Constraint Learning*

Luc De Raedt, Andrea Passerini, Stefano Teso

*senior member track paper in AAAI proceedings

Motivation

- Constraints are ubiquitous in AI
- It can be hard to acquire the model, that is, the right set of constraints
- This tutorial:

  Can we learn the constraints from data ?

- *It is about inductive learning, not about speed-up learning (such as "clause learning" or portfolio's)*

Tacle - Learning Constraints in Tabular Data

$$SERIES(T_1[:, 1])$$
$$T_1[:, 1] = RANK(T_1[:, 5])^*$$
$$T_1[:, 1] = RANK(T_1[:, 6])^*$$
$$T_1[:, 1] = RANK(T_1[:, 10])^*$$
$$T_1[:, 8] = RANK(T_1[:, 7])$$
$$T_1[:, 8] = RANK(T_1[:, 3])^*$$
$$T_1[:, 8] = RANK(T_1[:, 4])^*$$
$$T_1[:, 7] = SUM_{row}(T_1[:, 3{:}6])$$
$$T_1[:, 10] = SUMIF(T_3[:, 1], T_1[:, 2], T_3[:, 2])$$
$$T_1[:, 11] = MAXIF(T_3[:, 1], T_1[:, 2], T_3[:, 2])$$
$$T_2[1, :] = SUM_{col}(T_1[:, 3{:}7])$$
$$T_2[2, :] = AVERAGE_{col}(T_1[:, 3{:}7])$$
$$T_2[3, :] = MAX_{col}(T_1[:, 3{:}7]),$$
$$T_2[4, :] = MIN_{col}(T_1[:, 3{:}7])$$
$$T_4[:, 2] = SUM_{col}(T_1[:, 3{:}6])$$
$$T_4[:, 4] = PREV(T_4[:, 4]) + T_4[:, 2] - T_4[:, 3]$$
$$T_5[:, 2] = LOOKUP(T_5[:, 3], T_1[:, 2], T_1[:, 1])^*$$
$$T_5[:, 3] = LOOKUP(T_5[:, 2], T_1[:, 1], T_1[:, 2])$$

T1

ID	Salesperson	1st Quarter	2nd Quarter	3rd Quarter	4th Quarter	Total	Rank	Label	Items sold total	Max items sold
1	Diana Coolen	353	378	396	387	1514	2	Great	34	20
2	Marc Desmet	370	408	387	386	1551	1	Great	29	10
3	Kris Goossens	175	146	167	203	691	3	Low	19	19
4	Birgit Kenis	93	98	96	105	392	4	Low	17	15

B1 = T1[:, 1] B2 = T1[:, 2] B3 = T1[:, 3:8] B4 = T1[:, 9] B5 = T1[:, 10:11]

T2

Total	991	1030	1046	1081	4148
Average	247.75	257.5	261.5	270.25	1037
Max	370	408	396	387	1551
Min	93	98	96	105	392

B6 = T2[1:4, :]

T3

Salesperson	Items sold
Diana Coolen	5
Marc Desmet	10
Marc Desmet	8
Diana Coolen	9
Birgit Kenis	15
Marc Desmet	8
Birgit Kenis	2
Diana Coolen	20
Marc Desmet	3
Kris Goossens	19

B8 = T3[:, 1] B9 = T3[:, 2]

T4

Quarter	Income	Expenses	Total
Q1	991	212	779
Q2	1030	710	1099
Q3	1046	137	2008
Q4	1081	240	2849

B10 = T4[:, 1] B11 = T4[:, 2:4]

T5

Customer	Contact	Contact Name
Frank	1	Diana Coolen
Sarah	3	Kris Goossens
George	3	Kris Goossens
Mary	2	Diana Coolen
Tim	4	Birgit Kenis

B12 = T5[:, 1] B13 = T5[:, 2] B14 = T5[:, 3]

[Kolb et al. MLJ 17]

ModelSeeker

N. Beldiceanu and H. Simonis

Motivating Example

2	-1	4	-3	6	-5	8	-7	10	-9	12	-11	14	-13	16	-15	18	-17
-8	15	-10	7	-18	9	-4	1	-6	3	-14	13	-12	11	-2	17	-16	5
4	-17	14	-1	12	-13	10	-15	16	-7	18	-5	6	-3	8	-9	2	-11
7	11	-8	15	-16	-18	-1	3	-12	13	-2	9	-10	17	-4	5	-14	6
-3	-13	1	-11	10	16	-15	-17	14	-5	4	-18	2	-9	7	-6	8	12
15	9	-7	13	-14	-12	3	11	-2	17	-8	6	-4	5	-1	18	-10	-16
-17	-5	-15	-18	2	14	-9	-13	7	-11	10	-16	8	-6	3	12	1	4
11	18	5	9	-3	-10	17	12	-4	6	-1	-8	-15	16	13	-14	-7	-2
-13	-6	-17	-5	4	2	-11	-9	8	-16	7	14	1	-12	-18	10	3	15
9	16	11	6	-8	-4	13	5	-1	12	-3	-10	-7	18	17	-2	-15	-14
-5	-12	-13	-16	1	7	-6	-18	15	-14	17	2	3	10	-9	4	-11	8
6	14	9	12	-7	-1	5	16	-3	18	-15	-4	-17	-2	11	-8	13	-10
-18	-10	-12	-14	15	8	-16	-6	17	2	13	3	-11	4	-5	7	-9	1
12	-7	18	10	-17	-15	2	14	-11	-4	9	-1	16	-8	6	-13	5	-3
-14	4	-16	-2	11	17	-18	-10	13	8	-5	15	-9	1	-12	3	-6	7
10	-8	6	-17	-9	-3	12	2	5	-1	16	-7	18	-15	14	-11	4	-13
-16	3	-2	8	13	11	-14	-4	-18	15	-6	17	-5	7	-10	1	-12	9
-2	1	-4	3	-6	5	-8	7	-10	9	-12	11	-14	13	-16	15	-18	17
8	-15	10	-7	18	-9	4	-1	6	-3	14	-13	12	-11	2	-17	16	-5
-4	17	-14	1	-12	13	-10	15	-16	7	-18	5	-6	3	-8	9	-2	11
-7	-11	8	-15	16	18	1	-3	12	-13	2	-9	10	-17	4	-5	14	-6
3	13	-1	11	-10	-16	15	17	-14	5	-4	18	-2	9	-7	6	-8	-12
-15	-9	7	-13	14	12	-3	-11	2	-17	8	-6	4	-5	1	-18	10	16
17	5	15	18	-2	-14	9	13	-7	11	-10	16	-8	6	-3	-12	-1	-4
-11	-18	-5	-9	3	10	-17	-12	4	-6	1	8	15	-16	-13	14	7	2
13	6	17	5	-4	-2	11	9	-8	16	-7	-14	-1	12	18	-10	-3	-15
-9	-16	-11	-6	8	4	-13	-5	1	-12	3	10	7	-18	-17	2	15	14
5	12	13	16	-1	-7	6	18	-15	14	-17	-2	-3	-10	9	-4	11	-8
-6	-14	-9	-12	7	1	-5	-16	3	-18	15	4	17	2	-11	8	-13	10
18	10	12	14	-15	-8	16	6	-17	-2	-13	-3	11	-4	5	-7	9	-1
-12	7	-18	-10	17	15	-2	-14	11	4	-9	1	-16	8	-6	13	-5	3
14	-4	16	2	-11	-17	18	10	-13	-8	5	-15	9	-1	12	-3	6	-7
-10	8	-6	17	9	3	-12	-2	-5	1	-16	7	-18	15	-14	11	-4	13
16	-3	2	-8	-13	-11	14	4	18	-15	6	-17	5	-7	10	-1	12	-9

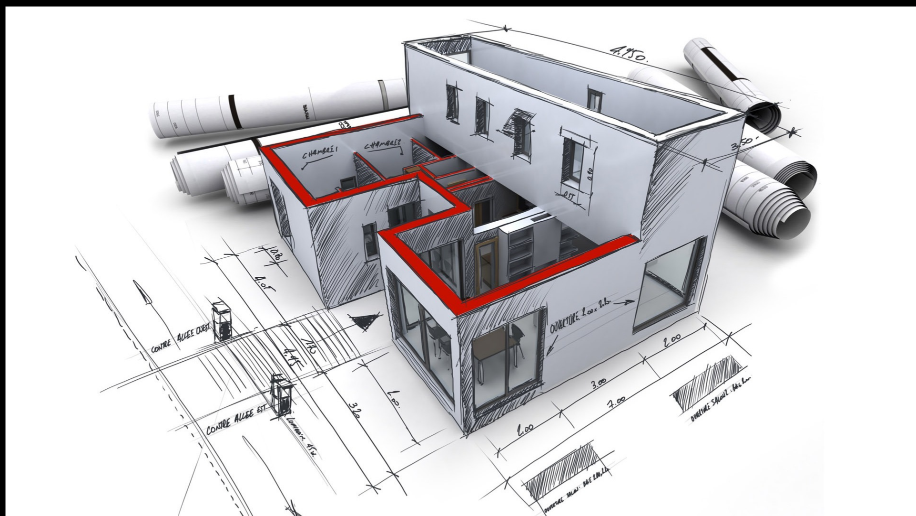
http://www.weltfussball.de/alle_spiele/bundesliga-2010-2011/

Result

J	Scheme	Ref	Trans	Constraint
1	scheme(612,34,18,1,18)	284	absolute_value	symmetric_alldifferent([1..18])*34
2	vector(612)	289	id	global_cardinality([-18.. -1-17,0-0,1..18-17])*1
3	scheme(612,34,18,34,1)	288	id	alldifferent*18
4	repart(612,34,18,17,18)	282	id	alldifferent*306
5	scheme(612,34,18,2,2)	286	id	alldifferent*153
6	scheme(612,34,18,1,18)	284	id	alldifferent*34
7	repart(612,34,18,34,9)	283	sign	alldifferent*306
8	scheme(612,34,18,17,1)	287	absolute_value	alldifferent*36
9	scheme(612,34,18,2,1)	285	absolute_value	alldifferent*306
10	repart(612,34,18,34,9)	283	id	sum_ctr(0)*306
11	repart(612,34,18,34,9)	283	id	sum_cubes_ctr(0)*306
12	scheme(612,34,18,1,18)	284	id	sum_squares_ctr(2109)*34
13	repart(612,34,18,34,9)	283	id	twin*1
14	repart(612,34,18,34,9)	283	id	elements([*i*,-*i*])*1
15	modulo(612,4)	281	id	all_differ_from_at_least_k_pos(152)*1
16	first(9,[1,3,5,7,9,11,13,15,17])	280	id	strictly_increasing*1
17	repart(612,34,18,34,9)	283	id	alldifferent_interval(2)*306
18	scheme(612,34,18,2,1)	285	id	alldifferent_interval(2)*306
19	repart(612,34,18,34,9)	283	sign	sum_ctr(0)*306
20	scheme(612,34,18,1,18)	284	sign	sum_ctr(0)*34
21	repart(612,34,18,34,9)	283	sign	twin*1
22	repart(612,34,18,34,9)	283	absolute_value	twin*1
23	repart(612,34,18,34,9)	283	sign	elements([*i*,-*i*])*1
24	repart(612,34,18,34,9)	283	absolute_value	elements([*i*,*i*])*1
25	first(9,[1,3,5,7,9,11,13,15,17])	280	absolute_value	strictly_increasing*1
26	first(6,[1,4,7,10,13,16])	279	absolute_value	strictly_increasing*1
27	repart(612,34,18,34,9)	283	sign	alldifferent_interval(2)*306
28	scheme(612,34,18,34,1)	288	sign	among_seq(3,[-1])*18

Slides N. Beldiceanu and H. Simonis

Layout Synthesis

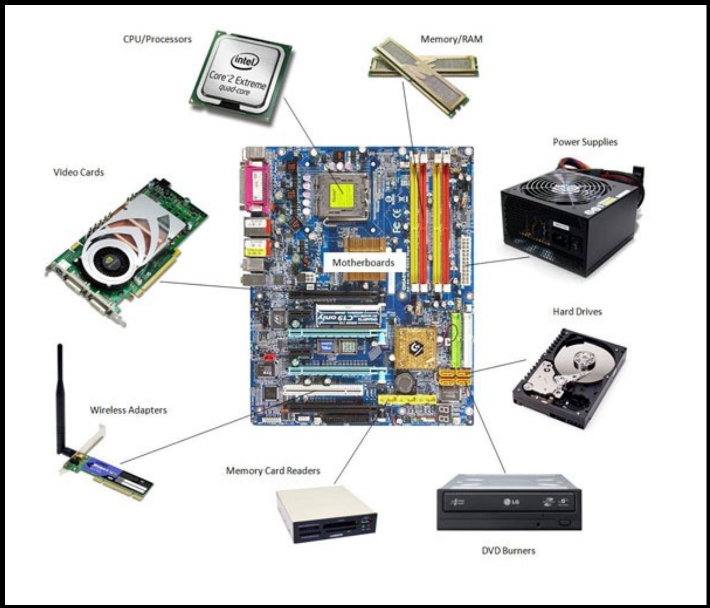


Building Design

Interior Design

Urban Planning

Preference Elictation



What is the best PC configuration based on constraints and preferences ?

Three parts

1. learning hard constraints from data (Luc)

    logical formulae

2. learning soft constraints (Andrea)

    probability, preferences and optimisation

3. interactive constraint learning (Stefano)

Part 1 : Learning Hard Constraints

Overview

- What are constraint satisfaction problems ?
- What is learning ?
- Boolean Concept-Learning for Constraints
- Variations and extensions
  - active learning
  - k-CNF and learning in first order logic
  - Actual system : equation discovery
  - Actual system : ModelSeeker and Tacle

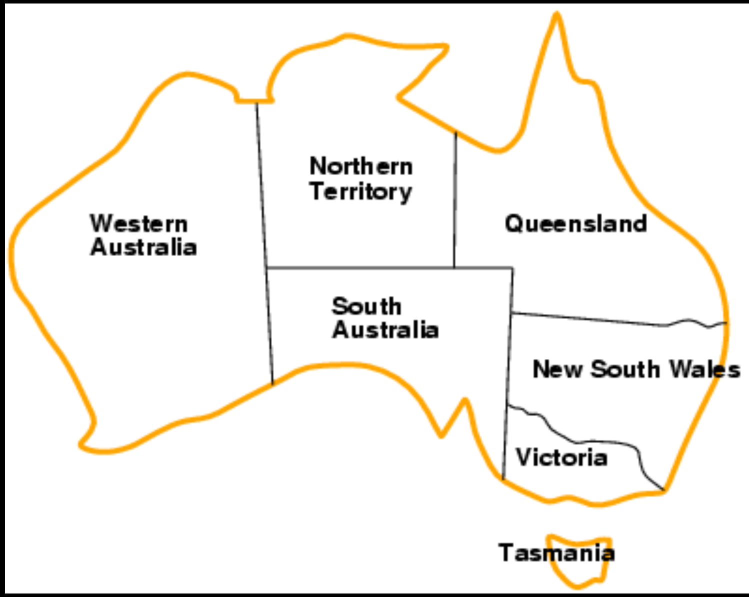
Why hard constraints ?

- Used in constraint programming, answer set programming, operations research, SAT and variants…
- Optimisation versions exist (cf. Parts II and III)

What are Constraint Satisfaction Problems ?

Map Colouring

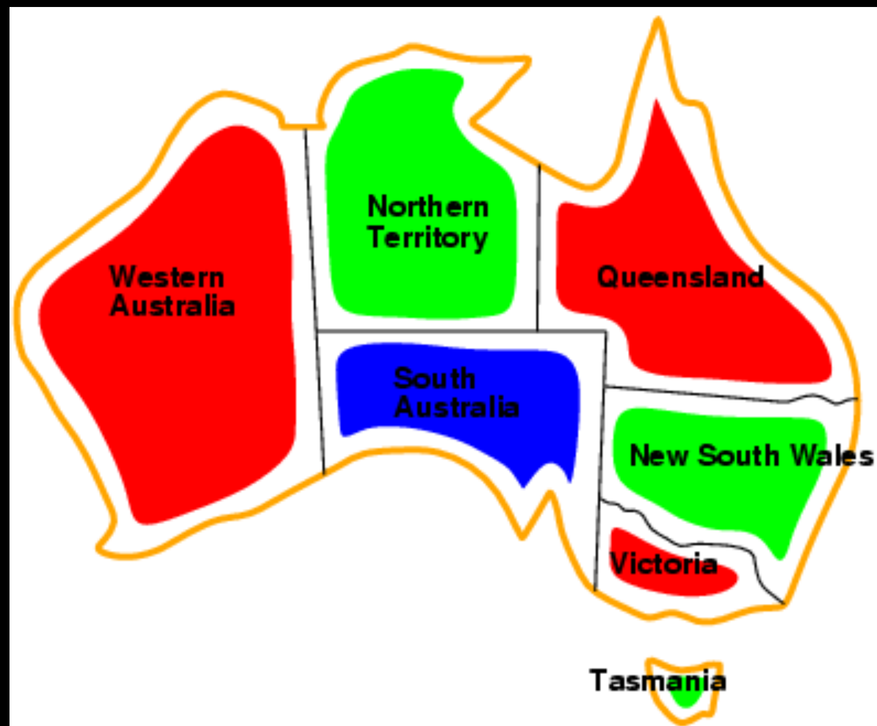
Traditional CSP(V,D,C)



- Variables WA, NT, Q, NSW, V, SA, T
- Domains D_i = {red,green,blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT, or (WA,NT) in {(red,green),(red,blue),(green,red),(green,blue),(blue,red),(blue,green)}
- Optimisation : *sometimes — find best solution according to some optimisation criterion* CSPo(V,D,C,f)

example + figures [Marriot & Stuckey]

Map Colouring



- Solutions are complete and consistent assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

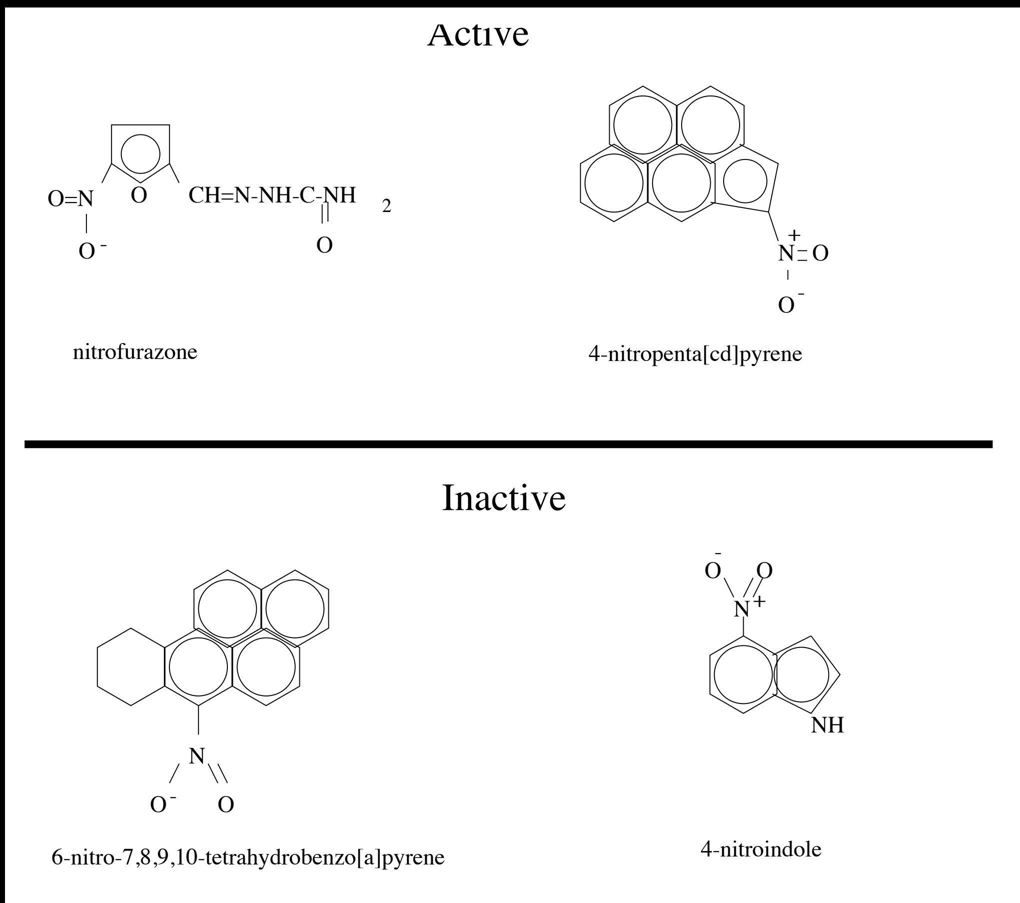
example + figures [Marriot & Stuckey]

Sudoku as a CSP

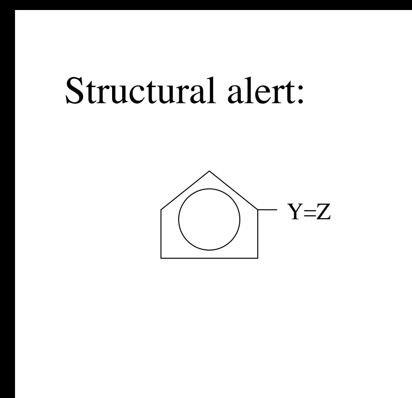
- 81 Variables
  - V= {A1, A2, … , I8, I9}
- Domain(V) = {1, …, 9}
  - for all variables V
- 27 Alldifferent Constraints
  - alldiff(A1, … ,A9)
  - for each row, collum and block

	1	2	3	4	5	6	7	8	9
A	1								6
B			6		2		7		
C	7	8	9	4	5		1		3
D				8		7			4
E					3				
F		9				4	2		1
G	3	1	2	9	7			4	
H		4			1	2		7	8
I	9		8						

What is machine learning ?



[Srinivasan et al. AIJ 96]



Data = Set of Small Graphs

Structure Activity Relationship Prediction

Machine Learning

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space H containing functions $X \rightarrow Y$
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$
- a loss function $loss(h,E) \rightarrow \mathbb{R}$ (or objective function)

Find $h \in H$ that minimizes $loss(h,E)$

supervised

Classification

Given - Molecular Data Sets

- a space of possible instances X -- Molecular Graphs
- an unknown target function f: X → Y -- {Active,Inactive}
- a hypothesis space L containing functions X → Y --
  H= {Active iff structural alert s covers instance x ∈ X|s ∈ X }
- a set of examples E = { (x, f(x)) | x ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$ |{ x ∈ E | f(x) ≠ h(x)}|

Find h ∈ H that minimizes *loss*(h,E)

If classes = {positive, negative} then this is concept-learning

Regression

Given - Molecular Data Sets

- a space of possible instances X -- Molecular Graphs
- an unknown target function f: X → Y -- $\mathbb{R}$
- a hypothesis space H containing functions X → Y -- a linear function of some features
- a set of examples E = { (x, f(x)) | x ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$ $\sqrt{\sum_{x \in E} f(x)^2 - h(x)^2}$

Find h ∈ H that minimizes *loss*(h,E)

Learning Preferences

Given

- a space of possible instances X
- an unknown target function Pref: X → Y   Y=Reals
- a hypothesis space H containing functions h: X → Y
- a set of examples

  E = { (x,y, Pref(x) > Pref(y)) | x,y ∈ X }

- a loss function *loss*(h,E) → $\mathbb{R}$

Find h ∈ H that minimizes *loss*(h,E)

Learning Probabilistic Models

Given

- a space of possible instances X
- an unknown target function P: X → Y Y=[0,1]
- a hypothesis space L containing functions X → Y (graphical models)
- a set of examples E = { (x, _) | x ∈ X } generative
- a loss function *loss*(h,E) → $\mathbb{R}$

$$\prod_{e \in E} P(e|h)$$

Find h ∈ L that minimizes *loss*(h,E)

maximize likelihood

generative

Boolean Concept-Learning

Boolean Concept-Learning

$X = \{(X_1, ..., X_n) \mid X_i = 0 / 1\}$

$Y = \{+,-\}$

H = boolean formulae

loss(h,E) = training set error

$= | \{e \mid e \in E, h(e) \neq f(e)\} | \;/\; |E|$

sometimes required to be 0

Simplest setting for learning, compatible with constraint learning and Valiant's PAC-learning

Dimensions

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space H containing functions X → Y

k-CNF ?
DNF ?
SMT ? etc

- a set of examples E = { (x, f(x)) | x ∈ X }

pos and neg ?
or pos only ?
or preferences ?

- a loss function *loss*(h,E) → $\mathbb{R}$ loss/error=0 required ?

Find h ∈ H that minimizes *loss*(h,E)

ability to ask questions ?

[Kearns and Vazzirani; Mitchell]

Various Settings and Algorithms

Depending on

- hypotheses space H (Part I)

  monomials vs k-CNF vs k-term DNF vs SMT …

- loss or objective function (Part II)

  completeness and consistency required

  (error training set =0)? or noise tolerated ?

- type of examples (Part II/III)

  positives only ? or positives and negatives ? or preferences ?

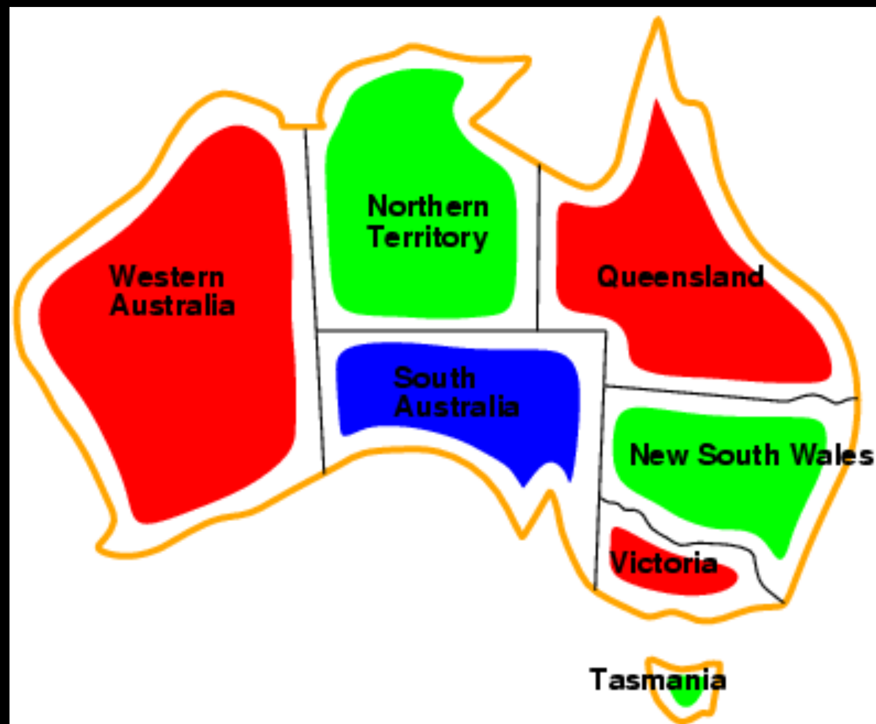
- interaction with user (oracle) / active learning (Part III)

Boolean concept-learning

	1	2	3	4	5		
ex 1	0	1	1	1	0	..	+
ex 2	1	1	1	1	1		+
ex 3	0	1	1	0	0		-
ex 4	1	0	0	1	0		-
...							

possible concept: (X_2 and X_4)

Map Colouring



- Solutions are complete and consistent assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

example + figures [Marriot & Stucket]

Why boolean concept-learning ? constraint networks

(V_1, V_2, V_3)	$V_1 < V_2$	$V_1 > V_2$	$V_1 = V_2$	$V_1 < V_3$	
(1,2,3)	1	0	0	1	
(2,3,1)	1	0	0	0	
(3,2,1)	0	1	0	0	
(1,3,2)	1	0	0	1	
...					

Bias

Propositionalization

Assignments

CONACQ example *[Bessiere et al.]*

Monomials

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$ monomials conjunctions
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$ pos only
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$ error = 0

Find $h \in L$ that minimizes *loss*(h,E)

Learning monomials

Represent each example by its set of literals

$\{\neg X_1, X_2, \neg X_3, X_4, \neg X_5\}$

Compute the intersection of all positive examples

intersection = *least general generalization*

A cautious algorithm - Find-S algorithm *[Mitchell, ML textbook 97]*

Makes prudent generalizations

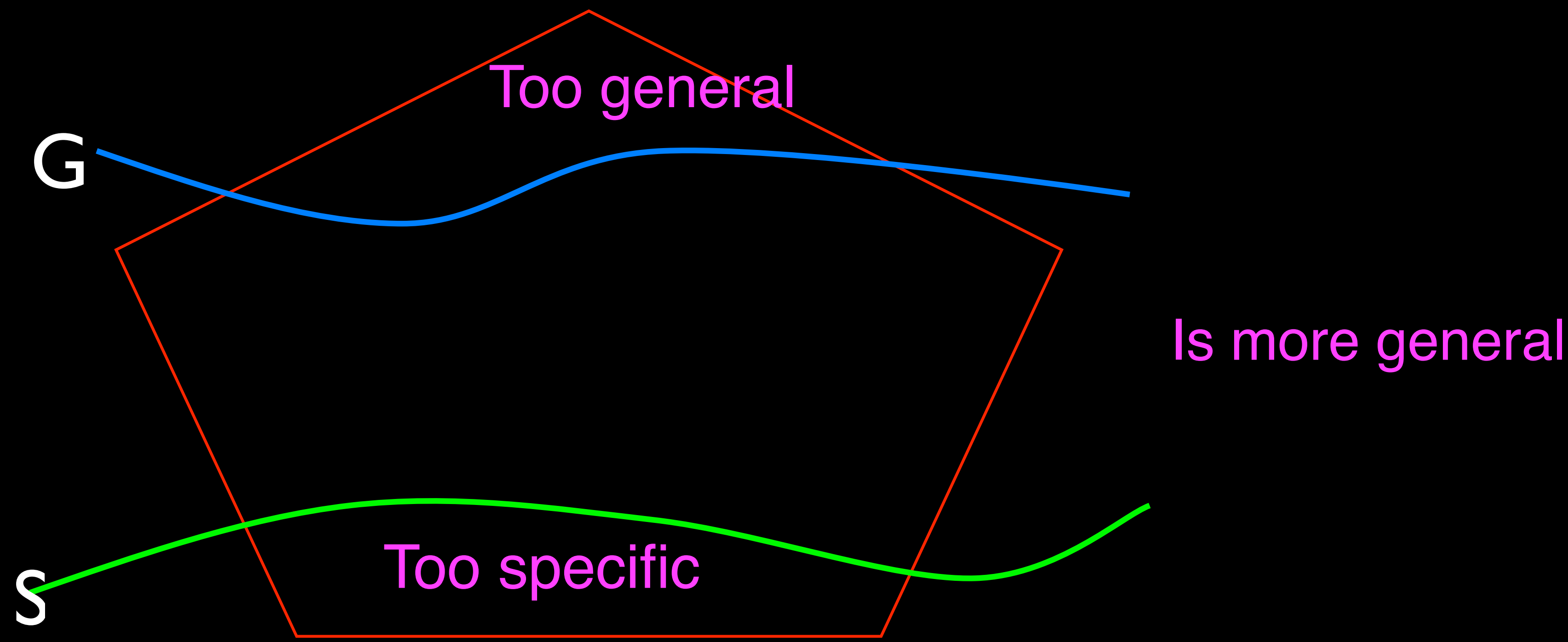
Complaints about Find-S

- Can't tell whether it has learned concept
- Can't tell when training data inconsistent
- Picks a maximally specific h (why?)
- Depending on H, there might be several.

- Could be alleviated with Versionspace Approach

Slide by [Mitchell 97]

Mitchell's Versionspace



G = $\{h \mid$ 1) $h \in$ H, 2) h is complete and consistent with all examples, and there is no g that satisfies 1) and 2) and is strictly more general than $h\}$

S = $\{h \mid$ 1) $h \in$ H, 2) h is complete and consistent with all examples, and there is no s that satisfies 1) and 2) and is strictly more specific than $h\}$

Generality for monomials

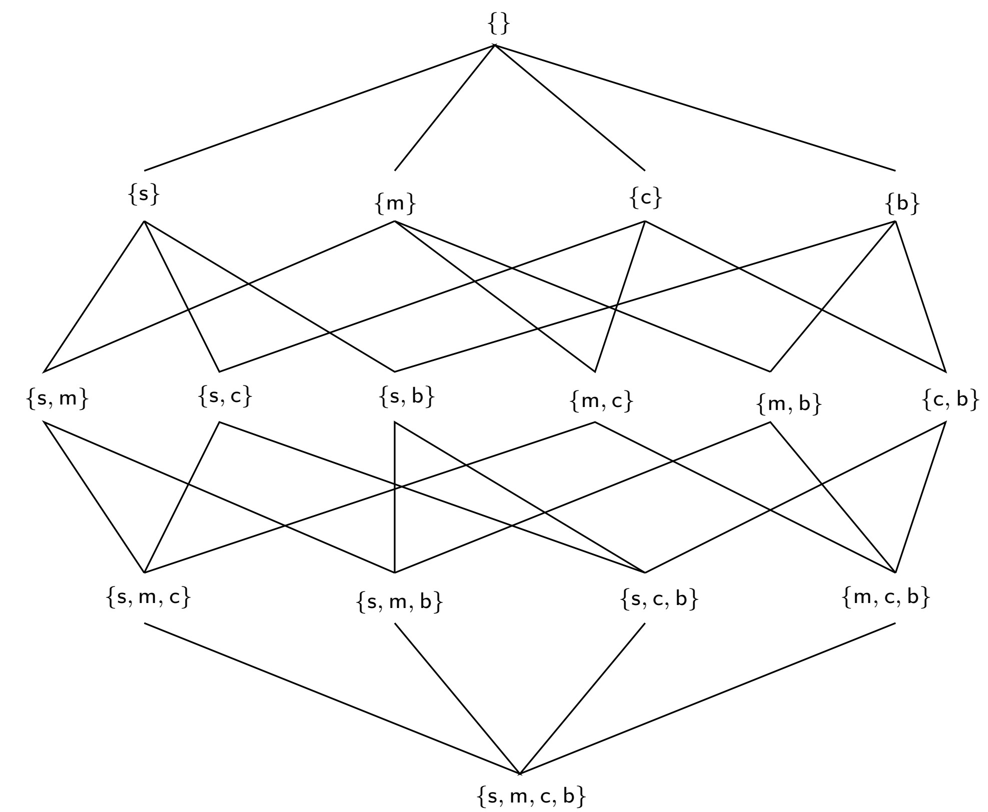


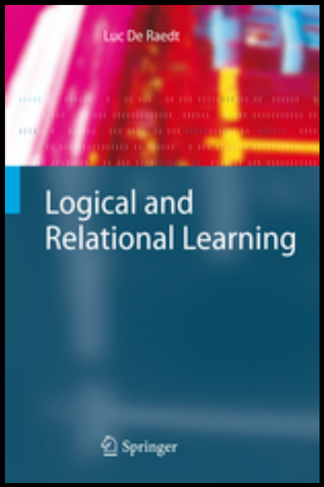
Fig. 3.5. The partial order over the item-sets

{s, m, c} = s and m and c

S ⊧ G if and only if G ⊆ S

just like item-sets in data mining

Generality



Two difficulties

1) $x = y \rightarrow x \geq y$ more general

many solutions can be found

preference for most specific one (Find-S)

2) $x = y$ and $y = z \rightarrow x = z$

therefore $x = y$ and $y = z$ and $x = z$ redundant

equivalent with $x = y$ and $y = z$

Consider now Sudoko ...

Many solutions syntactically different, but semantically equivalent

Variations and Extensions

Asking Queries
Interactive Learning

Provide the learner with the opportunity to ask questions

Let T be the (unknown) target theory

- Does x satisfy T ? (membership)
- Does T |= X ? (subset)
- Does X |= T ? (superset)
- Are T and X logically equivalent ? (equivalence)
- ...

The oracle has to provide a counter-example in case the answer is negative *[Angluin, MLJournal 88]*

How can we use this?

Reconsider learning monomials (cf. *[Mitchell],* Conacq *[Bessiere et al] and Quacq [Bessiere et al.]*)

Current hypothesis / conjunction

- $\{\neg X_1, X_2, \neg X_3, X_4, \neg X_5\}$
- generate example $\{X_1, X_2, \neg X_3, X_4, \neg X_5\}$
- if positive, delete X_1, if negative, keep
- only n+1 questions needed to converge on unique solution (mistake bound)
- a lot harder when there are redundancies (cf. generality)

Very interesting polynomial time algorithms for learning horn sentences *[Angluin et al. MLJ 92]* by asking queries

DNF / rule learning

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space H containing functions X → Y   DNF
- a set of examples E = { (x, f(x)) | x ∈ X } pos   pos and neg
- a loss function *loss*(h,E) → $\mathbb{R}$ error need not be 0

Find h ∈ H that minimizes *loss*(h,E)

[Fuernkranz, AI Review 99, book 12]

Rule learning

Learning from Positives and Negatives

Learn a formula in Disjunctive Normal Form

Rule learning algorithms (machine learning)

Rule learning is often heuristic

Set-covering algorithm

- repeatedly search for one rule (conjunction) that covers many positives and no negative
- discard covered positive examples and repeat

[Fuernkranz, AI Review 99, book 12]

- separate and conquer ... [Fuernkranz]

k-CNF

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space H containing functions X → Y   k-CNF
- a set of examples E = { (x, f(x)) | x ∈ X }   pos only
- a loss function *loss*(h,E) → $\mathbb{R}$

Find h ∈ H that minimizes *loss*(h,E)

Learning k-CNF

Naive Algorithm *[Valliant CACM 84]*

- Let S be the set of all clauses with k literals
- for each positive example e
  - for all clauses s in S
    - if e does not satisfy s then remove s from S

polynomial (for fixed k) -- PAC-learnable

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Initial Theory

A v B v C

not A v B v C

A v not B v C

A v B v not C

not A v not B v C

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & not C

A v B v C

not A v B v C

A v not B v C

A v B v not C

not A v not B v C

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & not C

A v B v C

not A v B v C

A v not B v C

A v B v not C

~~not A v not B v C~~

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & C

A v B v C

not A v B v C

A v not B v C

A v B v not C

~~not A v not B v C~~

A v not B v not C

not A v B v not C

not A v not B v not C

Example

Suppose k = 3 and three variables A, B, C and

target = A v B v C and not A v not B v not C

Example A & B & C

A v B v C

not A v B v C

A v not B v C

A v B v not C

~~not A v not B v C~~

A v not B v not C

not A v B v not C

~~not A v not B v not C~~

Formal Frameworks Exist

Probably Approximately Correct learning (PAC)

requires that learner finds with high probability approximately correct hypotheses

So, $P(\ loss_t(h,X) < \varepsilon) > 1\text{-}\delta$

Typically combined with complexity requirements

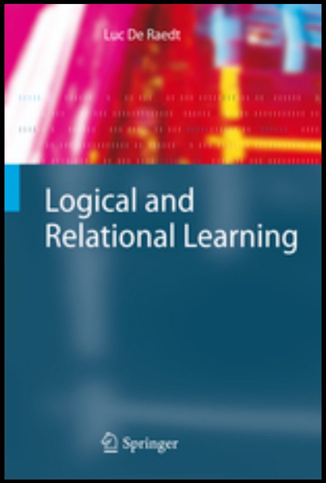
sample complexity: number of examples

computational complexity

Valliant proved polynomial PAC-learnability k-CNF (fixed k)

[Valiant 84, Kearns and Vazzirani]

Inductive Logic Programming



Instead of learning propositional formulae, learn first order logic formulae

Usually (definite) clausal logic

Generalizations of many algorithms exist

Rule learning, decision tree learning

Clausal discovery *[De Raedt MLJ 97, De Raedt AIJ 94]*

- generalizes k-CNF of Valliant to first order case
- enumeration process as for k-CNF with border ...

Clausal Discovery

train(utrecht, 8, 8, denbosch) ←
train(maastricht, 8, 10, weert) ←
train(utrecht, 9, 8, denbosch) ←
train(maastricht, 9, 10, weert) ←
train(utrecht, 8, 13, eindhoven) ←
train(utrecht, 8, 43, eindhoven) ←
train(utrecht, 9, 13, eindhoven) ←
train(utrecht, 9, 43, eindhoven) ←

train(tilburg, 8, 10, tilburg) ←
train(utrecht, 8, 25, denbosch) ←
train(tilburg, 9, 10, tilburg) ←
train(utrecht, 9, 25, denbosch) ←
train(tilburg, 8, 17, eindhoven) ←
train(tilburg, 8, 47, eindhoven) ←
train(tilburg, 9, 17, eindhoven) ←
train(tilburg, 9, 47, eindhoven) ←

From1 = From2 :- train(From1, Hour1, Min, To), train(From2, Hour2, Min, To)

Inducing constraints that hold in data points
here functional dependencies

*[De Raedt 97 MLJ, Flach AIComm 99,
Abdennaher CP 00, Lopez et al ICTAI 10, ...]*

Learning (k)-CNF

Alternative algorithm using principles

- generality for clauses $S \models G$ if and only if $S \subseteq G$
- clauses are disjunctions; monomials conjunctions
- monotonicity property :
  - if e satisfies clause C then e satisfies C v lit
  - interest in smallest clauses that satisfy all the examples
  - interest in all of them (the others are implied)
- find upper border ... (G set)

Generality for clauses

Direction of generality changes

Most general clause is bottom,

Top is unsatisfiable clause

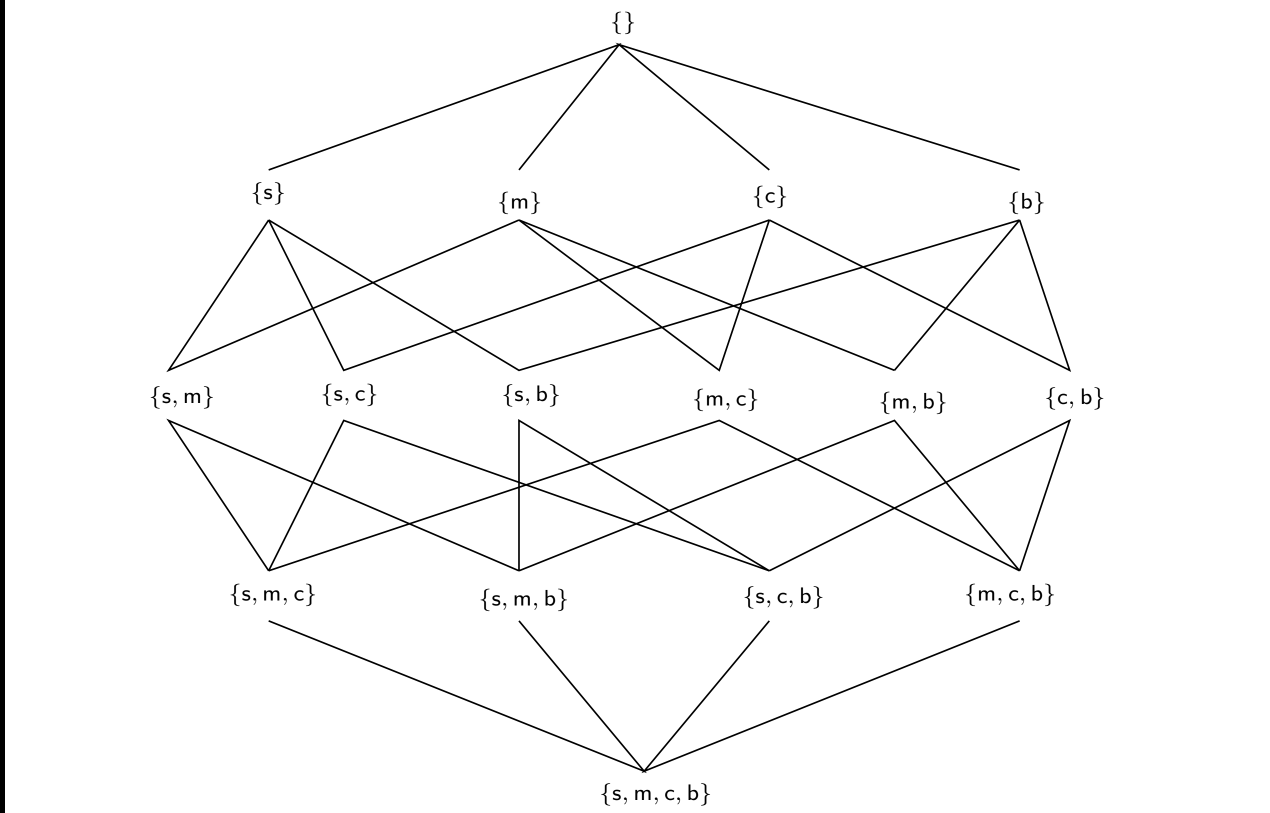


Fig. 3.5. The partial order over the item-sets

{s, m, c} = s or m or c

S ⊧ G if and only if S ⊆ G

Clausal Discovery

Example :

H : no constants/one variable

one interpretation :

{ human(luc), human(lieve), male(luc), female(lieve) }

Find :

human(X) :- male(X).

$$\forall X : human(X) \Leftarrow male(X)$$

human(X) :- female(X).

female(X); male(X) :- human(X)

false :- male(X), female(X).

so this is ako first order 3-CNF

Clausal Discovery

Observe :

if e does not satisfy $h_1; \ldots; h_n$:- $b_1, \ldots, b_m$

then there is an answer to

:-$b_1, \ldots, b_m$, not $h_1, \ldots$,not h_n

therefore consider refinements :

$h; h_1; \ldots, h_n$:- $b_1, \ldots, b_m$ and

$h_1; \ldots, h_n$:- $b_1, \ldots, b_m, b$

in all possible ways (adding a literal)

Clausal Discovery

Q := { false :- true }

h := {}

while Q is not empty do

    delete c from Q

    if c satisfies all p in P *(and h does not entail c)*

    then add c to h

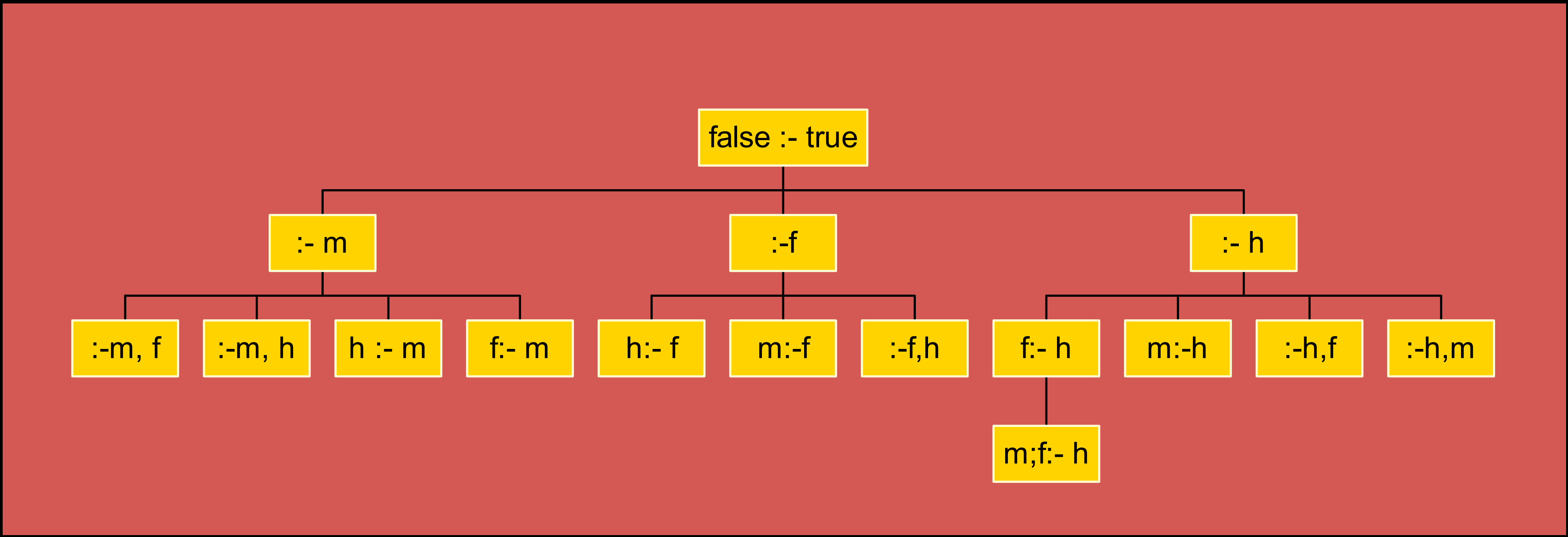
    else add all refinements of c to Q

    pruning :

        generate clauses in H at most once;

        remove entailed clauses

Clausal Discovery



Equation Discovery

Instead of learning clauses, learn equations *[Dzeroski and Todorovski; Langley and Bridewell]*.

As Valiant's algorithm

- generate and test candidate equations, e.g., ax + byz = c
  - fit parameters using regression
- possibly compute values for additional variables (partial derivatives w.r.t. time, etc.)
- include a grammar to specify "legal equations" (bias)

Ecological Modeling

Table 1
Variables used in the NPPc portion of the CASA model

NPPc is the net production of carbon by terrestrial plants at a site
E is the photosynthetic efficiency at a site after factoring various sources of stress
T1 is a temperature stress factor ($0 < T1 < 1$) for cold weather
T2 is a temperature stress factor ($0 < T2 < 1$), nearly Gaussian in form but falling off more quickly at higher temperatures
W is a water stress factor ($0.5 < W < 1$)
topt is the average temperature for the month at which *fas_ndvi* takes on its maximum value at a site
tempc is the average temperature at a site for a given month
eet is the estimated evapotranspiration (water loss due to evaporation and transpiration) at a site
PET is the potential evapotranspiration (water loss due to evaporation and transpiration given an unlimited water supply) at a site
pet_tw_m is a component of potential evapotranspiration that takes into account the latitude, time of year, and days in the month
A is a polynomial function of the annual heat index at a site
ahi is an annual heat index that takes the time of year into account
fas_ndvi is the relative greenness as measured from space
IPAR is the energy intercepted from the sun after factoring in the time of year and days in the month
FPAR_FAS is the fraction of energy intercepted from the sun that is absorbed photosynthetically after factoring in vegetation type
monthly_solar is the average radiation incoming for a given month at a site
SOL_CONV is 0.0864 times the num

$$NPPc = \max(0, E \cdot IPAR)$$
$$E = 0.312 \cdot T1^{1.36} \cdot T2^{0.728} \cdot W^{0}$$
$$T1 = 3.65 - 0.992 \cdot topt + 0.137 \cdot topt^2 - 0.00679 \cdot topt^3 + 0.000111 \cdot topt^4$$
$$T2 = 0.818/((1 + \exp(0.0521 \cdot (TDIFF - 10))) \cdot (1 + \exp(0 \cdot (-TDIFF - 10))))$$
$$TDIFF = topt - tempc$$
$$W = 0.5 + 0.5 \cdot eet/PET$$
$$PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^A \cdot pet\_tw\_m$$
$$A = 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239$$
$$IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5$$
$$FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95)$$
$$SR\_FAS = (1 + fas\_ndvi/750)/(1 - fas\_ndvi/750)$$
$$SOL\_CONV = 0.0864 \cdot days\_per\_month$$

Using equation discovery to revise an Earth ecosystem model of the carbon net production

Ljupčo Todorovski [a,*], Sašo Džeroski [a],
Pat Langley [b], Christopher Potter [c]

ModelSeeker

N. Beldiceanu and H. Simonis

Motivating Example

2	-1	4	-3	6	-5	8	-7	10	-9	12	-11	14	-13	16	-15	18	-17
-8	15	-10	7	-18	9	-4	1	-6	3	-14	13	-12	11	-2	17	-16	5
4	-17	14	-1	12	-13	10	-15	16	-7	18	-5	6	-3	8	-9	2	-11
7	11	-8	15	-16	-18	-1	3	-12	13	-2	9	-10	17	-4	5	-14	6
-3	-13	1	-11	10	16	-15	-17	14	-5	4	-18	2	-9	7	-6	8	12
15	9	-7	13	-14	-12	3	11	-2	17	-8	6	-4	5	-1	18	-10	-16
-17	-5	-15	-18	2	14	-9	-13	7	-11	10	-16	8	-6	3	12	1	4
11	18	5	9	-3	-10	17	12	-4	6	-1	-8	-15	16	13	-14	-7	-2
-13	-6	-17	-5	4	2	-11	-9	8	-16	7	14	1	-12	-18	10	3	15
9	16	11	6	-8	-4	13	5	-1	12	-3	-10	-7	18	17	-2	-15	-14
-5	-12	-13	-16	1	7	-6	-18	15	-14	17	2	3	10	-9	4	-11	8
6	14	9	12	-7	-1	5	16	-3	18	-15	-4	-17	-2	11	-8	13	-10
-18	-10	-12	-14	15	8	-16	-6	17	2	13	3	-11	4	-5	7	-9	1
12	-7	18	10	-17	-15	2	14	-11	-4	9	-1	16	-8	6	-13	5	-3
-14	4	-16	-2	11	17	-18	-10	13	8	-5	15	-9	1	-12	3	-6	7
10	-8	6	-17	-9	-3	12	2	5	-1	16	-7	18	-15	14	-11	4	-13
-16	3	-2	8	13	11	-14	-4	-18	15	-6	17	-5	7	-10	1	-12	9
-2	1	-4	3	-6	5	-8	7	-10	9	-12	11	-14	13	-16	15	-18	17
8	-15	10	-7	18	-9	4	-1	6	-3	14	-13	12	-11	2	-17	16	-5
-4	17	-14	1	-12	13	-10	15	-16	7	-18	5	-6	3	-8	9	-2	11
-7	-11	8	-15	16	18	1	-3	12	-13	2	-9	10	-17	4	-5	14	-6
3	13	-1	11	-10	-16	15	17	-14	5	-4	18	-2	9	-7	6	-8	-12
-15	-9	7	-13	14	12	-3	-11	2	-17	8	-6	4	-5	1	-18	10	16
17	5	15	18	-2	-14	9	13	-7	11	-10	16	-8	6	-3	-12	-1	-4
-11	-18	-5	-9	3	10	-17	-12	4	-6	1	8	15	-16	-13	14	7	2
13	6	17	5	-4	-2	11	9	-8	16	-7	-14	-1	12	18	-10	-3	-15
-9	-16	-11	-6	8	4	-13	-5	1	-12	3	10	7	-18	-17	2	15	14
5	12	13	16	-1	-7	6	18	-15	14	-17	-2	-3	-10	9	-4	11	-8
-6	-14	-9	-12	7	1	-5	-16	3	-18	15	4	17	2	-11	8	-13	10
18	10	12	14	-15	-8	16	6	-17	-2	-13	-3	11	-4	5	-7	9	-1
-12	7	-18	-10	17	15	-2	-14	11	4	-9	1	-16	8	-6	13	-5	3
14	-4	16	2	-11	-17	18	10	-13	-8	5	-15	9	-1	12	-3	6	-7
-10	8	-6	17	9	3	-12	-2	-5	1	-16	7	-18	15	-14	11	-4	13
16	-3	2	-8	-13	-11	14	4	18	-15	6	-17	5	-7	10	-1	12	-9

http://www.weltfussball.de/alle_spiele/bundesliga-2010-2011/

Result

J	Scheme	Ref	Trans	Constraint
1	scheme(612,34,18,1,18)	284	absolute_value	symmetric_alldifferent([1..18])*34
2	vector(612)	289	id	global_cardinality([-18.. -1-17,0-0,1..18-17])*1
3	scheme(612,34,18,34,1)	288	id	alldifferent*18
4	repart(612,34,18,17,18)	282	id	alldifferent*306
5	scheme(612,34,18,2,2)	286	id	alldifferent*153
6	scheme(612,34,18,1,18)	284	id	alldifferent*34
7	repart(612,34,18,34,9)	283	sign	alldifferent*306
8	scheme(612,34,18,17,1)	287	absolute_value	alldifferent*36
9	scheme(612,34,18,2,1)	285	absolute_value	alldifferent*306
10	repart(612,34,18,34,9)	283	id	sum_ctr(0)*306
11	repart(612,34,18,34,9)	283	id	sum_cubes_ctr(0)*306
12	scheme(612,34,18,1,18)	284	id	sum_squares_ctr(2109)*34
13	repart(612,34,18,34,9)	283	id	twin*1
14	repart(612,34,18,34,9)	283	id	elements([i,-i])*1
15	modulo(612,4)	281	id	all_differ_from_at_least_k_pos(152)*1
16	first(9,[1,3,5,7,9,11,13,15,17])	280	id	strictly_increasing*1
17	repart(612,34,18,34,9)	283	id	alldifferent_interval(2)*306
18	scheme(612,34,18,2,1)	285	id	alldifferent_interval(2)*306
19	repart(612,34,18,34,9)	283	sign	sum_ctr(0)*306
20	scheme(612,34,18,1,18)	284	sign	sum_ctr(0)*34
21	repart(612,34,18,34,9)	283	sign	twin*1
22	repart(612,34,18,34,9)	283	absolute_value	twin*1
23	repart(612,34,18,34,9)	283	sign	elements([i,-i])*1
24	repart(612,34,18,34,9)	283	absolute_value	elements([i,i])*1
25	first(9,[1,3,5,7,9,11,13,15,17])	280	absolute_value	strictly_increasing*1
26	first(6,[1,4,7,10,13,16])	279	absolute_value	strictly_increasing*1
27	repart(612,34,18,34,9)	283	sign	alldifferent_interval(2)*306
28	scheme(612,34,18,34,1)	288	sign	among_seq(3,[-1])*18

Slides N. Beldiceanu and H. Simonis

Key Points

Learning constraint models from positive examples

Start with **vector** of values

Group into **regular pattern**

Find constraint pattern that apply to group elements

Using ***Constraint Seeker*** for *Global Constraint Catalog*

Works for **highly structured** problems

Slides N. Beldiceanu and H. Simonis

Partition generators

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Structured groups of variables passed to

a conjunction of **identical** constraints

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_ctr(34)*4

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_ctr(34)*4

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

strictly_decreasing*2
sum_ctr(34)*2

Slides N. Beldiceanu and H. Simonis

Partition generators

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Structured groups of variables passed to

a conjunction of **identical** constraints

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_ctr(34)*4

surprise

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_squares_ctr(358)*2

surprise

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_squares_ctr(390)*2

surprise

Slides N. Beldiceanu and H. Simonis

Partition generators

Structured groups of variables passed to

a conjunction of **identical** constraints

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

16^1	3^2	2^3	13^4
5^5	10^6	11^7	8^8
9^9	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

Slides N. Beldiceanu and H. Simonis

Partition generators

sample

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Structured groups of variables passed to

a conjunction of **identical** constraints

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_ctr(34)*4

surprise

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

sum_squares_ctr(748)*2

surprise

16^{1}	3^{2}	2^{3}	13^{4}
5^{5}	10^{6}	11^{7}	8^{8}
9^{9}	6^{10}	7^{11}	12^{12}
4^{13}	15^{14}	14^{15}	1^{16}

alldifferent_interval(2)*8

Slides N. Beldiceanu and H. Simonis

Analysis

Clever enumeration of partitions (generate and test)

Per partition :

- clever generation of conjunction of constraints that holds

Multiple examples

- remove constraints that do not hold

Nicely deals with generality / redundancy

- relations amongst global constraints — builtin catalogue
- permutation -> alldifferent

Tacle - Learning Constraints in Tabular Data

$SERIES(T_1[:,1])$

$T_1[:,1] = RANK(T_1[:,5])^*$

$T_1[:,1] = RANK(T_1[:,6])^*$

$T_1[:,1] = RANK(T_1[:,10])^*$

$T_1[:,8] = RANK(T_1[:,7])$

$T_1[:,8] = RANK(T_1[:,3])^*$

$T_1[:,8] = RANK(T_1[:,4])^*$

$T_1[:,7] = SUM_{row}(T_1[:,3{:}6])$

$T_1[:,10] = SUMIF(T_3[:,1], T_1[:,2], T_3[:,2])$

$T_1[:,11] = MAXIF(T_3[:,1], T_1[:,2], T_3[:,2])$

$T_2[1,:] = SUM_{col}(T_1[:,3{:}7])$

$T_2[2,:] = AVERAGE_{col}(T_1[:,3{:}7])$

$T_2[3,:] = MAX_{col}(T_1[:,3{:}7]),$

$T_2[4,:] = MIN_{col}(T_1[:,3{:}7])$

$T_4[:,2] = SUM_{col}(T_1[:,3{:}6])$

$T_4[:,4] = PREV(T_4[:,4]) + T_4[:,2] - T_4[:,3]$

$T_5[:,2] = LOOKUP(T_5[:,3], T_1[:,2], T_1[:,1])^*$

$T_5[:,3] = LOOKUP(T_5[:,2], T_1[:,1], T_1[:,2])$

T1

ID	Salesperson	1st Quarter	2nd Quarter	3rd Quarter	4th Quarter	Total	Rank	Label	Items sold total	Max items sold
1	Diana Coolen	353	378	396	387	1514	2	Great	34	20
2	Marc Desmet	370	408	387	386	1551	1	Great	29	10
3	Kris Goossens	175	146	167	203	691	3	Low	19	19
4	Birgit Kenis	93	98	96	105	392	4	Low	17	15
B1 = T1[:, 1]	B2 = T1[:, 2]	B3 = T1[:, 3:8]						B4 = T1[:, 9]	B5 = T1[:, 10:11]	

T2

Total	991	1030	1046	1081	4148
Average	247.75	257.5	261.5	270.25	1037
Max	370	408	396	387	1551
Min	93	98	96	105	392

B6 = T2[1:4, :]

T3

Salesperson	Items sold
Diana Coolen	5
Marc Desmet	10
Marc Desmet	8
Diana Coolen	9
Birgit Kenis	15
Marc Desmet	8
Birgit Kenis	2
Diana Coolen	20
Marc Desmet	3
Kris Goossens	19
B8 = T3[:, 1]	B9 = T3[:, 2]

T4

Quarter	Income	Expenses	Total
Q1	991	212	779
Q2	1030	710	1099
Q3	1046	137	2008
Q4	1081	240	2849
B10 = T4[:, 1]	B11 = T4[:, 2:4]		

T5

Customer	Contact	Contact Name
Frank	1	Diana Coolen
Sarah	3	Kris Goossens
George	3	Kris Goossens
Mary	2	Diana Coolen
Tim	4	Birgit Kenis
B12 = T5[:, 1]	B13 = T5[:, 2]	B14 = T5[:, 3]

[Kolb et al. MLJ 17]

also type information
also CP solver to find constraints

Take Away

Essential components of a learner

- Traversing / enumerating the hypothesis space H
  - use of an operator (refinement operator)
  - naively (generate and test)
    - avoid generating the same constraint twice
    - consider generating partitions
  - pruning the space by generality
    - can be hard (cf. transitivity of =)

D. Angluin, M. Frazier, and L. Pitt.
Learning conjunctions of Horn clauses.
9:147–162, 1992.

D. Angluin.
Queries and concept-learning.
2:319–342, 1987.

Christian Bessiere, Remi Coletta, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, Toby Walsh, et al.
Constraint acquisition via partial queries.
In *IJCAI*, volume 13, pages 475–481, 2013.

Christian Bessiere, Remi Coletta, Frédéric Koriche, and Barry O'Sullivan.

Acquiring constraint networks using a sat-based version space algorithm.
In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 1565. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

Nicolas Beldiceanu and Helmut Simonis.
A model seeker: Extracting global constraint models from positive examples.
In *Principles and practice of constraint programming*, pages 141–157. Springer, 2012.

L. De Raedt and S. Džeroski.
First-Order jk-Clausal Theories are PAC-Learnable.
Artificial Intelligence, 70(1-2):375–392, 1994.

L. De Raedt and L. Dehaspe.

Clausal discovery.
Machine Learning, 26(2-3):99–146, 1997.

Luc De Raedt.
Logical and Relational Learning.
Springer, 2008.

Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrač.
Foundations of rule learning.
Springer Science & Business Media, 2012.

Johannes Fürnkranz and Eyke Hüllermeier.
Preference learning: An introduction.
In *Preference learning*, pages 1–17. Springer, 2010.

Johannes Fürnkranz.
Separate-and-conquer rule learning.
Artificial Intelligence Review, 13(1):3–54, 1999.

- Samuel Kolb, Sergey Paramonov, Tias Guns, and Luc De Raedt.
  Learning constraints in spreadsheets and tabular data.
  Machine Learning, 106(9-10):1441–1468, 2017.
- M. Kearns and U. Vazirani.
  An Introduction to Computational Learning Theory.
  1994.
- T. M. Mitchell.
  Machine Learning.
  McGraw-Hill, 1997.
- Kim Marriott and Peter J Stuckey.
  Programming with constraints: an introduction.
  MIT press, 1998.

Ljupčo Todorovski, Sašo Džeroski, Pat Langley, and Christopher Potter.
Using equation discovery to revise an earth ecosystem model of the carbon net production.
Ecological Modelling, 170(2):141–154, 2003.

L. Valiant.
A theory of the learnable.
Communications of the ACM, 27:1134–1142, 1984.

Thank you