Inductive Rule Learning

- Introduction
- Version Spaces
  - Generality Relations
  - Structured Hypothesis Spaces
  - Version Spaces
  - Candidate-Elimination Algorithm
- Separate-and-Conquer Rule Learning
  - Covering algorithm
  - Bottom-Up/Top-Down Learning
  - Rule Evaluation Heuristics
  - Pruning
  - Multi-Class Problems

Rule-based Classifiers

- A classifier basically is a function that computes the output (the *class*) from the input (the *attribute values*)
- Rule learning tries to represent this function in the form of (a set of) IF-THEN rules

  IF (att_i = val_{il}) AND (att_j = val_{jJ}) THEN $class_k$
- Coverage
  - A rule is said to ***cover*** an example if the example satisfies the conditions of the rule.
- Correctness
  - ***completeness***: Each example should be covered by (at least) one rule
  - ***consistency***: For each example, the predicted class should be identical to the true class.

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	falsch	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=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

A Better Solution

IF	Humidity = high	AND	Outlook = sunny	THEN	no
IF	Outlook = rain	AND	Windy = true	THEN	no
ELSE					yes

Rules vs. Trees

- 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)

Generality Relation

- Rule r1 is *more general* than r2
  - if it covers all examples that are covered by r1.
- Rule r1 is *more specific* than r2
  - if r2 is more general than r1.
- Rule r1 is equivalent to r2
  - if it is more specific and more general than r2.

- Examples:

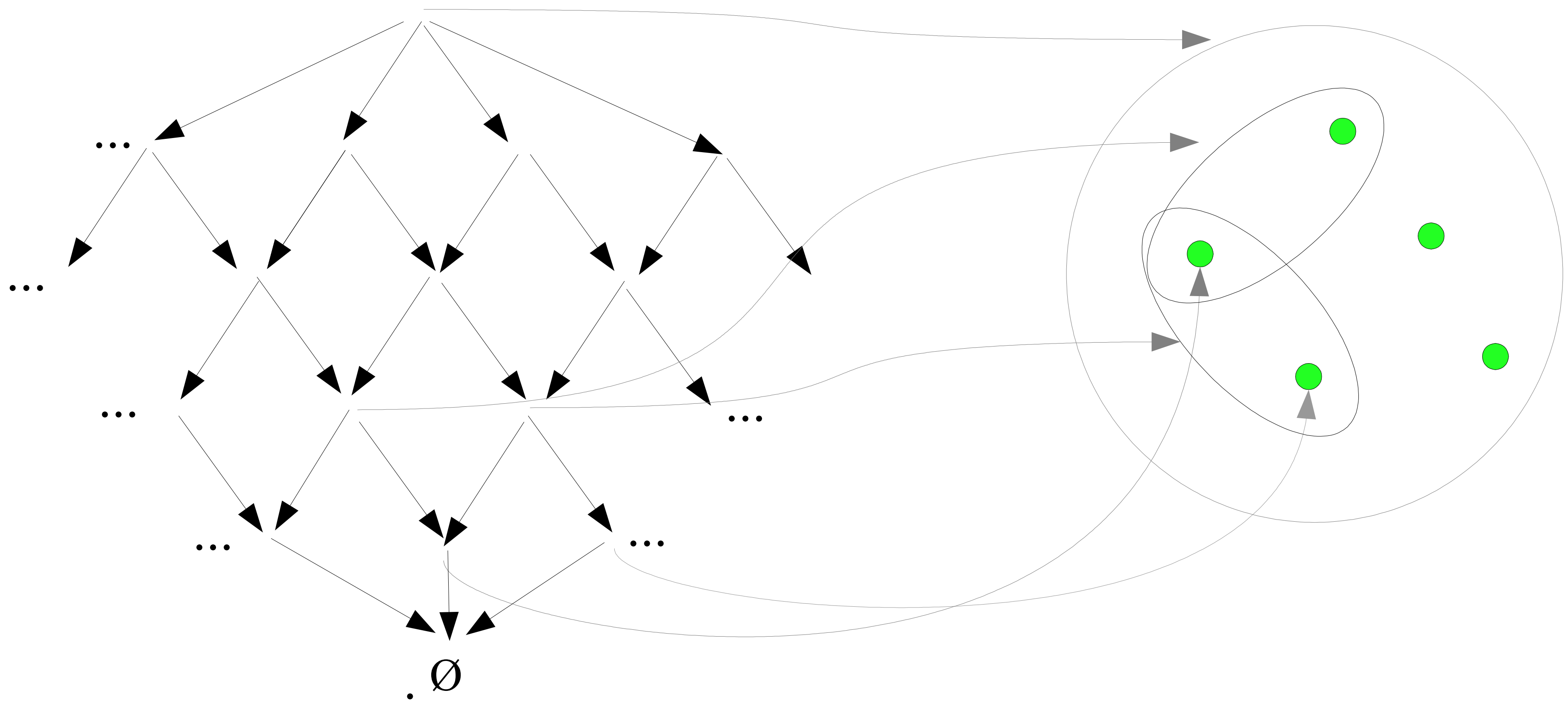
  IF size > 5 THEN +
  IF size > 3 THEN +

  IF outlook = sunny AND windy = false THEN +
  IF outlook = sunny THEN +

  IF animal = mammal THEN +
  IF feeds_children = milk THEN +

Structured Hypothesis Space

- The availability of a generality relation allows to structure the hypothesis space:



Structured Hypothesis Space
arrows represent „is more general than“

Instance Space

Testing for Generality

- In general, we cannot check the generality of theories
  - We do not have all examples of the domain available (and it would be too expensive to generate them)
- For single rules, we can approximate generality via a *syntactic generality check*:
  - Rule r1 is *more general* than r2 if the set of conditions of r1 forms a *subset* of the set of conditions of r2.
  - Why is this only an approximation?
- For the general case, computable generality relations will rarely be available
  - E.g., rule sets
- Structured hypothesis spaces and version spaces are also a theoretical model for learning

Algorithm Find-S

I. h = most specific hypothesis in H
   (covering no examples)

II. for each training example e

   a) if e is negative
      - do nothing

   b) if e is positive
      - for each condition c in h
        - if c does not cover e
          - delete c from h

III. return h

Note: when the first positive examples is encountered, step II.b) reduces to converting the example into a rule

Properties of Find-S

- completeness:
  - h covers all positive examples
- consistency:
  - h will not cover any negative training examples
  - but only if the hypothesis space contains a target concept
    (i.e., there is a single conjunctive rule that describes the target concept)
- Properties:
  - no way of knowing whether it has found the target concept
    (there might be more than one theory that are complete and consistent)
  - it prefers more specific hypothesis
    (it will never generalize unless forced by a training example)
  - it only maintains one specific hypothesis
    (in other hypothesis languages there might be more than one)

Version Space

- The Version Space V is the set of all hypotheses that
  - cover all positive examples (*completeness*)
  - do not cover any negative examples (*consistency*)
- For structured hypothesis spaces there is an efficient representation consisting of
  - the general boundary G
    - all hypotheses in V for which no generalization is in V
  - the specific boundary S
    - all hypotheses in V for which no specialization is in V
- a hypothesis that is neither in G nor in S is
  - a generalization of at least one hypothesis in S
  - a specialization of at least one hypothesis in G

Candidate Elimination Algorithm

- G = set of maximally general hypotheses
  S = set of maximally specific hypotheses
- For each training example e
  - if e is positive
    - For each hypothesis g in G that <u>does not cover</u> e
      - remove g from G
    - For each hypothesis s in S that <u>does not cover</u> e
      - remove s from S
      - S = S U all hypotheses h such that
        - h is a minimal <u>generalization</u> of s
        - h covers e
        - some hypothesis in G is more general than h
      - remove from S any hypothesis that is more general than another hypothesis in S

Candidate Elimination Algorithm (Ctd.)

- if e is negative
  - For each hypothesis s in S that <u>covers</u> e
    - remove s from S
  - For each hypothesis g in G that <u>covers</u> e
    - remove g from G
    - G = G U all hypotheses h such that
      - h is a minimal <u>specialization</u> of g
      - h does not e
      - some hypothesis in S is more specific than h
    - remove from G any hypothesis that is less general than another hypothesis in G

Example

No.	Sky	Temperature	Humidity	Windy	Water	Forecast	sport?
1	sunny	hot	normal	strong	warm	same	yes
2	sunny	hot	high	strong	warm	same	yes
3	rainy	cool	high	strong	warm	change	no
4	sunny	hot	high	strong	cool	change	yes

S_0: { <Ø,Ø,Ø,Ø,Ø,Ø> }
G_0: { <?, ?, ?, ?, ?, ?> }

S_1: { <sunny, hot, normal, strong, warm,same> }
G_1: { <?, ?, ?, ?, ?, ?> }

S_2: { <sunny, hot, ?, strong, warm,same> }
G_2: { <?, ?, ?, ?, ?, ?> }

S_3: { <sunny, hot, ?, strong, warm,same> }
G_3: { <sunny, ?, ?, ?, ?, ? >
<?, hot, ?, ?, ?, ? >
<?, ?, ?, ?, ?, same > }

S_4: { <sunny, hot, ?, strong, ?, ? > }
G_4: { <sunny, ?, ?, ?, ?, ? >
<?, hot, ?, ?, ?, ? >}

How to Classify these Examples?

- Version Space:

G <sunny, ?, ?, ?, ?, ? > <?, hot, ?, ?, ?, ? >

<sunny, ?, ?, strong, ?, ?> <sunny, hot, ?, ?, ?, ?> <?, hot, ?, strong, ?, ?>

S <sunny, hot, ?, strong, ?, ? >

- How to Classify these Examples?

No.	Sky	Temperature	Humidity	Windy	Water	Forecast	sport?
5	sunny	hot	normal	strong	cool	change	yes
6	rainy	cool	normal	light	warm	same	no
7	sunny	hot	normal	light	warm	same	?
8	sunny	cool	normal	strong	warm	same	maybe no

Properties

- Convergence towards target theory
  - If S = G
- Using partially learned concepts
  - an example that matches all elements in S must be a member of the target concept
  - an example that matches no element in G cannot be a member of the target concept
  - examples that match parts of S and G are undecidable
- no need to remember examples (*incremental* learning)
- Assumptions
  - no errors in the training set
  - the hypothesis space contains the target theory
  - practical only if generality relation is (efficiently) computable

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
	$p + n$	$P+N - (p+n)$	$P+N$

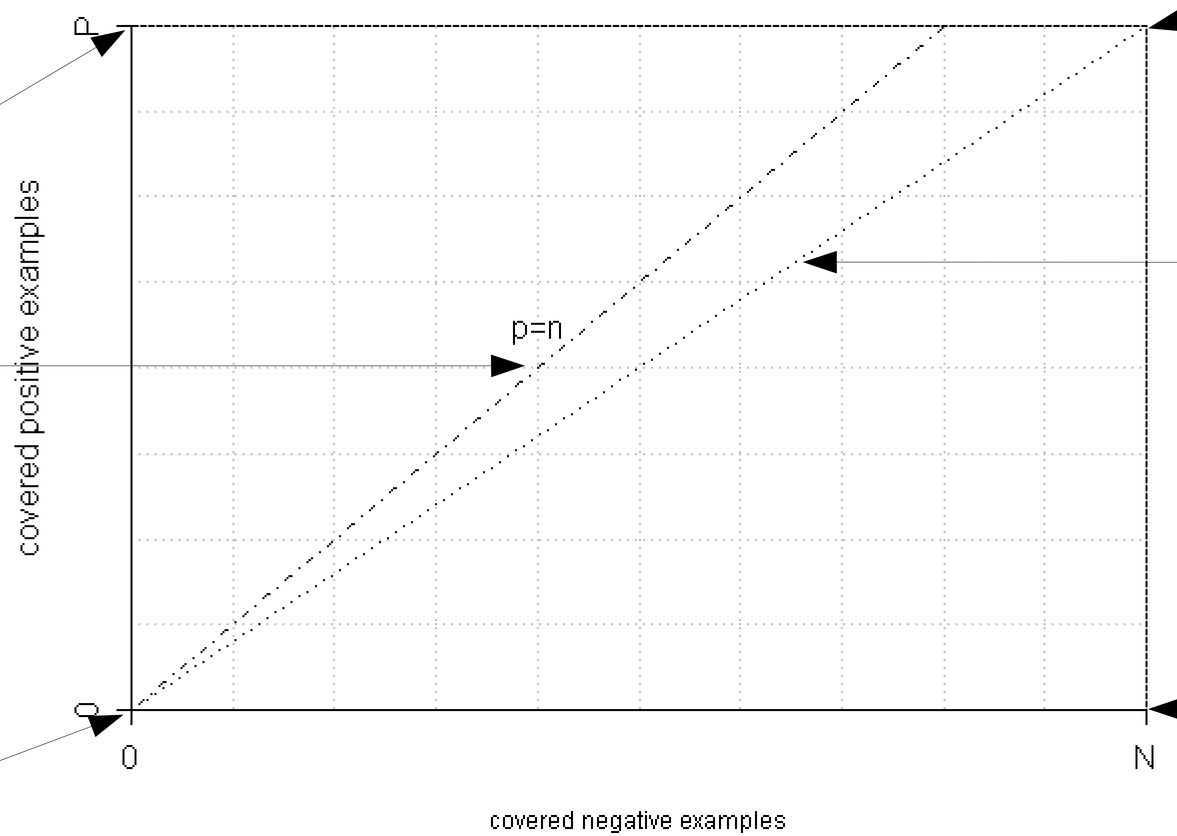
Coverage Spaces

- 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

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

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.

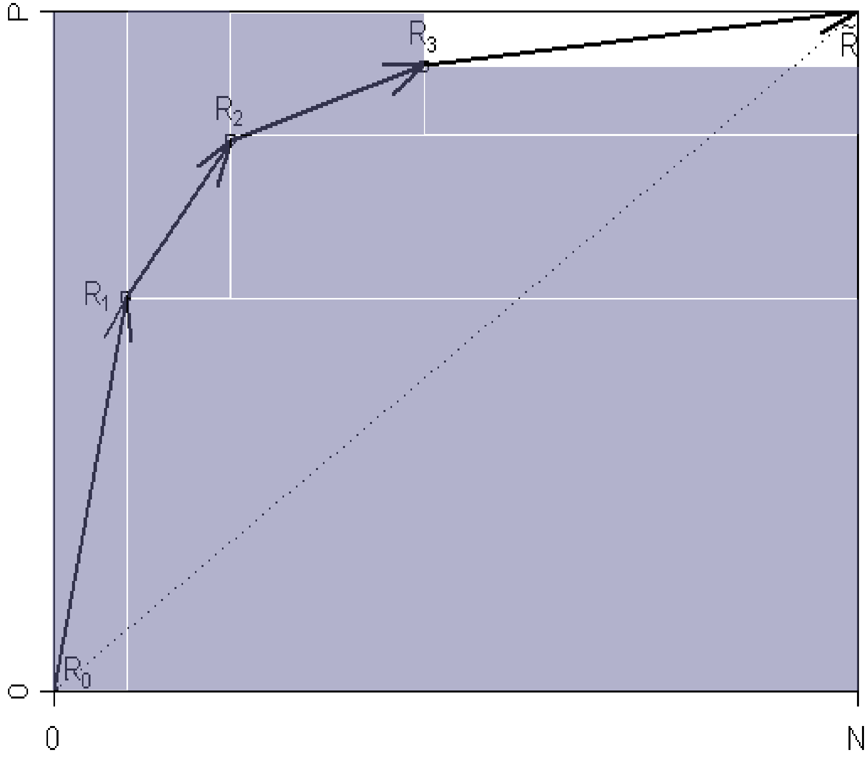
- One of the oldest family of learning algorithms
  - goes back AQ (Michalski, 60s)
  - FRINGE, PRISM and CN2: relation to decision trees (80s)
  - popularized in ILP (FOIL and PROGOL, 90s)
  - RIPPER brought in good noise-handling
- Different learners differ in how they find a single rule

	Language Bias							Search Bias							Overfitting Avoidance		
	Static					Dyn.		Algorithm				Strategy					
Algorithm	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

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*

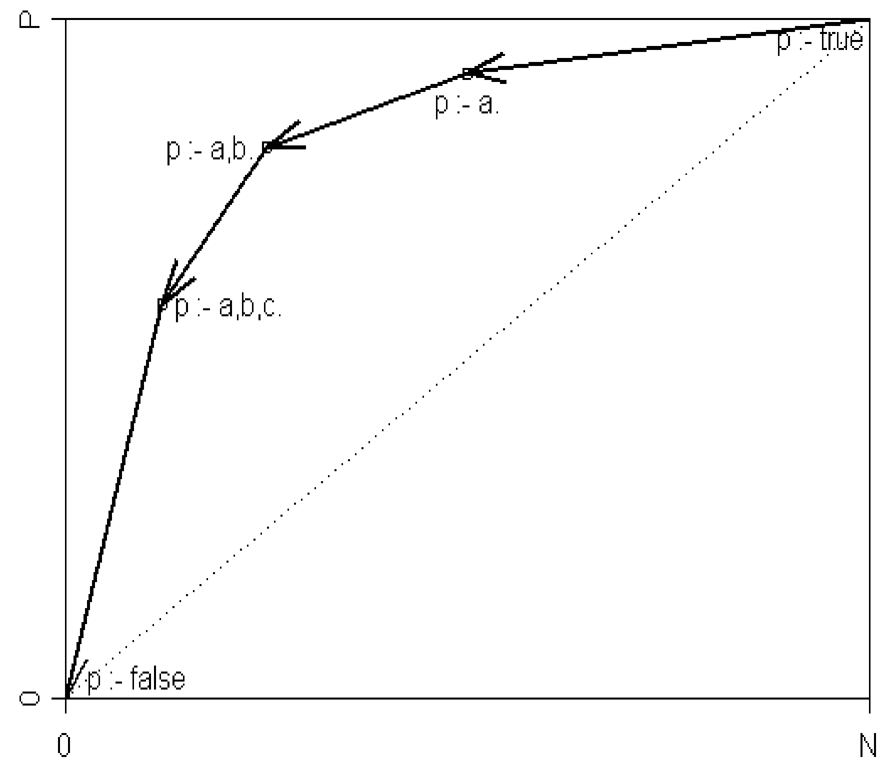
1. Start with an empty rule R that covers all examples
2. Evaluate all possible ways to add a condition to R
3. Choose the best one (according to some heuristic)
4. If R is satisfactory, return it
5. Else goto 2.

- 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 $p \to \infty$
    - grows with $n \to 0$
  - Note: Precision is not a good heuristic. Why?

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

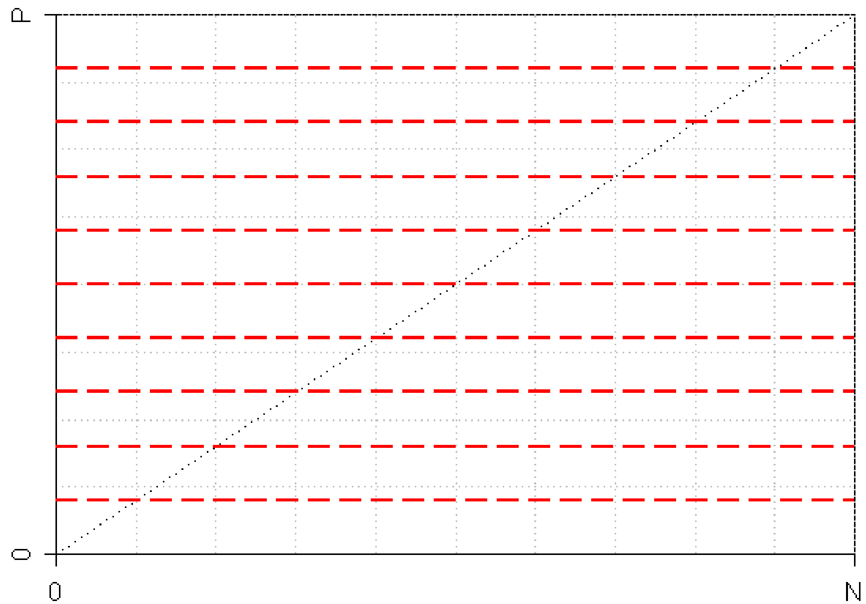
Example

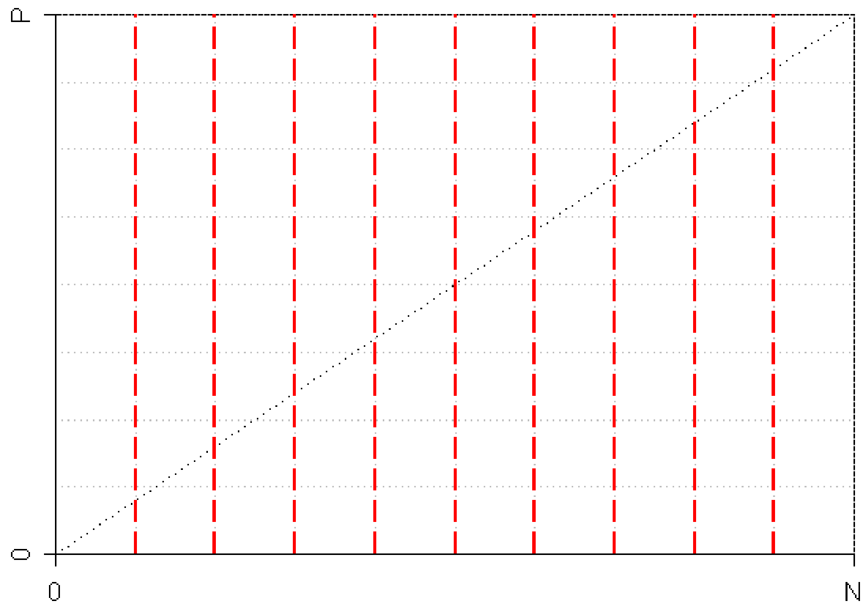
Condition		p	n	Precision	Laplace	p-n
Temperature =	Hot	2	2	0.5000	0.5000	0
	Mild	3	1	0.7500	0.6667	2
	Cold	4	2	0.6667	0.6250	2
Outlook =	Sunny	2	3	0.4000	0.4286	-1
	Overcast	4	0	1.0000	0.8333	4
	Rain	3	2	0.6000	0.5714	1
Humidity =	High	3	4	0.4286	0.4444	-1
	Normal	6	1	0.8571	0.7778	5
Windy =	True	3	3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

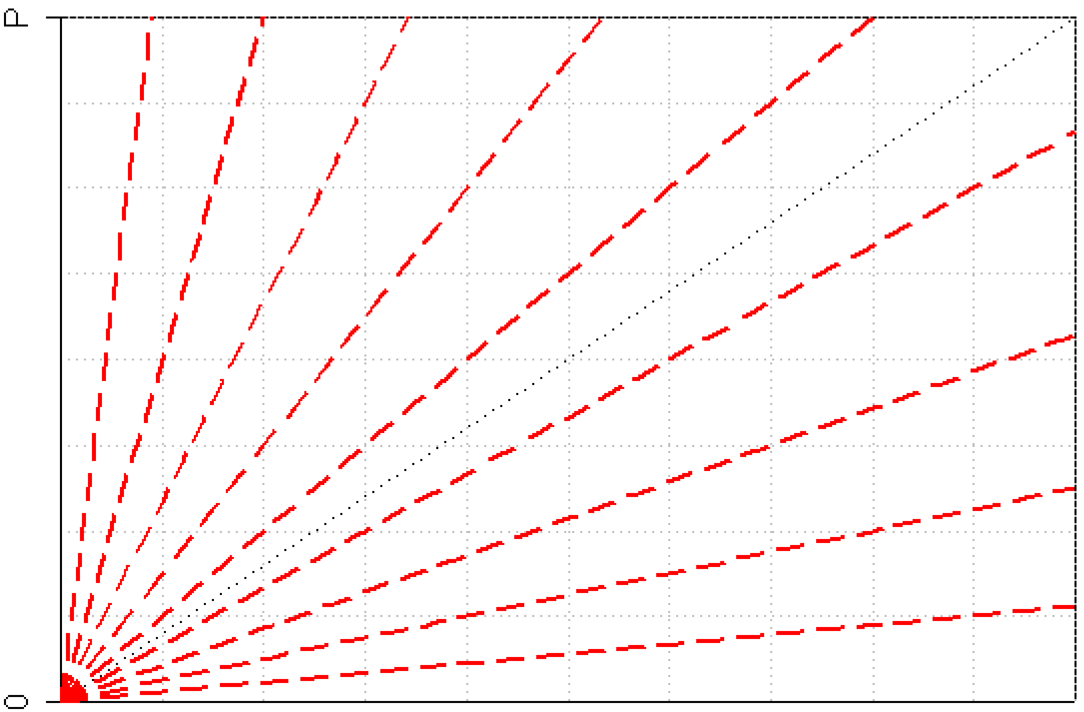




Precision (Confidence)

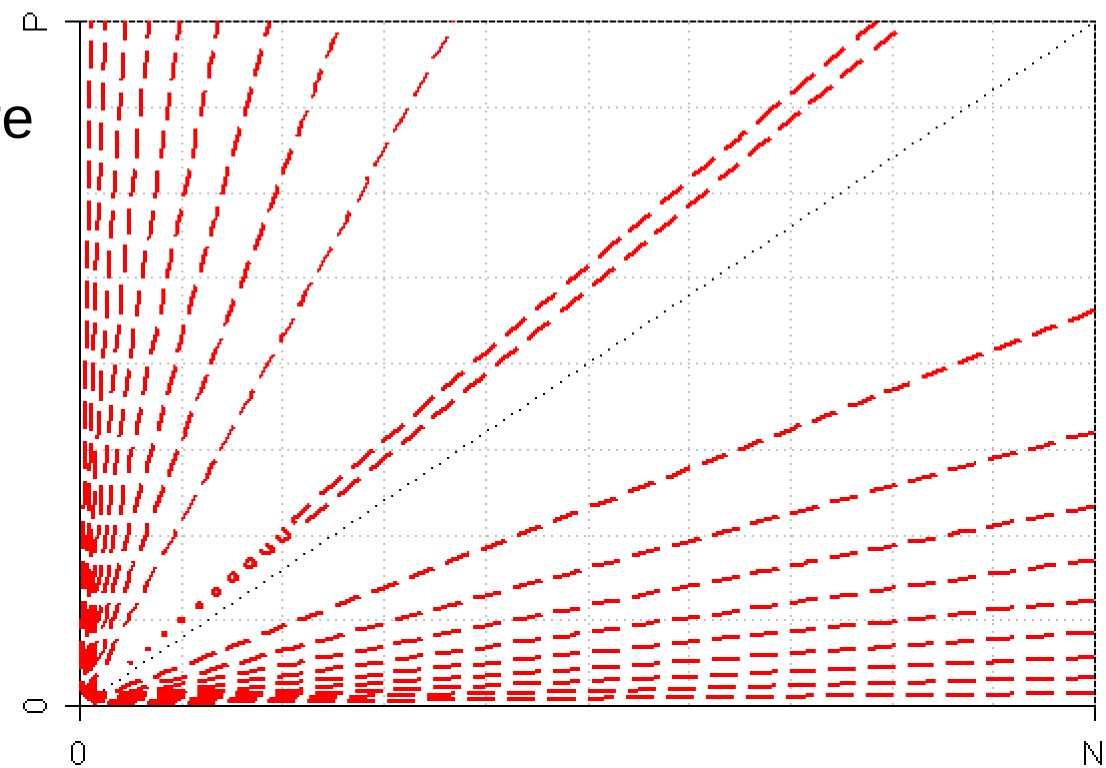
$$h_{Prec} = \frac{p}{p+n}$$

- *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

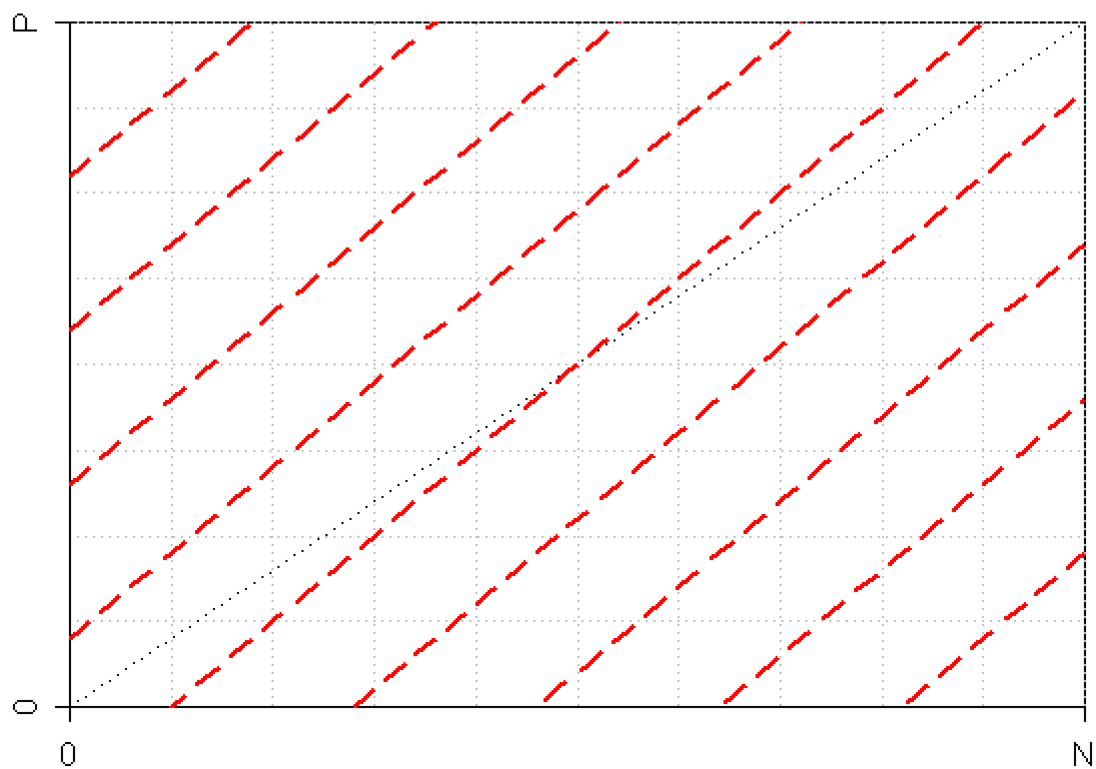
- $h_{Ent} = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n}\right)$
- $h_{Gini} = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2 \simeq \frac{pn}{(p+n)^2}$
- *effects:*
  - entropy and Gini index are equivalent
  - like precision, isometrics rotate around (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



Accuracy

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

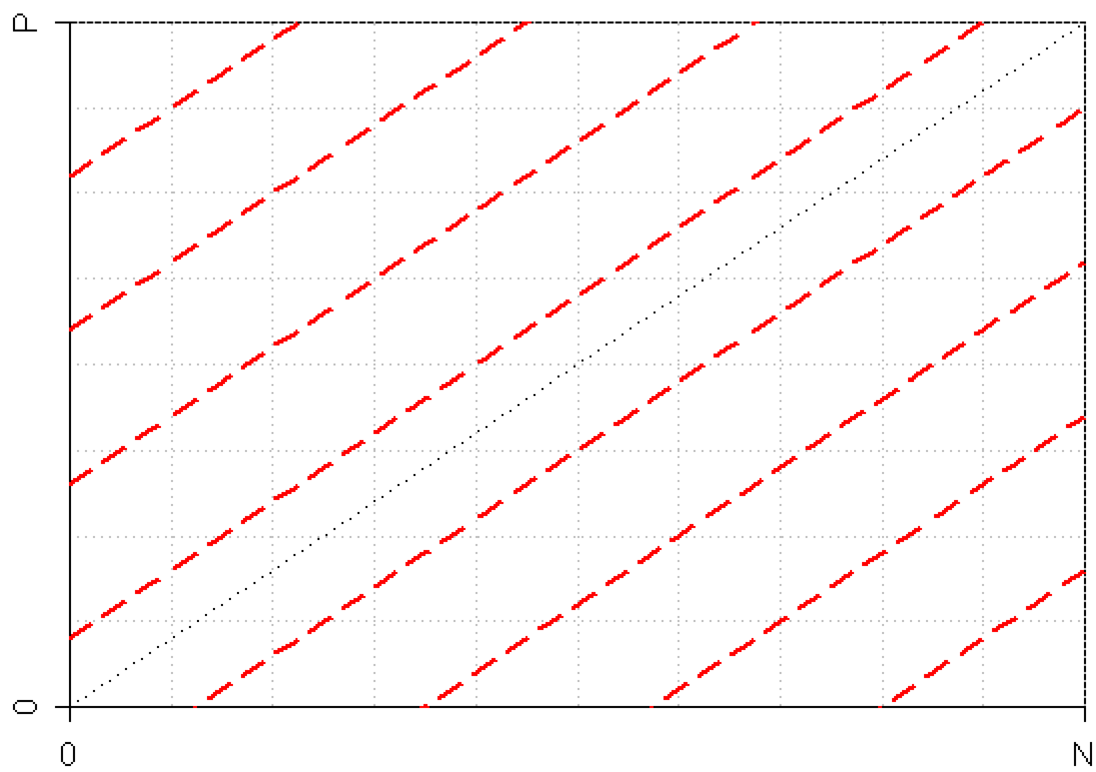
- *basic idea:*
  percentage of correct classifications
  (*covered positives* plus *uncovered negatives*)
- *effects:*
  - isometrics are parallel to 45° line
  - covering one positive example is as good as not covering one negative example



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

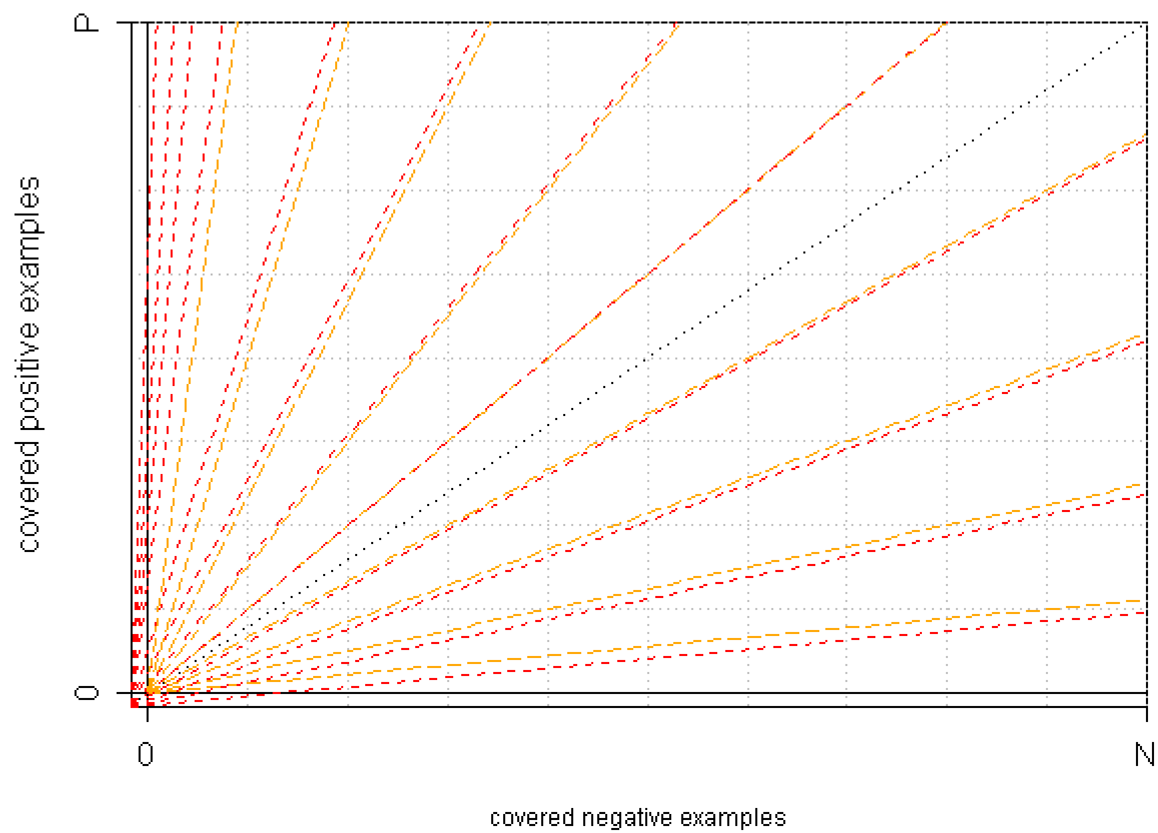
- Accuracy and weighted relative accuracy are only two special cases of the general case with linear costs:
  - costs c mean that covering 1 positive example is equivalent to not covering $(1\text{-}c)/c$ negative examples

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

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

Laplace-Estimate

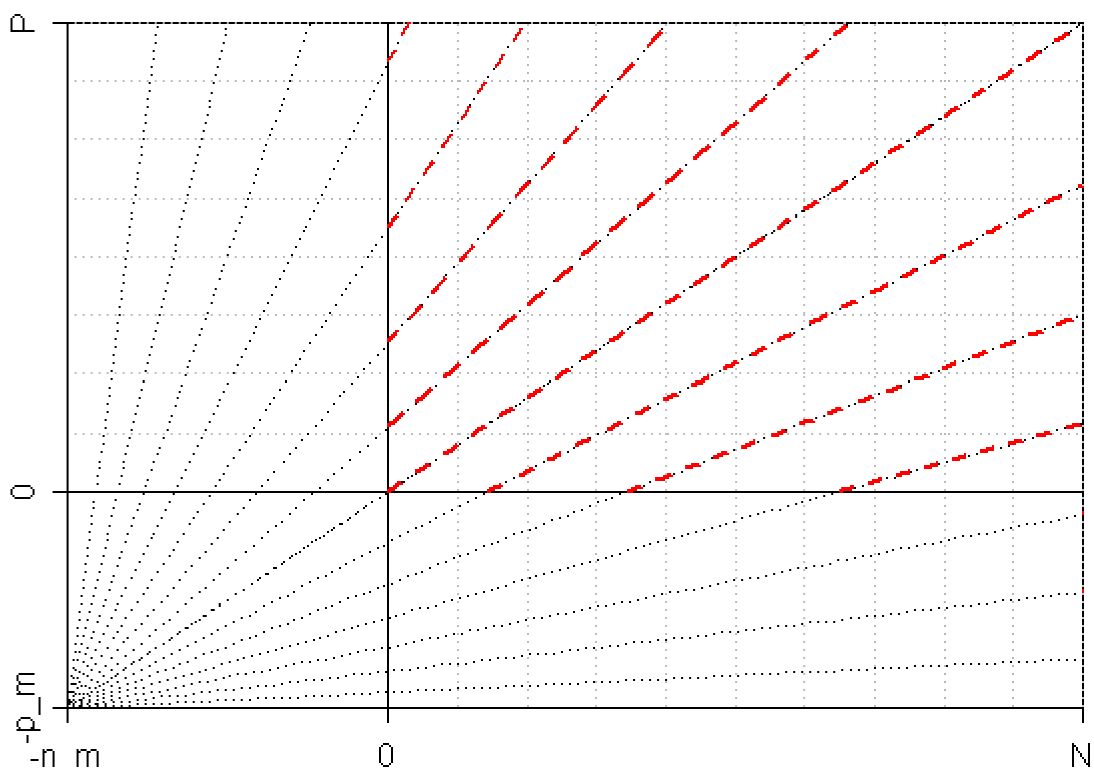
- $h_{Lap} = \dfrac{p+1}{p+n+2}$
- *basic idea:*
  precision, but count coverage for positive and negative examples starting with 1 instead of 0
- *effects:*
  - origin at $(-1,-1)$
  - different values on $p=0$ or $n=0$ axes
  - not equivalent to precision



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 confidence

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

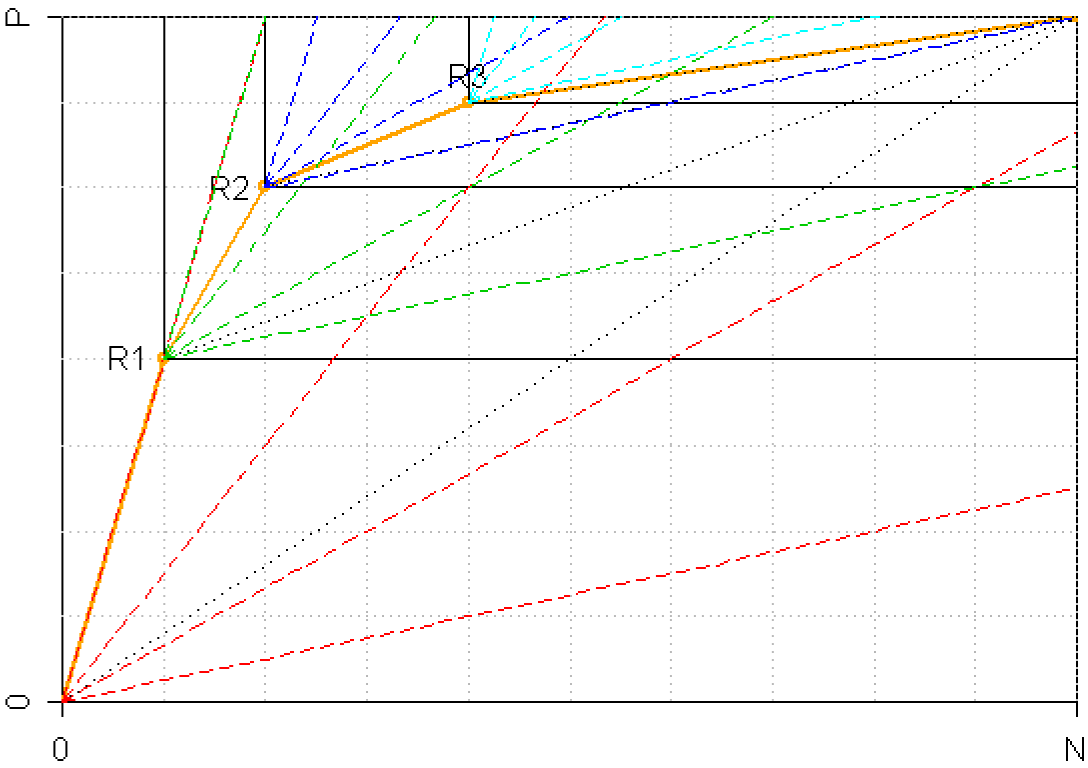


Generalized m-Estimate

- 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 m-Estimate may be viewed as a means of trading off between precision and the cost metric

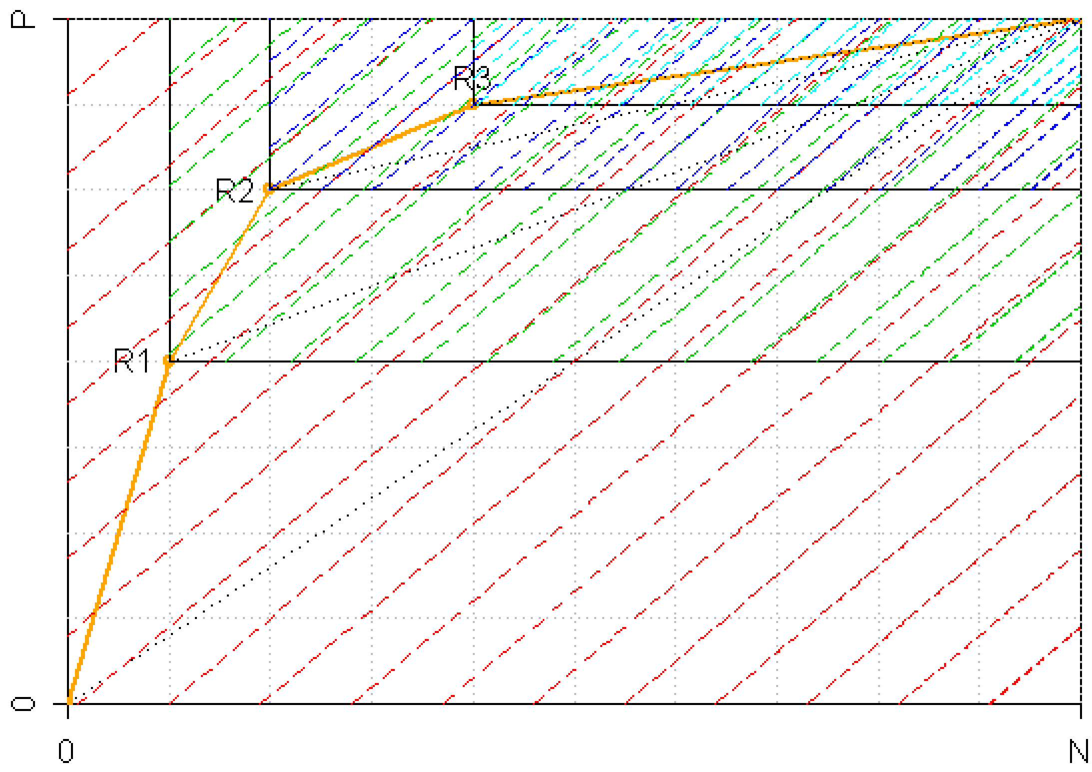
Optimizing Precision

- Precision tries to pick the steepest continuation of the curve
  - does not assume any costs



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

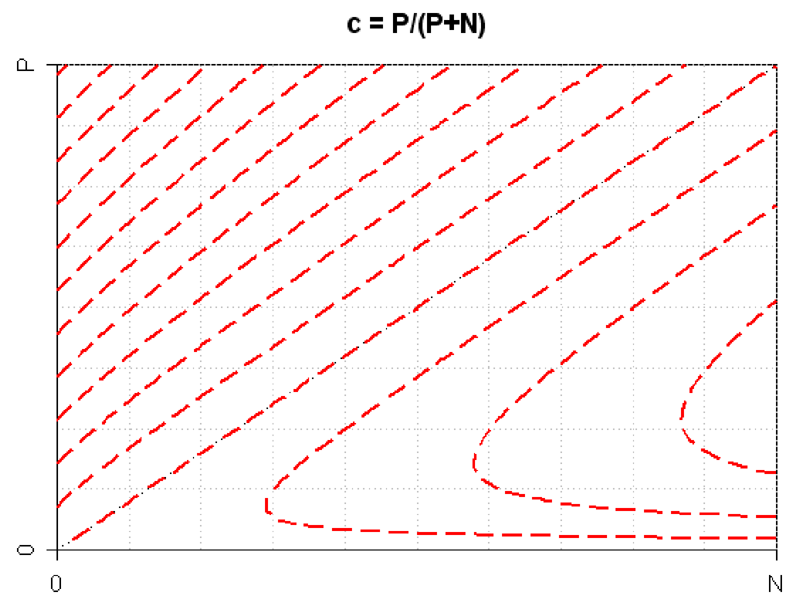
- There are <u>two basic types</u> of (linear) heuristics.
  - <u>*precision*</u>: rotation around the origin
  - <u>*cost metrics*</u>: parallel lines
- They have <u>different goals</u>
  - *precision* picks the steepest continuation for the curve for <u>unkown costs</u>
  - *linear cost metrics* pick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

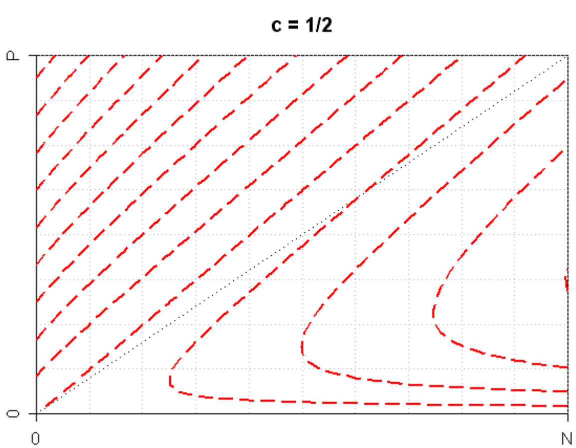
- The m-heuristic may be interpreted as a <u>trade-off</u> between the two prototypes
  - parameter c chooses the *cost model*
  - parameter m chooses the *“degree of parallelism”*

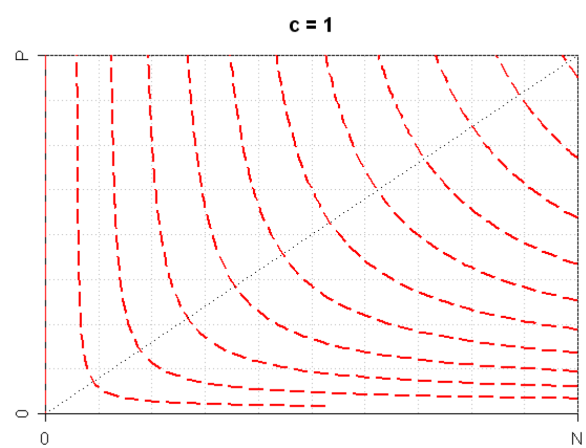
Foil Gain

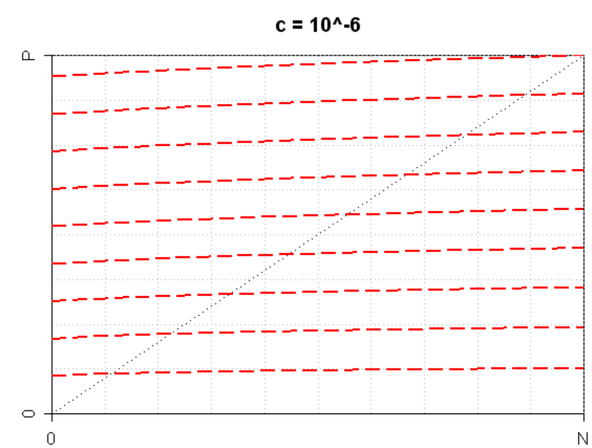
$$h_{foil} = -p\left(\log_2 c - \log_2 \frac{p}{p+n}\right)$$

(c is the precision of the parent clause)









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*

a fully specialized a single example

1. Start with ~~an empty~~ rule R that covers ~~all examples~~

delete

2. Evaluate all possible ways to ~~add~~ 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 a 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)

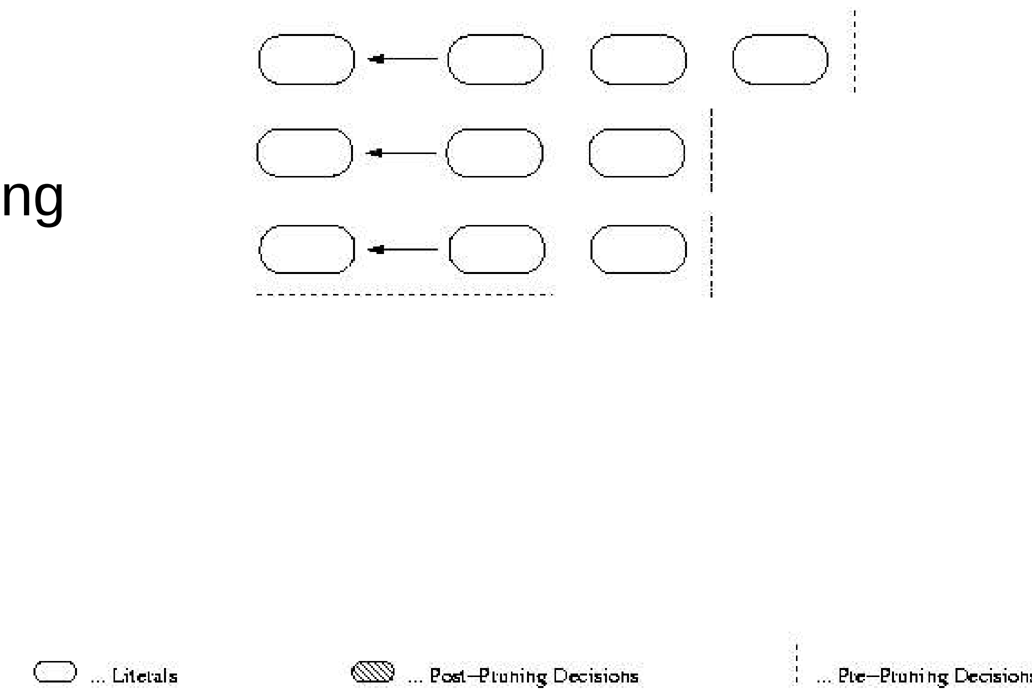
Overfitting

- 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

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



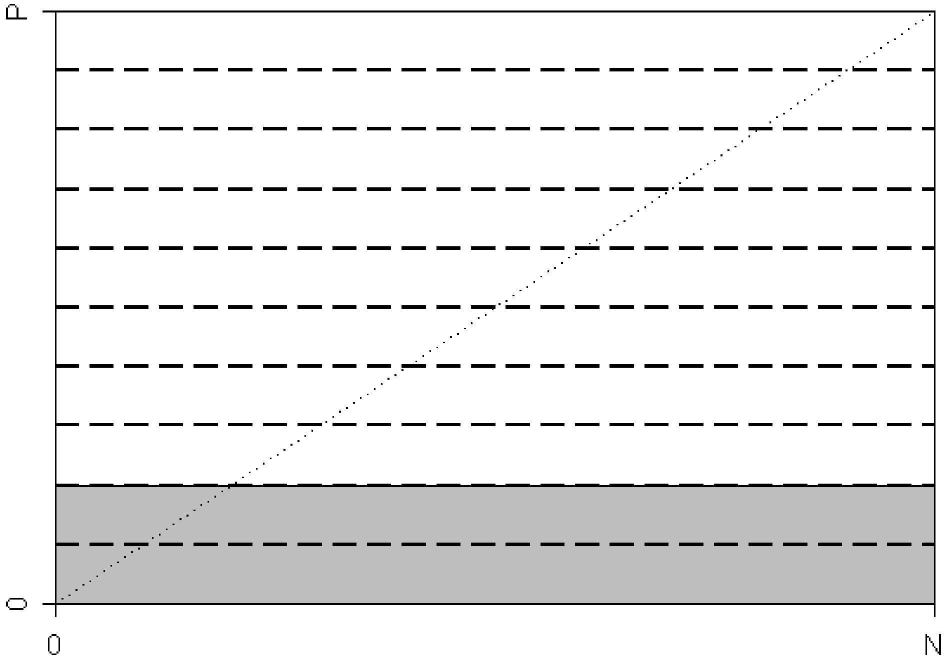
Pre-Pruning Heuristics

- Threshold
  - require a certain minimum value of the search heuristic
  - e.g.: Laplace > 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

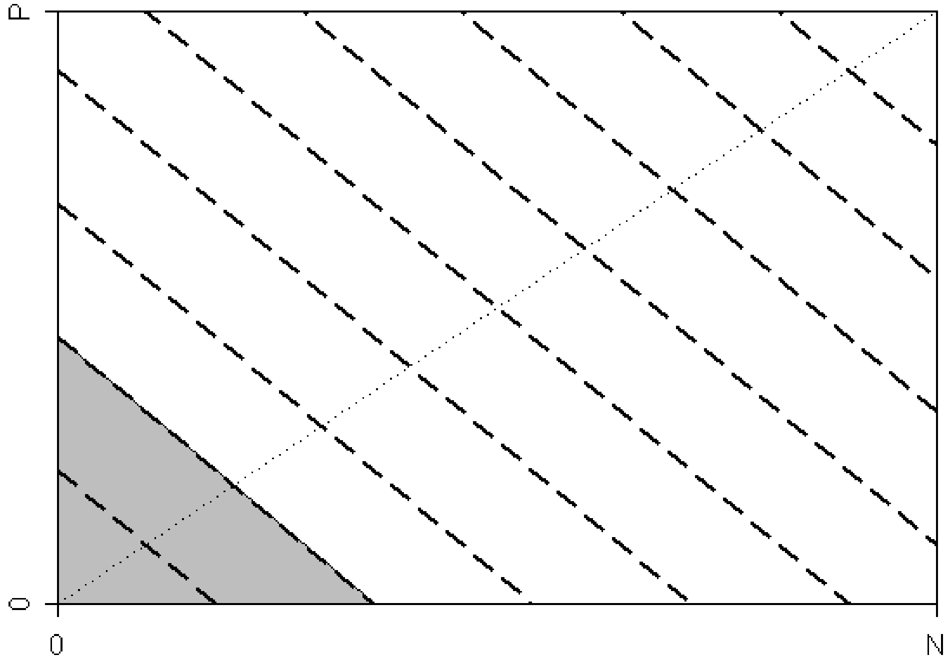
Minimum Coverage Filtering

filter rules that do not cover a minimum number of

positive examples

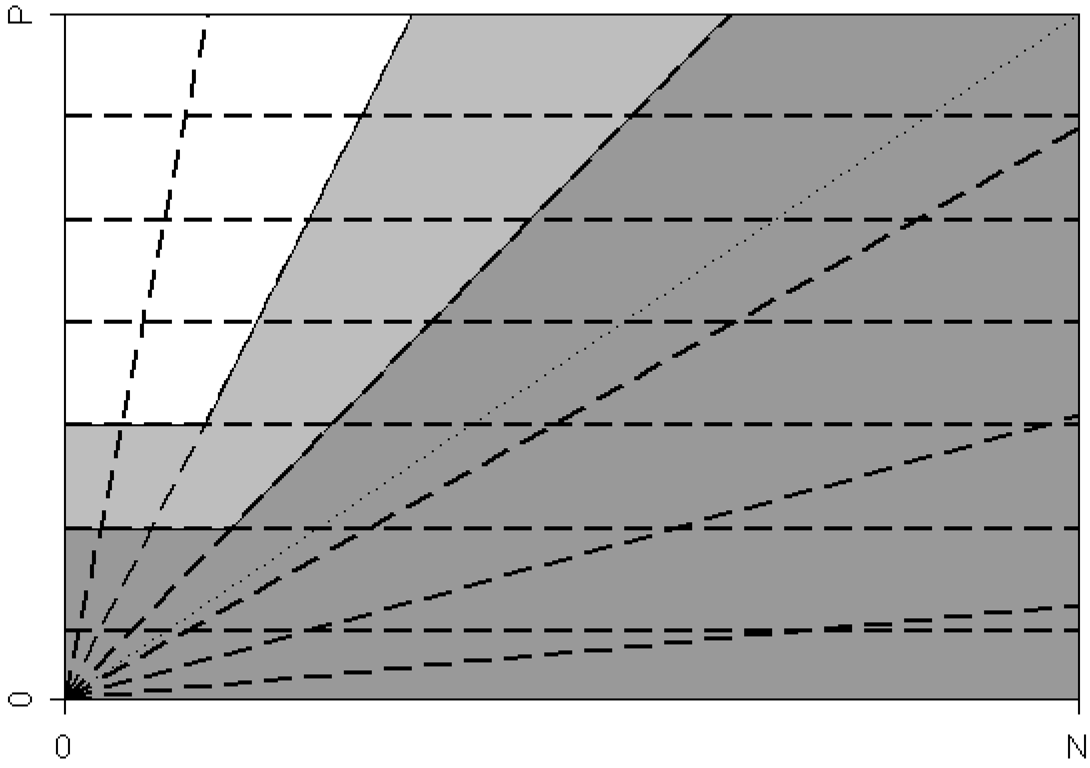


all examples



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

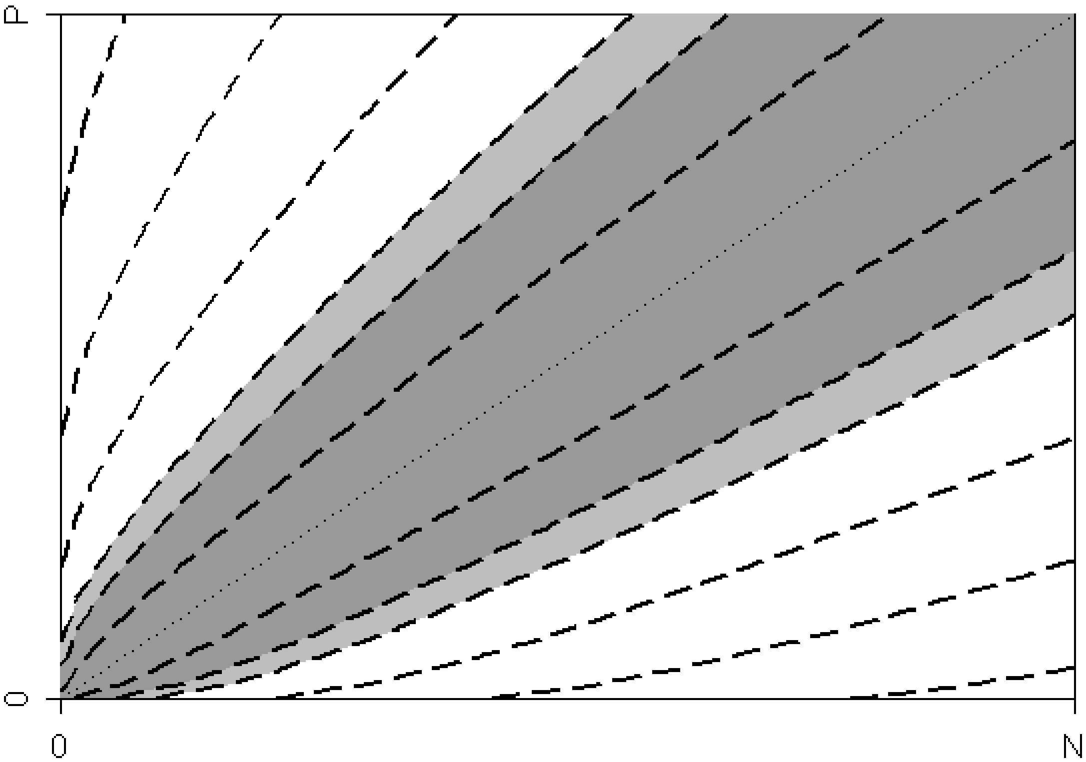


CN2's likelihood ratio statistics

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

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

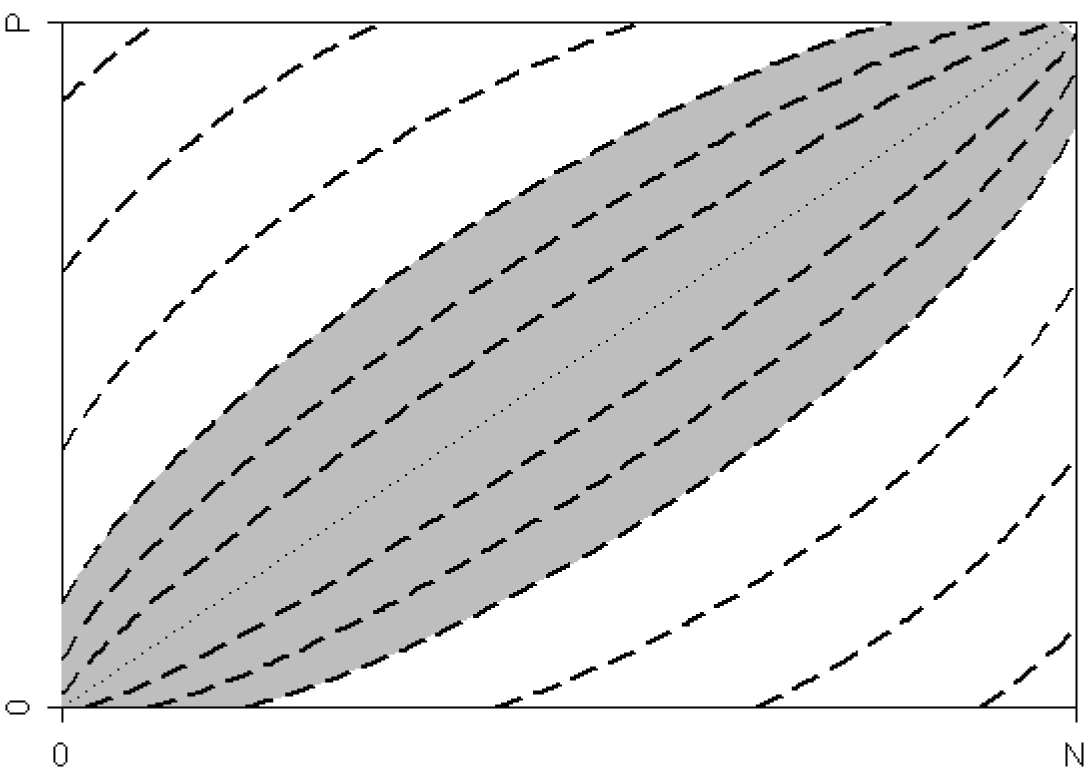
- *basic idea:*
  measure significant deviation from prior probability distribution
- *effects:*
  - non-linear isometrics
    - similar to m-estimate
    - but prefer rules near the edges
  - distributed χ^2
  - significance levels 95% (dark) and 99% (light grey)



Fossil's Correlation

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

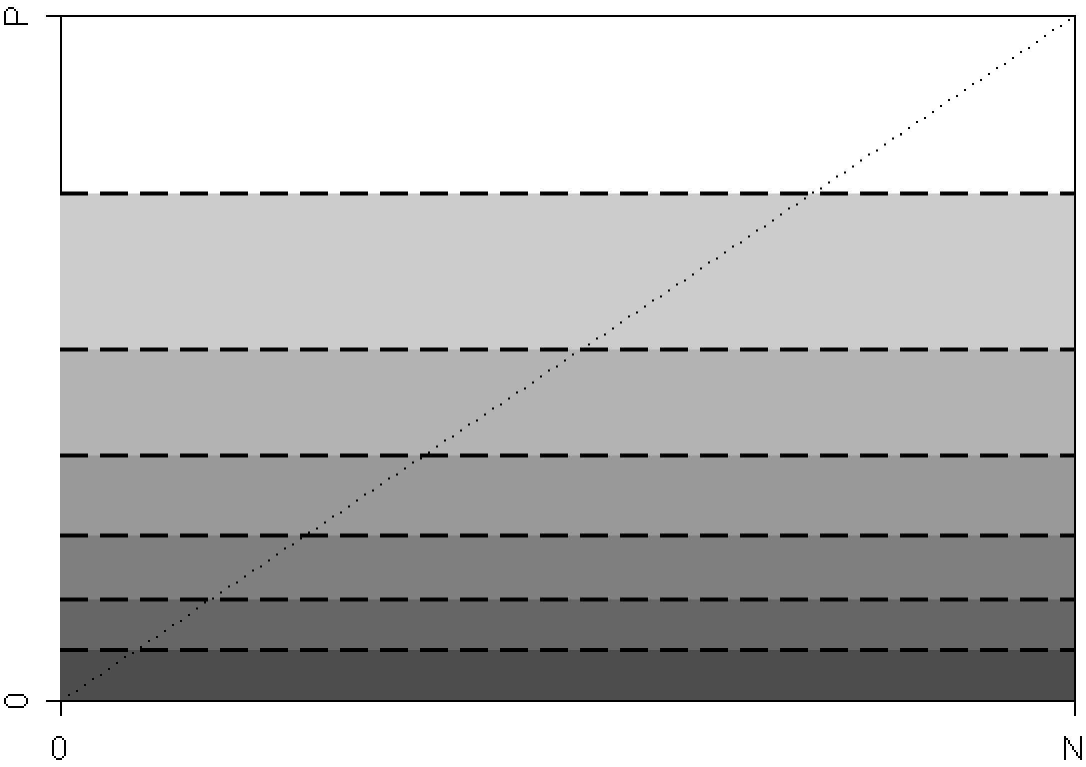
- *basic idea:*
  measure correlation coefficient of predictions with target
- *effects:*
  - non-linear isometrics
  - in comparison to WRA
    - prefers rules near the edges
    - steepness of connection of intersections with edges increases
  - equivalent to χ^2
  - grey area = cutoff of 0.3



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

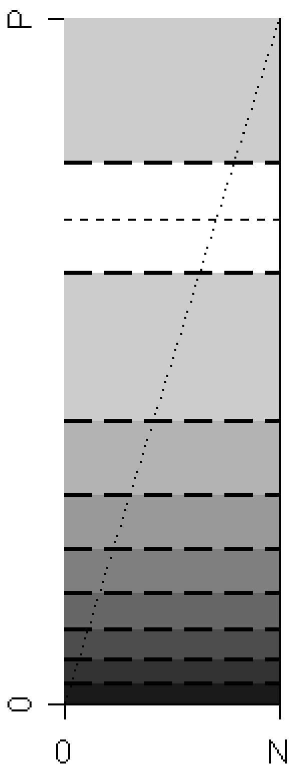


Anomaly of Foil's Stopping criterion

- 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!



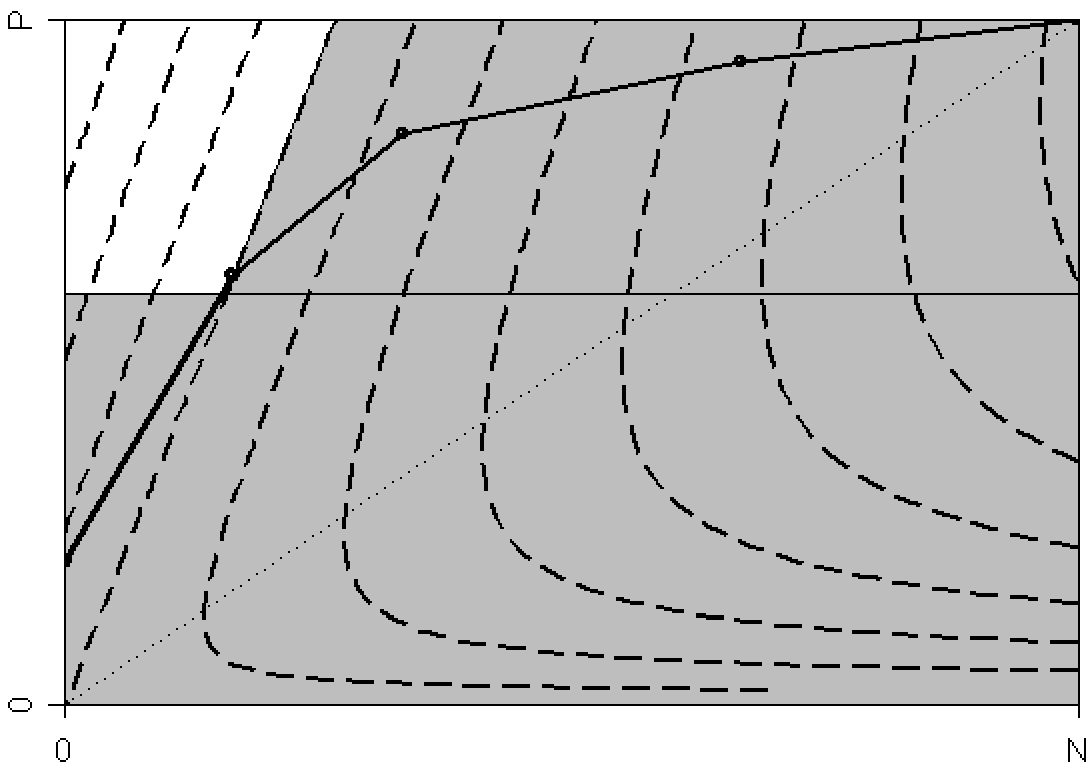
Pre-Pruning Systems

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

How Foil Works

→ Foil (almost) implements Support/Confidence Filtering

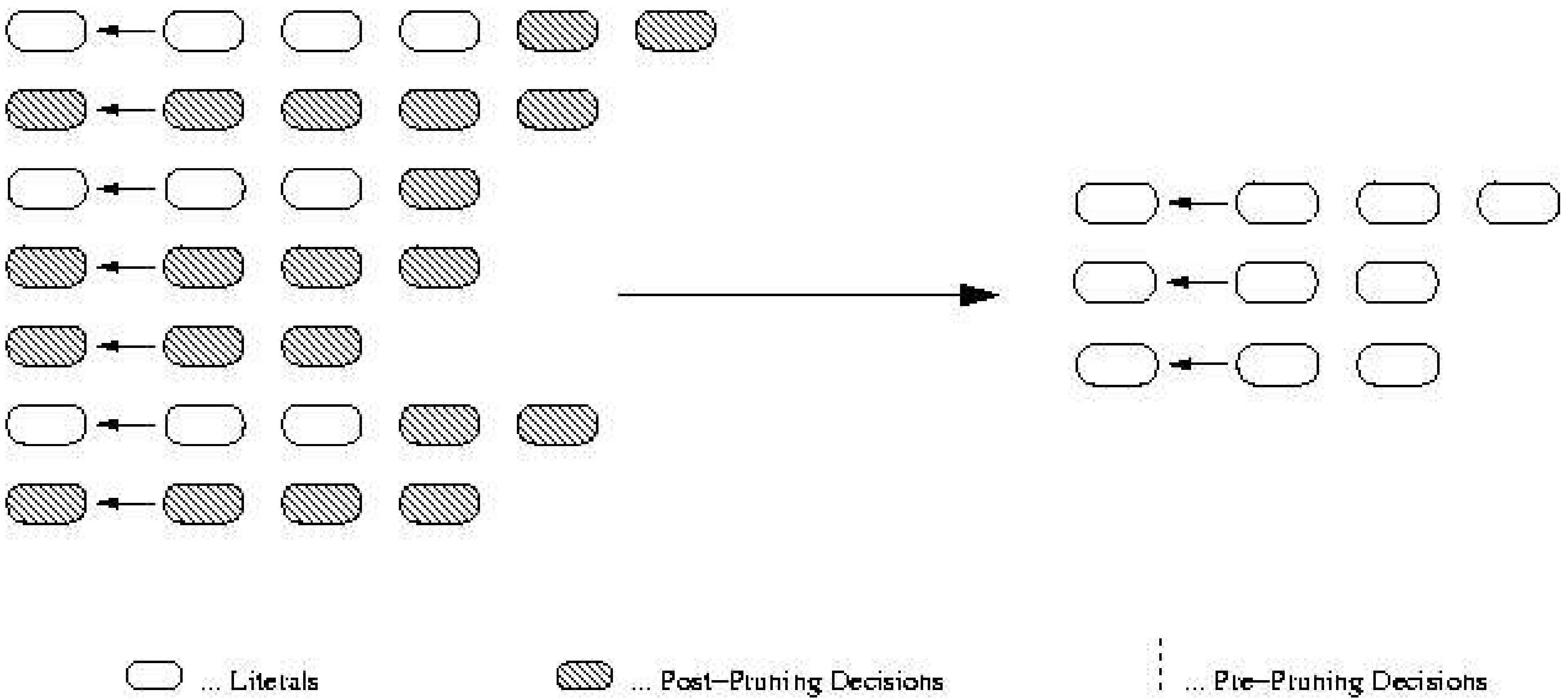
- filtering of rules with no information gain
  - after each refinement ste the region of acceptable rules is adjusted as in precision/confidence filtering
- filtering of rules that exceed the rule length
  - after each refinement ste the region of acceptable rules is adjusted as in support filtering

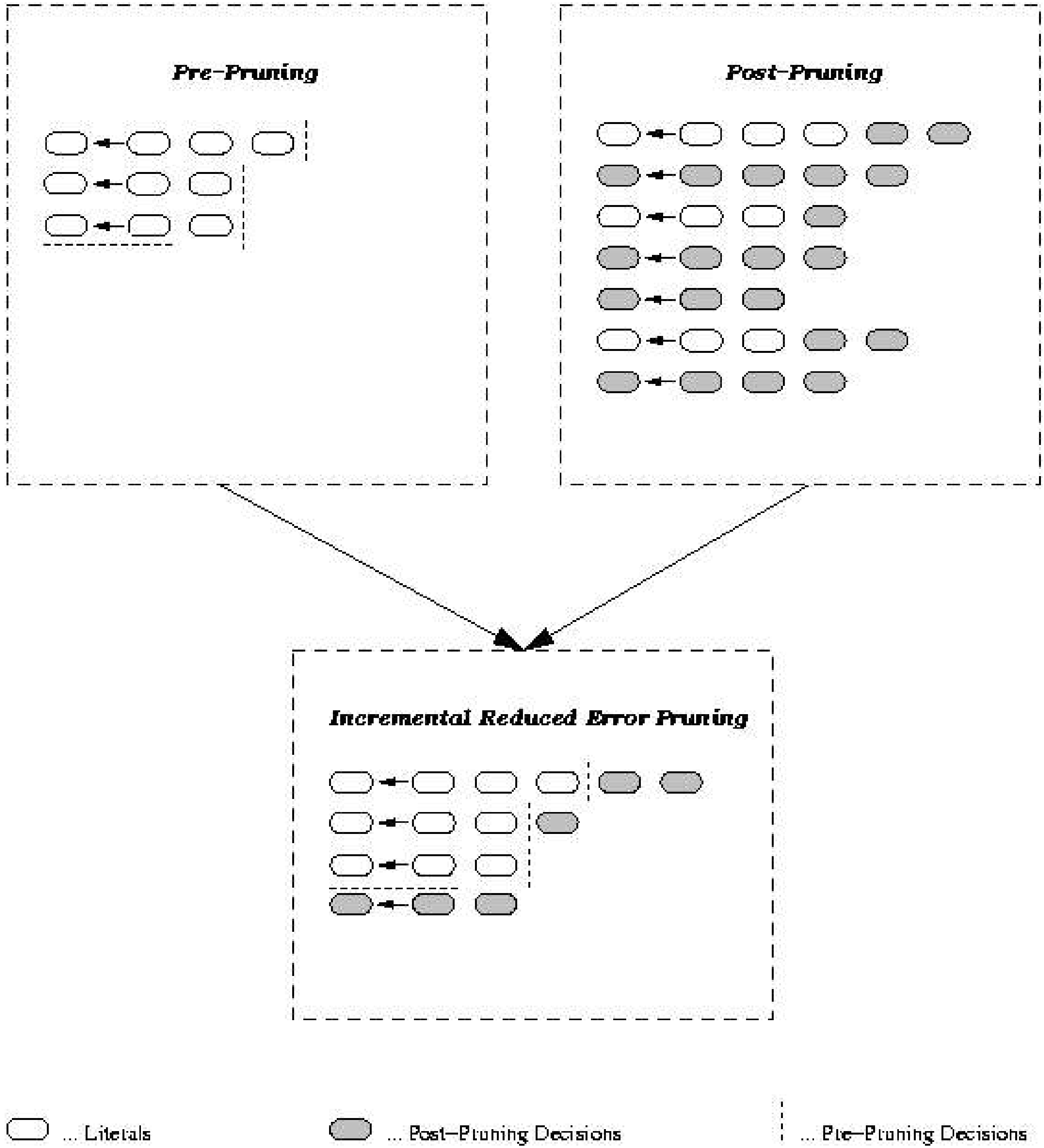


Post Pruning

- simplify a theory after it has been learned
- Reduced Error Pruning
  - anaologous to decision trees
    - Reserve part of the data for validation (pruning set)
    - Learn a rule set
    - Simplify rule set by deleting rules and conditions as long as this does not decrease accuracy on the validation set
- accurate but not efficient
  - $O(n^4)$

Reduced Error Pruning





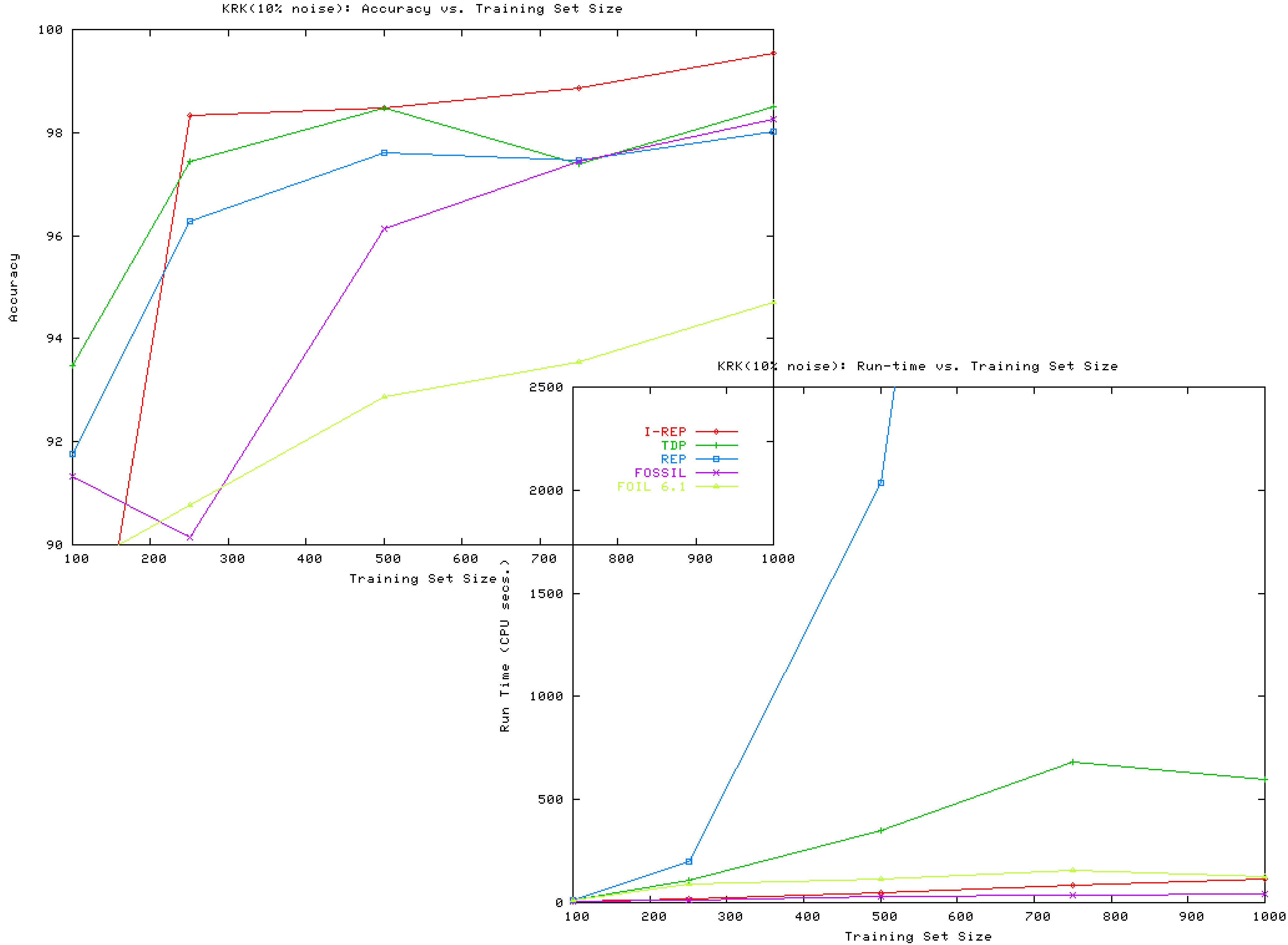
Pre-Pruning
Post-Pruning
Incremental Reduced Error Pruning
... Literals
... Post-Pruning Decisions
... Pre-Pruning Decisions

Incremental Reduced Error Pruning

- Prune each rule right after it is learned:

  1. split data into a training 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.

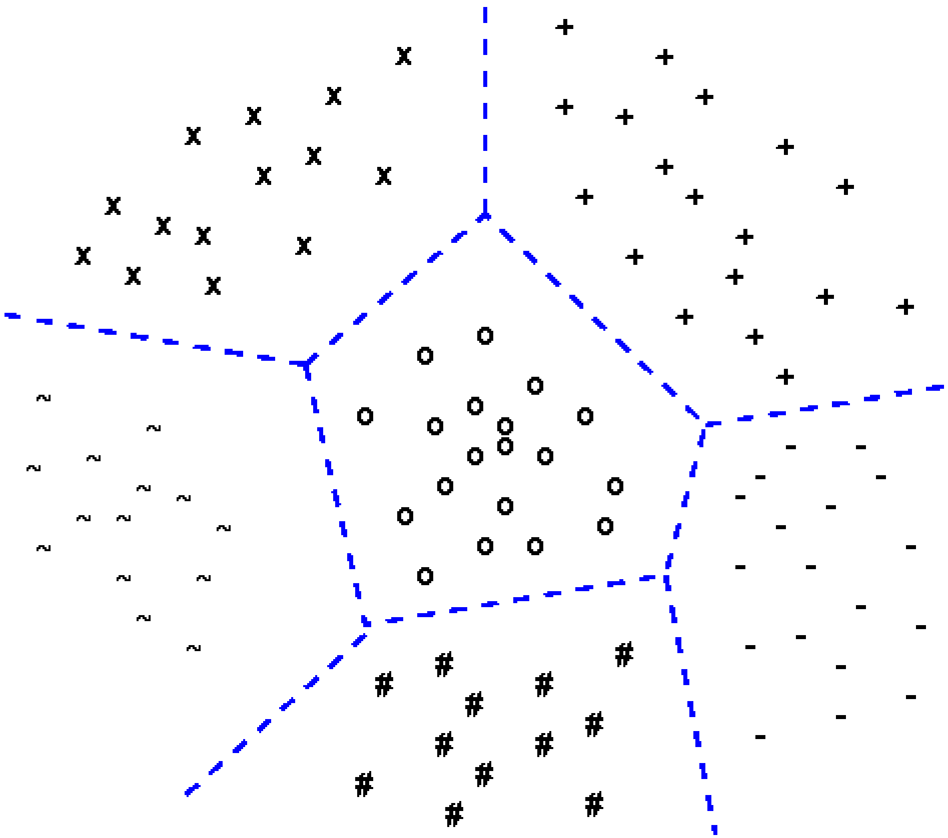
- More accurate, much more efficient
  - because it does not learn overly complex intermediate concept
  - REP: $O(n^4)$ I-REP: $O(n \log^2 n)$
- Subsequently used in the RIPPER (JRip in Weka) rule learner (Cohen, 1995)



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

Multi-class problems

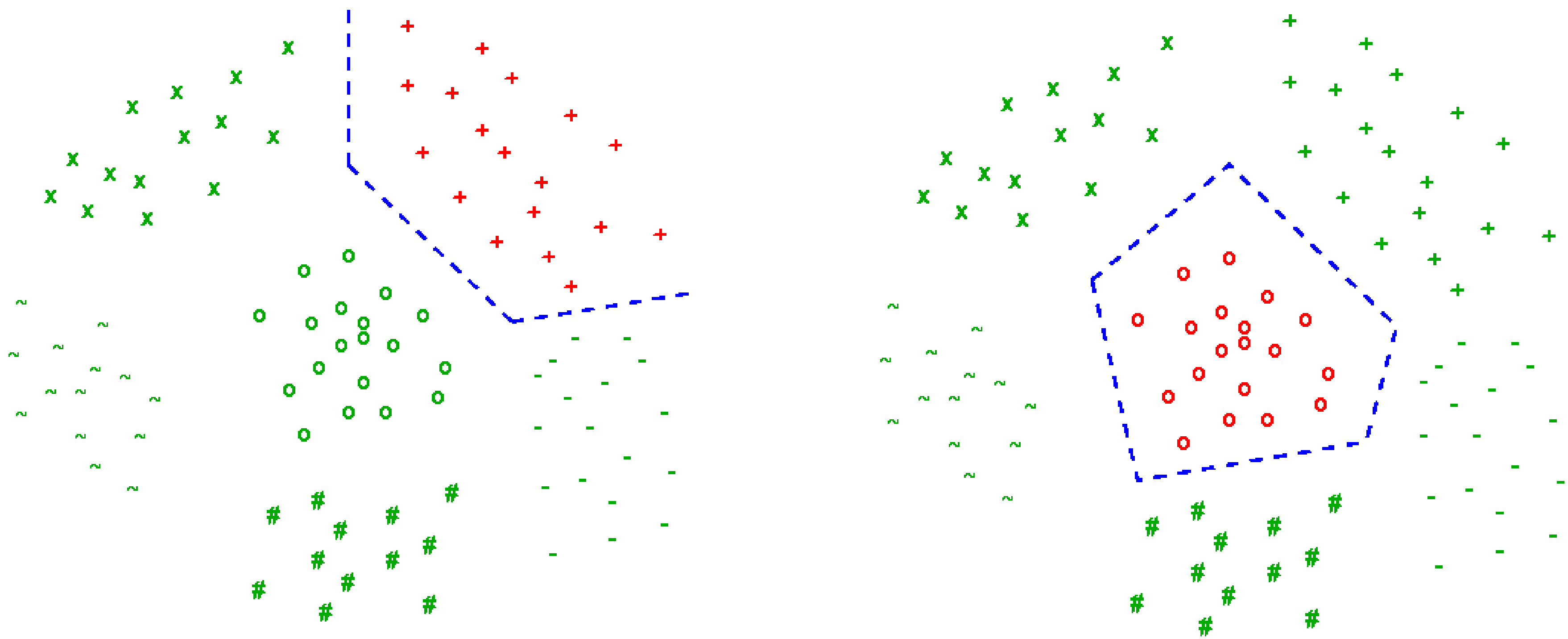
- GOAL: discriminate c classes from each other
- PROBLEM: many learning algorithms are only suitable for binary (2-class) problems
- SOLUTION: *"Class binarization":* Transform an c-class problem into a series of 2-class problems



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

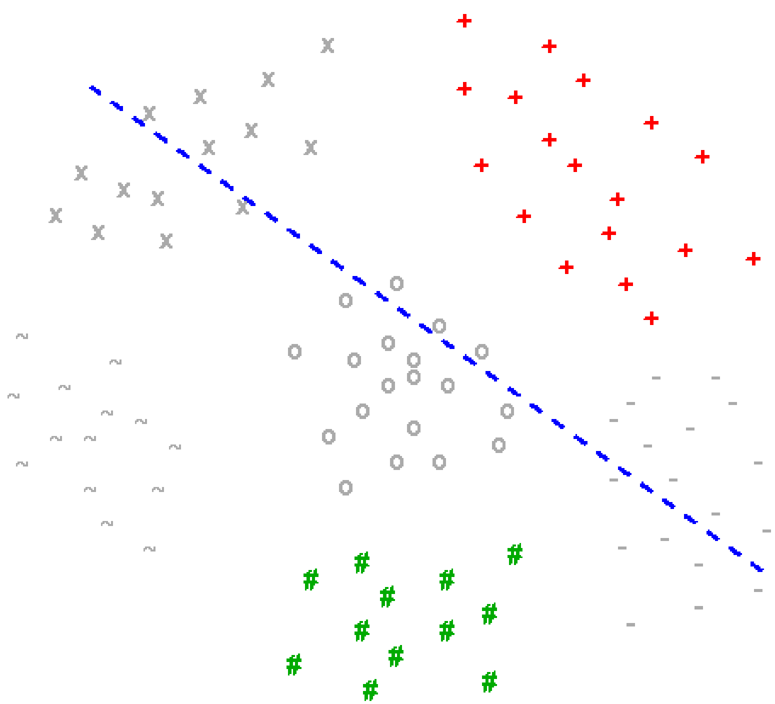


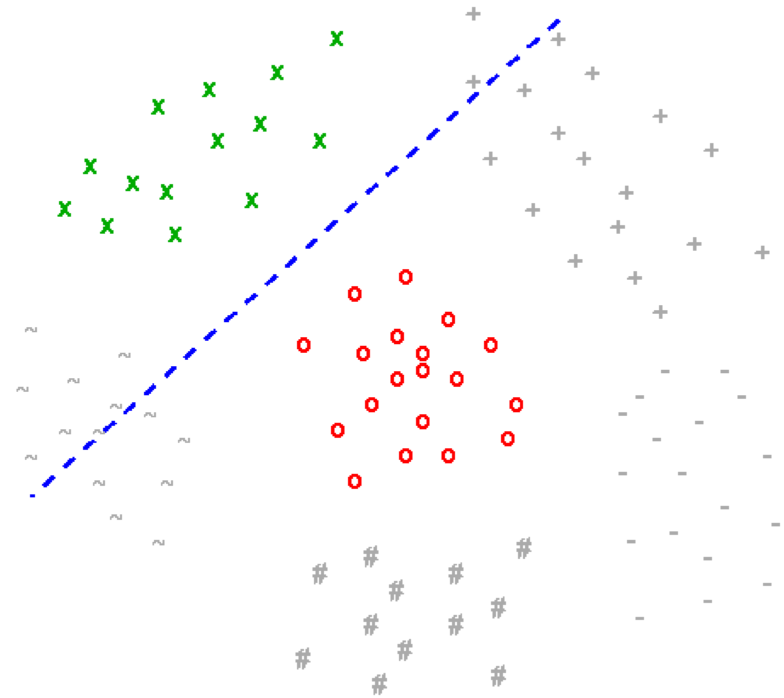
Treat each class as a separate concept:

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

Round Robin Learning
(aka *Pairwise Classification*)

- $c(c\text{-}1)/2$ problems
- each class against each other class





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

Accuracy

dataset	Ripper unord.	Ripper 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