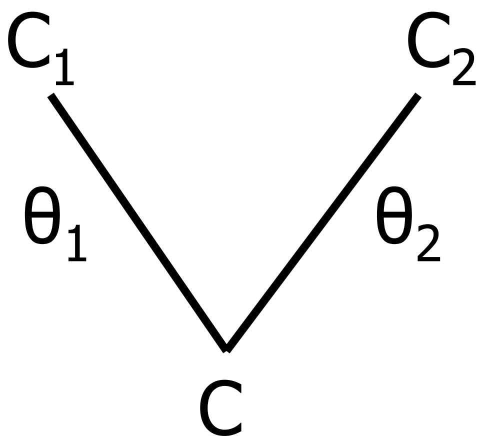
First-Order Resolution

- **Step 1:** Find literal L_1 from clause C_1, literal L_2 from clause C_2 and substitution Θ such that $L_1\Theta = \neg L_2\Theta$
- **Step 2:** Form resolvent C by including all literals
  - From $C_1\Theta$
  - From $C_2\Theta$
  - Except from $L_1\Theta$ and $\neg L_2\Theta$
- More precisely, the resolvent C is: $C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta$

First-Order Resolution Example



Inverting First-Order Resolution



$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta$

$\theta = \theta_1\theta_2$

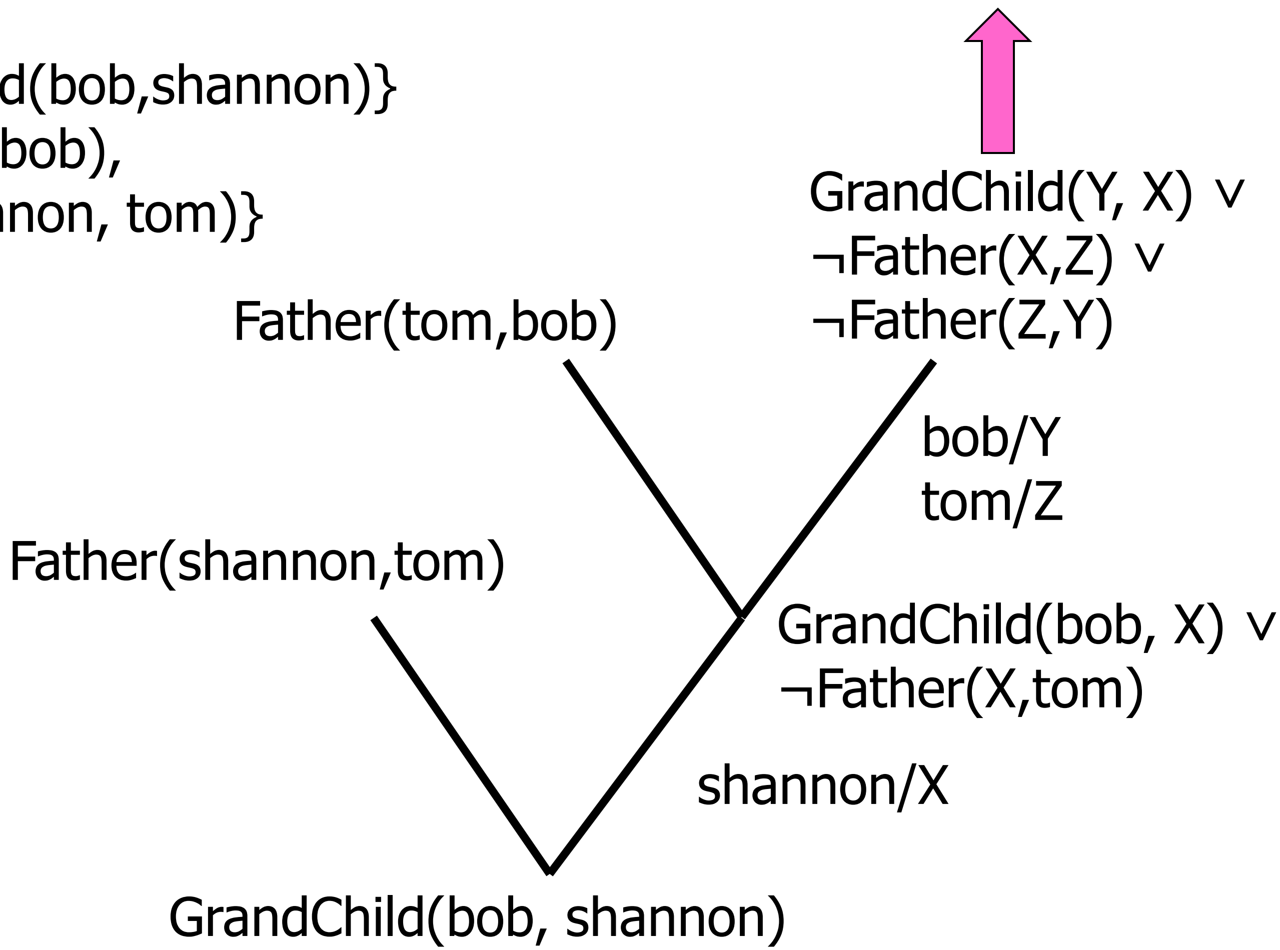
1. $C - (C_1 - \{L_1\})\theta_1 = (C_2 - \{L_2\})\theta_2$
2. $C_2 - \{L_2\} = (C - (C_1 - \{L_1\})\ \theta_1)\theta_2^{-1}$
3. $C_2 = (C - (C_1 - \{L_1\})\ \theta_1)\theta_2^{-1} \cup \{L_2\}$
4. $C_2 = (C - (C_1 - \{L_1\})\ \theta_1)\theta_2^{-1} \cup \{\neg L_1\theta_1\theta_2^{-1}\}$

Cigol

Target: GrandChild(Y, X)

GrandChild(Y, X) :- Father(X,Z), Father(Z,Y)

Pos = {GrandChild(bob,shannon)}
B = {Father(tom,bob),
Father(shannon, tom)}



Cigol: Pros and Cons

- Pros
  - Search can be more focused on effiecent
  - Can perform predicate invention
- Cons
  - Inverse resolution has many choices at each step
  - Does not make full use of data
  - Involves a human in the loop

Outline

- Propositional rule induction
- First-order rule induction
  - Motivation and first-order logic
  - FOIL
  - Inverting resolution: Cigol
  - Progol
  - Applications

Can We Combine Inverse Entailment and Top-Down Search?

- Can view this as mixed top-down and bottom-up approach
- Problem: It is undecidable in general whether:
  - $C1 \models C2$
  - $T \wedge C1 \models example$
- Solution: Use *subsumption* rather than implication

Subsumption for Literals

- Literal L1 subsumes L2 iff there exists a substitution θ such that L1θ= L2

- Examples

  p(f(X),X) subsumes p(f(a),a)

  p(f(X),X) does not subsume p(f(a),b)

Subsumption for Clauses

- Clause C_1 subsumes clause C_2 iff there exists a substitution θ such that $C_1\theta$ subset C_2

- Examples

  p(X,Y) v p(Y,Z) subsumes p(W,W)

  p(X,Y) v p(Y,Z) subsumes p(X,Y) v p(Y,Z) v q(Z)

- If C_1 subsumes C_2 then C_1 implies C_2 (opposite is not true)

Progol

- Learn hypothesis of definite clauses
- User provides sets of predicates, functions, forms of arguments for each
- While positive examples remain uncovered
  - Randomly pick positive example as seed, clause MUST cover this example
  - Build 'bottom' or most specific clause
  - General to specific search
    - If clause covers negative examples, then refine it
    - Score(clause) = |pos covered| - |neg covered|

Learning by Searching a Lattice of Hypotheses

active(X)

active(X) **if**
has-hydrophobic(X,A)

active(X) **if**
has-donor(X,A)

active(X) **if**
has-acceptor(X,A)

active(X) **if**
has-hydrophobic(X,A),
has-donor(X,B),
distance(X,A,B,5.0)

active(X) **if**
has-donor(X,A),
has-donor(X,B),
distance(X,A,B,4.0)

active(X) **if**
has-acceptor(X,A),
has-donor(X,B),
distance(X,A,B,6.0)

etc.

Lattice of Clauses

- We can construct a lattice of clauses
- Ordering is subsumption
  - We group clauses into variants;
  - Top: Call everything positive
  - Bottom: Bottom clause
- Intuition: Top-down search through space of clauses that inverse entailment could generate for an example

Refinement Operators

- Rewrite a clause to make it more specific

- Goals for a refinement operator
  - Finite
  - Complete
  - Not redundant

- Not possible to get all three!

Refinement Operator

- Ground a variable to a constant (function)
- Rename a variable to one that already exists in clause
- Add a new literal that introduces new variables

Note: This is complete and finite

Example

- First, consider IMDB as a problem
- Work in groups for 5 minutes
- Think about
  - What algorithmic approach would you use for this problem?
  - What are the pros and cons of the selected approach?

Outline

- Propositional rule induction

- First-order rule induction
  - Motivation and first-order logic
  - FOIL
  - Inverting resolution: Cigol
  - Progol
  - Applications

Application: Drug Design

- "Drugs" are small molecules that affect disease by binding to a target protein in the body

- Machine Learning is useful in drug design in several ways
  - Identify target proteins using gene expression analysis
  - Find target structure by analyzing X-ray crystallography data (DiMaio et al. 2006)
  - Find potential binding agents by analyzing known drugs (3D-QSAR) (Jain et al. 1994)

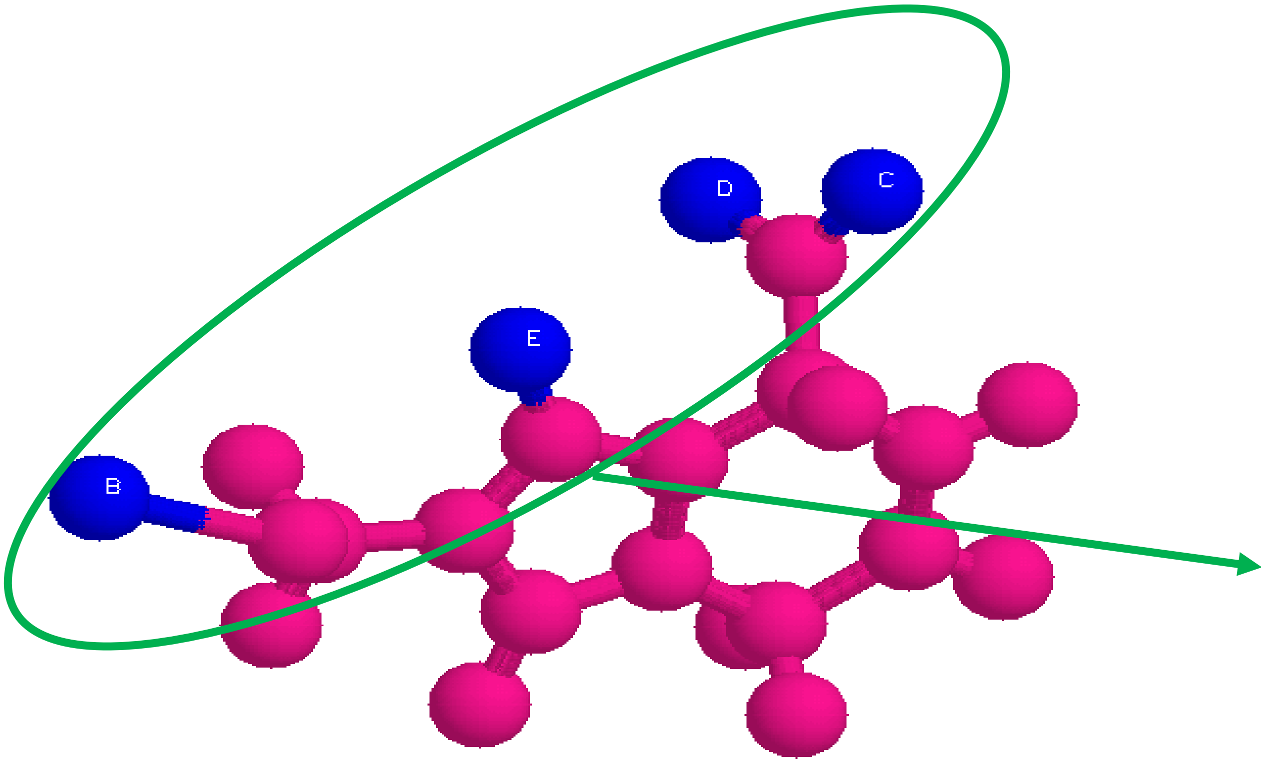
The Problem of 3D-QSAR

3-Dimensional **Q**uantitative **S**tructure-**A**ctivity **R**elationships

Given: 3-dimensional structures of low-energy conformations of molecules, and their known binding affinities to target

Do: Learn a model to predict binding affinity for new molecules to this target

Binding Mechanism

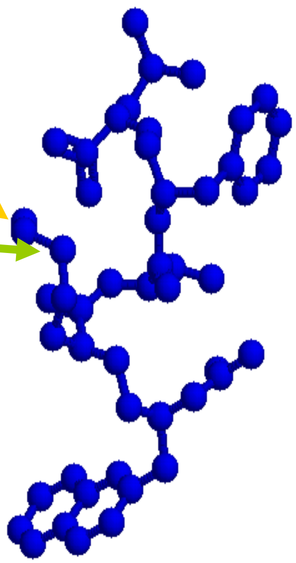


Pharmacophore – the 3D molecular substructure responsible for binding

Conformation of an Angiotensin-Converting Enzyme (ACE) inhibitor

Background Knowledge

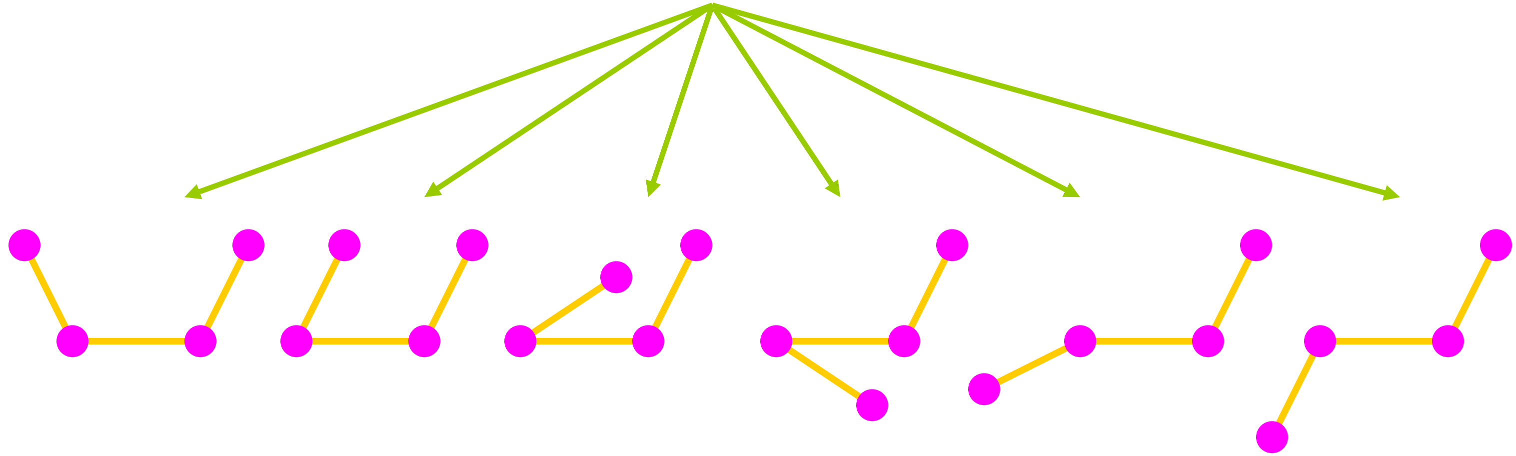
atom(m1,a1,o,6.0,-2.5,1.8).
atom(m1,a2,c,0.6,-2.7,0.3).
atom(m1,a3,s,0.4,-3.5,-1.3).
…
bond(m1,a1,a2,1).
bond(m1,a2,a3,1).
…



- Free to declare rules that use low level molecular descriptions
  - Benzene rings, hydrogen donors, hydrogen acceptors, hydrophobic groups, etc.

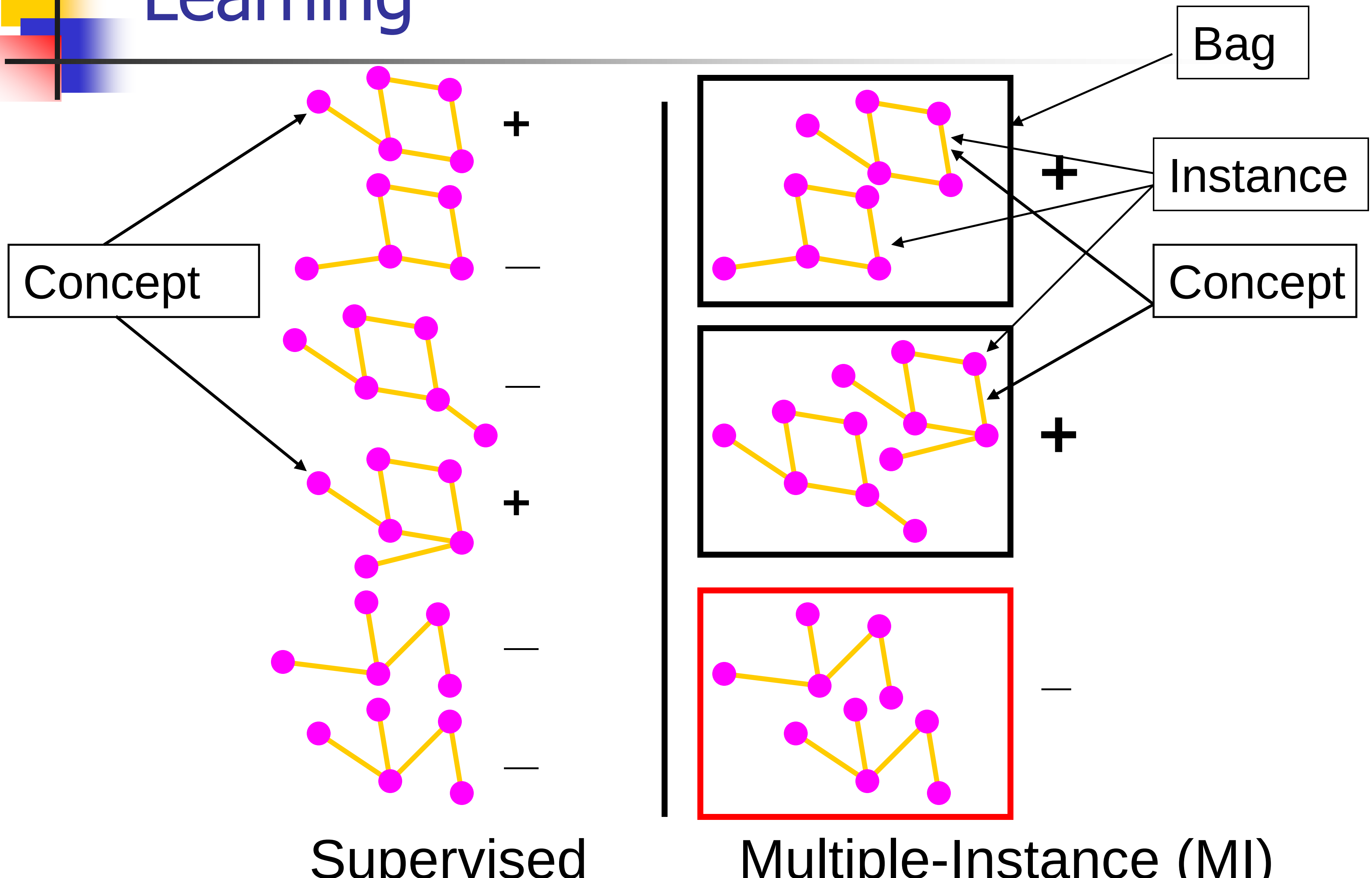
Challenge: Molecular Conformations

C_4H_{10}



Low energy conformations more likely

Supervised vs. Multiple Instance Learning



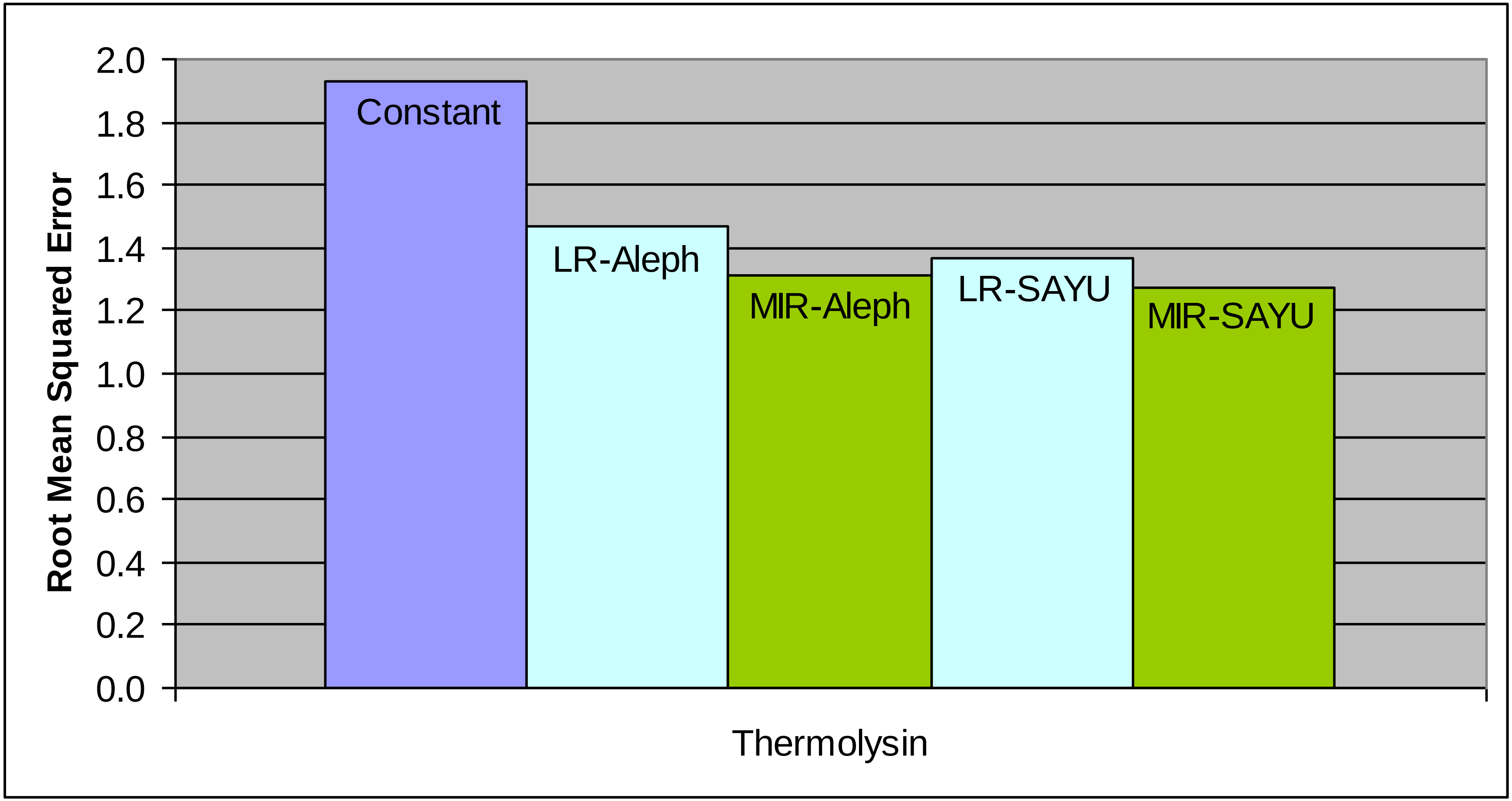
High-Level Idea

- ILP constructs features for a regression model

$$y_i = \sum_k b_k R_{ik} + b_0$$

- Used in practice
  - Gets good results
  - Better ‘hit’ rate
  - Discover biologically relevant information

Thermolysin: Blood Pressure



Application: Mammography

- Provide decision support for radiologists
- Variability due to differences in training and experience
- Experts have higher cancer detection and fewer benign biopsies
- **Shortage of experts**

Approach

Combine three things

- Forward feature selection
- Inductive logic programming to propose features
- Tree-augmented naïve Bayes as statistical model

Simple idea, but excellent performance

Rule Induction Summary

- Rules grown by adding one antecedent at a time to the rule body
  - Many search strategies
  - Many score functions
- Rule sets grown by adding rule at a time
- Rules can either be proposition or first-order
- Alternative idea: Learn by inverting resolution
- Rule learning applied to many real-world apps

Next class

- Homework 2 is due!
- Homework 1 review
- Perceptrons
- Neural networks
- Read Mitchell, Chapter 4

Questions?