Learning of Rule Sets

- Introduction
  - Learning Rule Sets
  - Terminology
  - Coverage Spaces
- Separate-and-Conquer Rule Learning
  - Covering algorithm
  - Top-Down Hill-Climbing
  - Rule Evaluation Heuristics
  - Overfitting and Pruning
  - Multi-Class Problems
  - Bottom-Up Hill-Climbing

Learning Rule Sets

- many datasets cannot be solved with a single rule
  - not even the simple weather dataset
  - they need a rule set for formulating a target theory
- finding a computable generality relation for rule sets is not trivial
  - adding a condition to a rule specializes the theory
  - adding a new rule to a theory generalizes the theory
- practical algorithms use different approaches
  - covering or separate-and-conquer algorithms
  - based on heuristic search

A sample task

Temperature	*Outlook*	*Humidity*	*Windy*	*Play Golf?*
hot	**sunny**	**high**	**false**	**no**
hot	**sunny**	**high**	**true**	**no**
hot	**overcast**	**high**	**false**	**yes**
cool	**rain**	**normal**	**false**	**yes**
cool	**overcast**	**normal**	**true**	**yes**
mild	**sunny**	**high**	**false**	**no**
cool	**sunny**	**normal**	**false**	**yes**
mild	**rain**	**normal**	**false**	**yes**
mild	**sunny**	**normal**	**true**	**yes**
mild	**overcast**	**high**	**true**	**yes**
hot	**overcast**	**normal**	**false**	**yes**
mild	**rain**	**high**	**true**	**no**
cool	**rain**	**normal**	**true**	**no**
mild	**rain**	**high**	**false**	**yes**

- Task:
  - Find a rule set that correctly predicts the dependent variable from the observed variables

A Simple Solution

IF T=hot AND H=high AND O=overcast AND W=false THEN yes
IF T=cool AND H=normal AND O=rain AND W=false THEN yes
IF T=cool AND H=normal AND O=overcast AND W=true THEN yes
IF T=cool AND H=normal AND O=sunny AND W=false THEN yes
IF T=mild AND H=normal AND O=rain AND W=false THEN yes
IF T=mild AND H=normal AND O=sunny AND W=true THEN yes
IF T=mild AND H=high AND O=overcast AND W=true THEN yes
IF T=hot AND H=normal AND O=overcast AND W=false THEN yes
IF T=mild AND H=high AND O=rain AND W=false THEN yes

- The solution is
  - a set of rules
  - that is complete and consistent on the training examples

  → it must be part of the version space
- but it does not generalize to new examples!

The Need for a Bias

- rule sets can be generalized by
  - generalizing an existing rule (as usual)
  - introducing a new rule (this is new)
- a minimal generalization could be
  - introduce a new rule that covers only the new example
- Thus:
  - The solution on the previous slide will be found as a result of the FindS algorithm
  - FindG (or Batch-FindG) are less likely to find such a bad solution because they prefer general theories
- But in principle this solution is part of the hypothesis space and also of the version space

⇒ *we need a search bias to prevent finding this solution!*

A Better Solution

IF Outlook = overcast THEN yes

IF Humidity = normal AND Outlook = sunny THEN yes

IF Outlook = rainy AND Windy = false THEN yes

Recap: Batch-Find

- Abstract algorithm for learning a single rule:

1. Start with an empty theory T and training set E
2. Learn a single (*consistent*) rule R from E and add it to T
3. return T

- Problem:
  - the basic assumption is that the found rules are complete, i.e., they cover all positive examples
  - What if they don't?
- Simple solution:
  - If we have a rule that covers part of the positive examples:
  - add some more rules that cover the remaining examples

Separate-and-Conquer Rule Learning

- Learn a set of rules, one by one

1. Start with an empty theory T and training set E
2. Learn a single (*consistent*) rule R from E and add it to T
3. If T is satisfactory (*complete*), return T
4. Else:
   - *Separate:* Remove examples explained by R from E
   - *Conquer:* If E is non-empty, goto 2.

- One of the oldest family of learning algorithms
  - goes back AQ (Michalski, 60s)
  - FRINGE, PRISM and CN2: relation to decision trees (80s)
  - popularized in ILP (FOIL and PROGOL, 90s)
  - RIPPER brought in good noise-handling
- Different learners differ in how they find a single rule

Relaxing Completeness and Consistency

- So far we have always required a learner to learn a complete and consistent theory
  - e.g., one rule that covers all positive and no negative examples
- This is not always a good idea (→ overfitting)
- Motivating Example:

Training set with 200 examples, 100 positive and 100 negative

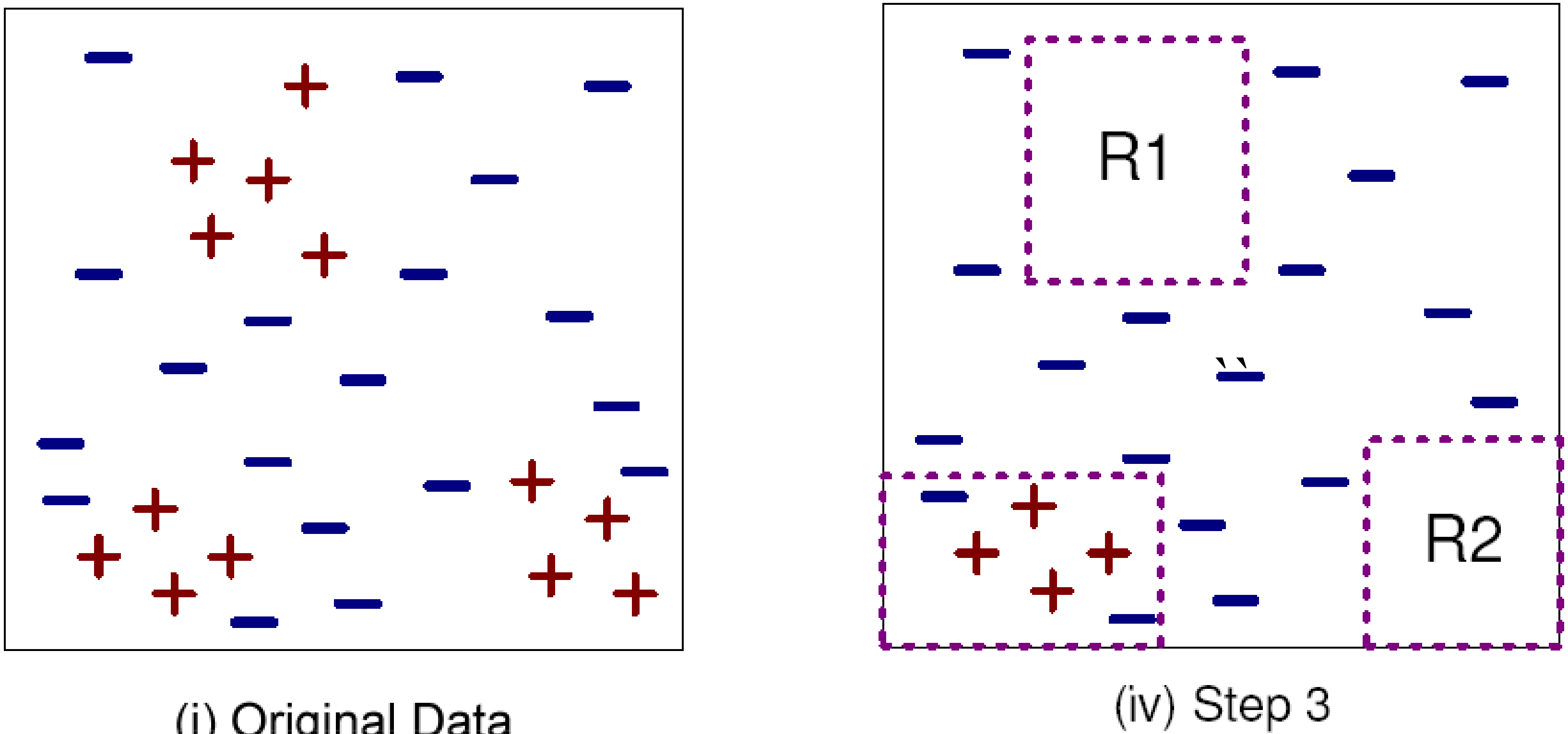
- **Theory A** consists of 100 complex rules, each covering a single positive example and no negatives

  → Theory A is complete and consistent on the training set
- **Theory B** consists of a single rule, covering 99 positive and 1 negative example

  → Theory B is incomplete and incosistent on the training set

Which one will generalize better to unseen examples?

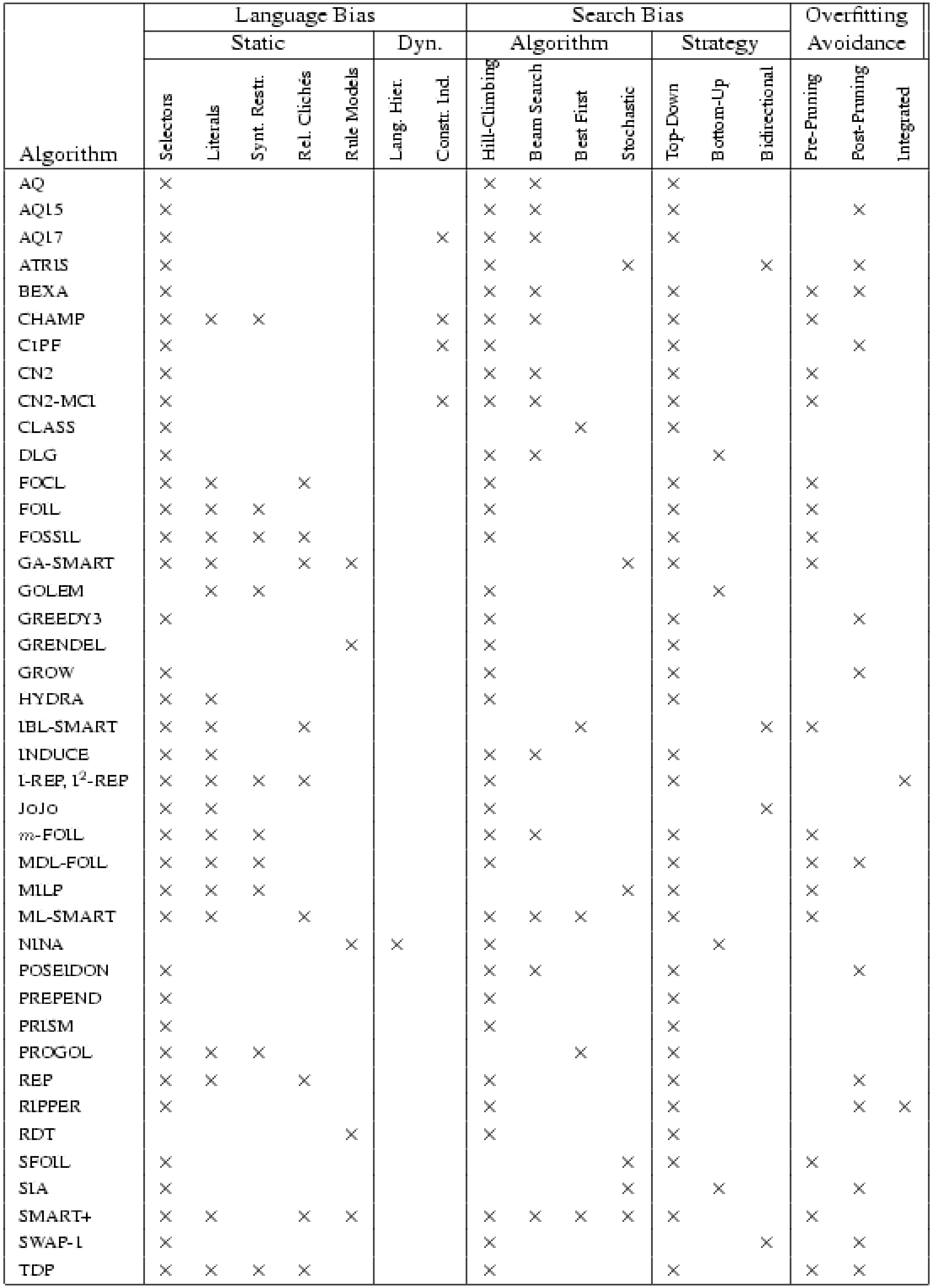
Separate-and-Conquer Rule Learning



(i) Original Data

(iv) Step 3

Quelle für Grafiken: http://www.cl.uni-heidelberg.de/kurs/ws03/einfki/KI-2004-01-13.pdf



Algorithm	Language Bias							Search Bias							Overfitting Avoidance		
	Static					Dyn.		Algorithm				Strategy					
	Selectors	Literals	Synt. Restr.	Rel. Cliches	Rule Models	Lang. Hier.	Constr. Ind.	Hill-Climbing	Beam Search	Best First	Stochastic	Top-Down	Bottom-Up	Bidirectional	Pre-Pruning	Post-Pruning	Integrated
AQ	×							×	×			×					
AQ15	×							×	×			×				×	
AQ17	×						×	×	×			×					
ATRIS	×							×			×			×		×	
BEXA	×							×	×			×			×	×	
CHAMP	×	×	×				×	×	×			×			×		
C1PF	×						×	×				×				×	
CN2	×							×	×			×			×		
CN2-MCI	×						×	×	×			×			×		
CLASS	×									×		×					
DLG	×							×	×				×				
FOCL	×	×		×				×				×			×		
FOIL	×	×	×					×				×			×		
FOSSIL	×	×	×	×				×				×			×		
GA-SMART	×	×		×	×						×	×			×		
GOLEM		×	×					×					×				
GREEDY3	×							×				×				×	
GRENDEL					×			×				×					
GROW	×							×				×				×	
HYDRA	×	×						×				×					
IBL-SMART	×	×		×						×				×	×		
INDUCE	×	×						×	×			×					
I-REP, I^2-REP	×	×	×	×				×				×					×
JoJo	×	×						×						×			
m-FOIL	×	×	×					×	×			×			×		
MDL-FOIL	×	×	×					×				×			×	×	
MILP	×	×	×								×	×			×		
ML-SMART	×	×		×				×	×	×		×			×		
NINA					×	×		×					×				
POSEIDON	×							×	×			×				×	
PREPEND	×							×				×					
PRISM	×							×				×					
PROGOL	×	×	×							×		×					
REP	×	×		×				×				×				×	
RIPPER	×							×				×				×	×
RDT					×			×				×					
SFOIL	×										×	×			×		
SIA	×										×		×			×	
SMART+	×	×		×	×			×	×	×	×	×			×		
SWAP-1	×							×						×		×	
TDP	×	×	×	×				×				×			×	×	

- language bias:
  - which type of conditions are allowed (static)
  - which combinations of condictions are allowed (dynamic)
- search bias:
  - search heuristics
  - search algorithm (greedy, stochastic, exhaustive)
  - search strategy (top-down, bottom-up)
- overfitting avoidance bias:
  - pre-pruning (stopping criteria)
  - post-pruning

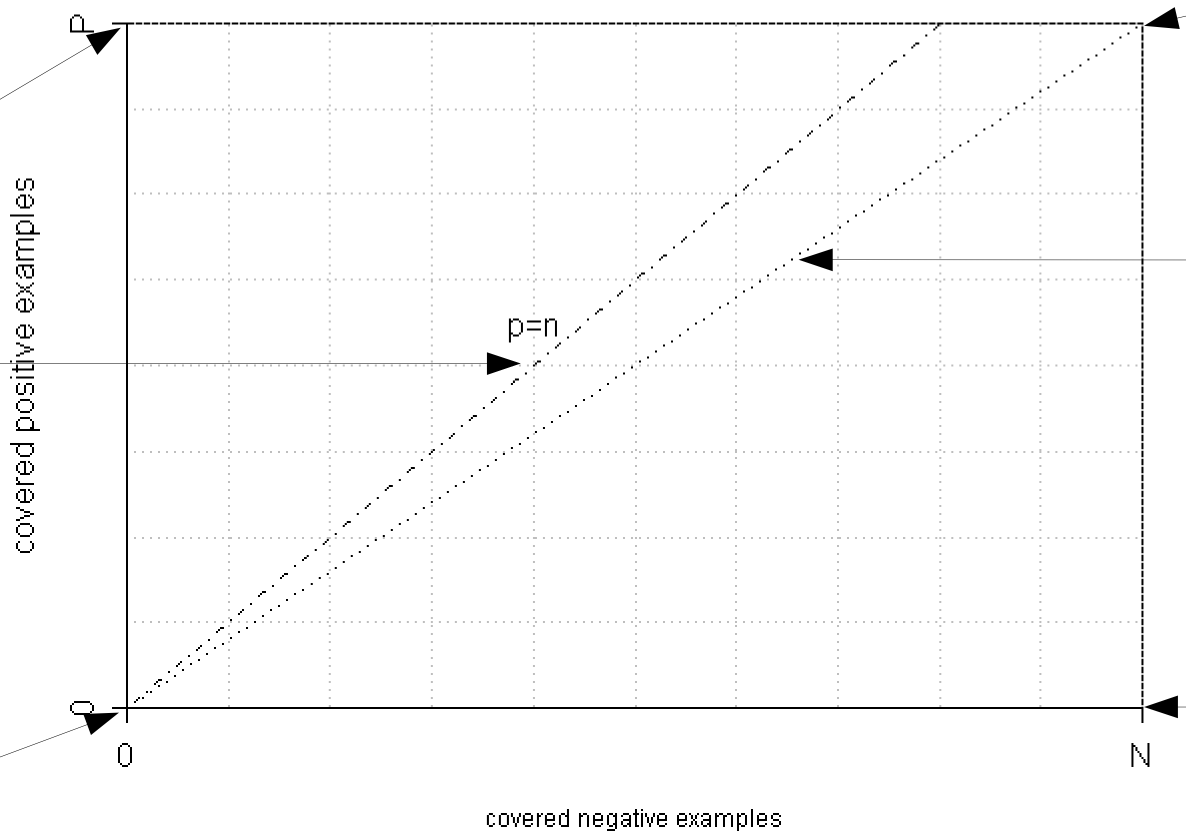
Terminology

- training examples
  - P: total number of positive examples
  - N: total number of negative examples
- examples covered by the rule (predicted positive)
  - true positives p: positive examples covered by the rule
  - false positives n: negative examples covered by the rule
- examples not covered the rule (predicted negative)
  - false negatives P-p: positive examples not covered by the rule
  - true negatives N-n: negative examples not covered by the rule

	predicted +	predicted -	
class +	p **(true positives)**	P-p **(false negatives)**	P
class -	n **(false positives)**	N-n **(true negatives)**	N
	$p + n$	$P+N-(p+n)$	$P+N$

Coverage Spaces

- good tools for visualizing properties of covering algorithms
  - each point is a theory covering p positive and n negative examples



perfect theory: all positive and no negative examples are covered

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

empty theory: no examples are covered

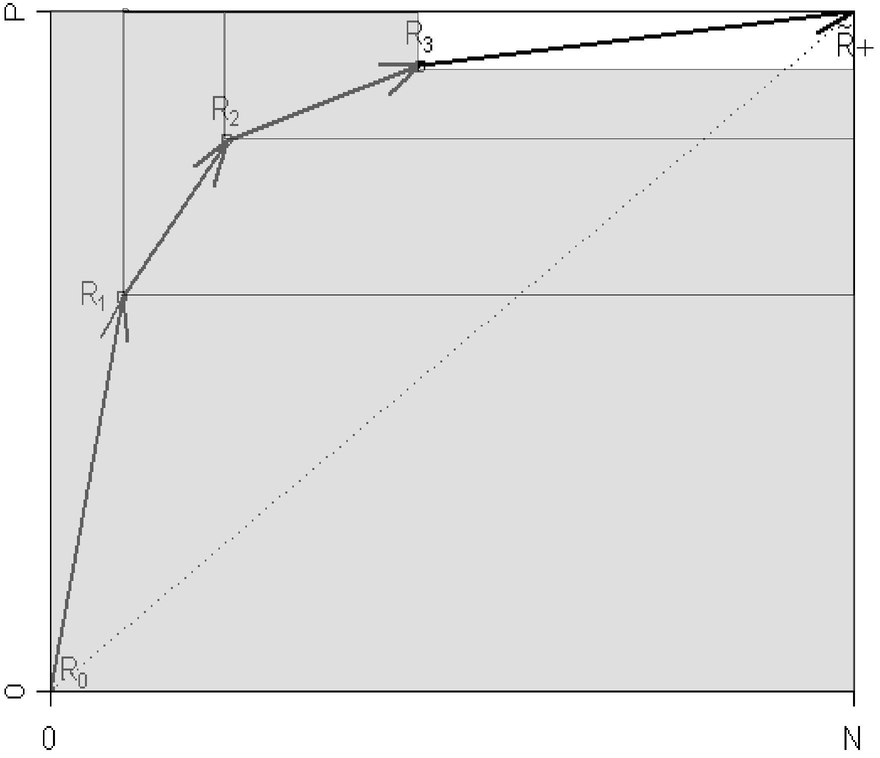
universal theory: all examples are covered

default distribution: maintain $P/(P+N)$ positive and $N/(P+N)$ negative examples

opposite theory: all negative and no positive examples are covered

Covering Strategy

- Covering or Separate-and-Conquer rule learning learning algorithms learn one rule at a time
- This corresponds to a path in coverage space:
  - The empty theory R_0 (no rules) corresponds to (0,0)
  - Adding one rule never decreases p or n because adding a rule covers *more* examples (generalization)
  - The universal theory R+ (all examples are positive) corresponds to (N,P)



Top-Down Hill-Climbing

- Top-Down: A rule is successively *specialized*

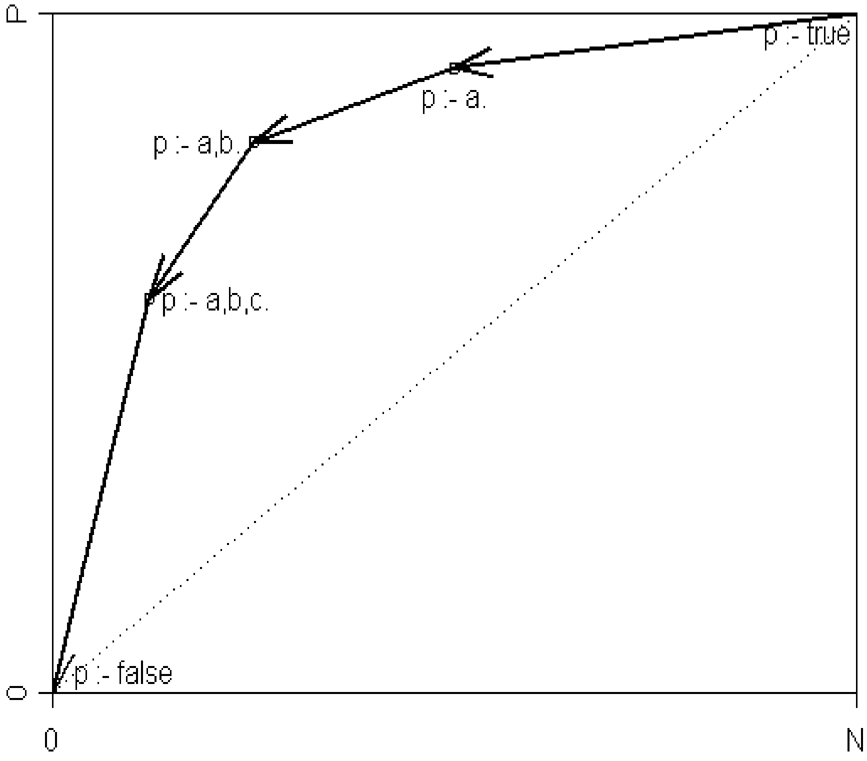
```
1. Start with an empty rule R that covers all examples
2. Evaluate all possible ways to add a condition to R
3. Choose the best one (according to some heuristic)
4. If R is satisfactory, return it
5. Else goto 2.
```

- Almost all greedy s&c rule learning systems use this strategy

Top-Down Hill-Climbing

- successively extends a rule by adding conditions

- This corresponds to a path in coverage space:
    - The rule `p:-true` covers all examples (universal theory)
    - Adding a condition never increases p or n (specialization)
    - The rule `p:-false` covers no examples (empty theory)



- which conditions are selected depends on a *heuristic function* that estimates the quality of the rule

Rule Learning Heuristics

- Adding a rule should
  - increase the number of covered negative examples as little as possible (do not decrease *consistency*)
  - increase the number of covered positive examples as much as possible (increase *completeness*)
- An evaluation heuristic should therefore trade off these two extremes
  - Example: Laplace heuristic $h_{Lap} = \frac{p+1}{p+n+2}$
    - grows with $p \rightarrow \infty$
    - grows with $n \rightarrow 0$
  - Note: Precision is not a good heuristic. Why?

$$h_{Prec} = \frac{p}{p+n}$$

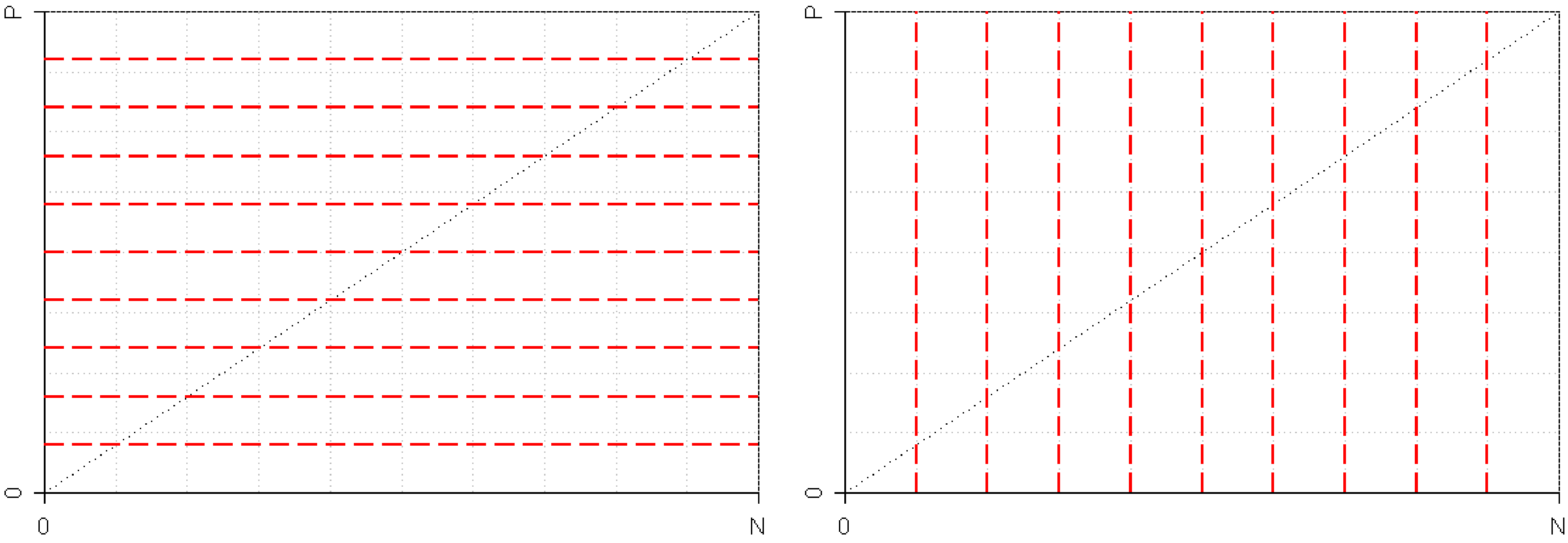
Example

Condition		p	n	Precision	Laplace	p-n
Temperature =	Hot	2	2	0.5000	0.5000	0
	Mild	3	1	0.7500	0.6667	2
	Cold	4	2	0.6667	0.6250	2
Outlook =	Sunny	2	3	0.4000	0.4286	-1
	Overcast	4	0	1.0000	0.8333	4
	Rain	3	2	0.6000	0.5714	1
Humidity =	High	3	4	0.4286	0.4444	-1
	Normal	6	1	0.8571	0.7778	5
Windy =	True	3	3	0.5000	0.5000	0
	False	6	2	0.7500	0.7000	4

- Heuristics Precision and Laplace
  - add the condition Outlook= Overcast to the (empty) rule
  - stop and try to learn the next rule
- Heuristic Accuracy / $p - n$
  - adds Humidity = Normal
  - continue to refine the rule (until no covered negative)

Isometrics in Coverage Space

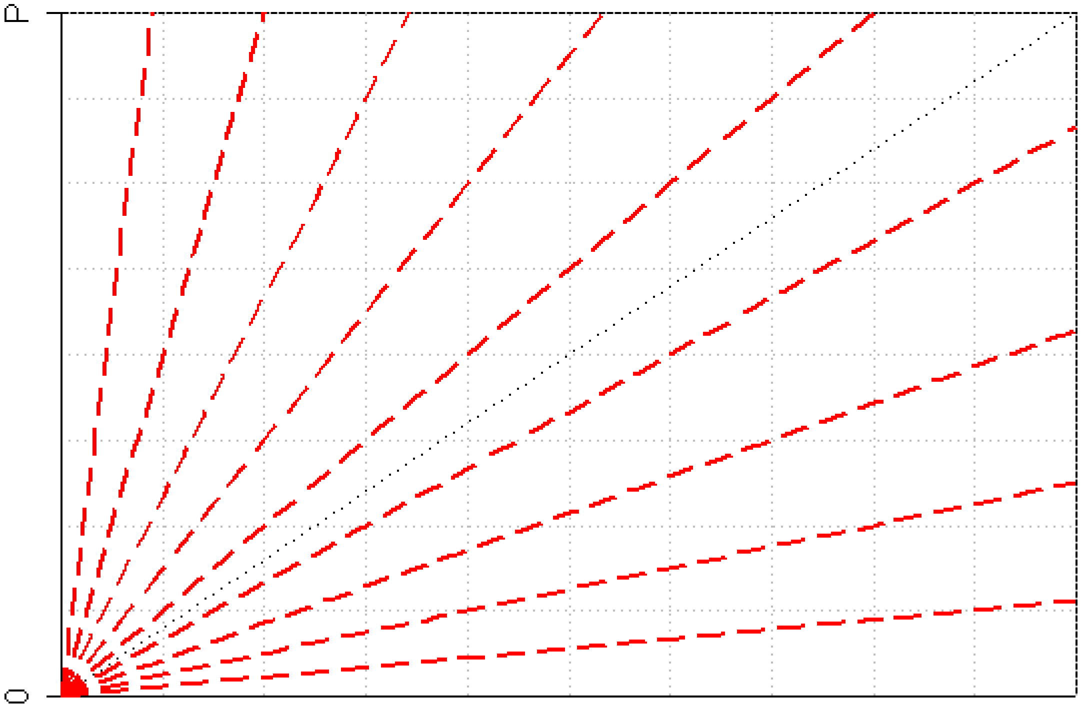
- Isometrics are lines that connect points for which a function in p and n has equal values
  - *Examples*: Isometrics for heuristics $h_p = p$ and $h_n = -n$



Precision (Confidence)

$$h_{Prec} = \frac{p}{p+n}$$

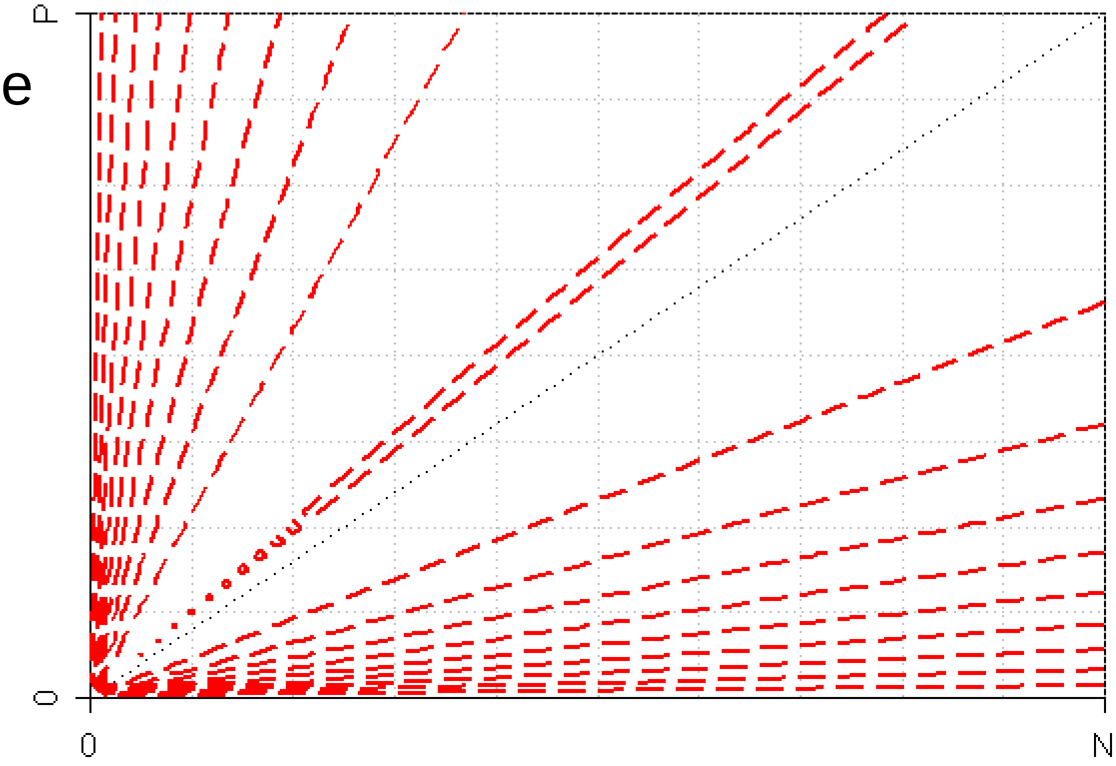
- *basic idea:*
  percentage of positive examples among covered examples
- *effects:*
  - rotation around origin (0,0)
  - all rules with same angle equivalent
  - in particular, all rules on P/N axes are equivalent



Entropy and Gini Index

- $$h_{Ent} = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n}\right)$$
- $$h_{Gini} = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2 \simeq \frac{pn}{(p+n)^2}$$
- *effects:*
  - entropy and Gini index are equivalent
  - like precision, isometrics rotate around (0,0)
  - isometrics are symmetric around 45º line
  - a rule that only covers negative examples is as good as a rule that only covers positives

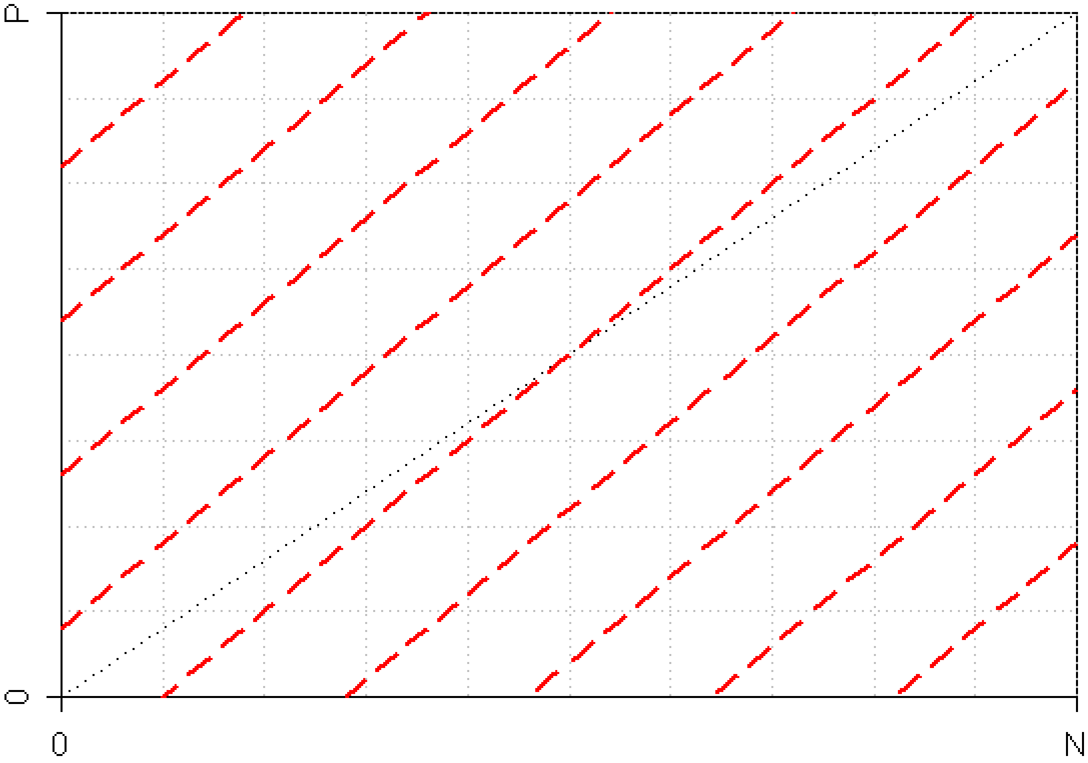
These will be explained later (decision trees)



Accuracy

$$h_{Acc} = \frac{p + (N - n)}{P + N} \simeq p - n$$

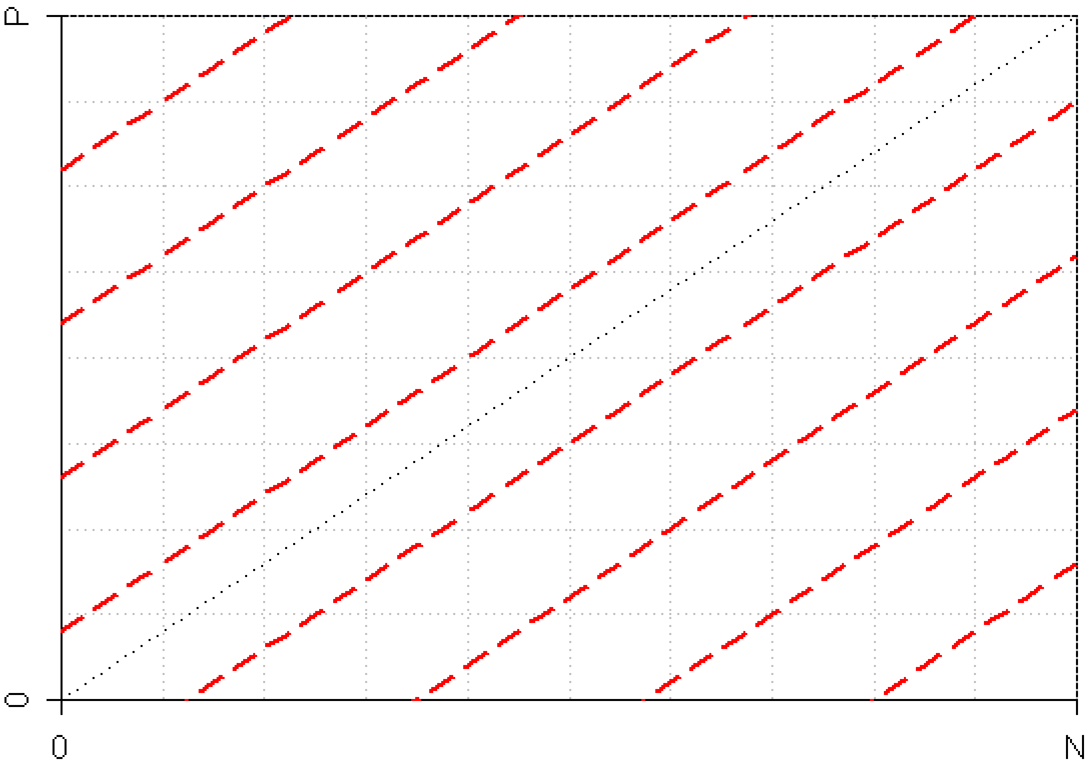
- *basic idea:*
  percentage of correct classifications
  (*covered positives* plus *uncovered negatives*)
- *effects:*
  - isometrics are parallel to 45^o line
  - covering one positive example is as good as not covering one negative example



Weighted Relative Accuracy

$$h_{Acc} = \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right) \simeq \frac{p}{P} - \frac{n}{N}$$

- *basic idea:*
  normalize accuracy with the class distribution
- *effects:*
  - isometrics are parallel to diagonal
  - covering $x\%$ of the positive examples is as good as not covering $x\%$ of the negative examples



Linear Cost Metric

- Accuracy and weighted relative accuracy are only two special cases of the general case with linear costs:
  - costs c mean that covering 1 positive example is as good as not covering $c/(1-c)$ negative examples

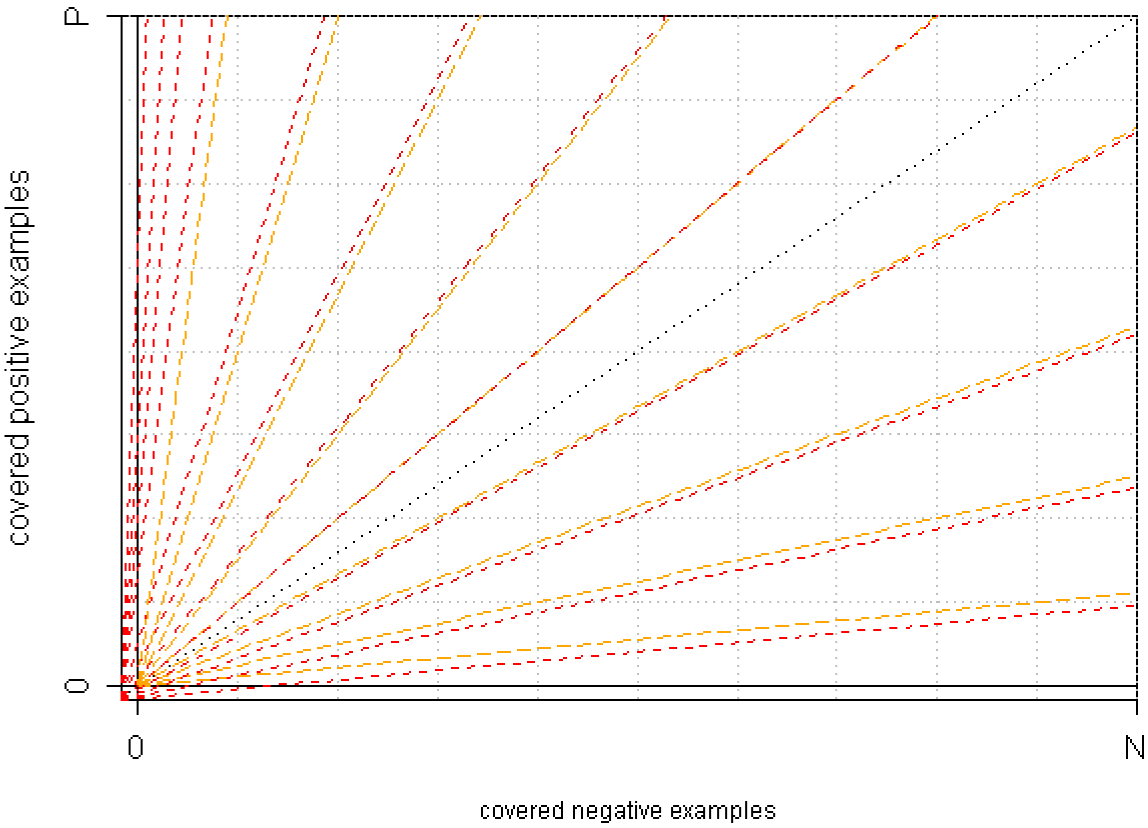
c	*measure*
½	accuracy
$N/(P+N)$	weighted relative accuracy
0	excluding negatives at all costs
1	covering positives at all costs

- The general form is then $h_{cost} = cp - (1-c)n$
  - the isometrics of h_{cost} are parallel lines with slope $(1-c)/c$

Laplace-Estimate

- $$h_{Lap}=\frac{p+1}{(p+1)+(n+1)}=\frac{p+1}{p+n+2}$$

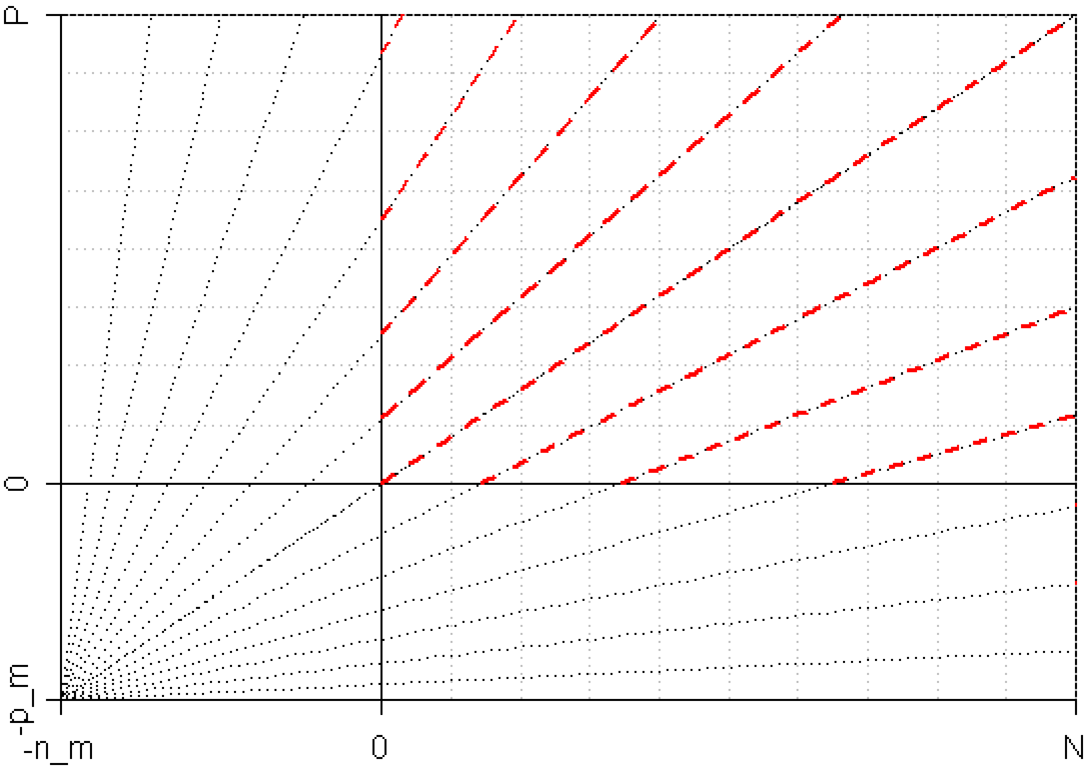
- *basic idea:*
  precision, but count coverage for positive and negative examples starting with 1 instead of 0
- *effects:*
  - origin at $(-1,-1)$
  - different values on $p=0$ or $n=0$ axes
  - not equivalent to precision



m-Estimate

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

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

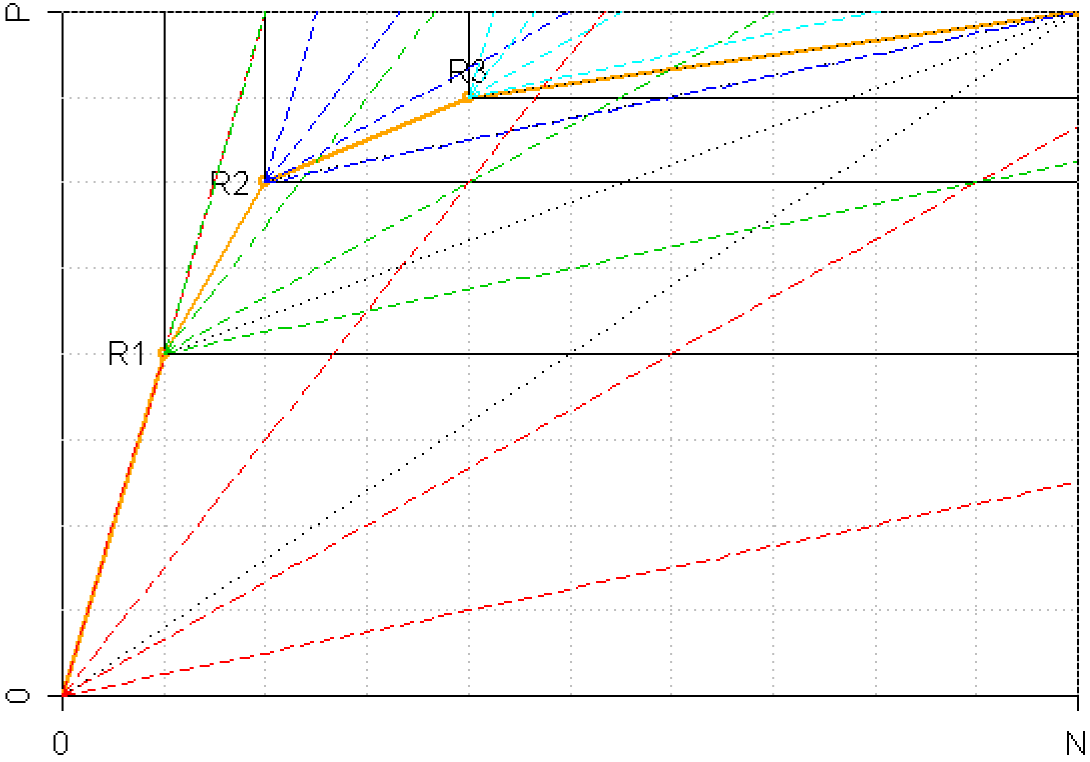


Generalized m-Estimate

- One can re-interpret the m-Estimate:
  - Re-interpret $c = N/(P+N)$ as a cost factor like in the general cost metric
  - Re-interpret m as a trade-off between precision and cost-metric
    - $m = 0$: precision (independent of cost factor)
    - $m \rightarrow \infty$: the isometrics converge towards the parallel isometrics of the cost metric
- Thus, the generalized m-Estimate may be viewed as a means of trading off between precision and the cost metric

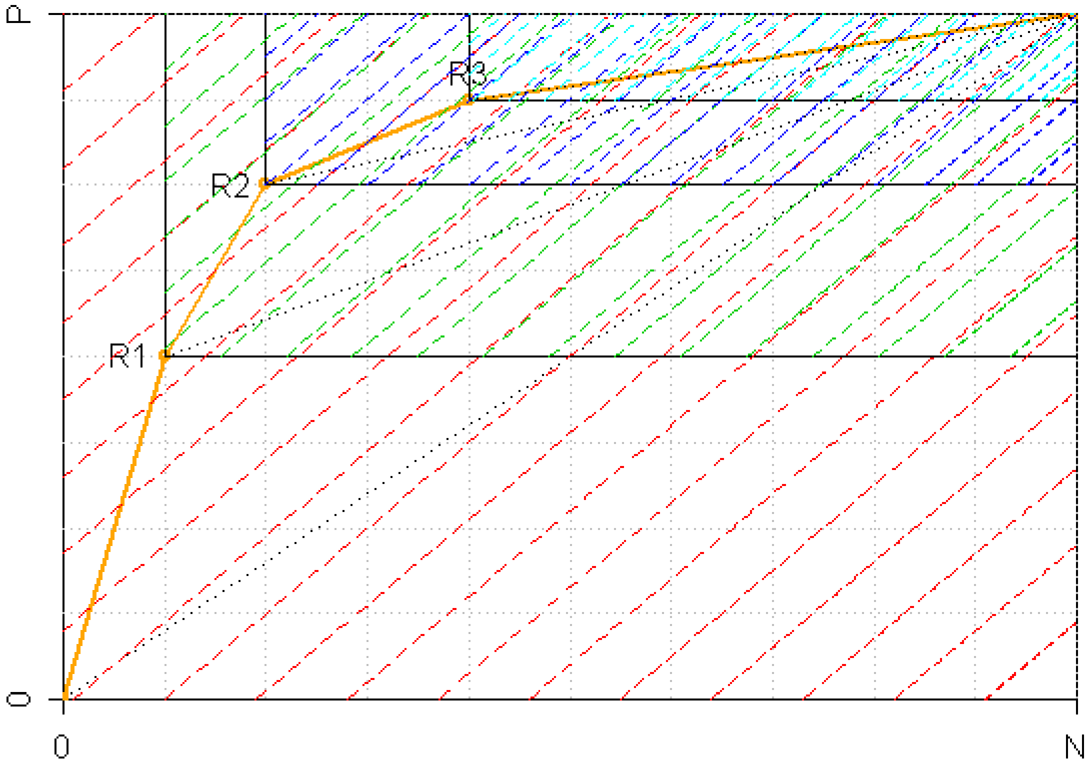
Optimizing Precision

- Precision tries to pick the steepest continuation of the curve
  - tries to maximize the area under this curve
    (→ AUC: Area Under the ROC Curve)
  - no particular angle of isometrics is preferred, i.e. no preference for a certain cost model



Optimizing Accuracy

- Accuracy assumes the same costs in all subspaces
  - a local optimum in a sub-space is also a global optimum in the entire space



Summary of Rule Learning Heuristics

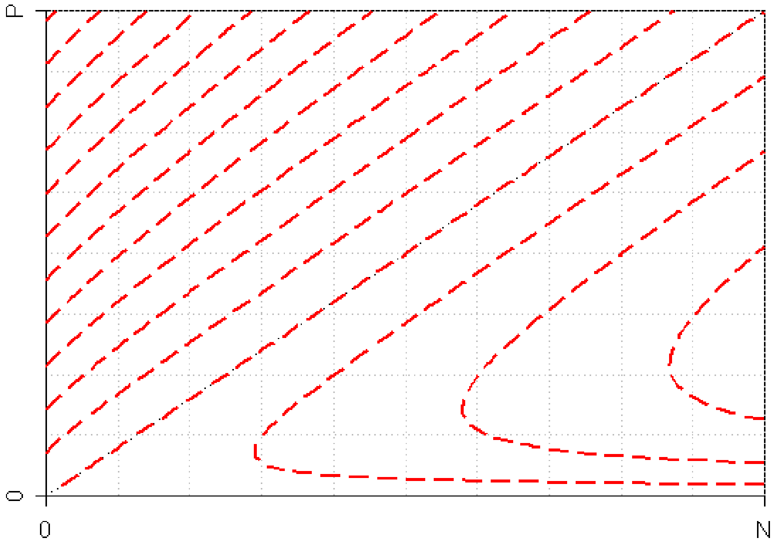
- There are two basic types of (linear) heuristics.
  - *precision*: rotation around the origin
  - *cost metrics*: parallel lines
- They have different goals
  - *precision* picks the steepest continuation for the curve for unkown costs
  - *linear cost metrics* pick the best point according to known or assumed costs

- The m-heuristic may be interpreted as a trade-off between the two prototypes
  - parameter c chooses the *cost model*
  - parameter m chooses the *"degree of parallelism"*

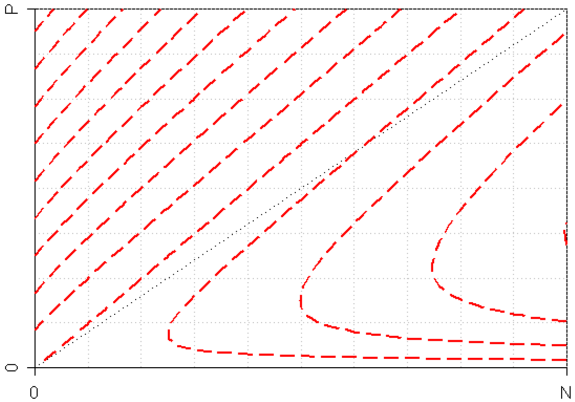
Foil Gain

$$h_{foil} = -p\left(\log_2 c - \log_2 \frac{p}{p+n}\right)$$

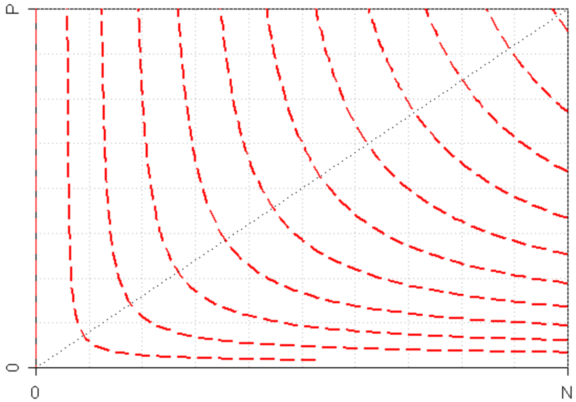
(c is the precision of the parent clause)



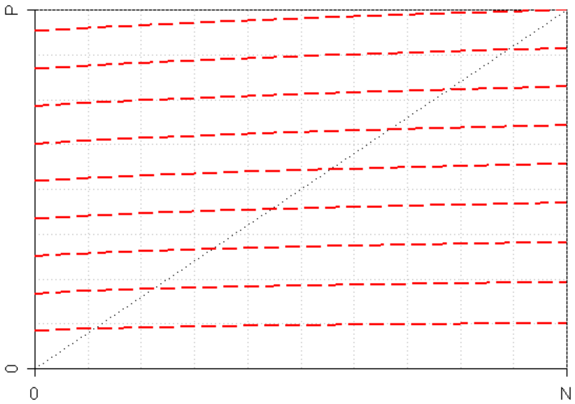
c = P/(P+N)



c = 1/2



c = 1



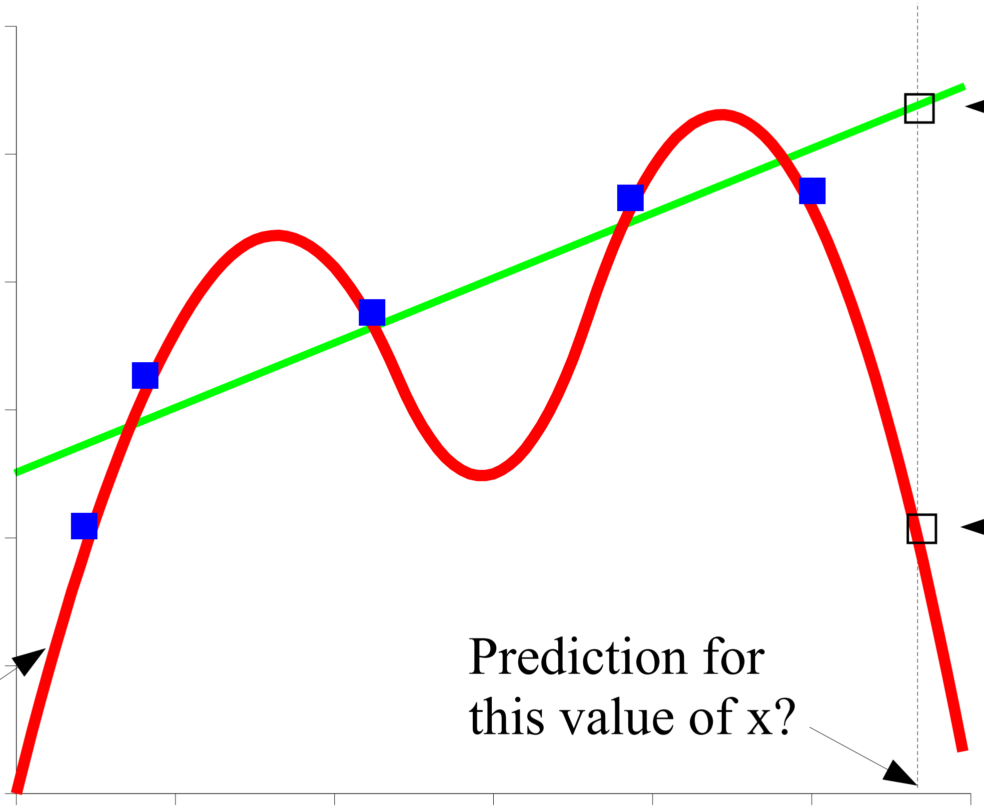
c = 10^-6

Overfitting

- Overfitting
  - Given
    - a fairly general model class
    - enough degrees of freedom
  - you can always find a model that explains the data
    - even if the data contains error (noise in the data)
    - in rule learning: each example is a rule

- Such concepts do not generalize well!
  - → Pruning

Overfitting - Illustration



or here ?

here

Prediction for
this value of x?

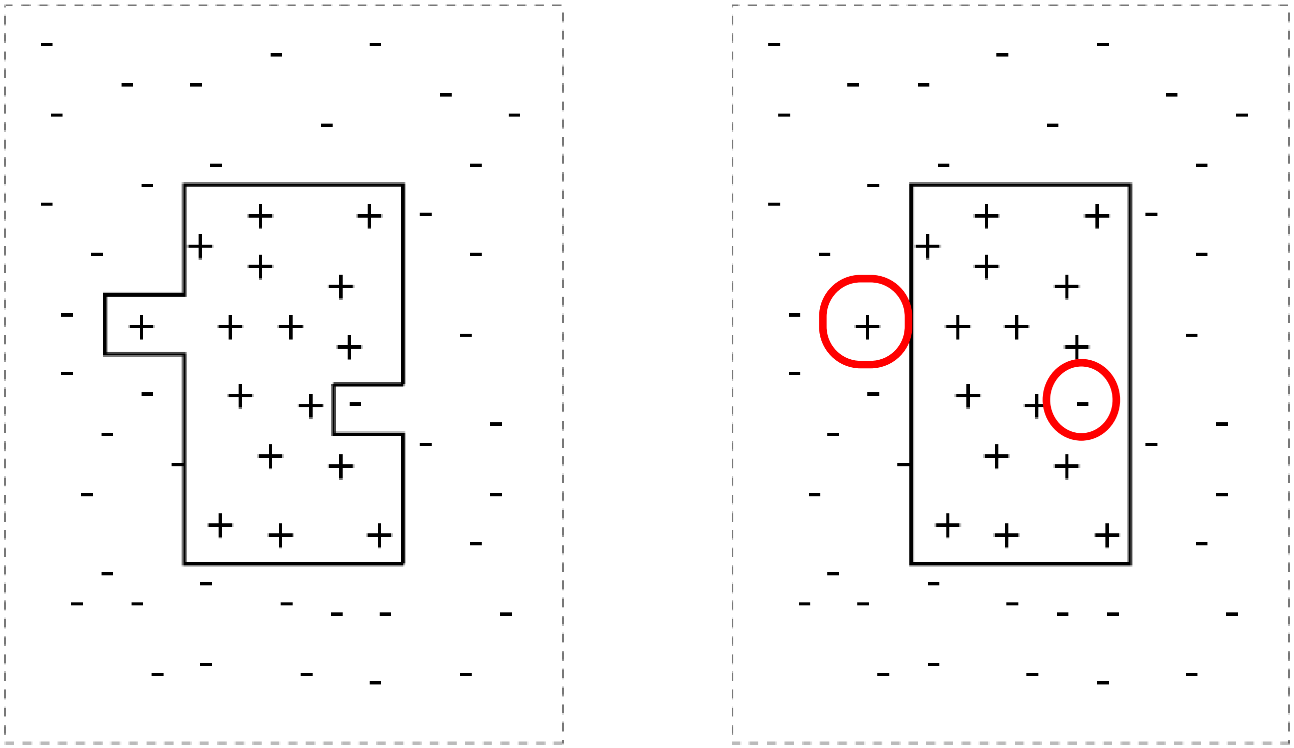
Polynomial degree 1
(linear function)

Polynomial degree 4
(n-1 degrees can always fit n points)

Overfitting

- Eine perfekte Anpassung an die gegebenen Daten ist nicht immer sinnvoll
  - Daten könnten fehlerhaft sein
    - z.B. zufälliges Rauschen (Noise)
  - Die Klasse der gewählten Funktionen könnte nicht geeignet sein
    - eine perfekte Anpassung an die Trainingsdaten ist oft gar nicht möglich
- Daher ist es oft günstig, die Daten nur ungefähr anzupassen
  - bei Kurven:
    - nicht alle Punkte müssen auf der Kurve liegen
  - beim Konzept-Lernen:
    - nicht alle positiven Beispiele müssen von der Theorie abgedeckt werden
    - einige negativen Beispiele dürfen von der Theorie abgedeckt werden

Overfitting



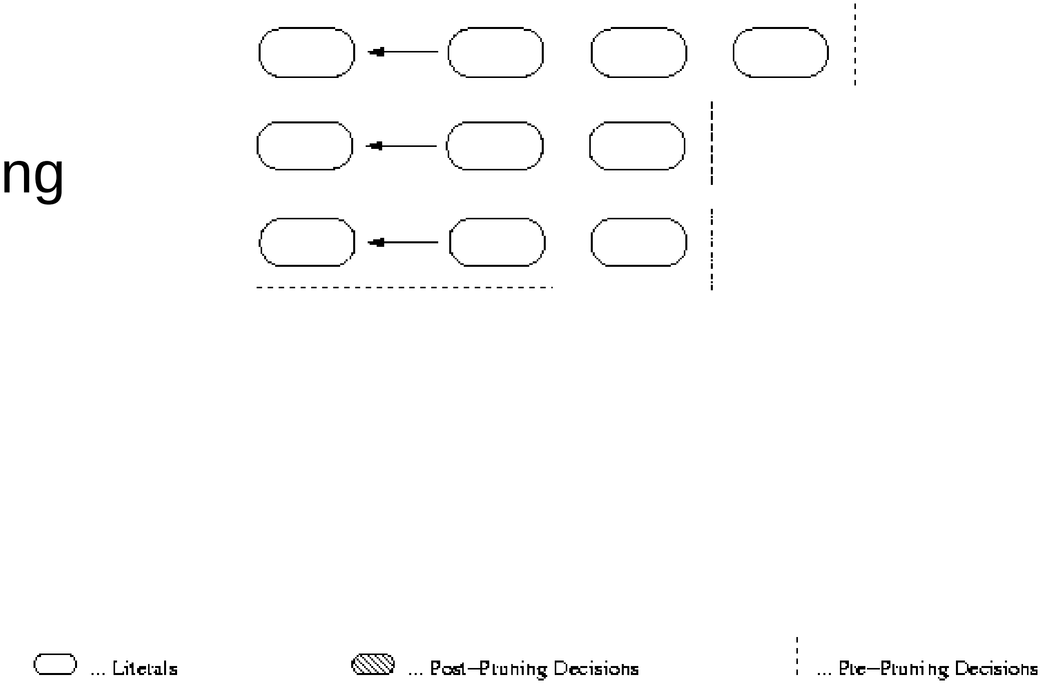
- beim Konzept-Lernen:
  - nicht alle positiven Beispiele müssen von der Theorie abgedeckt werden
  - einige negativen Beispiele dürfen von der Theorie abgedeckt werden

Komplexität von Konzepten

- Je weniger komplex ein Konzept ist, desto geringer ist die Gefahr, daß es sich zu sehr den Daten anpaßt
  - Für ein Polynom n-ten Grades kann man $n+1$ Parameter wählen, um die Funktion an alle Punkte anzupassen
- Daher wird beim Lernen darauf geachtet, die Größe der Konzepte klein zu halten
  - eine kurze Regel, die viele positive Beispiele erklärt (aber eventuell auch einige negative) ist oft besser als eine lange Regel, die nur einige wenige positive Beispiele erklärt.
- Pruning: komplexe Regeln werden zurechtgestutzt
  - **Pre-Pruning:**
    - während des Lernens
  - **Post-Pruning:**
    - nach dem Lernen

Pre-Pruning

- keep a theory simple *while* it is learned
    - decide when to stop adding conditions to a rule (*relax consistency* constraint)
    - decide when to stop adding rules to a theory (*relax completeness* constraint)
  - efficient but not accurate



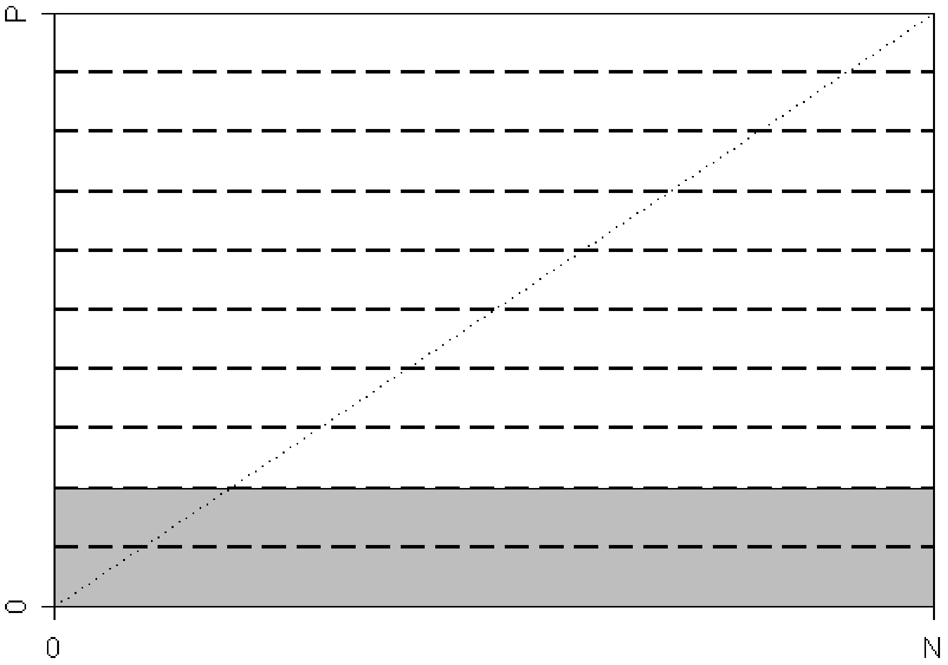
Pre-Pruning Heuristics

- Threshold
  - require a certain minimum value of the search heuristic
  - e.g.: Precision > 0.8.
- Foil's Minimum Description Length Criterion
  - the length of the theory plus the exceptions (misclassified examples) must be shorter than the length of the examples by themselves
  - lengths are measured in bits (information content)
- CN2's Significance Test
  - tests whether the distribution of the examples covered by a rule deviates significantly from the distribution of the examples in the entire training set
  - if not, discard the rule

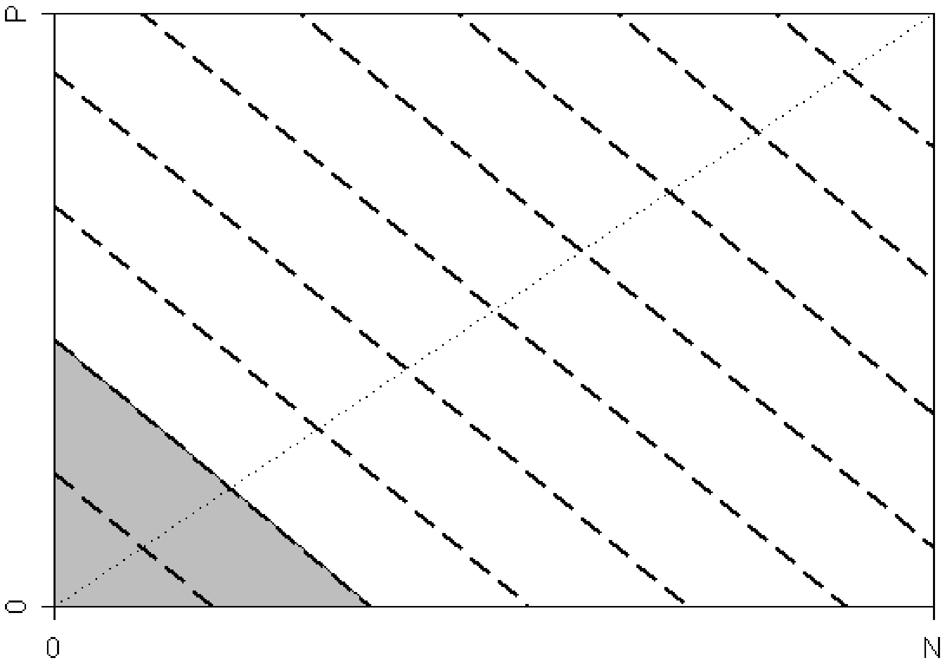
Minimum Coverage Filtering

filter rules that do not cover a minimum number of

positive examples (support)

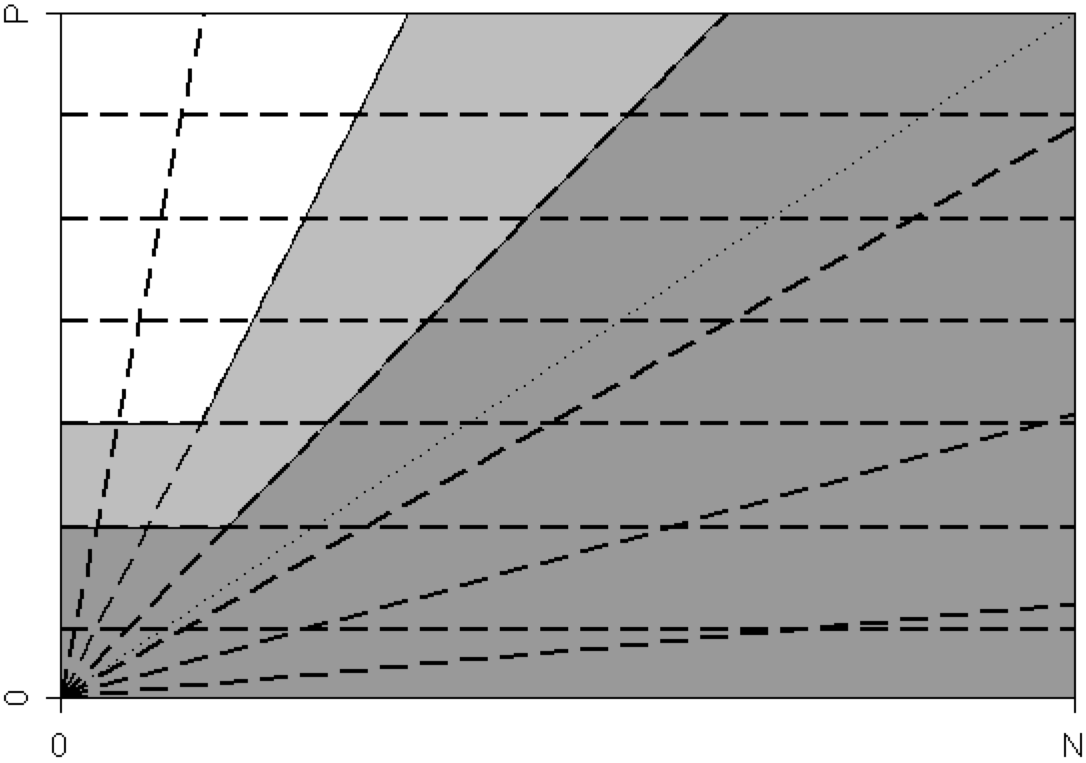


all examples (coverage)



Support/Confidence Filtering

- filter rules that
  - cover not enough positive examples ($p < supp_{min}$)
  - are not precise enough ($h_{prec} < conf_{min}$)
- *effects:*
  - all but a region around (0,P) is filtered



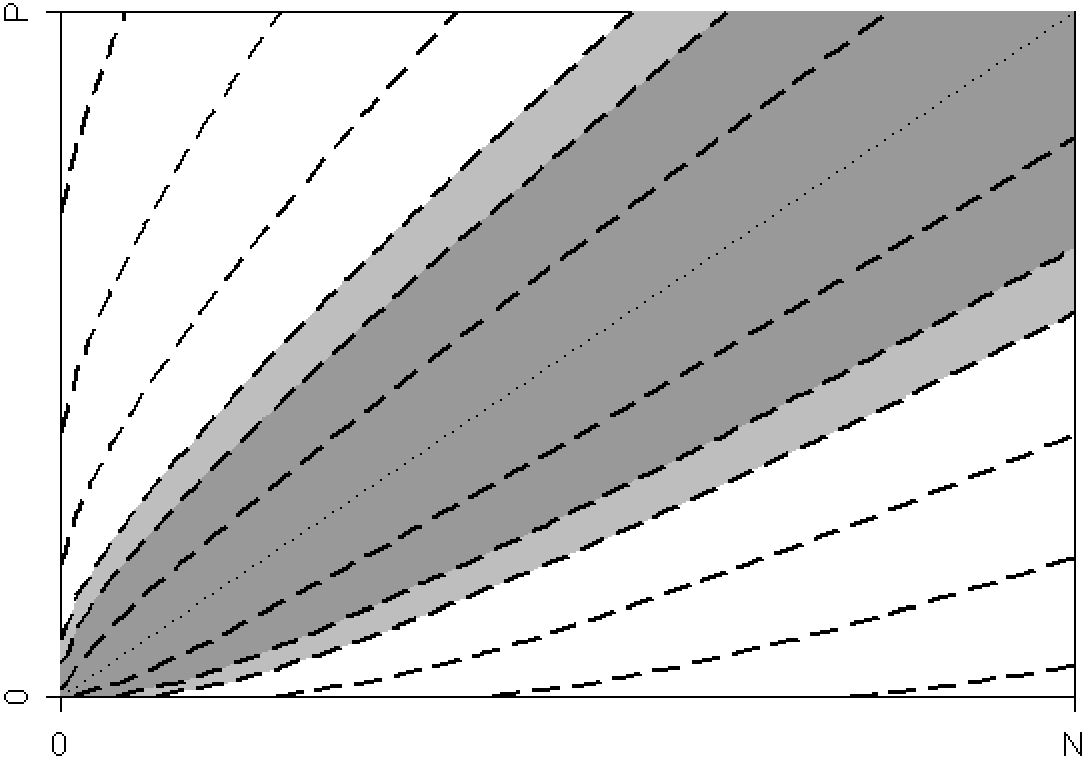
- we will return to support/confidence with association rule learning algorithms!

CN2's likelihood ratio statistics

$$h_{LRS} = 2\left(p \log \frac{p}{e_p} + n \log \frac{n}{e_n}\right)$$

$$e_p = (p+n)\frac{P}{P+N};\ e_n = (p+n)\frac{N}{P+N}$$

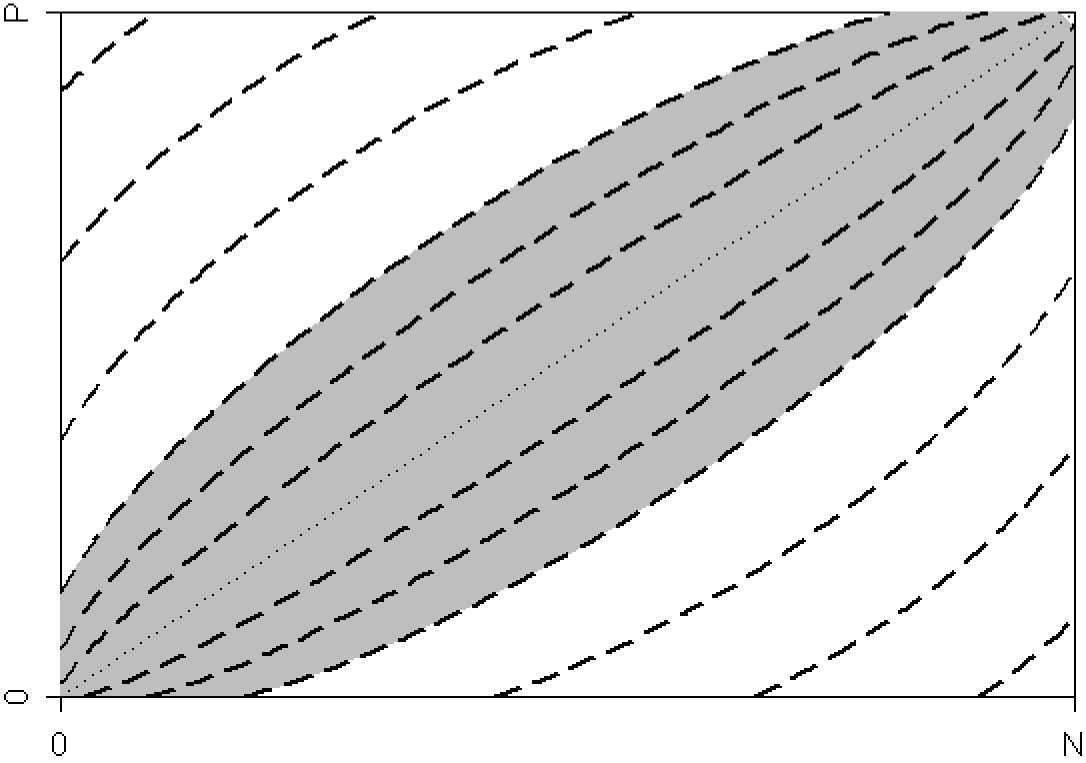
- *basic idea:*
  measure significant deviation from prior probability distribution
- *effects:*
  - non-linear isometrics
    - similar to m-estimate
    - but prefer rules near the edges
  - distributed χ^2
  - significance levels 95% (dark) and 99% (light grey)



Correlation

$$h_{Corr} = \frac{p(N-n)-(P-p)n}{\sqrt{PN(p+n)(P-p+N-n)}}$$

- *basic idea:*
  measure correlation coefficient of predictions with target
- *effects:*
  - non-linear isometrics
  - in comparison to WRA
    - prefers rules near the edges
    - steepness of connection of intersections with edges increases
  - equivalent to χ^2
  - grey area = cutoff of 0.3



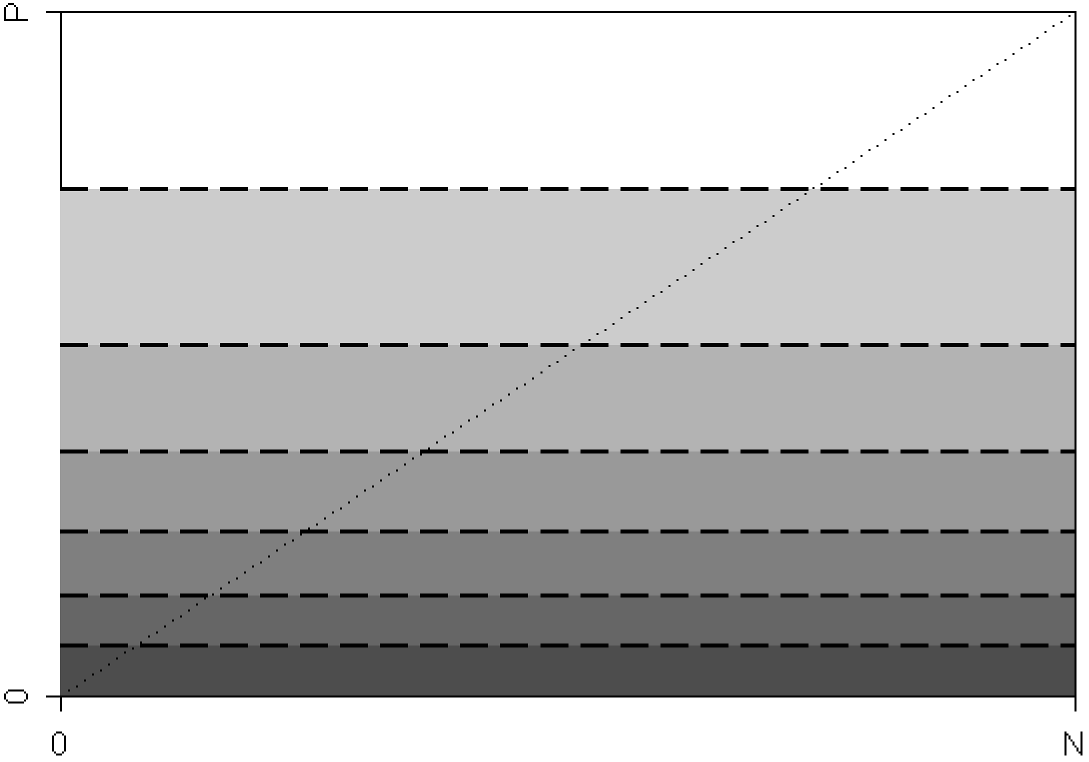
MDL-Pruning in Foil

- Basiert auf dem Minimum Description Length-Prinzip (MDL)
  - ist es effektiver die Regel oder die Beispiele zu übertragen?
    - der Informationsgehalt einer Regel wird berechnet (in Bits)
    - der Informationsgehalt aller Beispiele wird berechnet (in Bits)
    - wenn die Regel mehr Bits braucht als die Beispiele dann wird die Regel nicht weiter verfeinert
  - Details → (Quinlan, 1990)
- Funkioniert nicht perfekt
  - bei nicht perfekten Regeln müßte man noch die Kosten für die Ausnahmen kodieren
    - die müssen zusätzlich zur Regel übertragen werden
  - eine informations-theoretisch perfekte Kodierung einer Regel ist praktisch nicht möglich

Foil's MDL-based Stopping Criterion

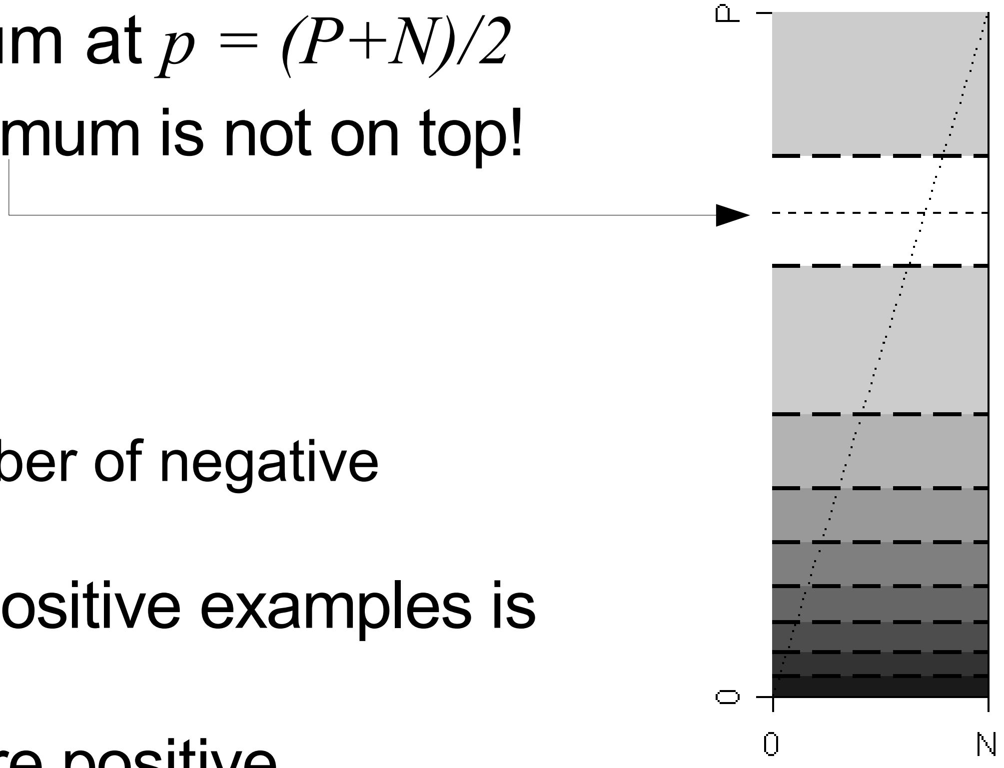
$$h_{MDL} = \log_2(P+N) + \log_2\binom{P+N}{p}$$

- *basic idea:*
  compare the encoding length of the rule $l(r)$ to the encoding length h_{MDL} of the example.
  - we assume $l(r) = c$ constant
- *effects:*
  - equivalent to filtering on support



Anomaly of Foil's Stopping criterion

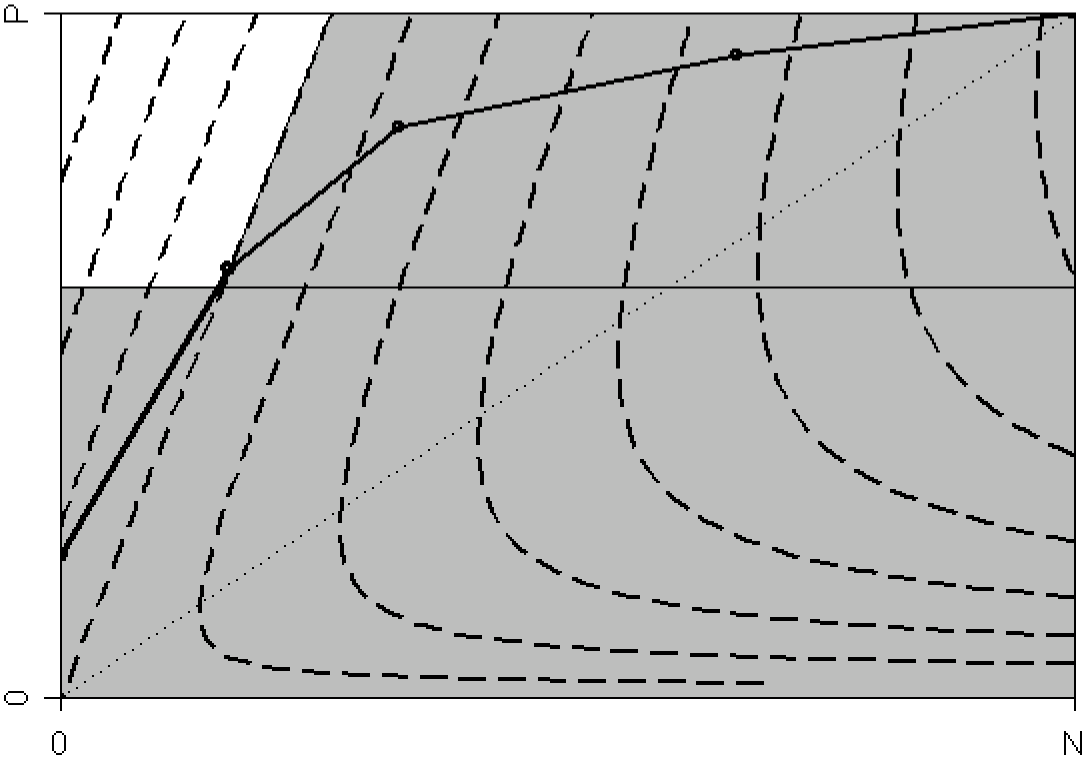
- We have tacitly assumed $N > P$...
- h_{MDL} assumes its maximum at $p = (P+N)/2$
  - thus, for $P > N$, the maximum is not on top!
- there may be rules
  - of equal length
  - covering the same number of negative examples
  - the rule covering fewer positive examples is acceptable
  - but the rule covering more positive examples is not!



How Foil Works

→ Foil (almost) implements Support/Confidence Filtering
(will be explained later → association rules)

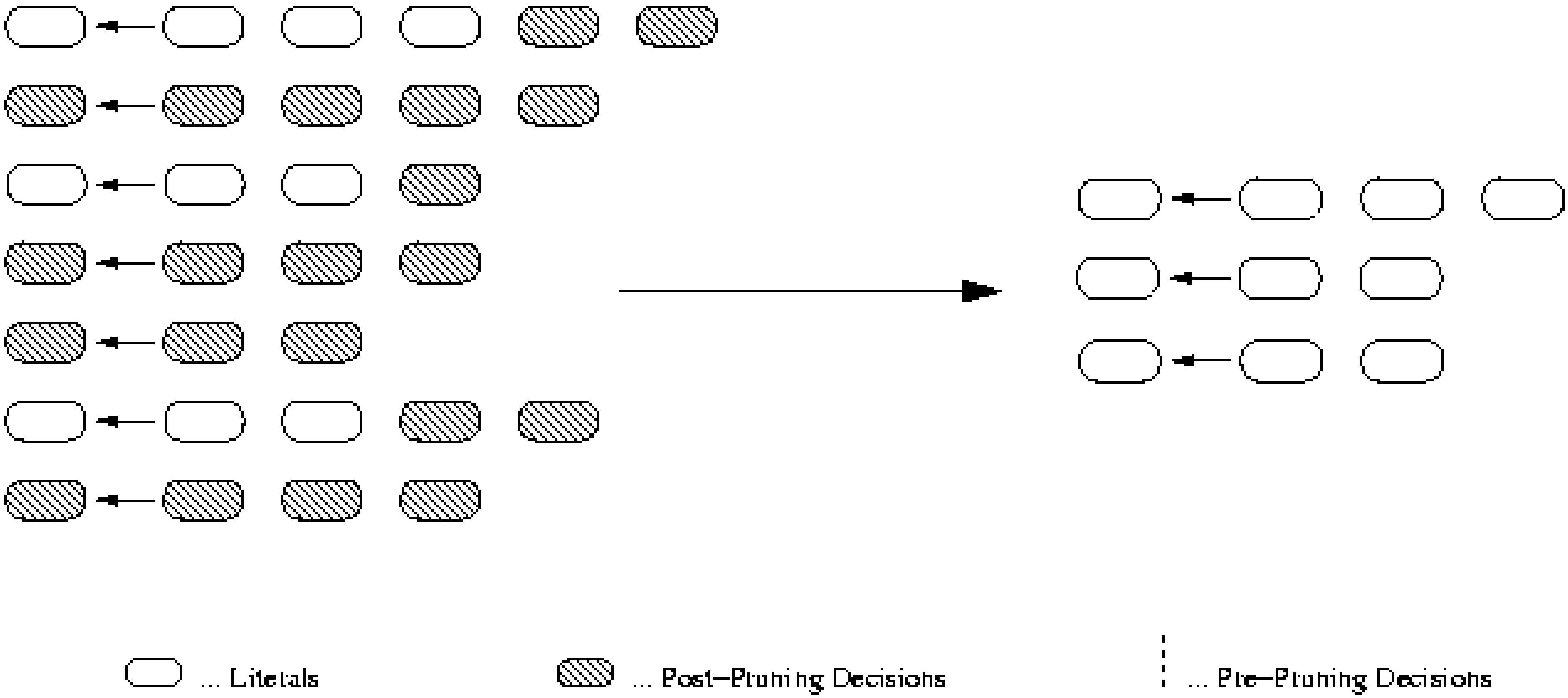
- filtering of rules with no information gain
  - after each refinement ste the region of acceptable rules is adjusted as in precision/confidence filtering
- filtering of rules that exceed the rule length
  - after each refinement ste the region of acceptable rules is adjusted as in support filtering



Pre-Pruning Systems

- Foil:
  - Search heuristic: Foil Gain
  - Pruning: MDL-Based
- CN2:
  - Search heuristic: Laplace/m-heuristic
  - Pruning: Likelihood Ratio
- Fossil:
  - Search heuristic: Correlation
  - Pruning: Threshold

Post Pruning

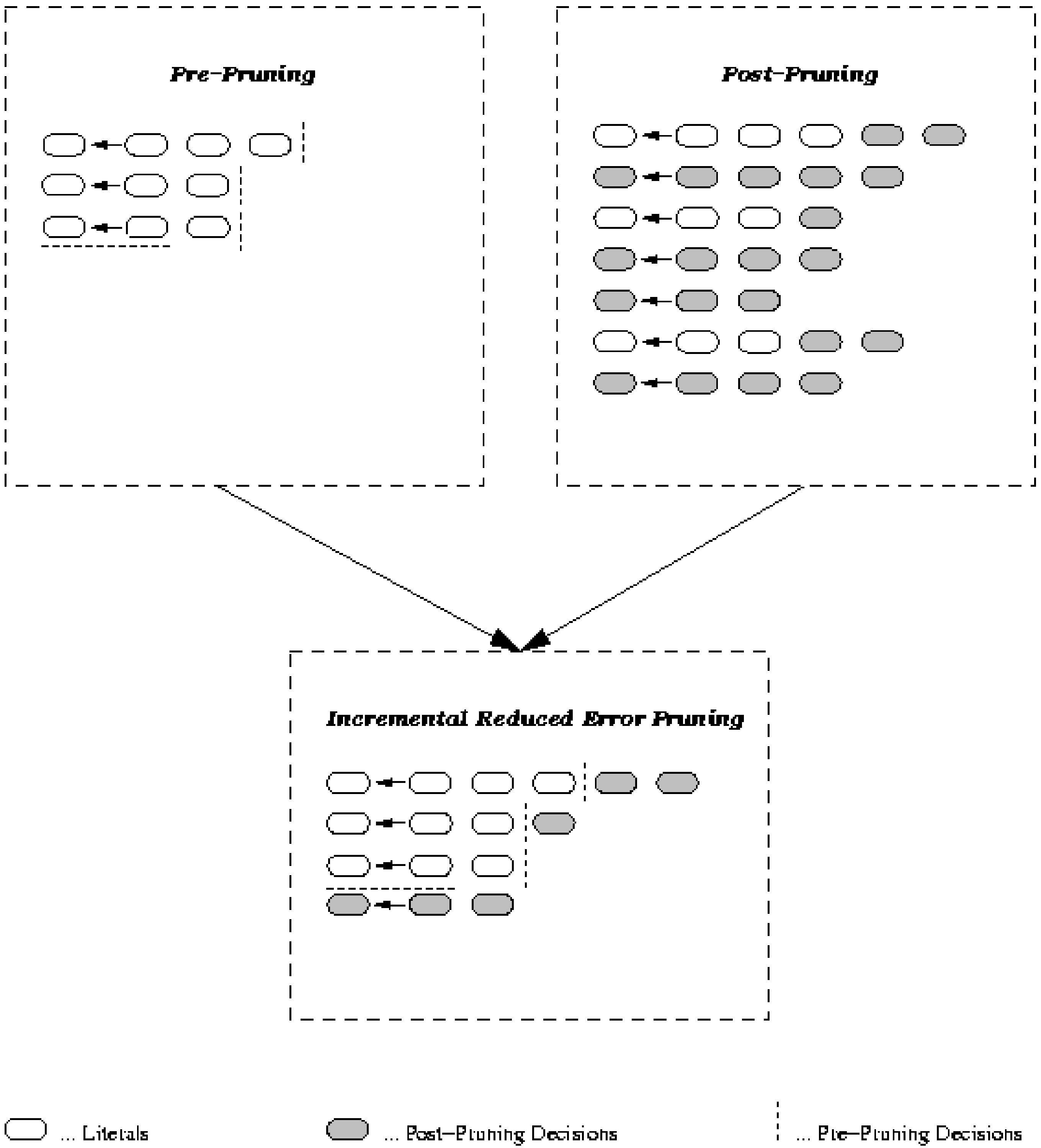


Reduced Error Pruning

- basic idea
  - optimize the accuracy of a rule set on a separate pruning set

1. split training data into a growing and a pruning set
2. learn a complete and consistent rule set covering all positive examples and no negative examples
3. as long as the error on the pruning set does not increase
   - delete condition or rule that results in the largest reduction of error on the pruning set
4. return the remaining rules

- accurate but not efficient
  - $O(n^4)$

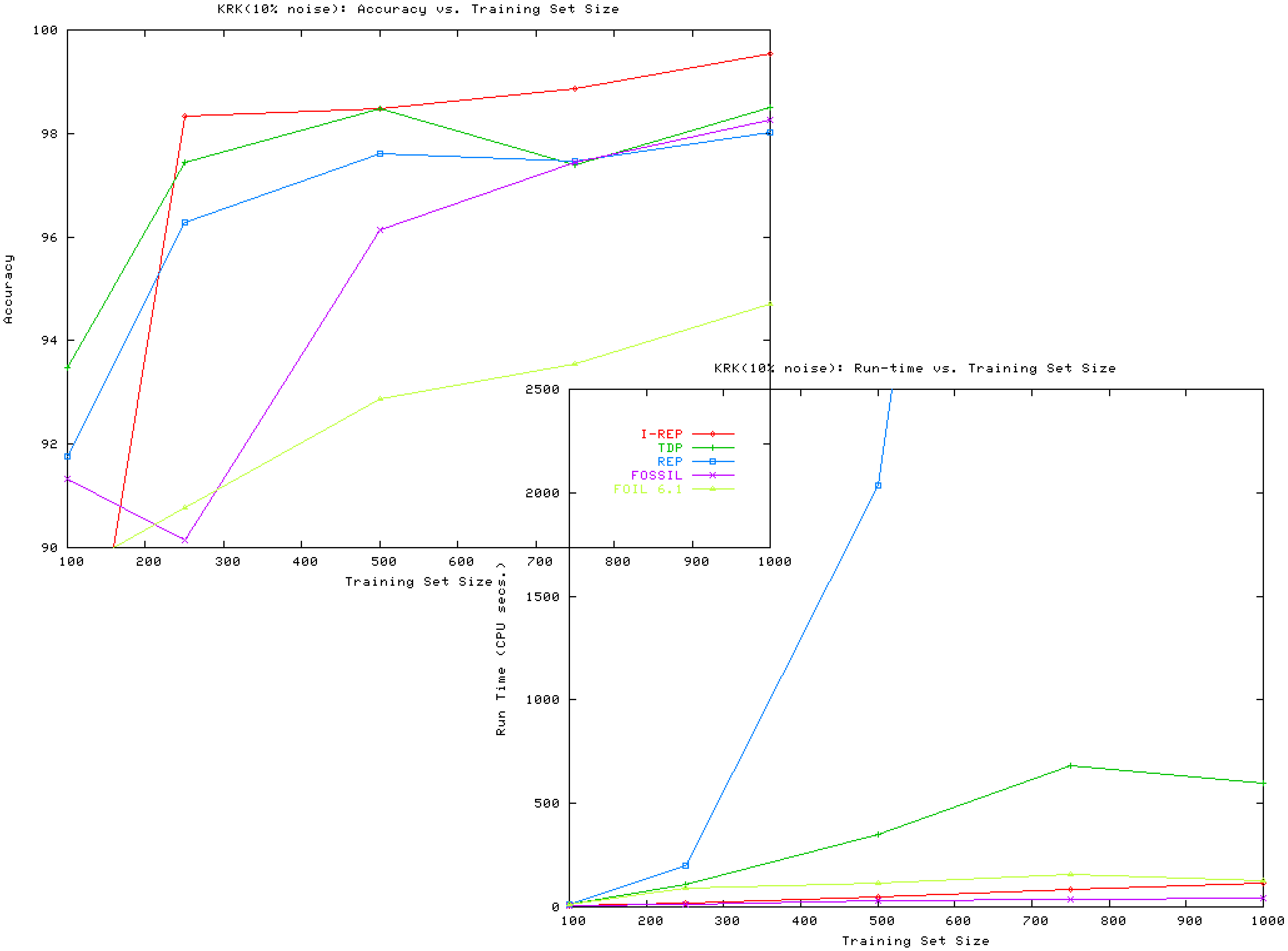

Pre-Pruning
Post-Pruning
Incremental Reduced Error Pruning
... Literals
... Post-Pruning Decisions
... Pre-Pruning Decisions

Incremental Reduced Error Pruning

- Prune each rule right after it is learned:

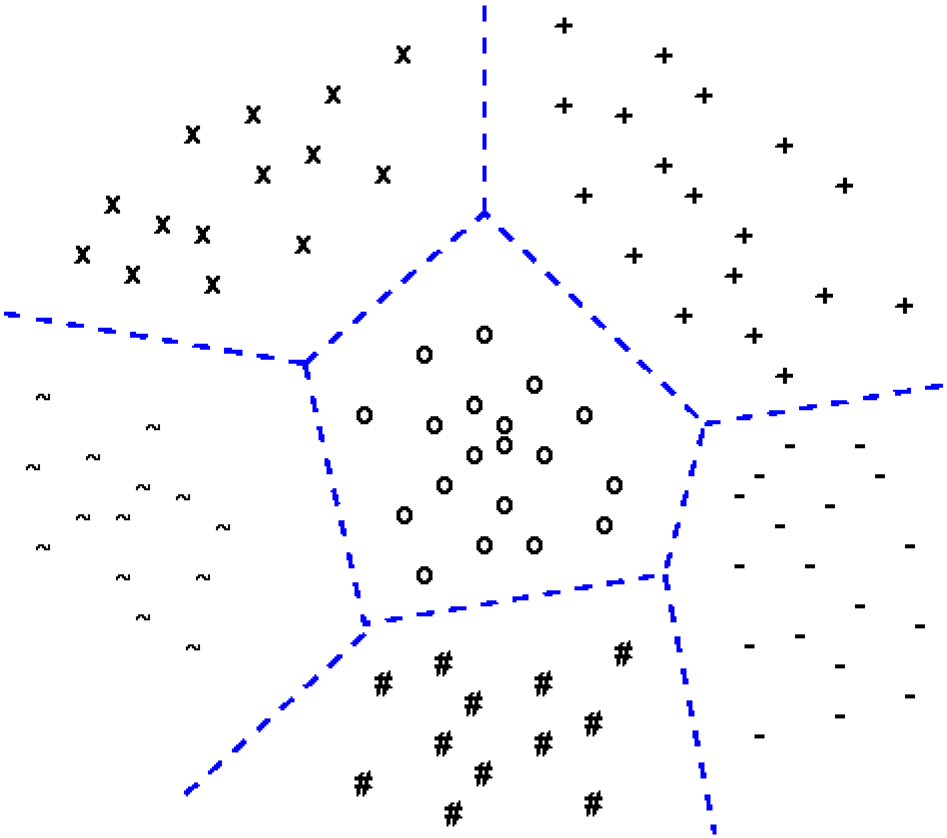
1. split training data into a growing and a pruning set
2. learn a consistent rule covering only positive examples
3. delete conditions as long as the error on the pruning set does not increase
4. if the rule is better than the default rule, add it to the rule set and goto 1.

- More accurate, much more efficient
  - because it does not learn overly complex intermediate concept
  - REP: $O(n^4)$ I-REP: $O(n \log^2 n)$
- Subsequently used in the RIPPER (JRip in Weka) rule learner (Cohen, 1995)


KRK(10% noise): Accuracy vs. Training Set Size
Accuracy
100
98
96
94
92
90
100
200
300
400
500
600
700
800
900
1000
Training Set Size
KRK(10% noise): Run-time vs. Training Set Size
Run Time (CPU secs.)
2500
2000
1500
1000
500
0
100
200
300
400
500
600
700
800
900
1000
Training Set Size
I-REP
TDP
REP
FOSSIL
FOIL 6.1

Multi-class problems

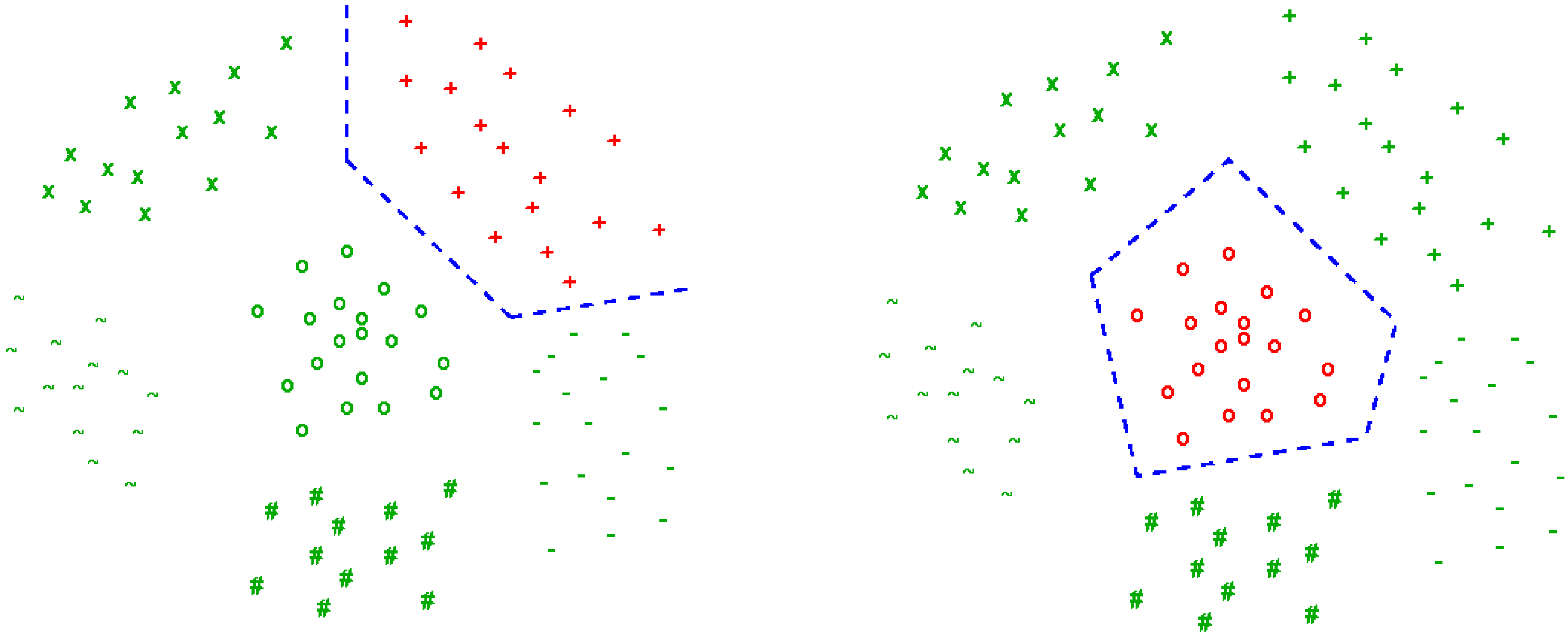
- GOAL: discriminate c classes from each other
- PROBLEM: many learning algorithms are only suitable for binary (2-class) problems
- SOLUTION: *"Class binarization":* Transform an c-class problem into a series of 2-class problems



Class Binarization for Rule Learning

- None
  - class of a rule is defined by the majority of covered examples
  - decision lists, CN2 (Clark & Niblett 1989)
- One-against-all / unordered
  - foreach class c: label its examples positive, all others negative
  - CN2 (Clark & Boswell 1991), Ripper -a unordered
- Ordered
  - sort classes - learn first against rest - remove first - repeat
  - Ripper (Cohen 1995)
- Error Correcting Output Codes (Dietterich & Bakiri, 1995)
  - generalized by (Allwein, Schapire, & Singer, JMLR 2000)

One-against-all binarization



Treat each class as a separate concept:

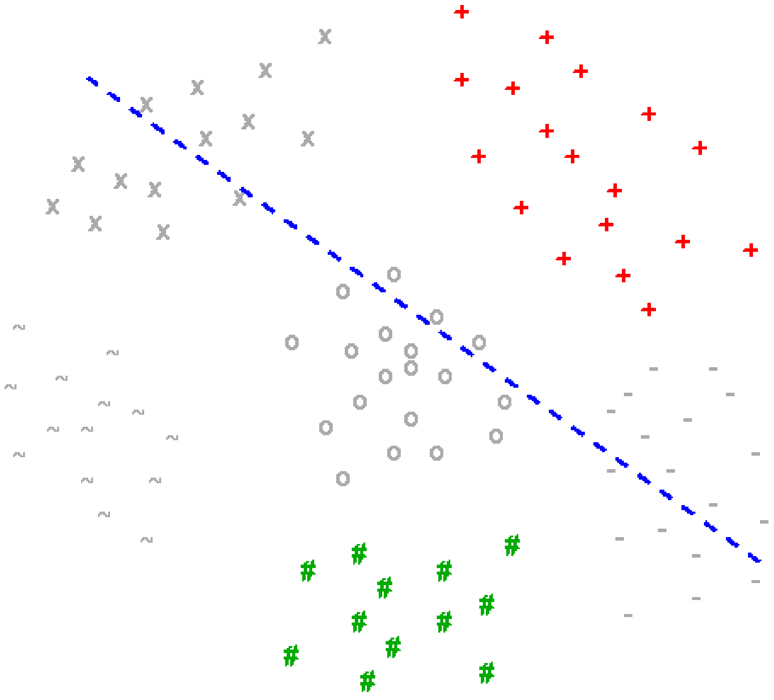
- c binary problems, one for each class
- label examples of one class positive, all others negative

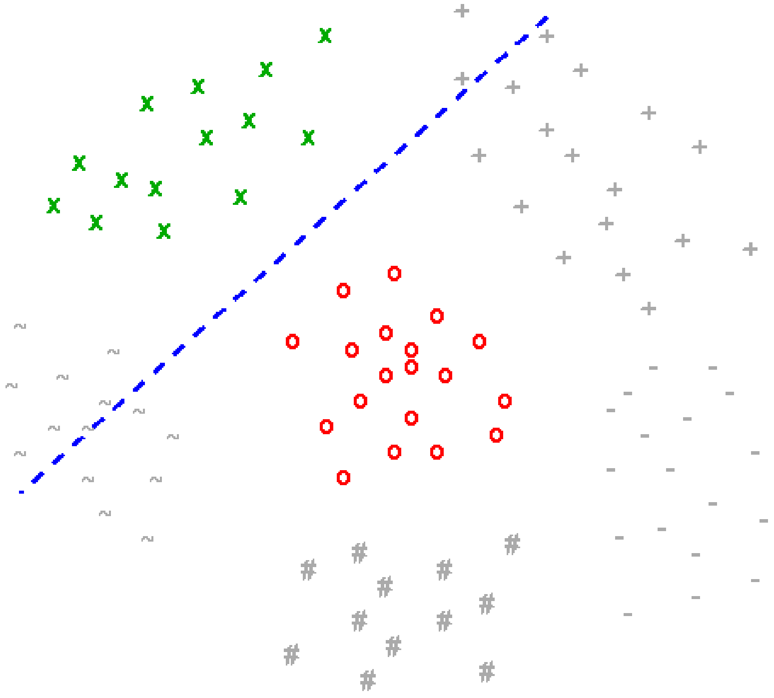
Prediction

- It can happen that multiple rules fire for a class
  - no problem for concept learning (all rules say +)
  - but problematic for multi-class learning
    - because each rule may predict a different class
  - Typical solution:
    - use rule with the highest precision for prediction
  - more complex approaches are possible: e.g., voting
- It can happen that no rule fires on a class
  - no problem for concept learning (the example is then -)
  - but problematic for multi-class learning
    - because it remains unclear which class to select
  - Typical solution: predict the largest class
  - more complex approaches:
    - e.g., rule stretching: find the most similar rule to an example

Round Robin Learning
(aka *Pairwise Classification*)

- $c(c-1)/2$ problems
- each class against each other class



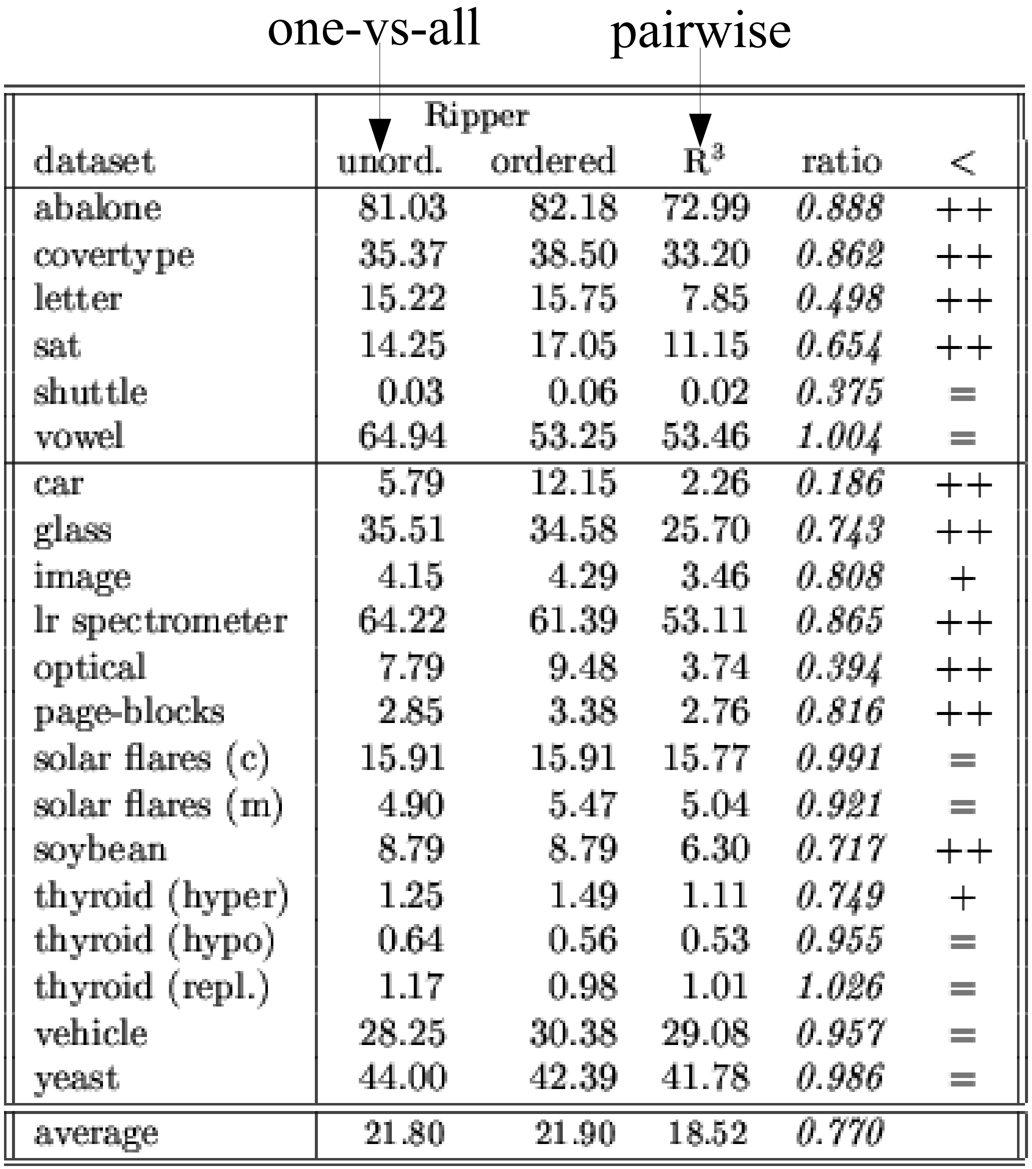


- ✔ smaller training sets
- ✔ simpler decision boundaries
- ✔ larger margins

Prediction

- Voting:
  - as in a sports tournament:
    - each class is a player
    - each player plays each other player, i.e., for each pair of classes we get a prediction which class „wins“
    - the winner receives a point
    - the class with the most points is predicted
      - tie breaks, e.g., in favor of larger classes
- Weighted voting:
  - the vote of each theory is proportional to its own estimate of its correctness
  - e.g., proportional to proportion of examples of the predicted class covered by the rule that makes the prediction

Accuracy



one-vs-all

pairwise

	Ripper				
dataset	unord.	ordered	R^3	ratio	<
abalone	81.03	82.18	72.99	*0.888*	++
covertype	35.37	38.50	33.20	*0.862*	++
letter	15.22	15.75	7.85	*0.498*	++
sat	14.25	17.05	11.15	*0.654*	++
shuttle	0.03	0.06	0.02	*0.375*	=
vowel	64.94	53.25	53.46	*1.004*	=
car	5.79	12.15	2.26	*0.186*	++
glass	35.51	34.58	25.70	*0.743*	++
image	4.15	4.29	3.46	*0.808*	+
lr spectrometer	64.22	61.39	53.11	*0.865*	++
optical	7.79	9.48	3.74	*0.394*	++
page-blocks	2.85	3.38	2.76	*0.816*	++
solar flares (c)	15.91	15.91	15.77	*0.991*	=
solar flares (m)	4.90	5.47	5.04	*0.921*	=
soybean	8.79	8.79	6.30	*0.717*	++
thyroid (hyper)	1.25	1.49	1.11	*0.749*	+
thyroid (hypo)	0.64	0.56	0.53	*0.955*	=
thyroid (repl.)	1.17	0.98	1.01	*1.026*	=
vehicle	28.25	30.38	29.08	*0.957*	=
yeast	44.00	42.39	41.78	*0.986*	=
average	21.80	21.90	18.52	*0.770*	

- error rates on 20 datasets with 4 or more classes
  - 10 significantly better ($p > 0.99$, McNemar)
  - 2 significantly better ($p > 0.95$)
  - 8 equal
  - never (significantly) worse

Yes, but isn't that expensive?

YES:

We have $O(c^2)$ learning problems...

but NO:

the total *training* effort is smaller than for the c learning problems in the one-against-all setting!

- Fine Print :
  - single round robin
    - more rounds add a constant factor
  - training effort only
    - test-time and memory are still quadratic
    - BUT: theories to test may be simpler

Advantages of Round Robin

- Accuracy
  - never lost against one-against-all
  - often significantly more accurate
- Efficiency
  - proven to be faster than, e.g., one-against-all, ECOC, boosting...
  - higher gains for slower base algorithms
- Understandability
  - simpler boundaries/concepts
  - similar to pairwise ranking as recommended by Pyle (1999)
- Example Size Reduction
  - each binary task is considerably smaller than original data
  - subtasks might fit into memory where entire task does not
- Easily parallelizable
  - each task is independent of all other tasks

A Pathology for Top-Down Learning

- Parity problems (e.g. XOR)
  - r relevant binary attributes
  - s irrelevant binary attributes
  - each of the $n = r + s$ attributes has values 0/1 with probability ½
  - an example is positive if the number of 1's in the relevant attributes is even, negative otherwise
- Problem for top-down learning:
  - by construction, each condition of the form $a_i = 0$ or $a_i = 1$ covers approximately 50% positive and 50% negative examples
  - irrespective of whether a_i is a relevant or an irrelevant attribute

  ➔ top-down hill-climbing cannot learn this type of concept
- Typical recommendation:
  - use *bottom-up learning* for such problems

Bottom-Up Approach: Motivation

IF	T=hot	AND	H=high	AND	O=sunny	AND	W=false	THEN	no
IF	T=hot	AND	H=high	AND	O=sunny	AND	W=true	THEN	no
IF	T=hot	AND	H=high	AND	O=overcast	AND	W=false	THEN	yes
IF	T=cool	AND	H=normal	AND	O=rain	AND	W=false	THEN	yes
IF	T=cool	AND	H=normal	AND	O=overcast	AND	W=true	THEN	yes
IF	T=mild	AND	H=high	AND	O=sunny	AND	W=false	THEN	no
IF	T=cool	AND	H=normal	AND	O=sunny	AND	W=false	THEN	yes
IF	T=mild	AND	H=normal	AND	O=rain	AND	W=false	THEN	yes
IF	T=mild	AND	H=normal	AND	O=sunny	AND	W=true	THEN	yes
IF	T=mild	AND	H=high	AND	O=overcast	AND	W=true	THEN	yes
IF	T=hot	AND	H=normal	AND	O=overcast	AND	W=false	THEN	yes
IF	T=mild	AND	H=high	AND	O=rain	AND	W=true	THEN	no
IF	T=cool	AND	H=normal	AND	O=rain	AND	W=true	THEN	no
IF	T=mild	AND	H=high	AND	O=rain	AND	W=false	THEN	yes

Bottom-Up Hill-Climbing

- Simple inversion of top-down hill-climbing
- A rule is successively *generalized*

1. Start with ~~an empty~~ a fully specialized rule R that covers ~~all examples~~ a single example
2. Evaluate all possible ways to ~~add~~ delete a condition to R
3. Choose the best one
4. If R is satisfactory, return it
5. Else goto 2.

A Pathology of Bottom-Up Hill-Climbing

	att1	*att2*	*att3*
+	1	1	1
+	1	0	0
–	0	1	0
–	0	0	1

- Target concept $att1 = 1$ not (reliably) learnable with bottom-up hill-climbing
  - because no generalization of any seed example will increase coverage
  - Hence you either stop or make an arbitrary choice (e.g., delete attribute 1)

Bottom-Up Rule Learning Algorithms

- AQ-type:
  - select a seed example and search the space of its generalizations
  - BUT: search this space top-down
  - Examples: AQ (Michalski 1969), Progol (Muggleton 1995)
- based on least general generalizations (lggs)
  - greedy bottom-up hill-climbing
  - BUT: expensive generalization operator (*lgg/rlgg* of *pairs* of seed examples)
  - Examples: Golem (Muggleton & Feng 1990), DLG (Webb 1992), RISE (Domingos 1995)
- Incremental Pruning of Rules:
  - greedy bottom-up hill-climbing via deleting conditions
  - BUT: start at point previously reached via top-down specialization
  - Examples: I-REP (Fürnkranz & Widmer 1994), Ripper (Cohen 1995)

Rules vs. Trees

- Each decision tree can be converted into a rule set

→ Rule sets are at least as expressive as decision trees

  - a decision tree can be viewed as a set of non-overlapping rules
  - typically learned via *divide-and-conquer* algorithms (recursive partitioning)

- Many concepts have a shorter description as a rule set
  - exceptions: if one or more attributes are relevant for the classification of *all* examples (e.g., parity)
- Learning strategies:
  - Separate-and-Conquer vs. Divide-and-Conquer