Towards Deep and Interpretable Rule Learning

Johannes Fürnkranz

Johannes Kepler University, Linz
Institute for Application-Oriented Knowledge Processing
Computational Data Analytics Group

juffi@faw.jku.at

Joint Work with
Florian Beck, Van Quoc Phuong Hyunh, Paulus Seip, Martina Seidl, Clemens Hofstadler, Peter Pfeiffer, Robert Peharz, Stefan Szeider et al.

A Sample Database

No.	Education	Marital S.	Sex.	Children?	**Approved?**
1	Primary	Single	M	N	-
2	Primary	Single	M	Y	-
3	Primary	Married	M	N	+
4	University	Divorced	F	N	+
5	University	Married	F	Y	+
6	Secondary	Single	M	N	-
7	University	Single	F	N	+
8	Secondary	Divorced	F	N	+
9	Secondary	Single	F	Y	+
10	Secondary	Married	M	Y	+
11	Primary	Married	F	N	+
12	Secondary	Divorced	M	Y	-
13	University	Divorced	F	Y	-
14	Secondary	Divorced	M	N	+

Property of Interest
(“class variable”)

Could be viewed as a Constraint Problem

JOHANNES KEPLER UNIVERSITÄT LINZ

- Positive

```
p :- primary,    male,   married,  no_child.
p :- university, female, divorced, no_child.
p :- university, female, married,  children.
p :- university, female, single,   no_child.
p :- secondary,  female, divorced, no_child.
p :- secondary,  female, single,   children.
p :- secondary,  male,   married,  children.
p :- primary,    female, married,  no_child.
p :- secondary,  male,   divorced, no_child.
```

- Negative

```
:- p,  primary,    male,   single,   no_child.
:- p,  primary,    male,   single,   children.
:- p,  secondary,  male,   single,   no_child.
:- p,  secondary,  male,   divorced, children.
:- p,  university, female, divorced, children.
```

Goal:
Find a definition for `p` that satisfies all the constraints

… that is more compact

… and generalizes well to new instances from this domain

… while possibly violating some of the training constraints

Could be viewed as a Constraint Problem

JOHANNES KEPLER UNIVERSITÄT LINZ

- Positive

```
p :- primary,    male,   married,  no_child.
p :- university, female, divorced, no_child.
p :- university, female, married,  children.
p :- university, female, single,   no_child.
p :- secondary,  female, divorced, no_child.
p :- secondary,  female, single,   children.
p :-
p :-
p :-
```

```
p :- married.
p :- single, female.
p :- divorced, no_child.
```

- Negative

```
:- p,  primary,    male,   single,   no_child.
:- p,  primary,    male,   single,   children.
:- p,  secondary,  male,   single,   no_child.
:- p,  secondary,  male,   divorced, children.
:- p,  university, female, divorced, children.
```

Goal:
Find a definition for `p` that satisfies all the constraints

… that is more compact

… and generalizes well to new instances from this domain

… while possibly violating some of the training constraints

Biases in Machine Learning

JOHANNES KEPLER
UNIVERSITÄT LINZ

The Need for Biases in Learning Generalizations

Tom M. Mitchell
Computer Science Department
Rutgers University
New Brunswick, NJ 08904

May, 1980

Abstract

The ability to make an appropriate "inductive leap" when generalizing from a small set of training instances is possible only under a priori biases for choosing an appropriate generalization out of the many possible. Understanding the origins and justification of such biases is critical to further progress in the field of machine learning. The notion of an UNbiased learner is defined, then the notion of bias, its usefulness, and some classes of justifiable biases are considered.

The Need for Biases in Learning Generalizations

(Mitchell, 1980)

JOHANNES KEPLER UNIVERSITÄT LINZ

- Mitchell introduced the term bias into machine learning

> In this paper, we use the term *bias* to refer to *any basis for choosing one generalization over another, other than strict consistency with the observed training instances.*

- As possible biases he suggested
  - domain knowledge for limiting the hypothesis space
  - intended use of the learned theories (e.g., misclassification costs)
  - knowledge about the source of the training data (e.g., sample bias)
  - analogy with previously learned generalizations
  - **bias towards simplicity and generality**

Interpretability vs. Complexity

JOHANNES KEPLER UNIVERSITÄT LINZ

- Conventional Rule learning algorithms tend to learn short rules
  - They favor to add conditions that exclude many negative examples
- **Typical intuition:** Short rules and rule sets are better
  - long rules are less understandable, therefore short rules are preferable
  - short rules are more general, therefore (statistically) more reliable and would have been easier to falsify on the training data
- **Claim:** Shorter rules or rule sets are not always better
  - **Predictive Performance:** Longer rules often cover the same number of examples than shorter rules so that (statistically) there is no preference for choosing one over the other
  - **Understandability:** In many cases, longer rules may be much more intuitive than shorter rules

→ *we need to understand understandability*!

The LORD Rule Learner

(Huynh, Fürnkranz, Beck 2023)

- **Key idea**
  - aim at learning the best rule for each training example
    - local optimum in a local neighborhood around the training example
    - motivated by the XAI idea of providing explanations for each example
  - the result is one rule for each training example
    - almost, because suboptimal and duplicate rules are removed
- **Implementation characteristics**

https://github.com/vqphuynh/LORD

  - Make use of efficient data structures known from association rule mining like PPC-trees and N-lists
    - can efficiently summarize the dataset in one pass
  - Use a rule learning heuristic for guiding its greedy search
    - e.g. the m-estimate
  - Inherently parallel search for locally optimal rules
    - LORD can efficiently tackle very large example sets

Empirical Results

- 24 datasets with various sizes
- the largest with 5 million examples and 19 attributes

#	Datasets	# Exs.	# Attr.	Attr. Types	Missing Values	Class Distributions (%)
1	*lymph*	148	19	categorical	no	54.7; 41.2; 2.7; 1.3
2	*wine*	178	14	numeric	no	33.2; 39.9; 26.9
3	*vote*	435	17	categorical	yes	54.8; 45.2
4	*breast-cancel*	699	10	numeric	yes	65.5; 34.5
5	*tic-tac-toe*	958	10	categorical	no	65.3; 34.7
6	*german*	1,000	21	mix	no	70; 30
7	*car-eval*	1,728	7	categorical	no	22.3; 3.9; 70; 3.8
8	*hypo*	3,163	26	mix	yes	95.2; 4.8
9	*kr-vs-kp*	3,196	37	categorical	no	52.2; 47.8
10	*waveform*	5,000	22	numeric	no	33.2; 32.9; 33.9
11	*mushroom*	8,124	23	categorical	yes	51.7; 48.3
12	*nursery*	12,960	9	categorical	no	33.3; 32.9; 31.2; 2.5; 0.01
13	*adult*	48,842	14	mix	yes	76; 24
14	*bank*	45,211	17	mix	no	11.7; 88.3
15	*skin*	245,057	4	numeric	no	20.7; 79.3
16	*s-mushroom*	61,069	21	mix	yes	44.5; 55.5
17	*connect-4*	67,557	42	categorical	no	65.8; 24.6; 9.6
18	*PUC-Rio*	165,632	19	mix	no	28.6; 26.2; 7.5; 7.1; 30.6
19	*census*	299,285	41	mix	yes	93.8; 6.2
20	*gas-sensor-11*	919,438	11	numeric	no	32.9; 29.8; 37.3
21	*gas-sensor-12*	919,438	12	numeric	no	32.9; 29.8; 37.3
22	*cover-type*	581,012	55	mix	no	36.4; 48.8; 6.2; 0.5; 1.6; 3; 3.5
23	*pamap2*	1,942,872	33	numeric	yes	9.9; 6; 9.5; 5.4; 2.5; 9.8; 12.3; 9; 5.1; 12.3; 8.5; 9.7
24	*susy*	5,000,000	19	numeric	no	54.2; 45.8

Empirical Results – Accuracy and Run-time

- Accuracy
  - better than Ripper and other modern rule learner (not ensembles)

#	Datasets	Lord (m = 0.1)	Lord (best m)	Lord* (m = 0.1)	Ripper (o = 0)	Ripper (o = 2)	CMAR	CG	PyIDS (k = 50)	PyIDS (k = 150)
Avg. acc. (3-6,8-9,11,13-16)		**0.9416**	**0.9436**	**0.9415**	0.9365	0.9374	0.916	0.9222	0.8137	0.8312
Avg. acc. (1-22)		**0.9268**	**0.9311**	**0.9266**	0.9073	0.9152	0.8056	//	0.7077	0.7287
Avg. ranks (1-22)		**3.14**	**1.84**	**3.3**	4.48	3.59	5.2	//	7.57	6.89

- Run-time
  - only few algorithms could tackle the largest datasets

#	Datasets	Lord (m = 0.1)	Lord (best m)	Lord* (m = 0.1)	Ripper (o = 0)	Ripper (o = 2)	CMAR	CG	PyIDS (k = 50)	PyIDS (k = 150)
23	pamap2	6063	6044	386	Out of memory	Out of memory	50.4	//	Out of memory	Out of memory
24	susy	52592	51218	15350	Out of memory	Out of memory	97.4	Out of time	9435	29109
Avg. runtime (1-22)		**94**	**95.1**	**31.5**	342	8642.8	116	//	274.7	2568.6
Avg. ranks (1-22)		3.5	3.75	**1.73**	2.95	5.09	4.89	//	6.27	7.82

Empirical Results – Number of Rules

JOHANNES KEPLER UNIVERSITÄT LINZ

- Number of learned rules
  - enormous, e.g., 1.6 million rules for the susy dataset

#	Datasets	Lord (m = 0.1)		Lord (best m)		Lord* (m = 0.1)		Ripper (o = 0)		Ripper (o = 2)		CMAR		CG		PyIDS (k = 50)		PyIDS (k = 150)	
23	pamap2	16827	3.07	14137	3.09	15824	3.05	Out of memory		Out of memory		486	2.54	//		Out of memory		Out of memory	
24	susy	1611856	4.30	1201338	4.10	976522	4.30	Out of memory		Out of memory		637	1.40	Out of time		18	2.0	63	2.0
Avg. values (1-22)		8390.6	3.54	8261.4	3.5	6361.2	3.49	104.6	4.12	111.5	3.74	1945.1	3.06	//		**16.8**	**2.06**	**50.4**	**2.1**
Avg. ranks (1-22)		6.82	5.86	6.39	5.82	5.25	4.95	2.73	5.16	2.23	3.95	6.45	4.86	//		**1.91**	**2.45**	4.23	**2.93**

- The rule sets are certainly not interpretable
  - However, each rule is the perfect explanation for one of the training examples
- Ongoing Work:
  - LORD as a post-hoc XAI tool
  - transductive learning of rules (this is harder than you may think…)

Coverage Spaces

(Fürnkranz & Flach 2005)

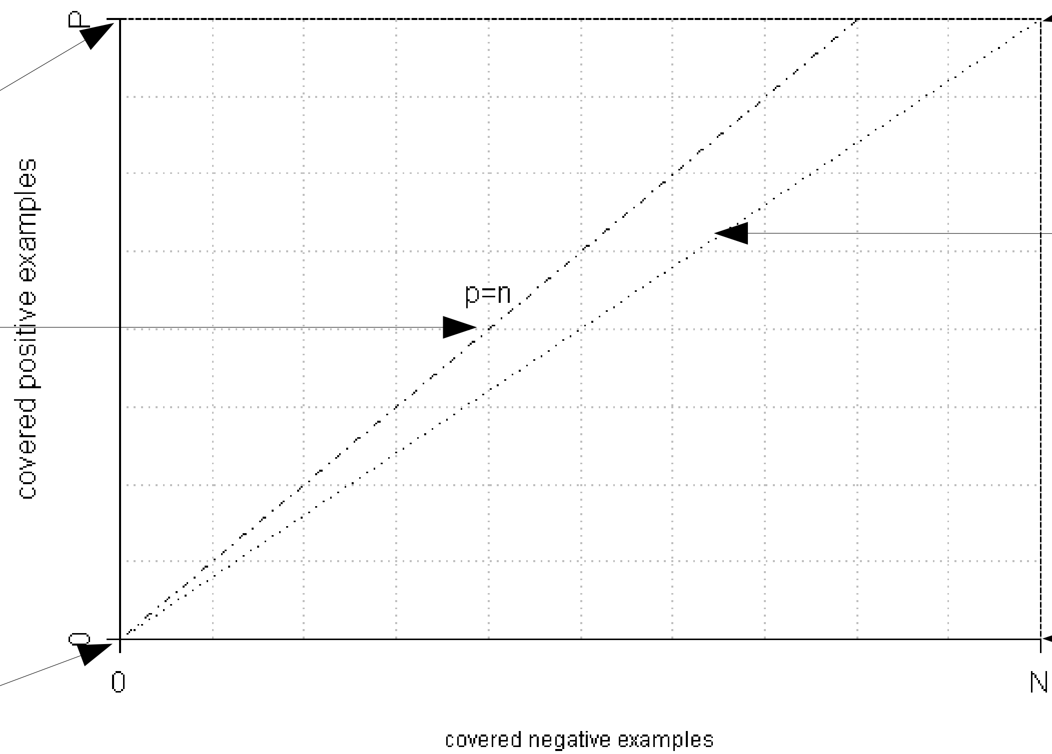
JOHANNES KEPLER UNIVERSITÄT LINZ

- good tool for visualizing rule learning algorithms
  - each point is a rule covering p positive and n negative examples

perfect rule: all positive and no negative examples are covered

iso-accuracy: cover same amount of positive and negative examples

empty rule: no examples are covered (most specific)



universal rule: all examples are covered (most general)

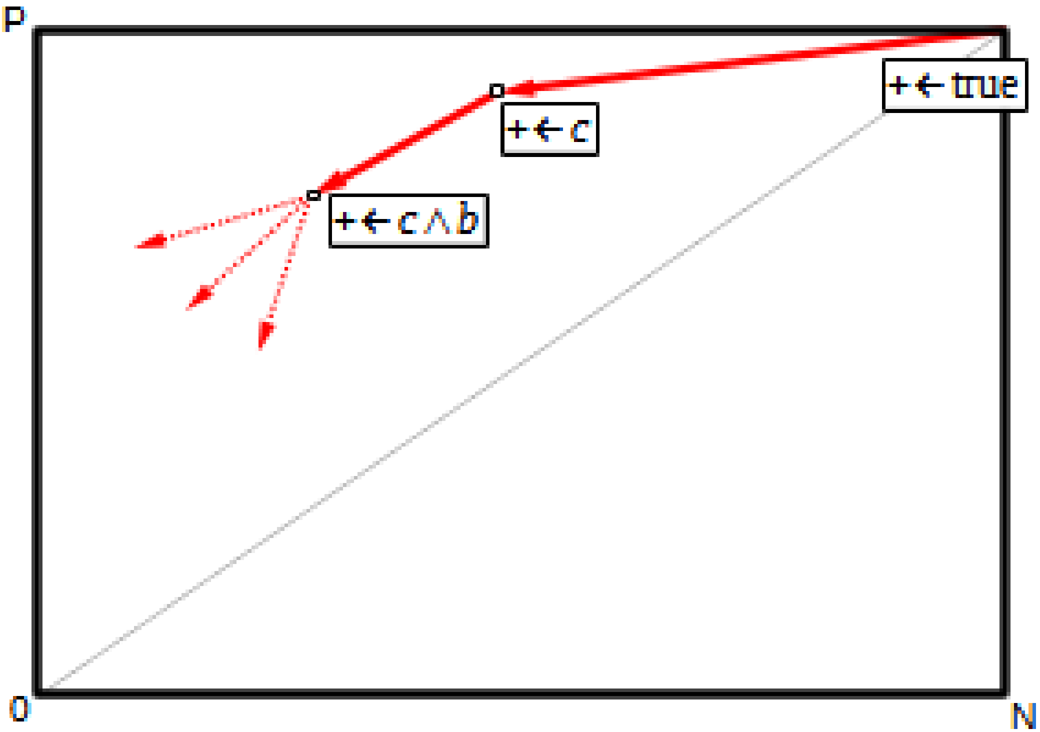
random rules: predict with coin tosses with fixed probability

opposite rule: all negative and no positive examples are covered

Learning Conjunctive Rules

Most algorithms learn conjunctive rules

- by greedily adding one conjunct at a time
- so that it maximizes some heuristic function $h(p,n)$



CN2-Style Rule Induction

JOHANNES KEPLER UNIVERSITÄT LINZ

- Most popular type of rule induction (Clark & Niblett, 1989)
  - used in most covering (separate-and conquer) rule learning algorithms

Algorithm 2 CN2-type rule induction

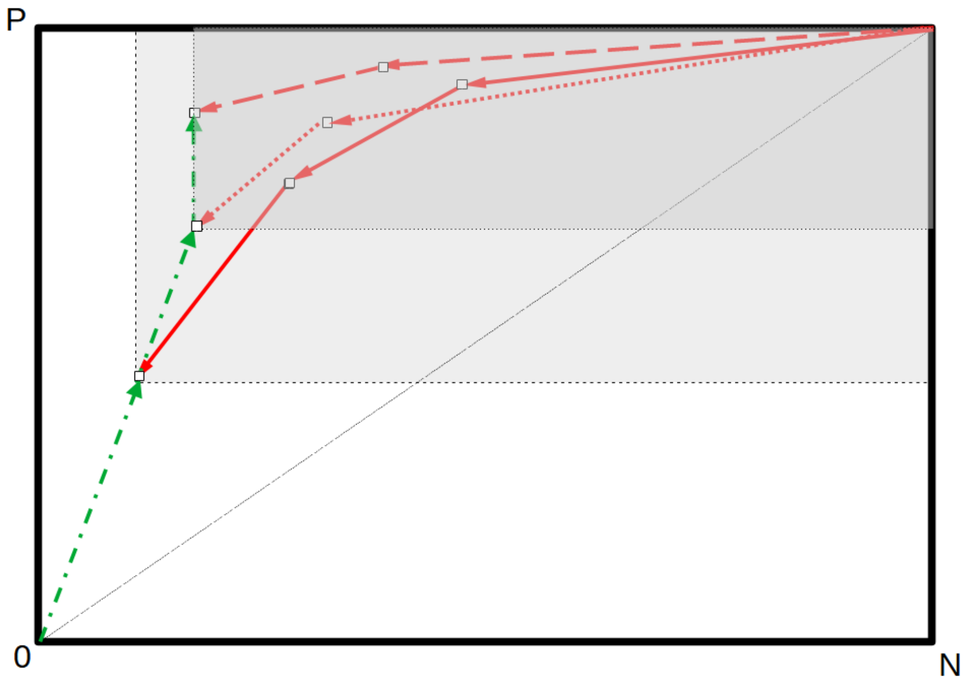
```
function CN2(E,F,h)
    R = ∅
    while E ≠ ∅ do
        r ← arg max_{r'=(B→y), B⊂F, y∈C} h(r')
        R ← R ∪ {r}
        E ← E \ E_r
    end while
    return R
end function
```

(Greedily) find a subset B of all features F so that some quality function h is optimized

Covering: Repeat until all examples are covered by one (or more) rule

Learning DNF Expressions

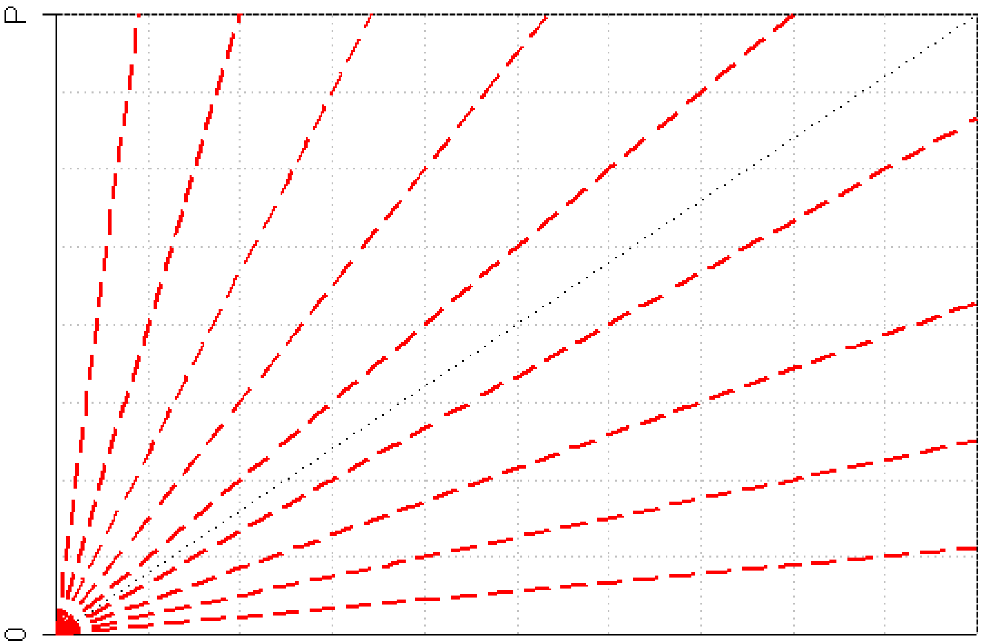
- successive refinement of individual rules (red)
- reductions in coverage space by removing covered examples (shades of grey)
- bulding up the DNF by adding one conjunct at a time (green)



Heuristics: Precision

- *basic idea:*
  - percentage of positive examples among covered examples
- *effects:*
  - rotation around origin (0,0)
  - all rules with same angle equivalent
  - in particular, all rules on P/N axes are equivalent
- typically **overfits**

$$h_{Prec}(p,n)=\frac{p}{p+n}$$

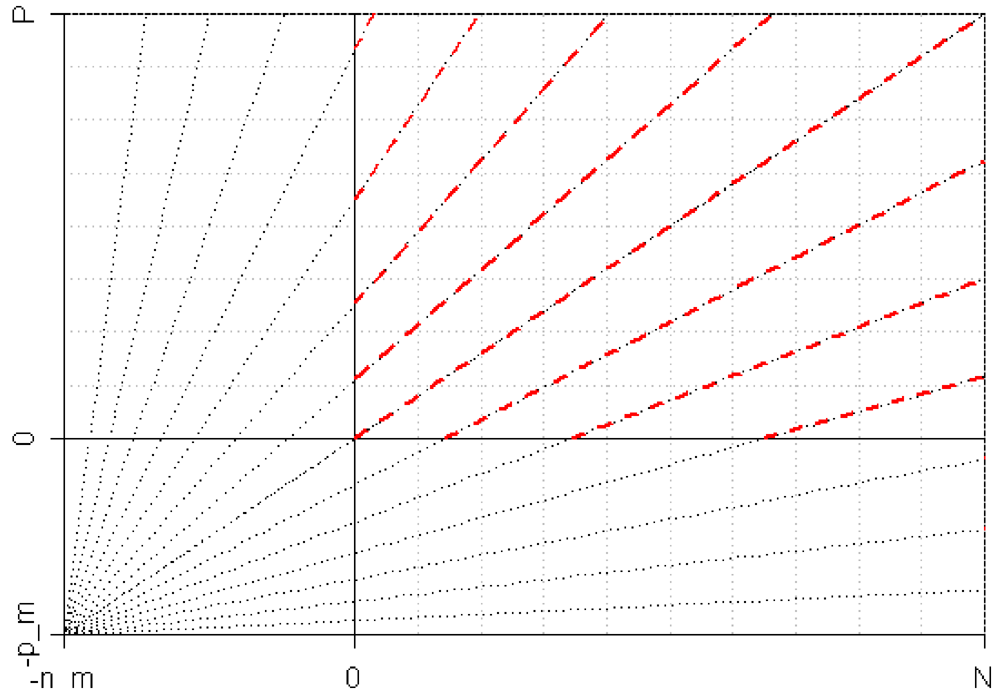


Heuristics: m-Estimate

- *basic idea:*
  initialize the counts with m examples in total, distributed according to the prior distribution $P/(P+N)$ of p and

$$h_m(p,n)=\frac{p+m\frac{P}{P+N}}{\left(p+m\frac{P}{P+N}\right)+\left(n+m\frac{N}{P+N}\right)}=\frac{p+m\frac{P}{P+N}}{p+n+m}$$

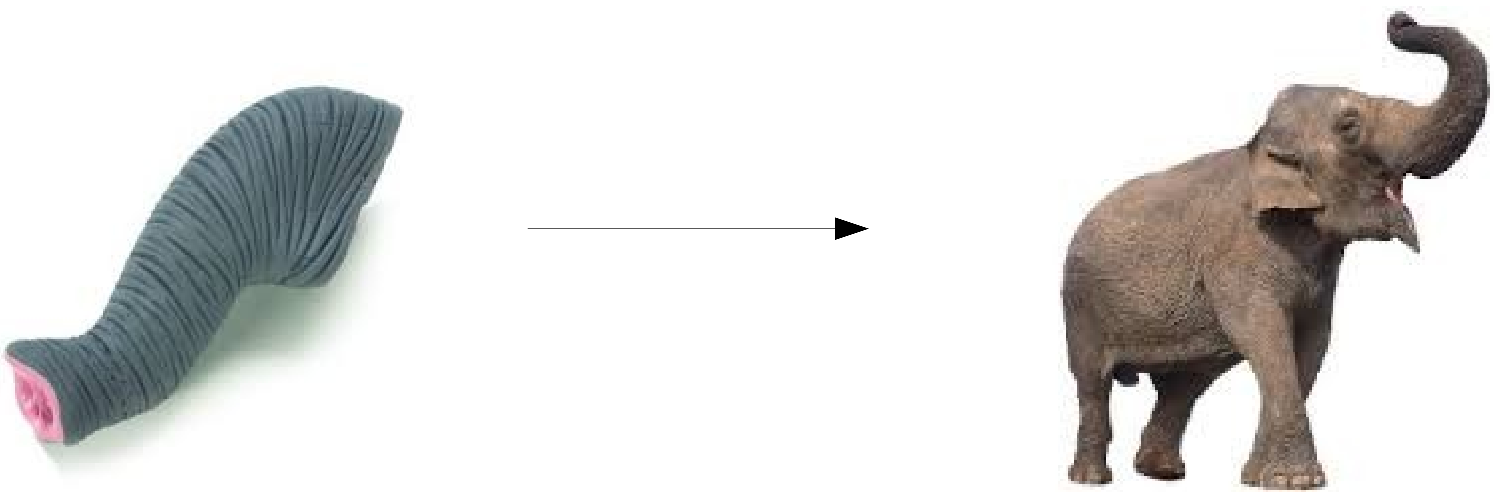
- *effects:*
  - origin shifts to $(-mP/(P+N),-mN/(P+N))$
  - with increasing m, the lines become more and more parallel
  - can be re-interpreted as a trade-off between WRA an precision/confidence



Discriminative Rules

- Allow to quickly **discriminate an object** of one category from objects of other categories
- Typically a few properties suffice

Example:



Characteristic Rules

- Allow to characterize an object of a category
- Focus is on all properties that are **representative** for objects of that category

Example:

Note: Characteristic rules tend to be more complex

Discriminative Rules vs. Characteristic Rules

Michalski (1983) discerns two kinds of classification rules:

- **Discriminative Rules:**
  - A way to distinguish the given class from other classes

```
Features → Class
```

  - Most interesting are *minimal discriminative rules*.
- **Characteristic Rules:**
  - A conjunction of all properties that are common to all objects in the class

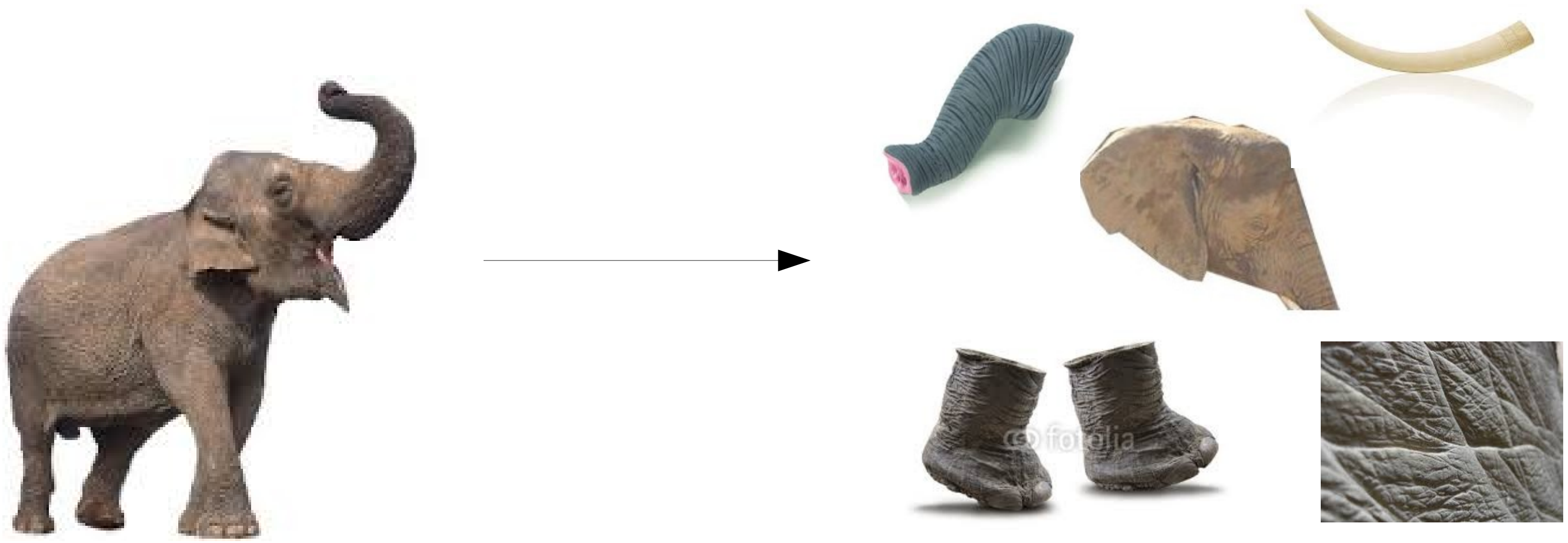
```
Class → Features
```

  - Most interesting are *maximal characteristic rules*.

Characteristic Rules

- An alternative view of characteristic rules is to invert the implication sign
- All properties that are implied by the category

Example:

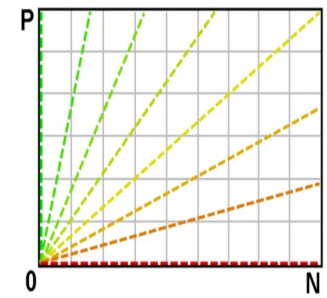


Inverted Coverage Spaces

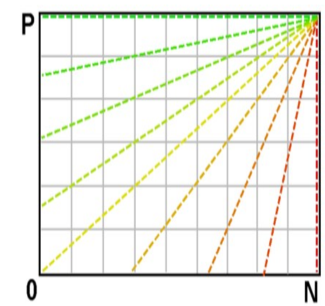
JOHANNES KEPLER UNIVERSITÄT LINZ

(Stecher, Janssen, Fürnkranz 2014)

- **Regular rule learning heuristics** quickly exclude negative examples
  - e.g., precision: $h(p,n)=\frac{p}{p+n}$
    - is maximal if no negative examples are covered (regardless of the number of positive examples)



- **Inverted heuristics** instead maintain a high coverage of positive examples
  - e.g., inverted precision $\text{ɥ}(p,n)=\frac{N-n}{(P-p)+(N-n)}$
    - is maximal if all positive examples are covered (regardless of the number of negative examples)

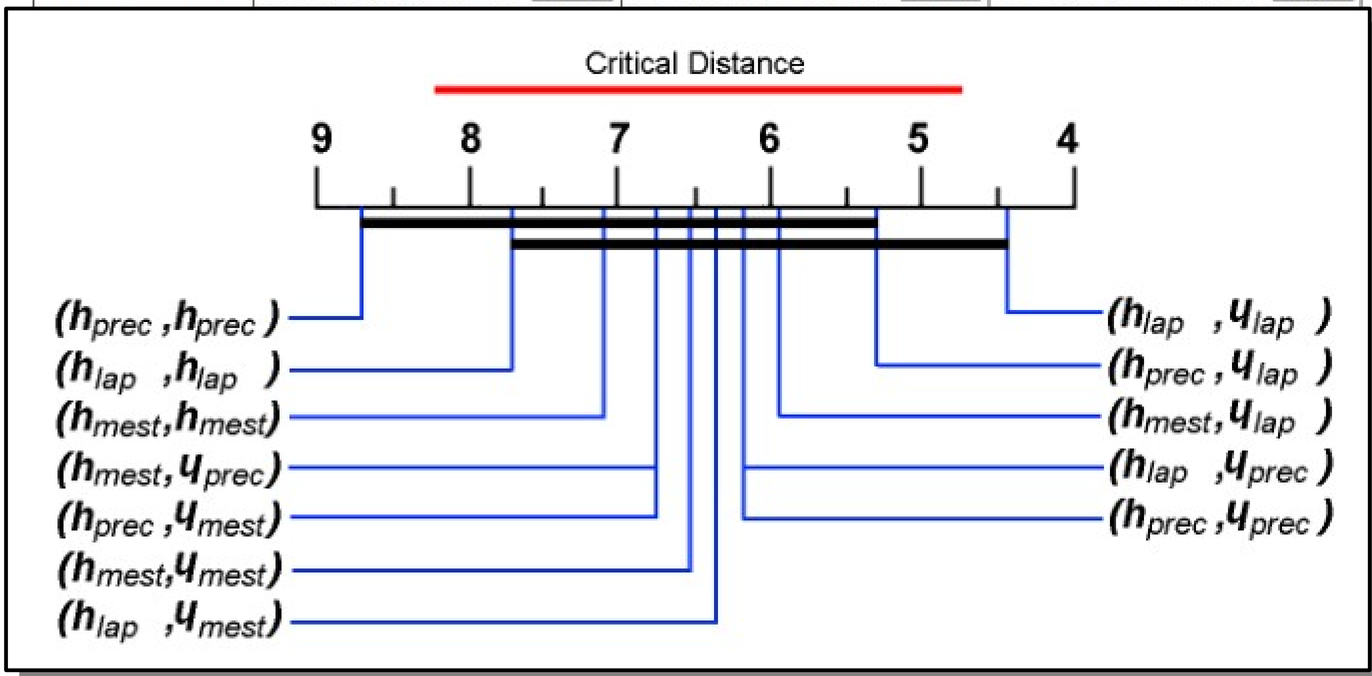


- **General approach:** $\text{h}(N-n, P-p) = \text{ɥ}(p,n)$
  - swap the role of positive and negative examples
  - negate all the inputs
    - i.e., learn conjunctions of negated features that predict the negative class

This is similar to inverting the implication
→ **characteristic rules**

Empirical Results:
Inverted heuristics tend to work better

Dataset	$(h_{prec}, .)$				$(h_{lap}, .)$				$(h_{mest}, .)$			
	h_{prec}	$ɥ_{prec}$	$ɥ_{lap}$	$ɥ_{mest}$	h_{lap}	$ɥ_{prec}$	$ɥ_{lap}$	$ɥ_{mest}$	h_{mest}	$ɥ_{prec}$	$ɥ_{lap}$	$ɥ_{mest}$
breast-cancer	68.53	72.38	72.03	73.43	69.58	70.63	71.33	72.73	71.33	72.03	72.38	73.78
tic-tac-toe	97.39	98.02	97.60	97.81	97.60	98.02	97.60	97.91	98.12	98.02	97.60	97.81
vote	94.94	93.56	94.25	94.48	95.40	94.25	94.25	94.94	93.33	93.56	94.71	96.09
zoo	84.16	88.12	92.08	90.01	86.14	88.12	92.08	90.10	89.11	88.12	92.08	90.10
average rank	3.075	2.400	1.975	2.550	3.000	2.500	1.975	2.525	2.700	2.625	2.225	2.450



Inverted Heuristics – Rule Length

- Inverted Heuristics tend to learn longer rules
  - If there are conditions that can be added without decreasing coverage on the positive examples, inverted heuristics will add them first (before adding discriminative conditions)

	(h_{lap}, h_{lap})		(h_{lap}, h'_{lap})			(h_{lap}, h_{lap})		(h_{lap}, h'_{lap})	
Dataset	R	L	R	L	**Dataset**	R	L	R	L
breast-cancer	25	67	38	173	ionosphere	17	25	8	42
car	107	495	107	506	labor	5	7	3	12
contact-lenses	5	14	5	15	lymphography	18	42	11	47
futebol	4	7	2	5	monk3	13	38	11	32
glass	50	103	14	83	mushroom	11	13	7	35
hepatitis	13	26	7	46	primary-tumor	80	319	72	518
horse-colic	44	114	19	111	soybean	62	134	45	195
hypothyroid	27	65	9	69	tic-tac-toe	22	84	16	69
iris	7	15	5	17	vote	13	48	12	58
idh	4	5	2	5	zoo	19	19	6	14
averages						28.2	85.6	20.6	106.2

Example Rules – Mushroom dataset

- The best three rules learned with conventional heuristics

```
poisonous :- odor = foul.              (2160,0)
poisonous :- gill-color = buff.        (1152,0)
poisonous :- odor = pungent.            (256,0)
```

- The best three rules learned with inverted heuristics

```
poisonous :- veil-color = white, gill-spacing = close,
             no bruises, ring-number = one,
             stalk-surface-above-ring = silky.  (2192,0)
poisonous :- veil-color = white, gill-spacing = close,
             gill-size = narrow, population = several,
             stalk-shape = tapering.             (864,0)
poisonous :- stalk-color-below-ring = white,
             ring-type = pendant, ring-number = one,
             stalk-color-above-ring = white,
             cap-surface = smooth, stalk-root = bulbuous,
             gill-spacing = close.               (336,0)
```

Example Rules – Coronary Heart Disease

(Stecher, Janssen, Fürnkranz 2016)

JOHANNES KEPLER UNIVERSITÄT LINZ

```
[17| 0] class 1 :- holst < 0.0001, vkgq = 1, ergfr = 1, ergrt = 1.
[17| 0] class 1 :- ergg < 180.0001, ergst < 0.2001, vkgq = 1,
                   ehoef >= 68, ergfr = 1.
[14| 0] class 1 :- holst < 0.2001, vkgq = 1, ecgfr >= 70, ergd >= 100.
```

Longer rules with **higher coverage** (compared to h_{Lap})

```
[32|0] class 1 :- vkgq = 1, ergkp = 1, ergny = 1, ergrt = 1,
                  hight >= 154, ergfr = 1, holst < 0.3001, ecgpr = 1,
                  holfr = 1, ehoef >= 65.
[28|0] class 1 :- ergst < 0.3001, vkgq = 1, ergny = 1, hdl >= 0.72,
                  ergfr = 1, ecgrt = 1, ecgpr = 1, fib < 4.5001,
                  vkghl = 1, holst < 0.2001, ecgst = 1, holrt = 1, ldl < 4.7601.
[25|0] class 1 :- ergst < 0.2001, vkgq = 1, ergny = 1, ergrt = 1,
                  ergfr = 1, ecgpr = 1, ergkp = 1, ecgrt = 1, holfr = 1,
                  ehoef >= 64, ua < 308.0001.
```

Example Rules – Brain Ischemia

```
[149| 0] ischemia :- b.i. < 60.0001.
[140| 0] ischemia :- b.i. < 70.0001, fibrin. >= 3.8.
[137| 0] ischemia :- b.i. < 75.0001, fibrin. >= 3.9.
```

Regular heuristics find **Barthel index** and **fibrinogen value** as relevant for a brain stroke.

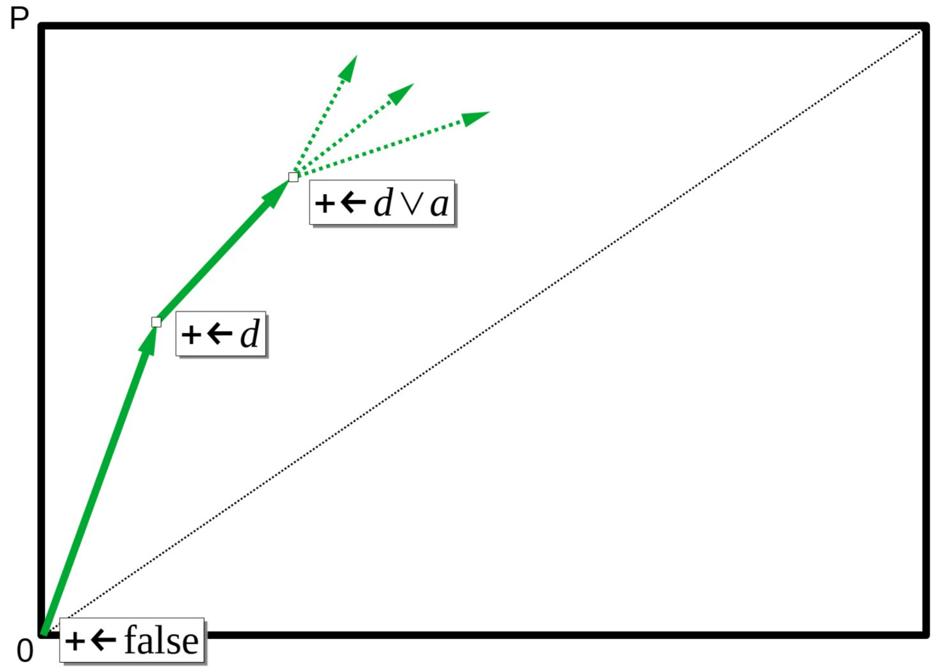
Inverted heuristics in addition refer to **age**, **diastolic blood pressure**, and **cholesterol**

```
[147| 0] ischemia :- rrrrdyast.. >= 70, fibrin. >= 2.8, b.i. < 60.0001.
[139| 0] ischemia :- age >= 58, rrrrdyast.. >= 80, b.i. < 60.0001.
[107| 0] ischemia :- rrrrdyast.. >= 80, fibrin. >= 3.5, b.i. < 65.0001, chol. >= 5.2.
```

Learning Disjunctive Rules

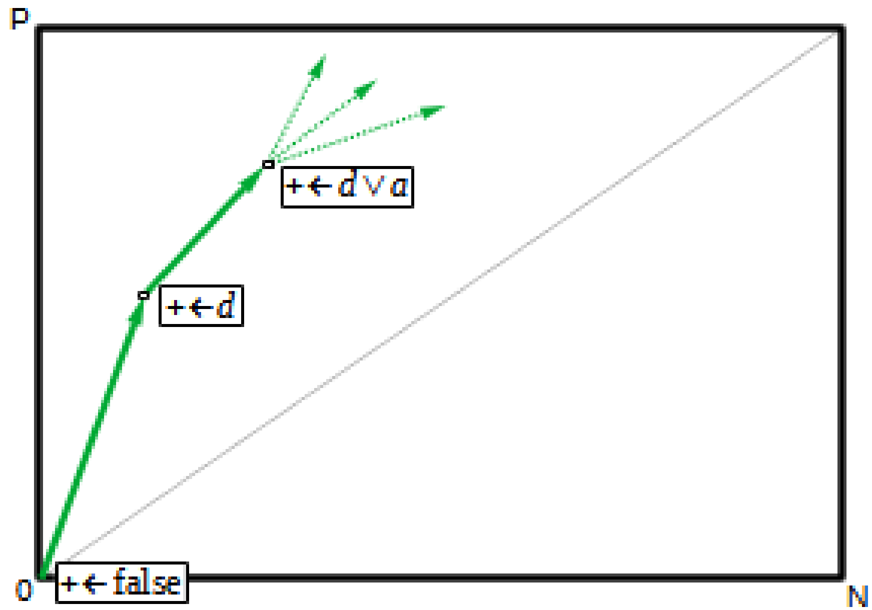
JOHANNES KEPLER UNIVERSITÄT LINZ

- Disjunctive rules can be learned analogously to conjunctive ones
  - when these are combined conjunctively, it effective learns a CNF definition for the concept
- Learning a disjunctive single rule in coverage space:

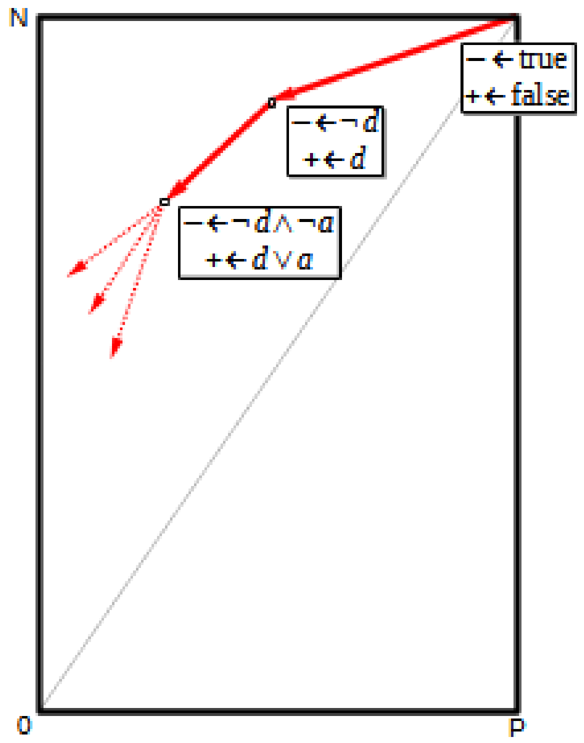


The Duality of Conjunctive and Disjunctive Learning

- By Boolean Algebra, disjunctive rules can be learned as conjunctions



disjunctive learning of a disjunctive rule

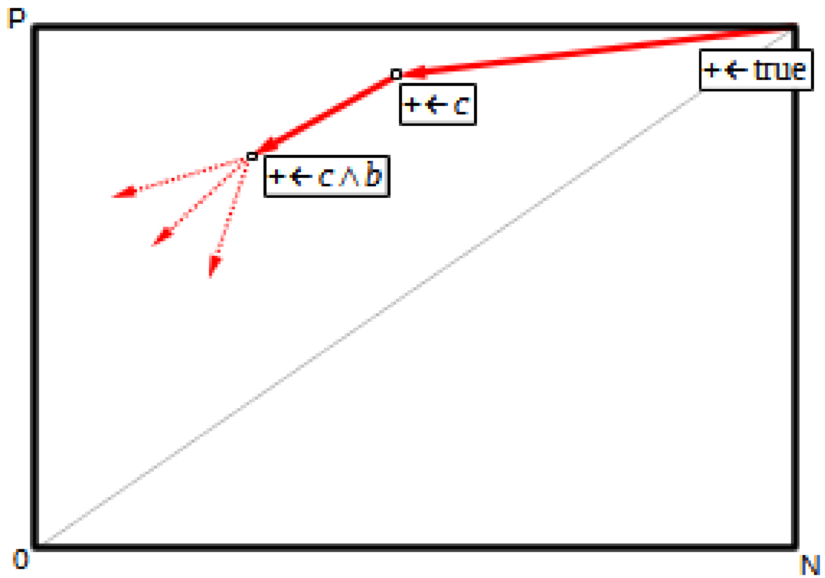


conjunctive learning of a disjunctive rule

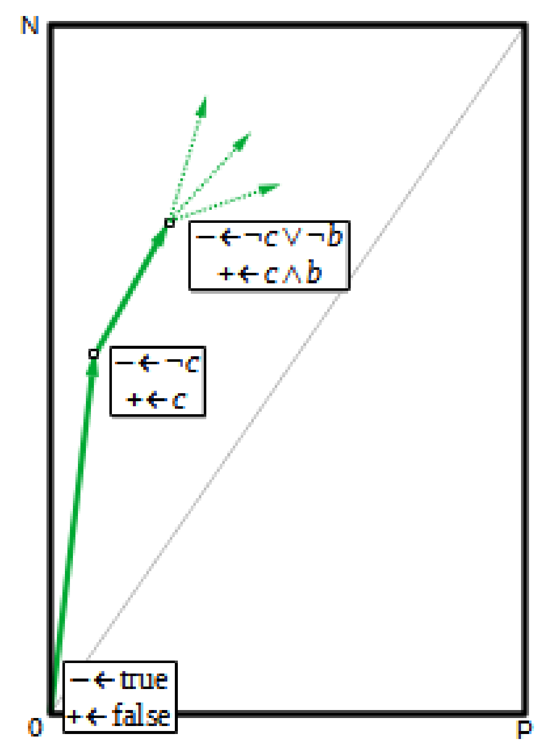
The Duality of Conjunctive and Disjunctive Learning

JOHANNES KEPLER UNIVERSITÄT LINZ

- Analogously conjunctive rules can be learned as disjunctions



conjunctive learning of a conjunctive rule



disjunctive learning of a conjunctive rule

Learning DNFs and CNFs

- DNF: disjunctively combining conjunctive rule bodies
  - often learned via covering on the positives,
  - i.e., conjunctive rules that cover
    - some examples of the class to learn
    - (almost) no examples of the other classes
  - are successively added to the rule set until all examples of the class to learn are covered
- CNF: conjunctively combining disjunctive rule bodies
  - can also be learned via excluding on the negatives
  - i.e., disjunctive rules that exclude
    - some examples of the other classes
    - (almost) no examples of the class to learn
  - are successively added to the rule set until no examples of other classes are covered

Inversion of Biases

- Note that while the learned DNF and CNF expressions are equivalent w.r.t. the training examples, they are not equivalent w.r.t. unseen examples
- in fact, the learning biases will be inverted as well!
  - if the conjunctive learner has a bias towards learning very specific concepts, it will also learn a very specific description for the negated class
  - inverting this results in a very general description for the positive class
- Explains previous results on learning with inverted heuristics:
  - using inverted heuristics results in longer rules with higher coverage
  - because more general conditions are selected
    - the focus for condition selection is maintaining a good coverage of positives, not on quickly excluding many negatives

NegLORD

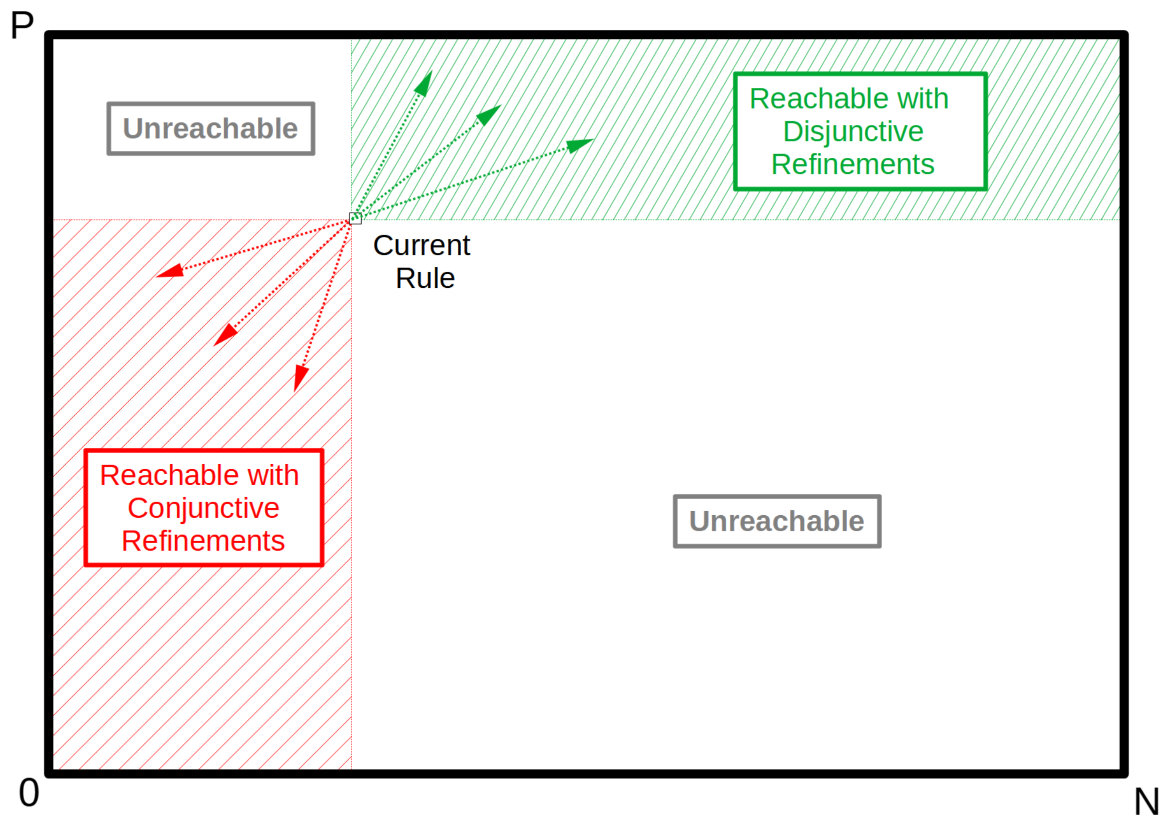
- Learn CNF descriptions using LORD as a DNF learner
- Generally works well (albeit no miracles)

DATA SET	RIPPER	K-CNF	LORD	$\overline{\text{LORD}}$
AVG. RANK	2.67	2.64	2.50	2.19

  - some datasets seem to be better learnable with CNF, others with DNF
    - more or less confirming the results of previous works
- A few caveats:
- Efficiency:
  - data structures of LORD are optimized for sparse data
  - negated features are dense → NegLORD is considerably slower
- Multi-class:
  - we have to learn descriptions for every *negated* class
  - negated classes have more examples than non-negated classes

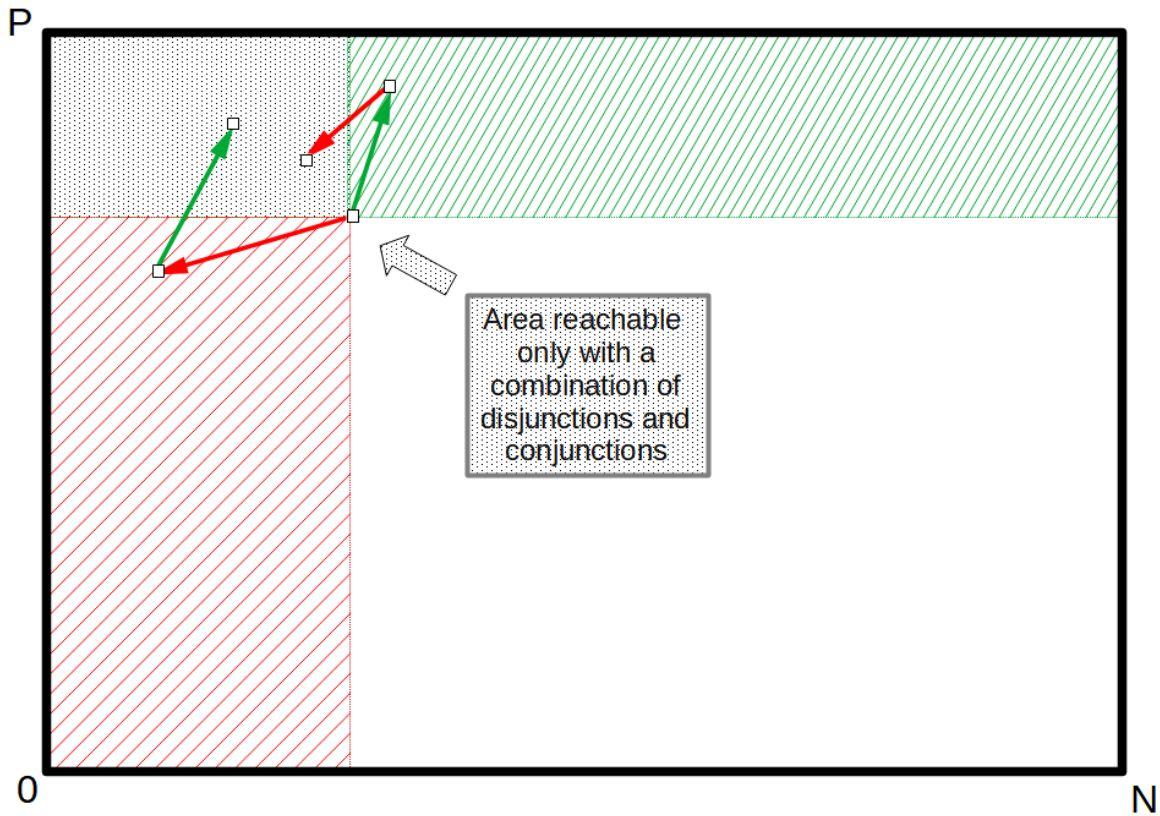
Limitations of Uni-Directional Refinements

→ The regions in coverage space that can be reached with successive (conjunctive or disjunctive) refinements are limited



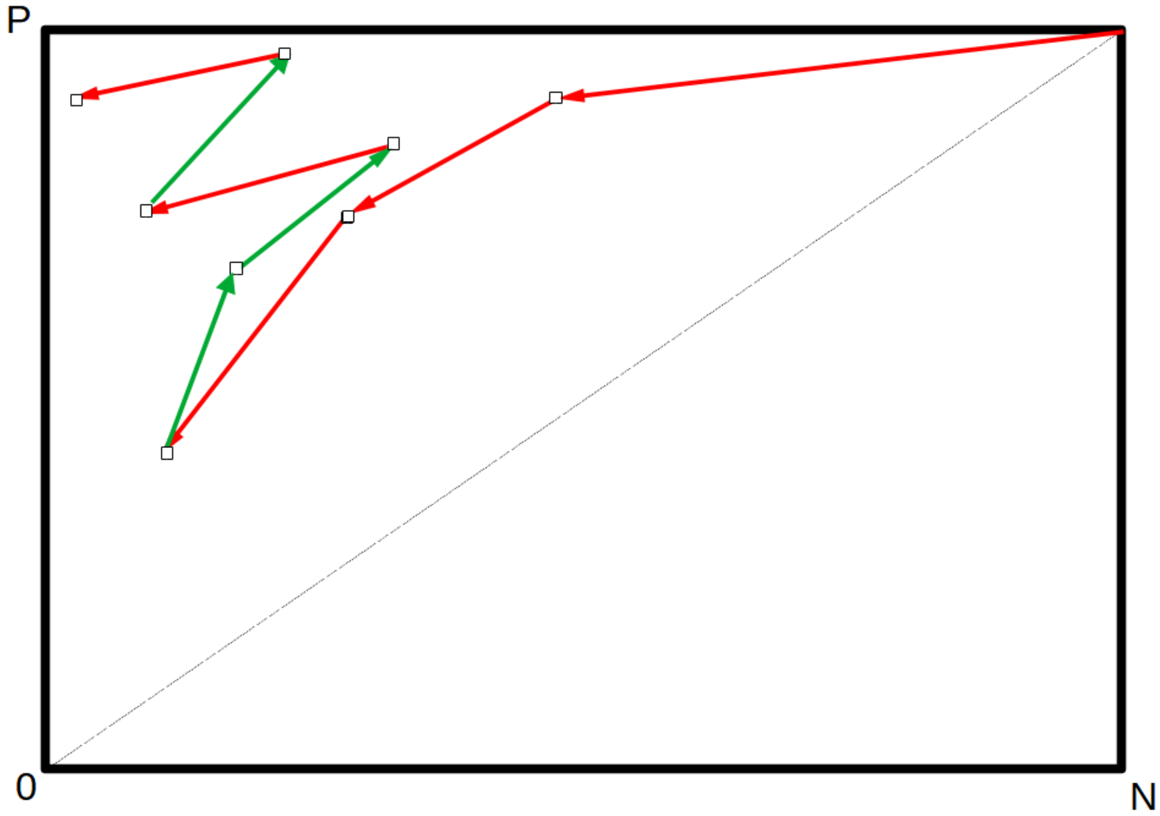
Bi-Directional Refinements

- This can be overcome with by allowing successive alternations of conjunctions and disjunctions



Bi-Directional Refinements

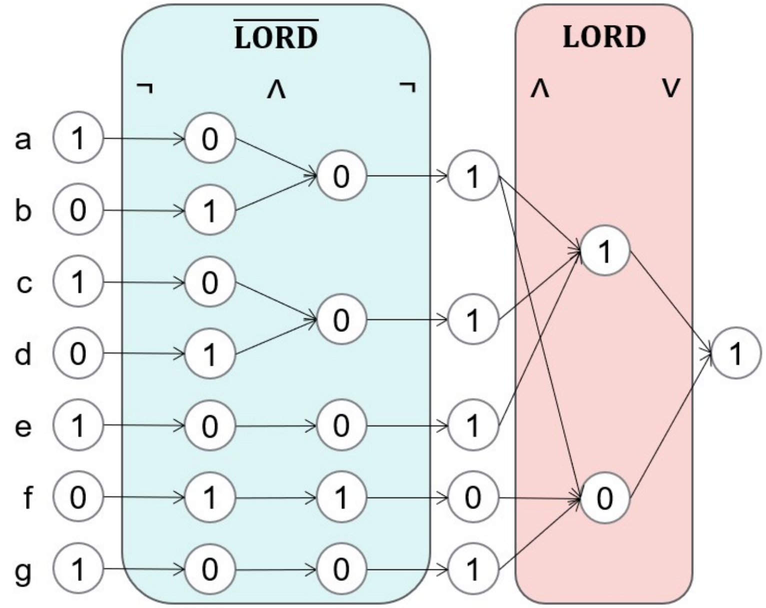
- ...which essentially corresponds to multiple alternating AND/OR layers



Learning Mixed Conjunctive and Disjunctive Rules

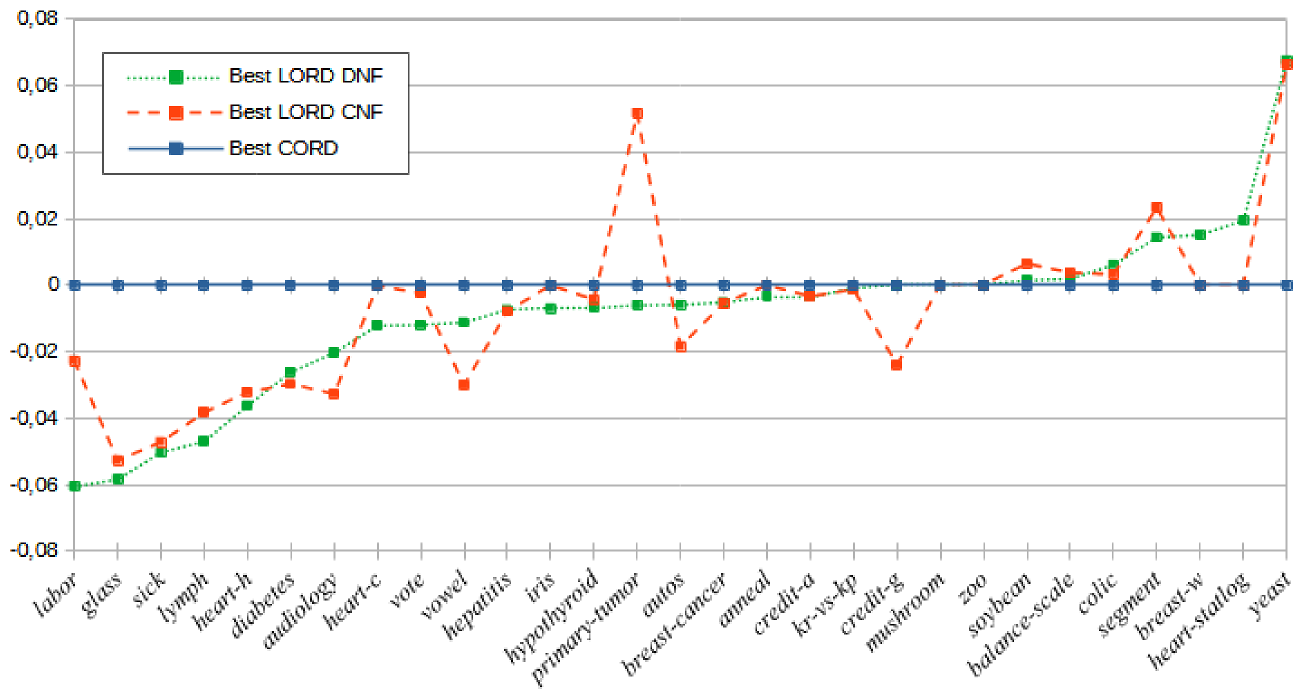
(Beck, Fürnkranz, Huynh 2023)

- LORD: A (powerful) conventional rule learner (i.e., DNF learner)
- NegLORD: Learn a CNF by inverting the problem to learn a DNF on the negated classes and negated inputs
- CORD: Allow a combination of conjunctive and disjunctive layers to potentially learn the best of both worlds



Results

- As known from previous works, some concepts can be better learned in CNF, some in DNF
- CORD is in most (but not all) cases better than either



Going Deeper

- CORD has 3 layers by default (disj./conj./disj.)
- More layers could be added with the same setup
- Results show modest but not consistent improvements for carefully tuned networks

From → to	2 → 3	2 → 4	2 → 5	3 → 4	3 → 5	4 → 5
# Impr.	6219	6189	6788	4407	4877	3189
# Det.	5274	5301	6057	4452	5007	3289
% Impr.	24.75	24.63	27.01	17.54	19.41	12.69
% Det.	20.99	21.09	24.10	17.72	19.92	13.09
Values for best five-layered Cord:						
# Impr.	126	139	144	86	97	40
# Det.	48	53	52	62	56	17
% Impr.	43.45	47.93	49.66	29.66	33.45	13.79
% Det.	16.55	18.28	17.93	21.38	19.31	5.86

Analysis of Deeper Networks

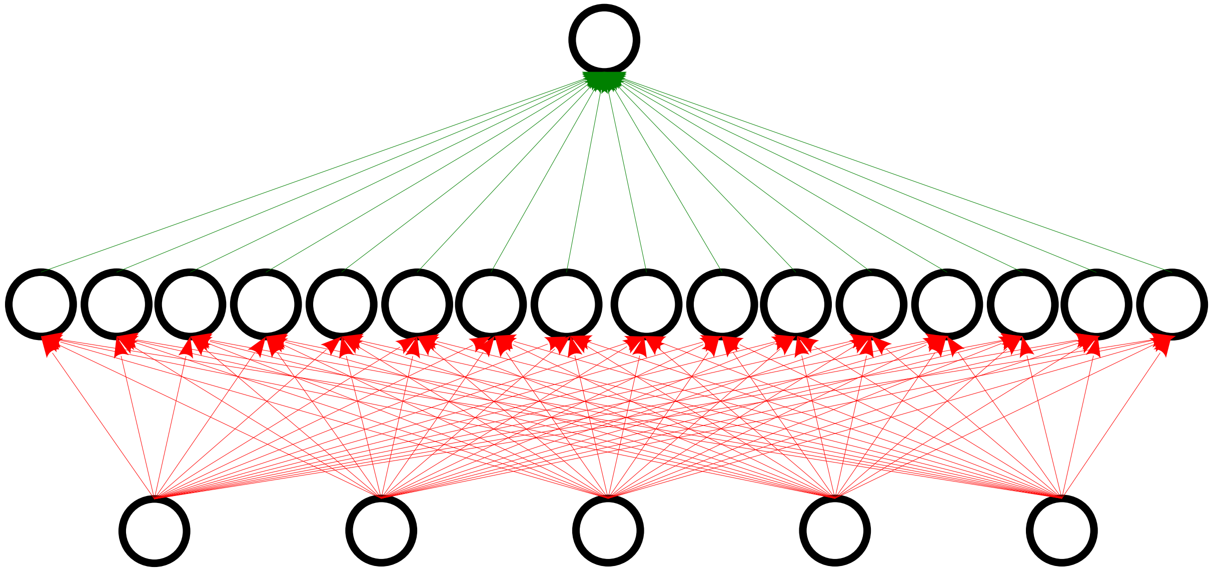
- positive and negative correlation of various properties in the conjunctive and disjunctive layers of 5-layer networks with overall accuracy

	CORD				DORC			
	D_1	C_2	D_3	C_4	C_1	D_2	C_3	D_4
m	0.154	0.020	-0.101	-0.131	0.081	0.175	0.019	-0.098
# Rules	-0.189	-0.145	-0.092	-0.043	-0.084	-0.253	-0.134	-0.081
# Concepts	-	0.095	0.045	0.008	-	0.060	0.151	0.074
Avg. Depth	-	0.111	0.057	-0.018	-	0.117	0.159	0.107
Accuracy	0.203	0.520	0.690	-	-0.041	0.342	0.564	-

  - e.g., higher values of the m-parameter (yielding more general rules) are good in early layers, wheras lower values are better in later layers
  - accuracy increases in later layers

Flat Rule Learning

- Flat Rule Sets can be converted into a network using a single AND and a single OR layer (→ a DNF expression)

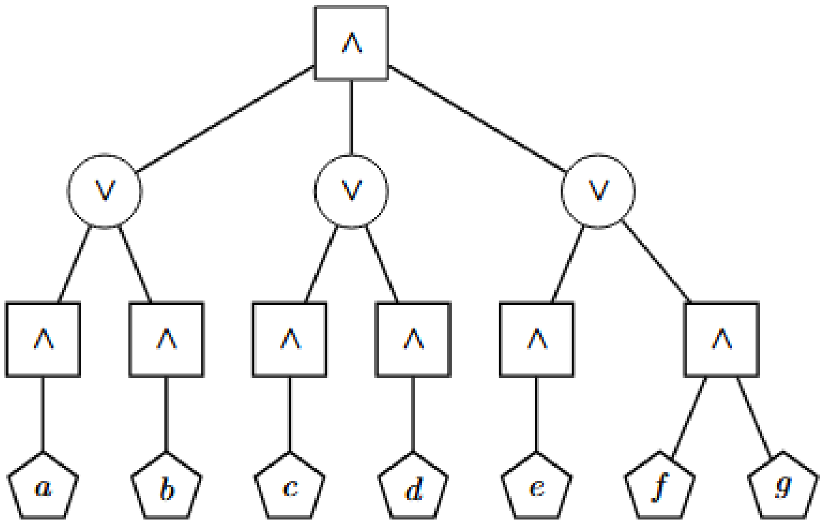


- Each node in the hidden layer corresponds to one rule
  - typically it is a local pattern, covering part of the target

Deep Rule Learning

- Deep Rule Networks with alternating AND and OR layers corresponds to multiple rule layers
  - each conjunctive node corresponds to one rule
  - each disjunctive node corresponds to a rule set

Negated Normal Form (NNF)



$(a \vee b) \wedge (c \vee d) \wedge [e \vee (f \wedge g)]$

Deep Rule Representation

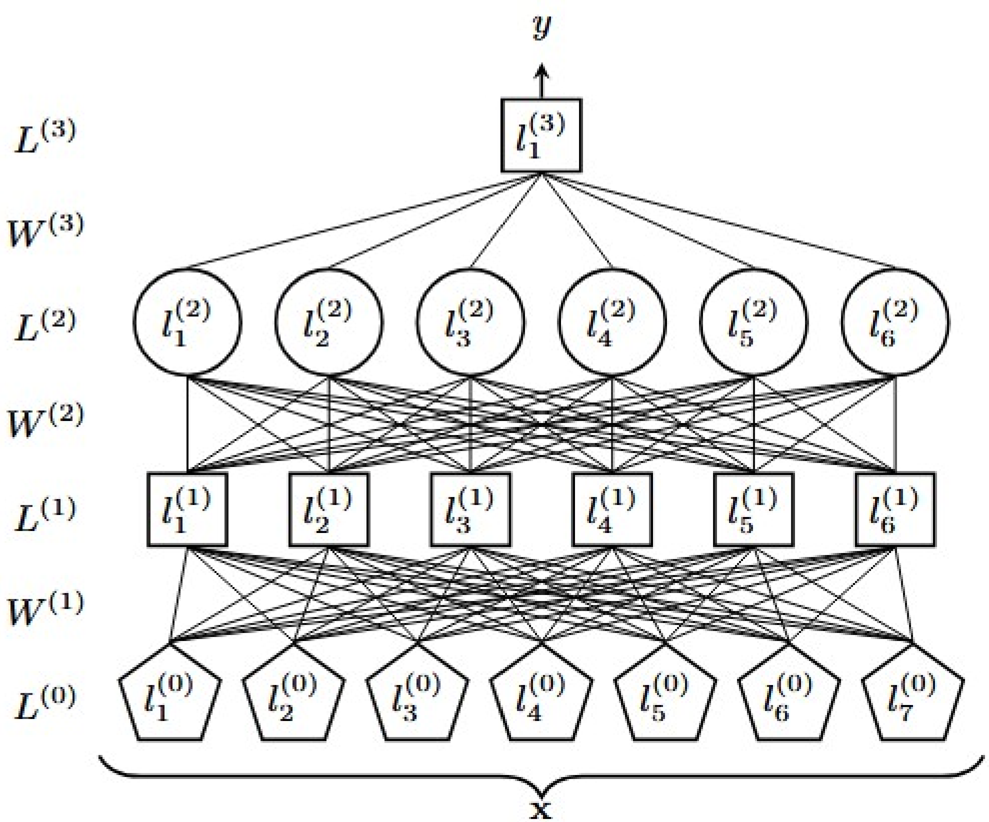
```
+ :- h21, h22, h23.
h21 :- a.     h22 :- c.     h23 :- e.
h21 :- b.     h22 :- d.     h23 :- f, g.
```

Flat Rule Representation (DNF)

```
+ :- a, c, e.          + :- b, c, e.
+ :- a, c, f, g.       + :- b, c, f, g.
+ :- a, d, e.          + :- b, d, e.
+ :- a, d, f, g.       + :- b, d, f, g.
```

General Approach

- Provide a fully connected network structure
- find binary weights for the edges



- 7 input features
- 2 hidden layers of size 6

$\rightarrow 7\times6+6\times6+6\times1=84$ weights

- we also need to store and propagate the activation of each node for each training example

$\rightarrow (7+6+6+1)\times N$ variables

Simple Greedy Local Search

(Beck & Fürnkranz 2020)

JOHANNES KEPLER UNIVERSITÄT LINZ

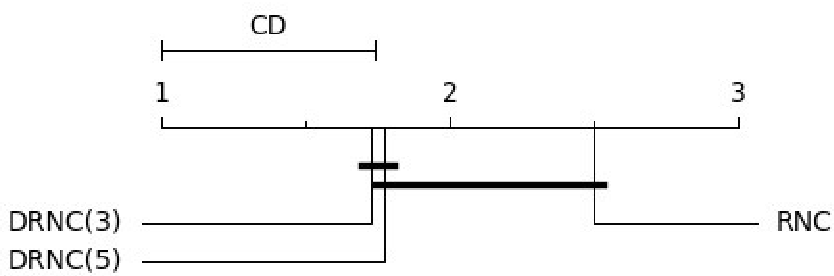
- Find the weights using a **simple optimization algorithm** to learn both, deep and shallow representations
  1) Fix a network architecture
     - Shallow, single layer network RNC: [20]
     - Deep 3-layer network DRNC(3): [32, 8, 2]
     - Deep 5-layer network DRNC(5): [32, 16, 8, 4, 2]
  2) Initialize Boolean weights probabilistically
  3) Use stochastic local search to find best weight „flip“ on a mini-batch of data until convergence
  4) Optimize finally on whole training set

Main goal: See if deep structure can be useful

Results on Artificial Datasets

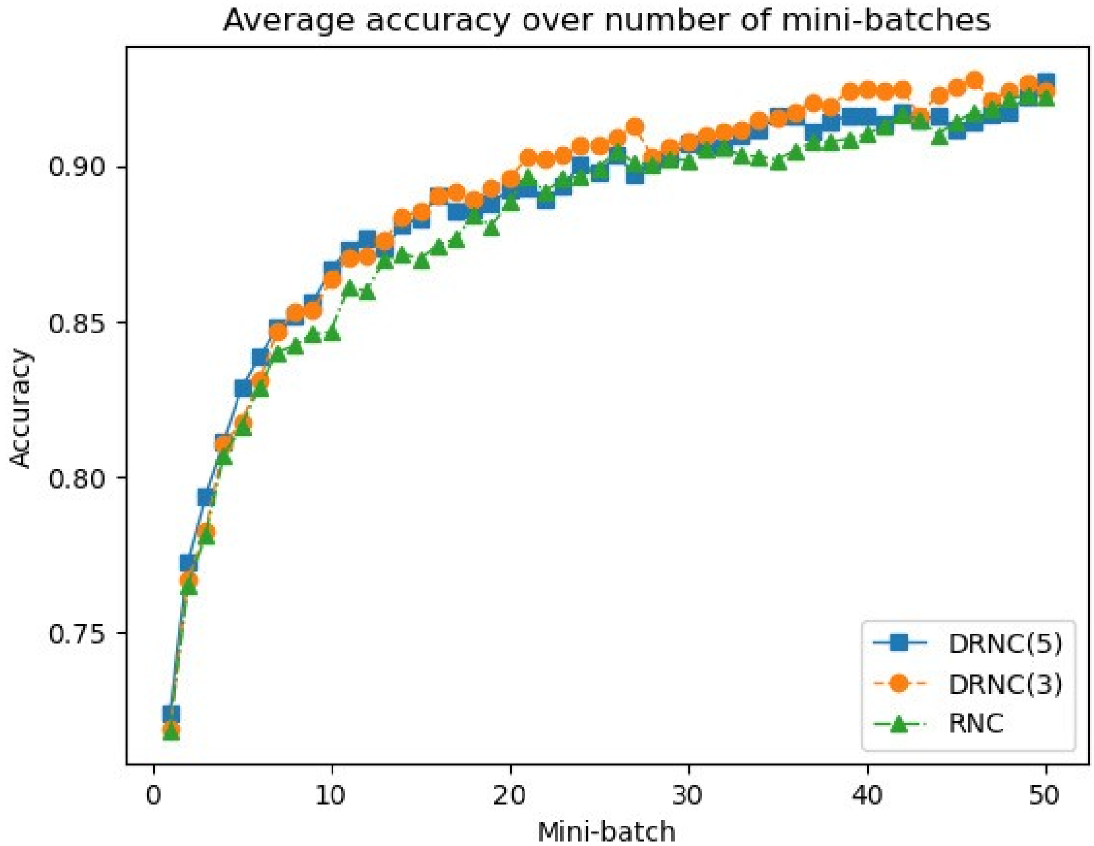
- 20 artificial datasets with 10 Boolean inputs, 1 Boolean output
  - generated from a randomly initialized (deep) Boolean network

seed %(+)	DRNC(5)	DRNC(3)	RNC	RIPPER	CART
Ø Accuracy	0.9467	0.9502	0.9386	0.9591	*0.9644*
Ø Rank	1.775	1.725	2.5		



- DRNC(3) [DRNC(5)] outperforms RNC on a significance level of more than 95% [90%]

Learning Curves (Artificial Datasets)



- DRNC(3) and DRNC(5) converge faster than RN

Results on Real-World (UCI) Datasets

(Beck & Fürnkranz 2021) JOHANNES KEPLER UNIVERSITÄT LINZ

dataset	%(+)	DRNC(5)	DRNC(3)	RNC	RIPPER	CART
car-evaluation	0.7002	0.8999	**0.9022**	0.8565	*0.9838*	0.9821
connect-4	0.6565	**0.7728**	0.7712	0.7597	0.7475	*0.8195*
kr-vs-kp	0.5222	0.9671	0.9643	**0.9725**	0.9837	*0.989*
monk-1	0.5000	***1***	0.9982	0.9910	0.9478	0.8939
monk-2	0.3428	0.7321	**0.7421**	0.7139	0.6872	*0.7869*
monk-3	0.5199	**0.9693**	0.9603	0.9567	0.9386	*0.9729*
mushroom	0.784	***1***	0.978	0.993	0.9992	*1*
tic-tac-toe	0.6534	0.8956	0.9196	**0.9541**	*1*	0.9217
vote	0.6138	***0.9655***	0.9288	0.9264	0.9011	0.9287
Ø Rank		1.556	2	2.444		

- DRNC(5) has the best performance on these real-world datasets, followed by DRNC(3)

Optimization Using SAT

(Seip et al. 2025)

If we ignore noise etc., and (for simplicity here, but not in general) assume that all hidden layers have size m, a more precise formulation could be the following:

- Given features $f_{i,j}$ $(1 \leq i \leq n,\ 1 \leq j \leq d)$
- Find $N_W = m^2 \cdot (h-1) + m \cdot (d+1)$ Boolean values
  - $W^{(1)} = w_{i,j}^{(1)}$ $(1 \leq i \leq d,\ 1 \leq j \leq m)$
  - $W^{(l)} = w_{i,j}^{(l)}$ $(2 \leq l \leq h,\ 1 \leq i,j \leq m)$
  - $W^{(h+1)} = w_i^{(h+1)}$ $(1 \leq i \leq m)$
- such that
  - $a_{i,k}^{(1)} = \bigwedge_{j=1}^{d}(f_{i,j} \wedge w_{j,k}^{(1)})$ $(1 \leq i \leq n, 1 \leq k \leq m)$
  - $a_{i,k}^{(l)} = \bigwedge_{j=1}^{m}(a_{i,j}^{(l-1)} \wedge w_{j,k}^{(l)})$ **if** l is odd $(2 \leq l \leq h)$

    $a_{i,k}^{(l)} = \bigvee_{j=1}^{m}(a_{i,j}^{(l-1)} \wedge w_{j,k}^{(l)})$ **if** l is even
  - $y_i = \bigwedge_{j=1}^{m}(a_{i,j}^{(h)} \wedge w_j^{(h+1)})$ **if** $h+1$ is odd

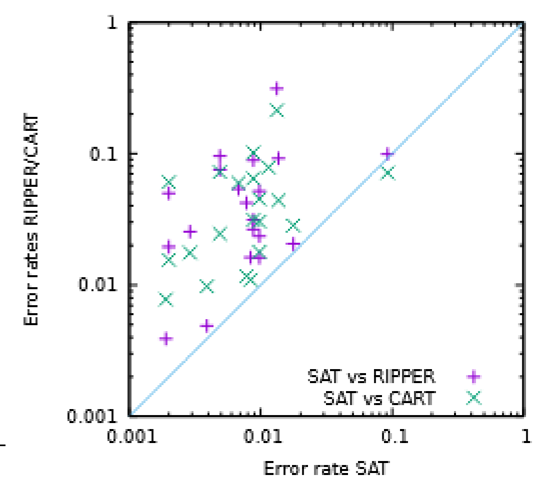
    $y_i = \bigvee_{j=1}^{m}(a_{i,j}^{(h+1)} \wedge w_j^{(h+1)})$ **if** $h+1$ is even

Results

JOHANNES KEPLER UNIVERSITÄT LINZ

	NNF(5)		NNF(3)		NNF(1)		RIPPER	CART
	SAT	SHS	SAT	SHS	SAT	SHS		
Artificial Datasets								
Ø acc.	**0.9857**	0.9467	**0.9852**	0.9502	***0.9939***	0.9386	0.9591	0.9644
Ø rank	2.7	5.9	2.7	6.05	1.45	7	4.3	4.4
UCI Datasets								
Ø acc.	**0.9682**	0.9229	**0.9722**	0.9190	***0.9807***	0.9153	0.9290	0.9289
Ø rank	4.1429	4.4286	3.1429	5.7143	2.1429	5.8571	5.5714	4.2857

- Optimization works well
  - SAT trained outperform greedily trained networks
  - as well as Ripper and CART
- but deep structures do not seem to be helpful
  - because flat networks have fewer parameters?
- … and scalability is a huge problem



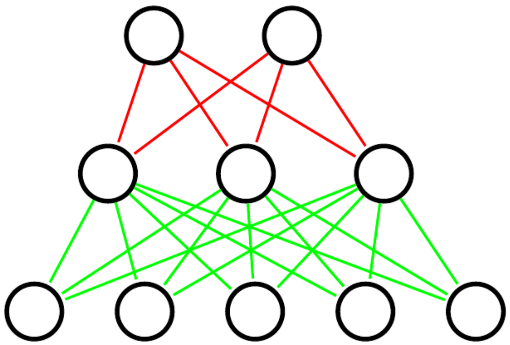
Various Ideas for Speed-Up

JOHANNES KEPLER UNIVERSITÄT LINZ

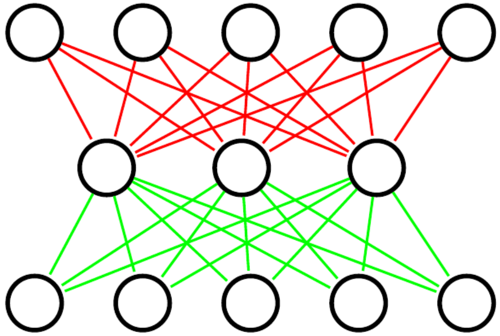
- incremental freezing of weights via mini-batch optimization
  - find optimal weight settings for various small mini-batches
  - combine then so that successively stable hidden concepts emerge
- use feedback from the SAT solver to recognize potentially noisy examples and remove them
- use LORD to focus on relevant features
  - each example can be represented by the rule LORD generates
  - may also help to cope with noise
- pre-training using flat Boolean auto-encoders
  - in case these are feasible...

Further Ideas to Explore

- Joint learning of multiple outputs with the same network
  - rule learning algorithms learn each output independently
  - joint optimization should yield smaller formulas
  - even for flat rule sets



- Boolean auto-encoders
  - compressing Boolean data by learning a function that can reconstruct the data from fewer variables (→ embedding)
  - layer-wise pre-training with auto-encoders was one of the first successful deep learning approaches



Conclusions

- Learning more complex rule sets
  - Locally optimal rule induction (LORD)
  - Learns one rule per example (in analogy to XAI approaches)
- Learning more complex rules
  - Characteristic Rules vs. Discriminative Rules
  - Related to learning disjunctive vs. conjunctive concepts
- Learning deeper rule sets
  - greedy training is possible but not very effective
  - SAT-based optimization is better but quite inefficient

- Main bottleneck is scalability
  - Various ideas for improvements currently under investigation

References

- Seip P., Fürnkranz J., Beck F., Hofstadler C., Pfeiffer P., Seidl M., Peharz R., Szeider S. Towards SAT-Based Learning of NNF Networks, *Proc. 27th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2025.
- Beck F., Fürnkranz J.: An Empirical Investigation into Deep and Shallow Rule Learning. *Frontiers in Artificial Intelligence* 4 (2021). doi:10.3389/frai.2021.689398
- Huynh P. V. Q., Fürnkranz, J., Beck, F.: Efficient learning of large sets of locally optimal classification rules. *Machine Learning* 112(2): 571-610 (2023). doi:10.1007/s10994-022-06290-w.
- Beck F., Fürnkranz J., Huynh P.V.C.: Layerwise Learning of Mixed Conjunctive and Disjunctive Conference on Rules and Reasoning (RuleML) 2023:95-109
- Beck F., Fürnkranz J.: Beyond DNF: On the Incremental Construction of Deep Rule Theories, in *Proceedings of the 22nd Conference Information Technologies - Applications and Theory (ITAT)*, pp. 61--68, 2022.
- Beck F., Fürnkranz J.: An Investigation into Mini-Batch Rule Learning, in *Proceedings of ,* Rule Sets – A Challenge for Deep Discrete Learning, in *Proceedings of the 2nd Workshop on Deep Continuous-Discrete Machine Learning (DeCoDeML)*, 2020.