Rule Learning

From Subgroup Discovery to Preference Learning

Johannes Fürnkranz
TU Darmstadt
Knowledge Engineering Group
Hochschulstrasse 10
D-64289 Darmstadt
06151/166238
juffi@ke.tu-darmstadt.de

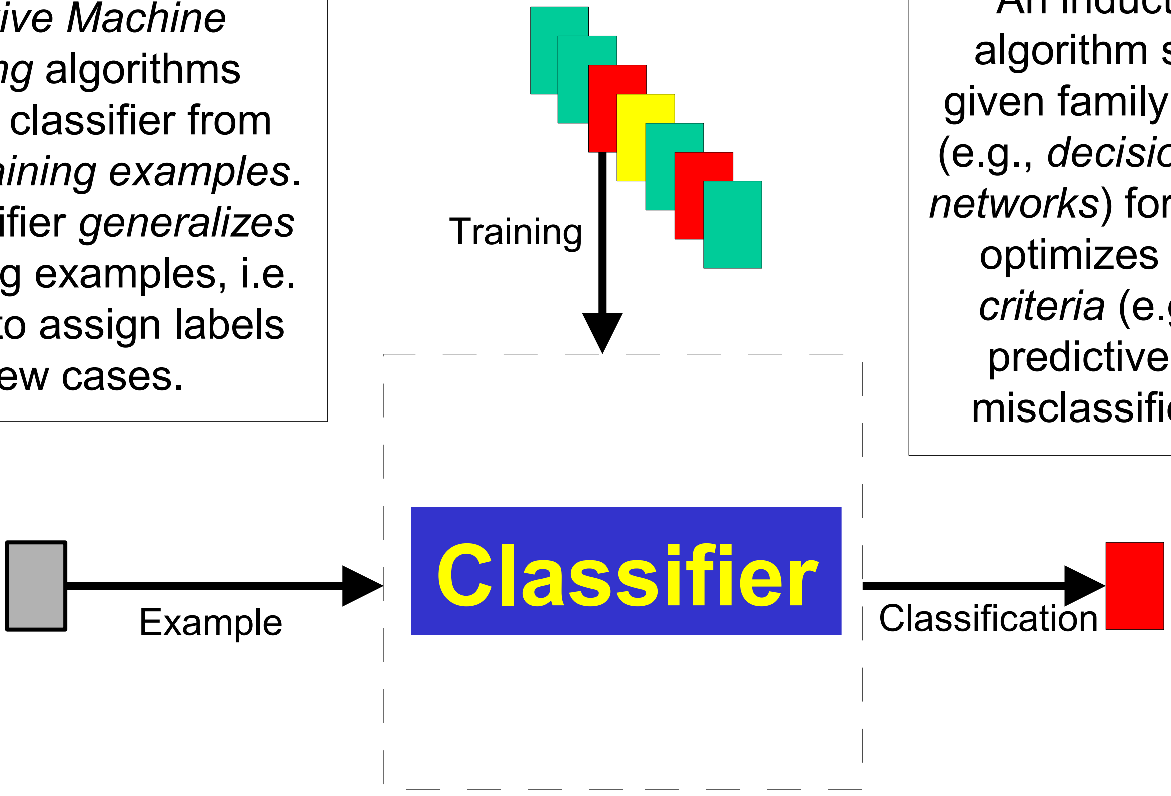
Overview

- Introduction
  - Features
  - Conjunctive Rules
- Subgroup Discovery
  - Rule Learning Heuristics
  - Inverted Heuristics
- Concept Learning
  - Covering Strategy
  - LeGo Framework
- Overfitting Avoidance
  - Pruning
  - Incremental REP
- Multi-Class Classification
  - One-Against-All Strategy
  - Pairwise Classification
- Preference Learning
  - Label Ranking
  - Pairwise Label Ranking
- Multi-label Classification
  - Calibrated Label Ranking
- Regression
  - Reduction to Classification
- Summary

Supervised Learning - Induction of Classifiers

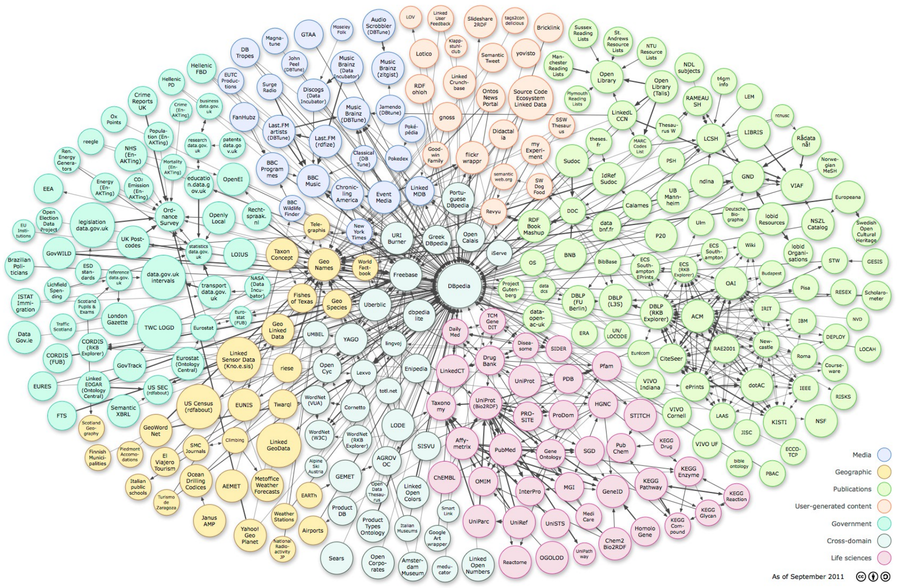
Inductive Machine Learning algorithms induce a classifier from *labeled training examples*. The classifier *generalizes* the training examples, i.e. it is able to assign labels to new cases.

An inductive learning algorithm searches in a given family of hypotheses (e.g., *decision trees, neural networks*) for a member that optimizes given *quality criteria* (e.g., estimated predictive accuracy or misclassification costs).



Why Rules?

- Rules provide a good (the best?) trade-off between
  - human understandability
  - machine executability
- Used in many applications which will gain importance in the near future
  - Security
  - Spam Mail Filters
  - Semantic Web
- But they are not a universal tool
  - e.g., learned rules sometimes lack in predictive accuracy → challenge to close or narrow this gap

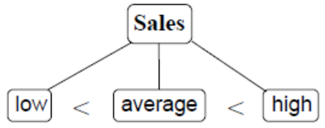


Features

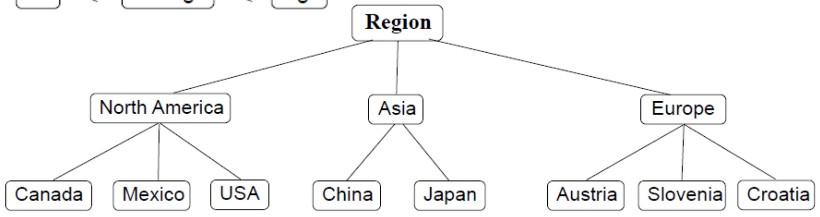
- A feature is a Boolean property of an object

Feature types

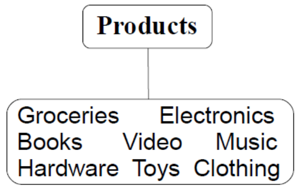
- Selectors
  - select a nominal value: **Sex** = female
  - compare to a numerical value: **Salary** > 100,000
- Ordered features
  - the nominal values form an ordered set



- Hierarchical features
  - the nominal values form a hierarchy



- Relational features
  - relate two or more values to each other **Length** > **Height**
- Set-valued features
  - compare to a set of values (e.g., a set of words)



Conjunctive Rule

$\underbrace{\texttt{if (feature 1) and (feature 2)}}_{} \quad \underbrace{\texttt{then +}}_{}$

Body of the rule (IF-part)

- contains a conjunction of conditions
- a condition is a binary feature

Head of the rule (THEN-part)

- contains a prediction
- typically + if object belongs to concept, – otherwise

- Coverage
  - A rule is said to ***cover*** an example if the example satisfies the conditions of the rule.
- Prediction
  - If a rule covers an example, the rule's head is predicted for this example.

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")

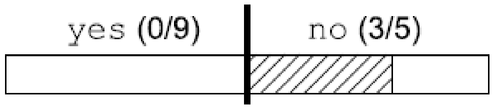
Subgroup Discovery

- Definition

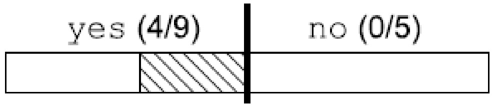
> "Given a population of individuals and a property of those individuals that we are interested in, **find population subgroups** that are statistically 'most interesting', e.g., are as large as possible and have the most unusual distributional characteristics with respect to the property of interest"
>
> (Klösgen 1996; Wrobel 1997)

- Examples

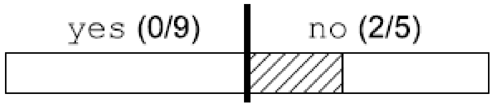
```
IF   MaritalStatus = single
 AND Sex = male
THEN Approved = no
```



```
IF   MaritalStatus = married
THEN Approved = yes
```



```
IF   MaritalStatus = divorced
 AND HasChildren = yes
THEN Approved = no
```



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

Algorithms for Subgroup Discovery

Objective

- Find the best subgroups / rules according to some measure h

Algorithms

- Greedy search
  - top-down hill-climbing or beam search
    - successively add conditions that increase value of h
    - most popular approach
- Exhaustive search
  - efficient variants
    - avoid to search permutations of conditions more than once
    - exploit monotonicity properties for pruning of parts of the search space
- Randomized search
  - genetic algorithms etc.

Top-Down Hill-Climbing

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

1. Start with the universal 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.

- Most greedy s&c rule learning systems use a top-down strategy

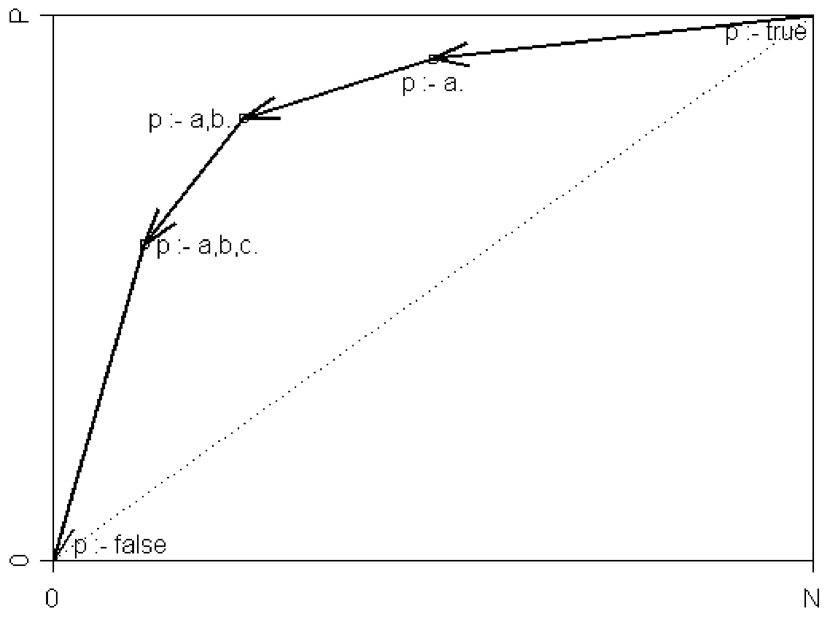
Beam Search:

- Always remember (and refine) the best b solutions in parallel

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

- How can we measure the quality of a rule?
  - should cover as few negative examples as possible (*consistency*)
  - should cover as many positive examples as possible (*completeness*)

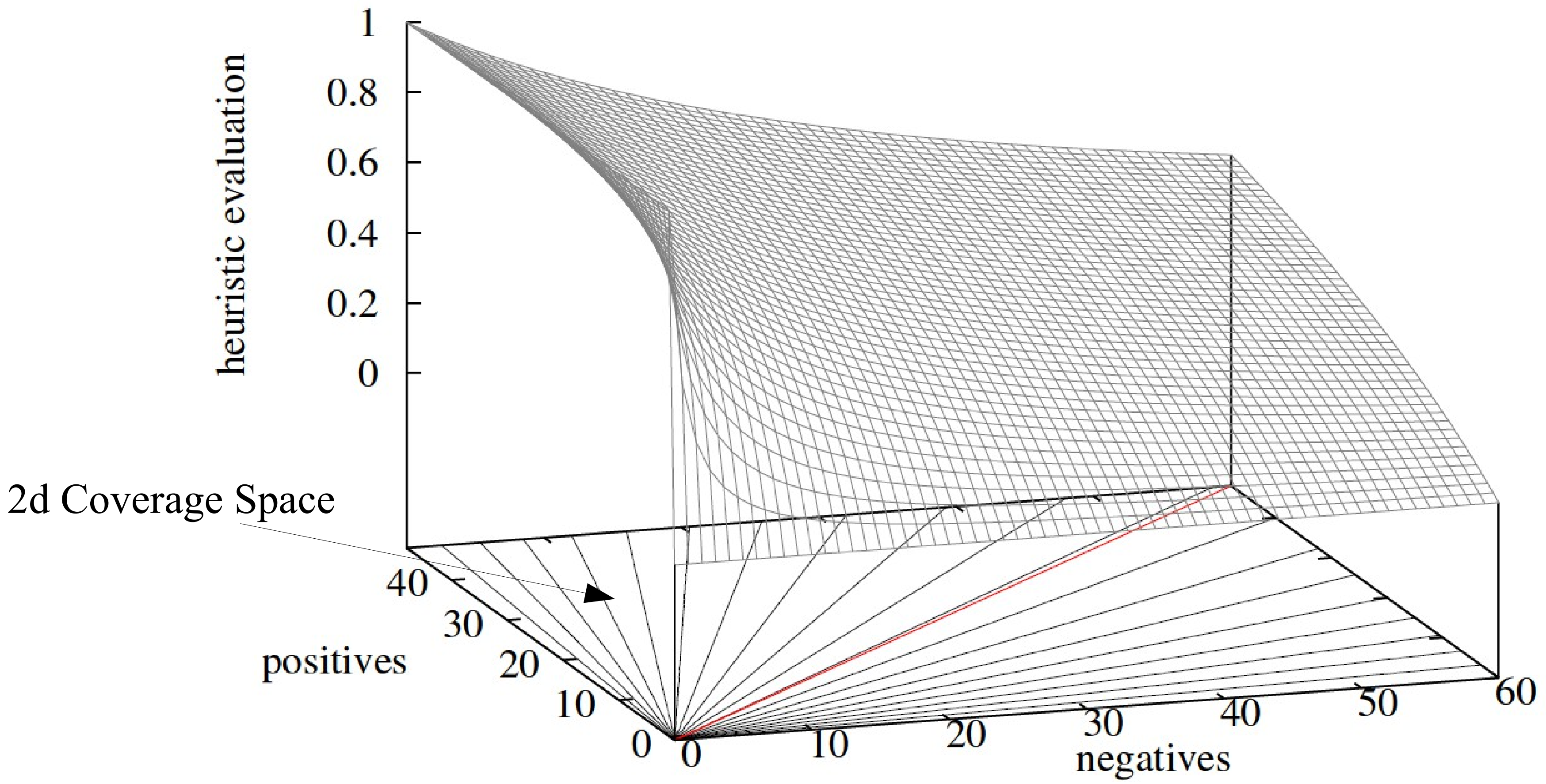
- An evaluation heuristic should therefore trade off these two properties
  - Example: Laplace heuristic $h_{Lap} = \frac{p+1}{p+n+2}$
    - grows with $p \rightarrow \infty$
    - grows with $n \rightarrow 0$
  - Example: Precision $h_{Prec} = \frac{p}{p+n}$
    - is not a good heuristic. Why?

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)

3d-Visualization of Precision



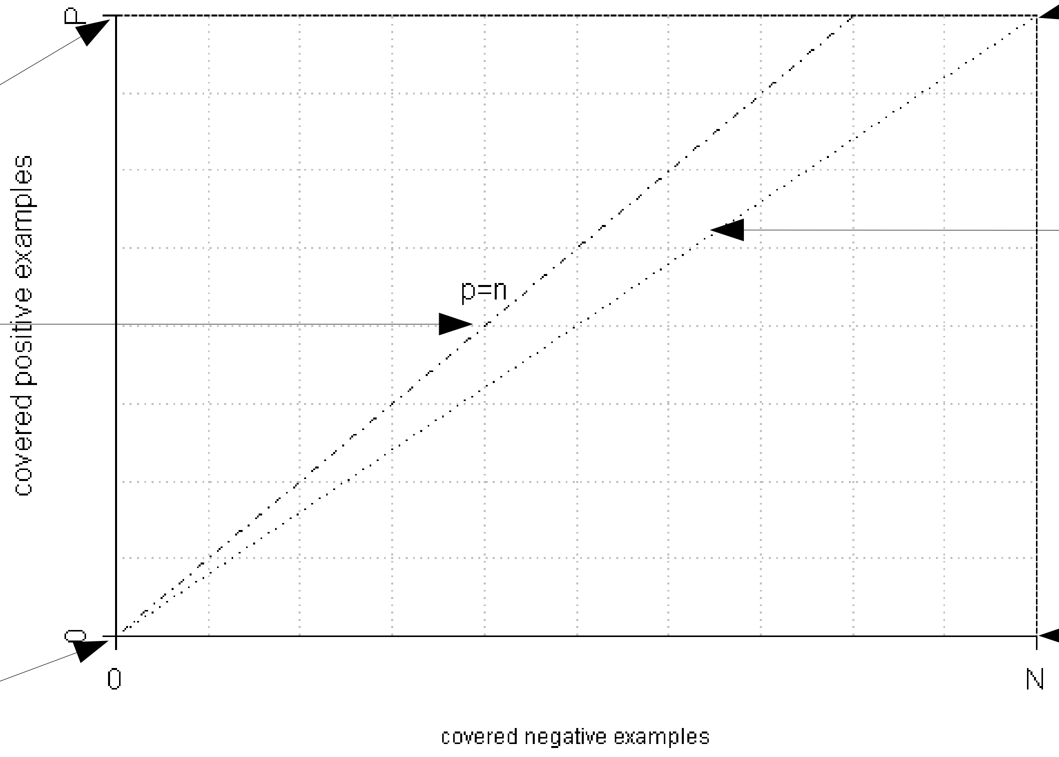
Coverage Spaces

- good tool for visualizing properties of rule evaluation heuristics
  - 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)



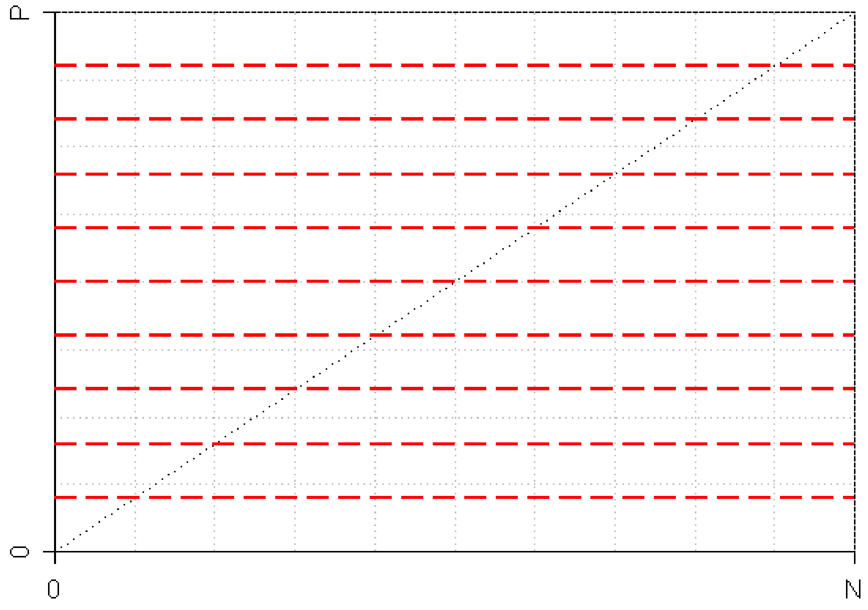
universal rule:
all examples are covered (most general)

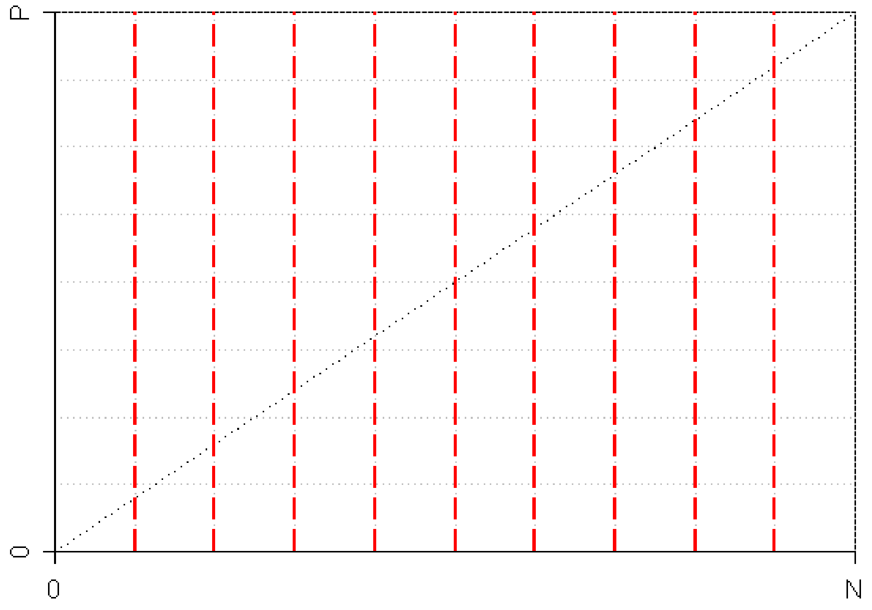
random rules:
predict with coin tosses with fixed probability

opposite rule:
all negative and no positive examples are covered

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 $h_n = -n$

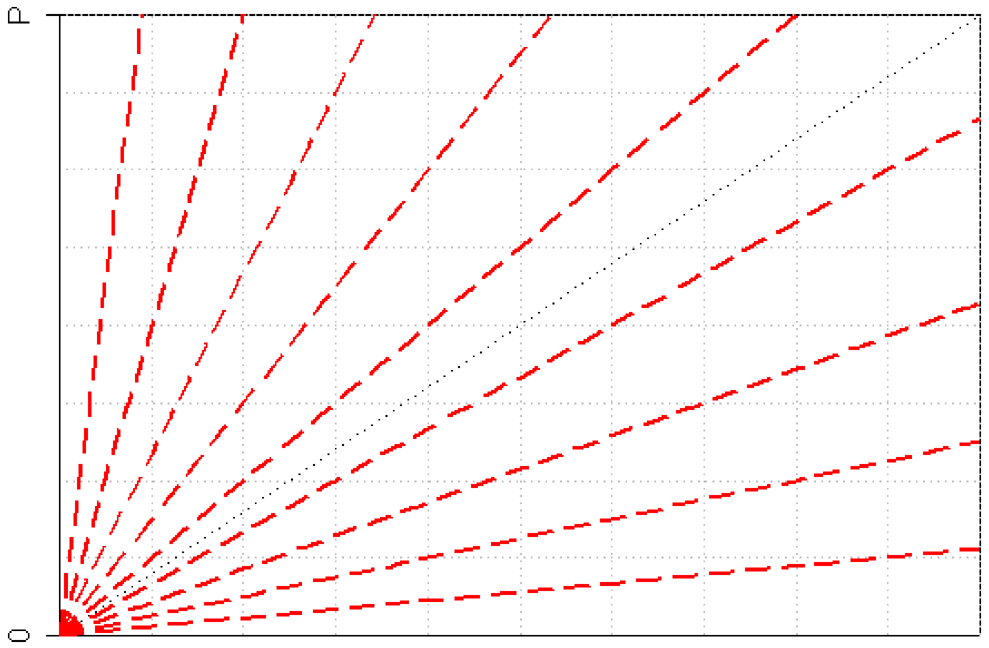




Precision

- *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**

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

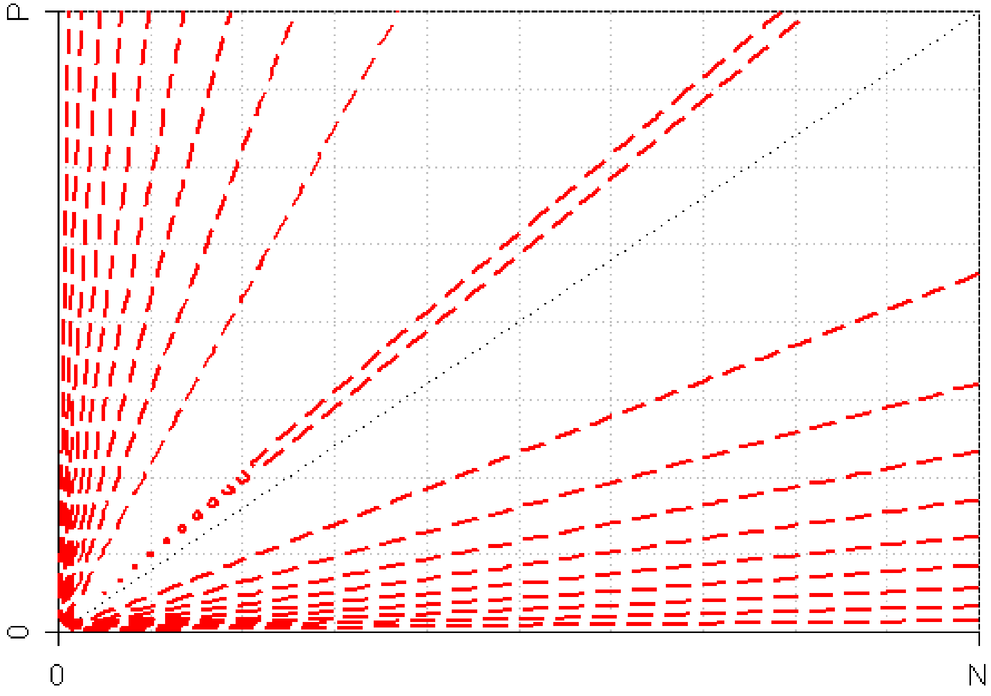


Entropy and Gini Index

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

$$h_{Gini} = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2 \simeq \frac{pn}{(p+n)^2}$$

- *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

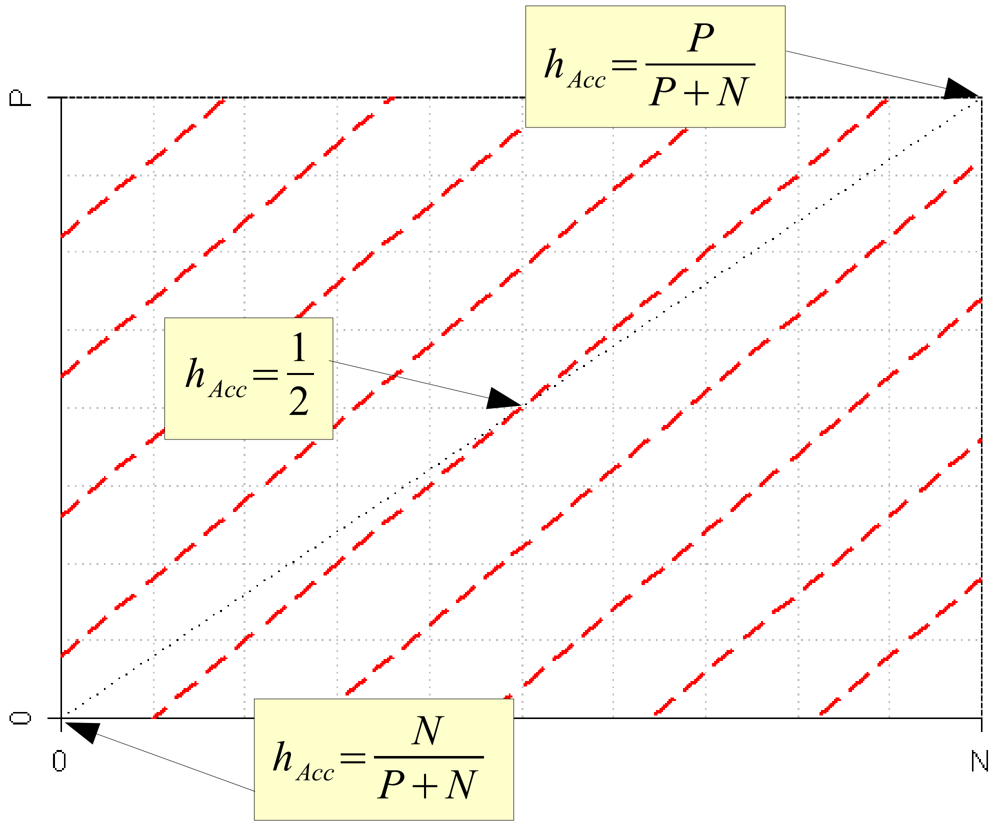


Accuracy

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

Why are they equivalent?

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



Weighted Relative Accuracy

- Two Basic ideas:
  - **Precision Gain:** compare precision to precision of a rule that classifies all examples as positive

$$\frac{p}{p+n} - \frac{P}{P+N}$$

  - **Coverage:** Multiply with the percentage of covered examples

$$\frac{p+n}{P+N}$$

- Resulting formula:

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

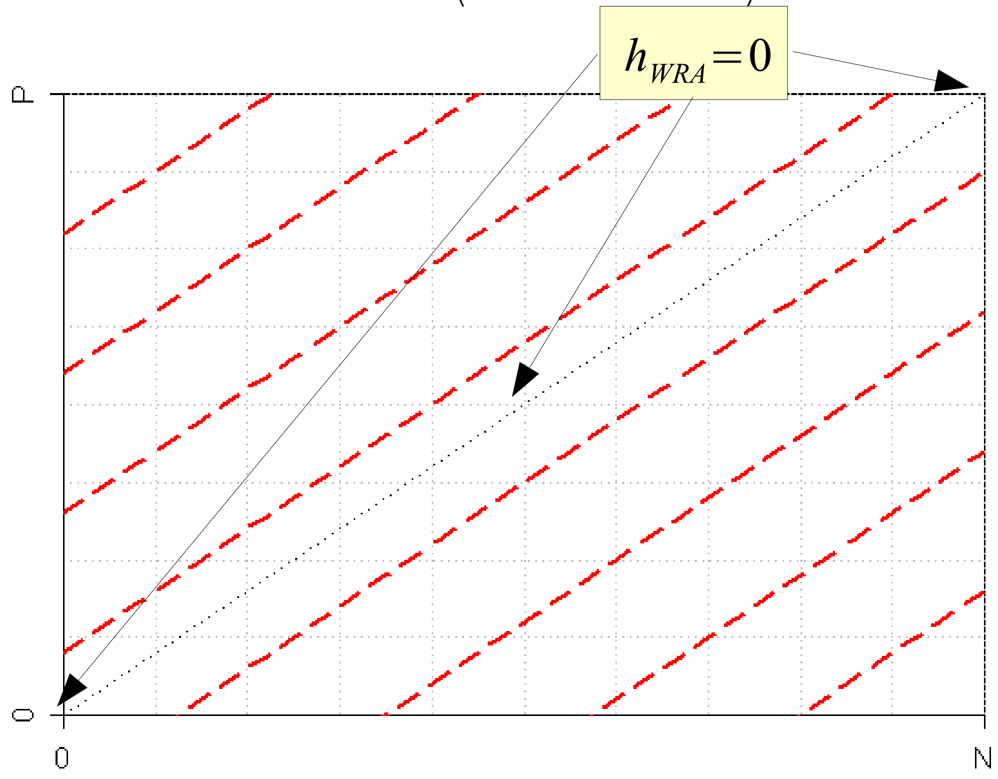
  - one can show that sorts rules in exactly the same way as

$$h_{WRA}' = \frac{p}{P} - \frac{n}{N}$$

Weighted relative accuracy

- *basic idea:*
  compte the distance from the diagonal (i.e., from random rules)
- *effects:*
  - isometrics are parallel to diagonal
  - covering $x\%$ of the positive examples is considered to be as good as not covering $x\%$ of the negative examples
- typically **over-generalizes**

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



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 $c/(1\text{-}c)$ negative examples

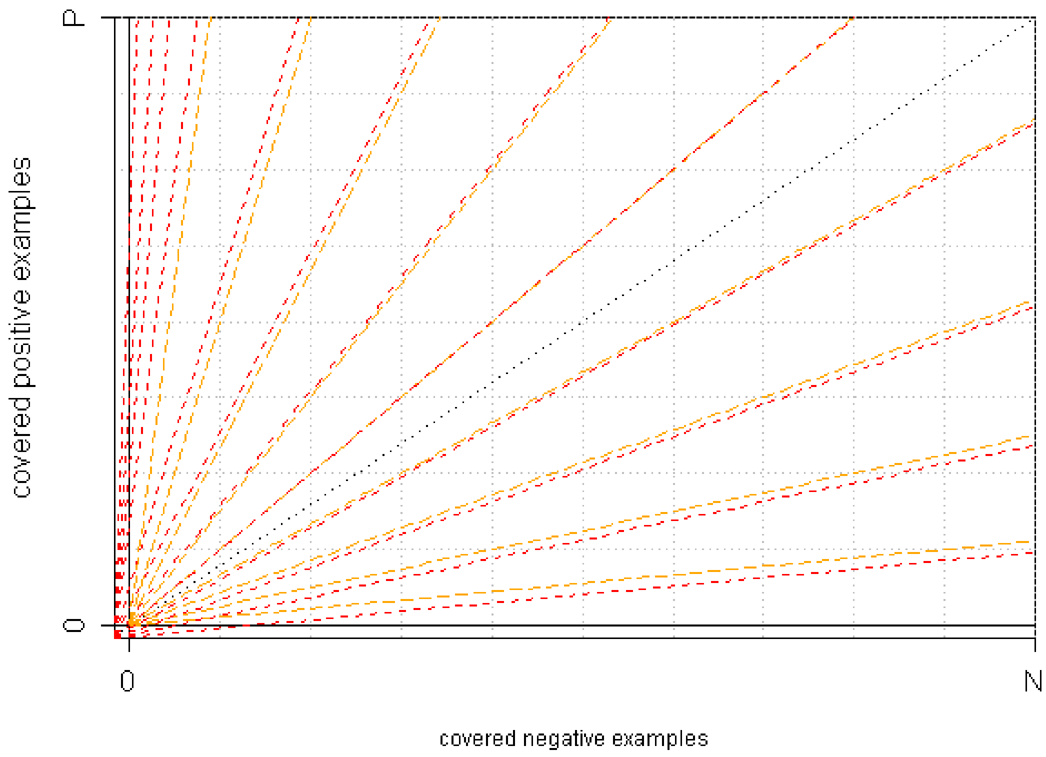
c	*measure*
½	accuracy
$N/(P+N)$	weighted relative accuracy
0	excluding negatives at all costs
1	covering positives at all costs

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

Laplace-Estimate

- *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

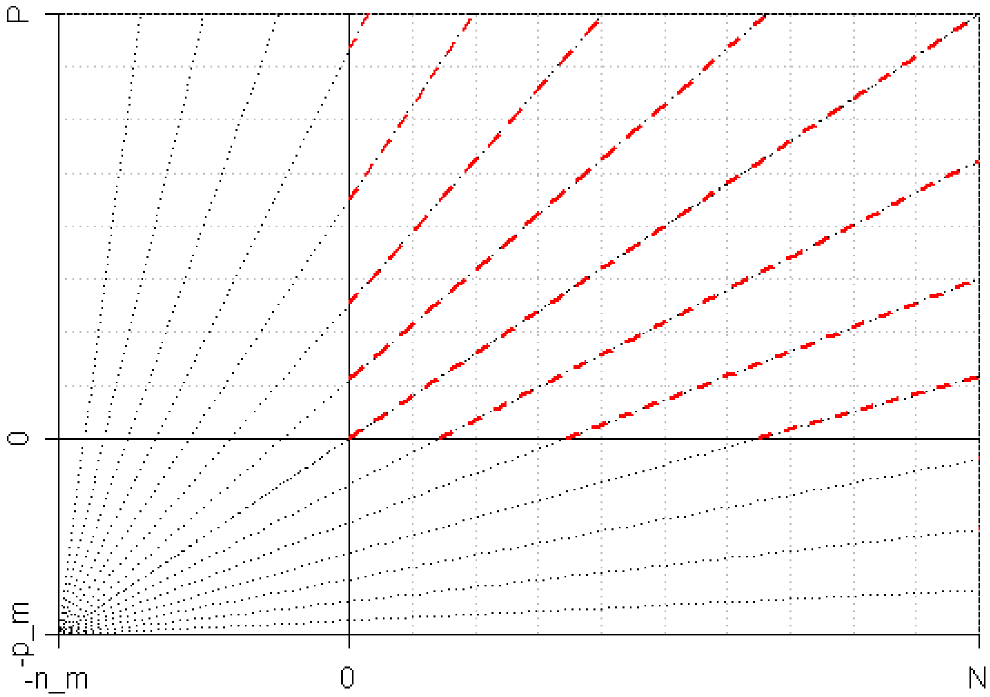
$$h_{Lap} = \frac{p+1}{(p+1)+(n+1)} = \frac{p+1}{p+n+2}$$



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**

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



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 for trading off between precision and the cost metric

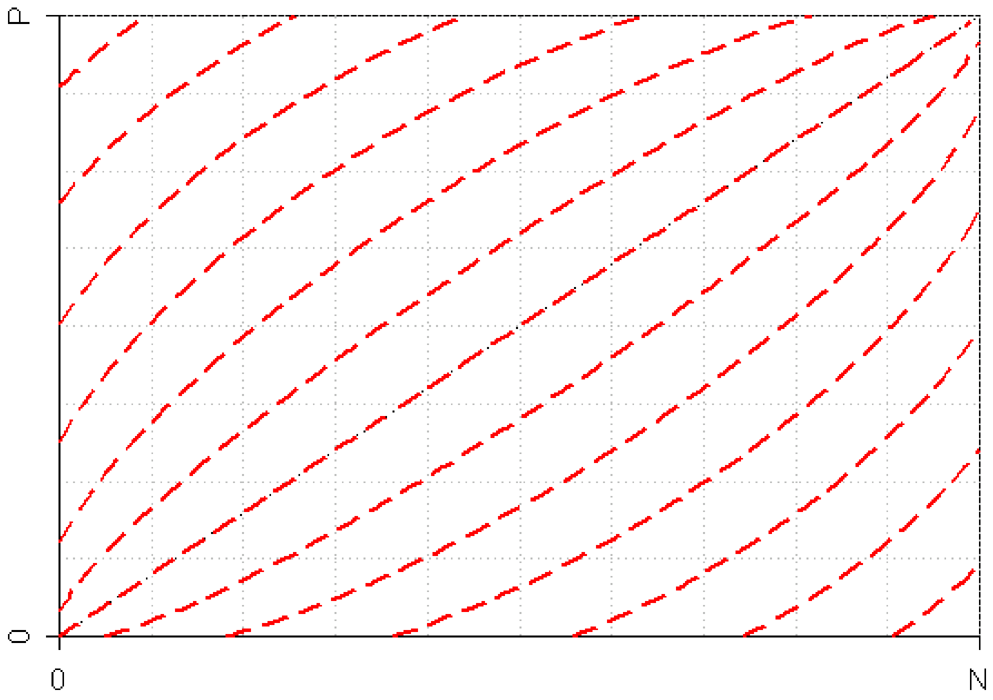
Summary of Linear Rule Learning Heuristics

- 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"*

Non-Linear Isometrics – Correlation

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

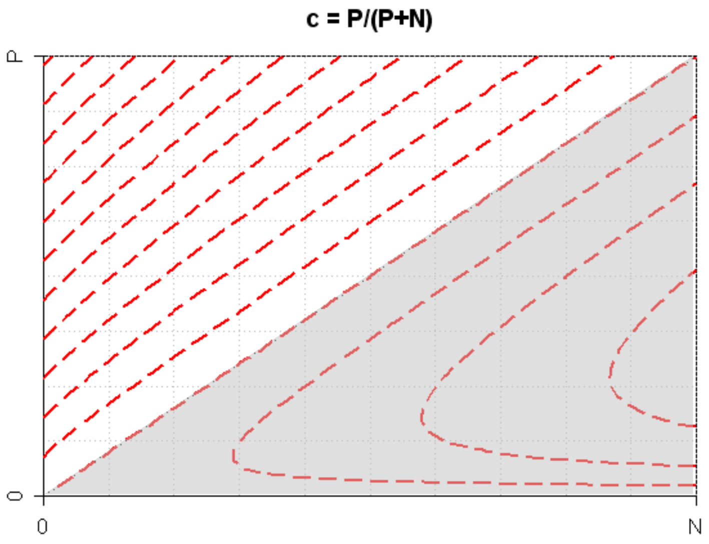
- *basic idea:*
  measure correlation coefficient of predictions with target
- *effects:*
  - non-linear isometrics
  - in comparison to WRA
    - prefers rules near the edges
    - steepness of connection of intersections with edges increases
  - equivalent to χ^2

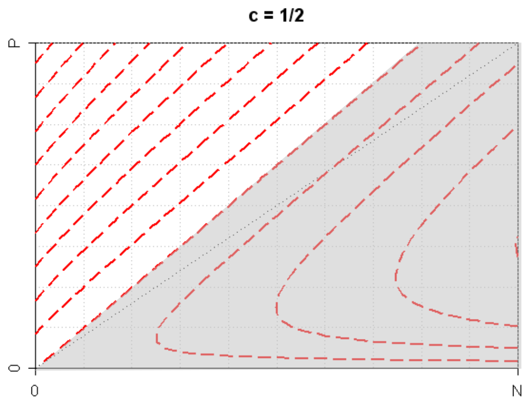


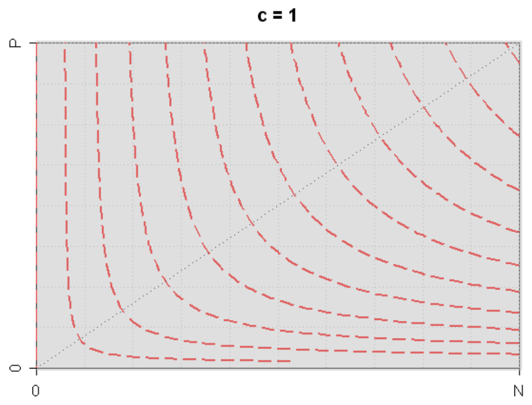
Non-Linear Isometrics – Foil's Information Gain

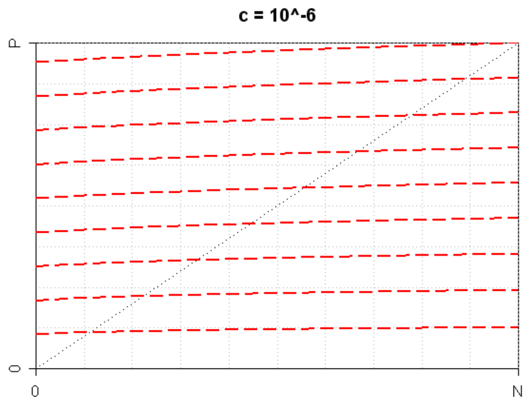
$$h_{foil} = -p(\log_2 c - \log_2 \frac{p}{p+n})$$

(c is the precision of the parent rule)





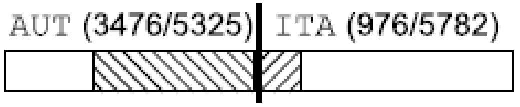




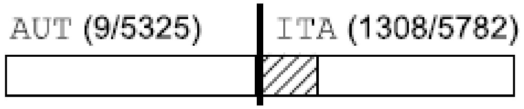
Application Study: Life Course Analysis

- Data:
  - Fertility and Family Survey 1995/96 for Italians and Austrians
  - Features based on general descriptors and variables that describes whether (quantum), at which age (timing) and in what order (sequencing) typical life course events have occurred.
- Objective:
  - Find subgroups that capture typical life courses for either country
- Examples:

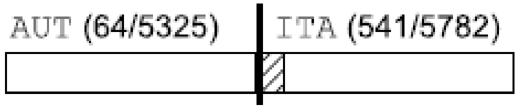
```
IF   LeftHome < Marriage
THEN AUT
```



```
IF   Union = Marriage
 AND Education <= 14
THEN ITA
```

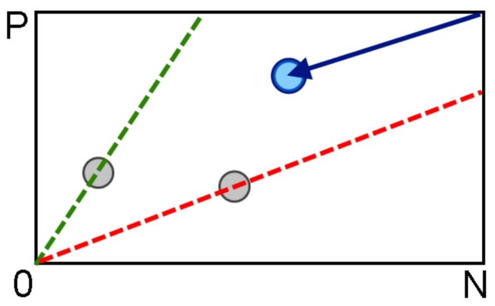


```
IF   Union = Marriage
 AND Education >= 22
THEN ITA
```

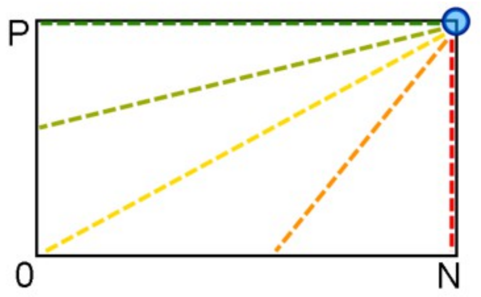


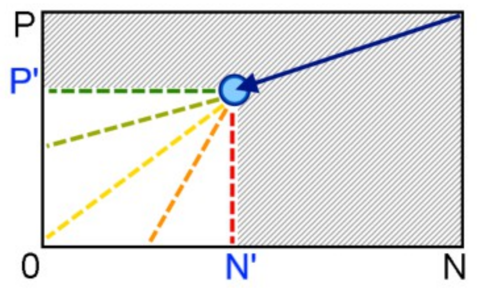
Inverted Heuristics – Motivation

- While the search proceeds top-down
- the evaluation of refinements happens from the point of view of the origin (bottom-up)



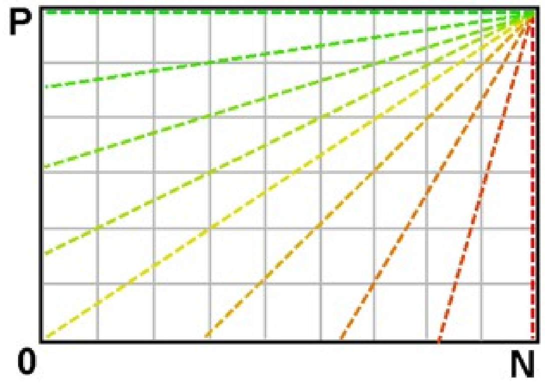
- Instead, we want to evaluate the refinement from the point of view of the predecessor

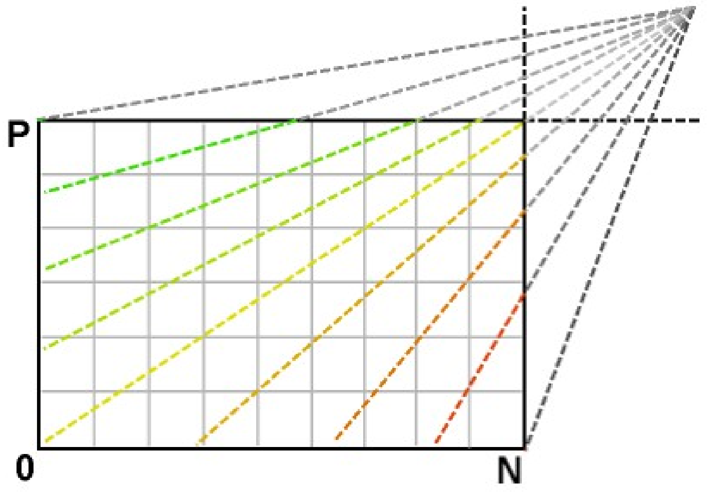




Inverted Heuristics

- Many heuristics can be “inverted” by replacing changing their angle point from the origin to the current rule





$$h'_{precision}(p,n,P,N)=\frac{N-n}{(P+N)-(p+n)}$$

$$h'_{m-Estimate}(p,n,P,N)=\frac{N-n+m\cdot\frac{P}{P+N}}{(P+N)-(p+n-m)}$$

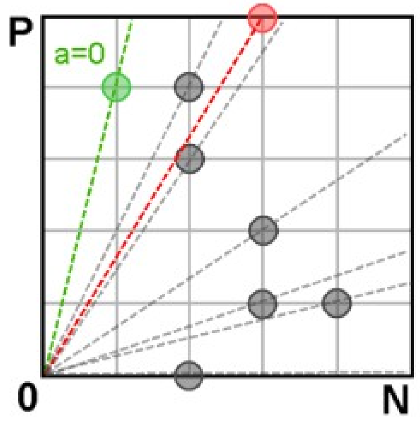
- **Note:** not all heuristics can be inverted
  - e.g. WRA is invariant w.r.t. inversion (because of symmetry)

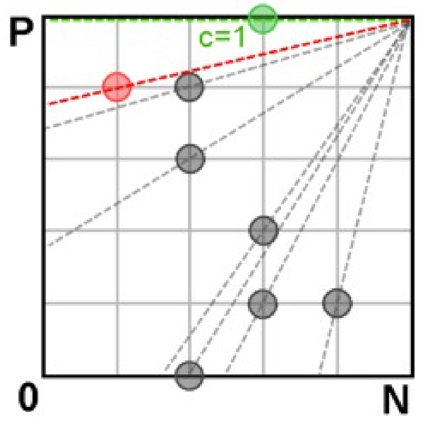
Inverted Heuristics – Example

First refinement step in small example dataset

- 4 Attributes, 10 data points, binary-class

a	b	c	d	*C*
0	1	1	1	+
0	1	1	1	+
0	0	1	0	-
1	1	1	0	-
1	0	0	1	-
0	1	1	0	+
0	0	1	1	+
1	1	1	0	-
1	0	1	1	+
1	0	0	1	-





Inverted heuristic function (right image) selects preferable refinement condition $c=1$ with coverage of $(p, n)=(5,3)$

Inverted Heuristics – Rule Length

- Inverted Heuristics tend to learn longer rules
  - If there are conditions that can be added without decreasing coverage, inverted heuristics will add them first (before adding discriminative conditions)
- Typical intuition:
  - long rules are less understandable, therefore short rules are preferable
  - short rules are more general, therefore (statistically) more reliable
- Should shorter rules be preferred?
  - Claim: Not necessarily, because longer rules may capture more information about the object
  - Related to discriminative rules vs. characteristic rules
  - Open question...

Example: Mushroom dataset

- The best three rules learned with conventional heuristics

```
IF odor = f          THEN poisonous (2160,0)
IF gill-color = b    THEN poisonous (1152,0)
IF odor = p          THEN poisonous (256,0)
```

- The best three rules learned with inverted heuristics

```
IF veil-color = w, gill-spacing = c, bruises? = f,
   ring-number = o, stalk-surface-above-ring = k
THEN poisonous (2192,0)
IF veil-color = w, gill-spacing = c, gill-size = n,
   population = v, stalk-shape = t
THEN poisonous (864,0)
IF stalk-color-below-ring = w, ring-type = p,
   stalk-color-above-ring = w, ring-number = o,
   cap-surface = s, stalk-root = b, gill-spacing = c
THEN poisonous (336)
```

From Descriptive Rules to Predictive Theories

- Descriptive Learning
  - Focus on discovering patterns that describe (parts of) the data
- Predictive Learning
  - Focus on finding patterns that allow to make predictions about the data

From Descriptive Rules to Predictive Rule Sets

- **Rule Diversity and Completeness:**
  - We need to be able to make a prediction for every possible instance
- **Predictive Evaluation:**
  - It is important how well rules are able to predict the dependent variable on new data

Concept Learning

- Given:
  - Positive Examples E^+
    - examples for the concept to learn (e.g., days with golf)
  - Negative Examples E^-
    - counter-examples for the concept (e.g., days without golf)
  - Hypothesis Space H
    - a (possibly infinite) set of candidate hypotheses
    - e.g., rules, rule sets, decision trees, linear functions, neural networks, ...

- Find:
  - Find the target hypothesis $h \in H$
  - the target hypothesis is the concept that was used (or could have been used) to generate the training examples

A Learned Rule Set

```
IF  E=primary    AND S=male   AND M=married  AND C=no     THEN yes
IF  E=university AND S=female AND M=divorced AND C=no     THEN yes
IF  E=university AND S=female AND M=married  AND C=yes    THEN yes
IF  E=university AND S=female AND M=single   AND C=no     THEN yes
IF  E=secondary  AND S=female AND M=divorced AND C=no     THEN yes
IF  E=secondary  AND S=female AND M=single   AND C=yes    THEN yes
IF  E=secondary  AND S=male   AND M=married  AND C=yes    THEN yes
IF  E=primary    AND S=female AND M=married  AND C=no     THEN yes
IF  E=secondary  AND S=male   AND M=divorced AND C=no     THEN yes
ELSE no
```

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

$\rightarrow$ it must be part of the version space

- but it does not generalize to new examples!

A Better Solution

```
IF Marital = married                           THEN   yes
IF Marital = single   AND Sex = female         THEN yes
IF Marital = divorced AND Children = no        THEN yes
ELSE no
```

Separate-and-Conquer Rule Learning

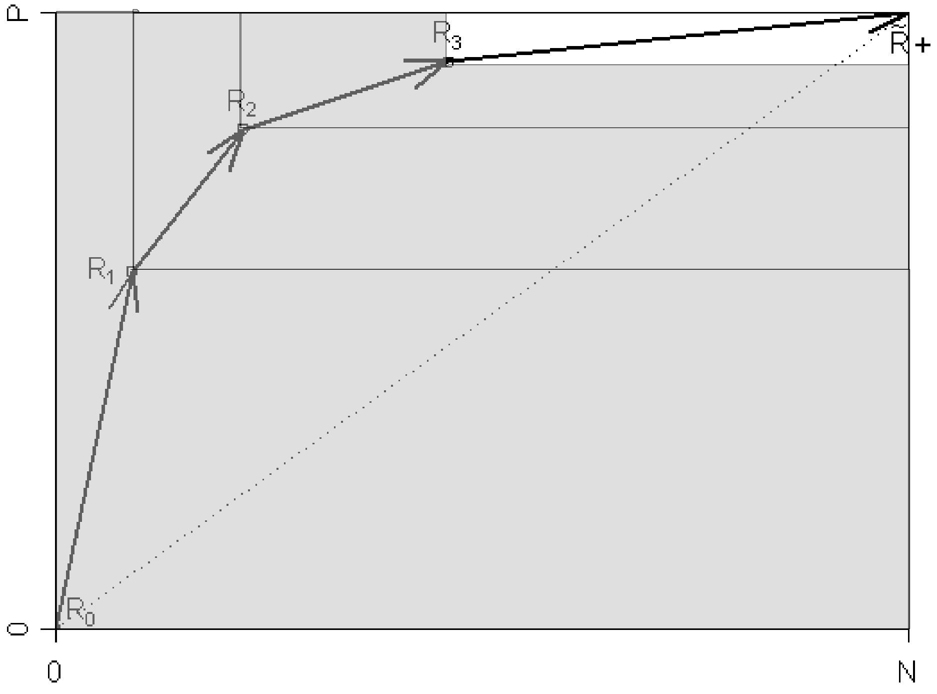
- Learn a set of rules, one rule after the other using greedy covering

```
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:  goto 2.
```

- One of the oldest family of learning algorithms
- Different learners differ in how they find a single rule
- Completeness and consistency requirements are typically loosened

Covering Strategy

- Covering or Separate-and-Conquer rule learning learning algorithms learn one rule at a time
  - and then removes the examples covered by this rule
- 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)



Which Heuristic is Best?

- There have been many proposals for different heuristics
  - and many different justifications for these proposals
  - some measures perform better on some datasets, others on other datasets

- Large-Scale Empirical Comparison:
  - 27 training datasets
    - on which parameters of the heuristics were tuned)
  - 30 independent datasets
    - which were not seen during optimization
  - Goals:
    - see which heuristics perform best
    - determine good parameter values for parametrized functions

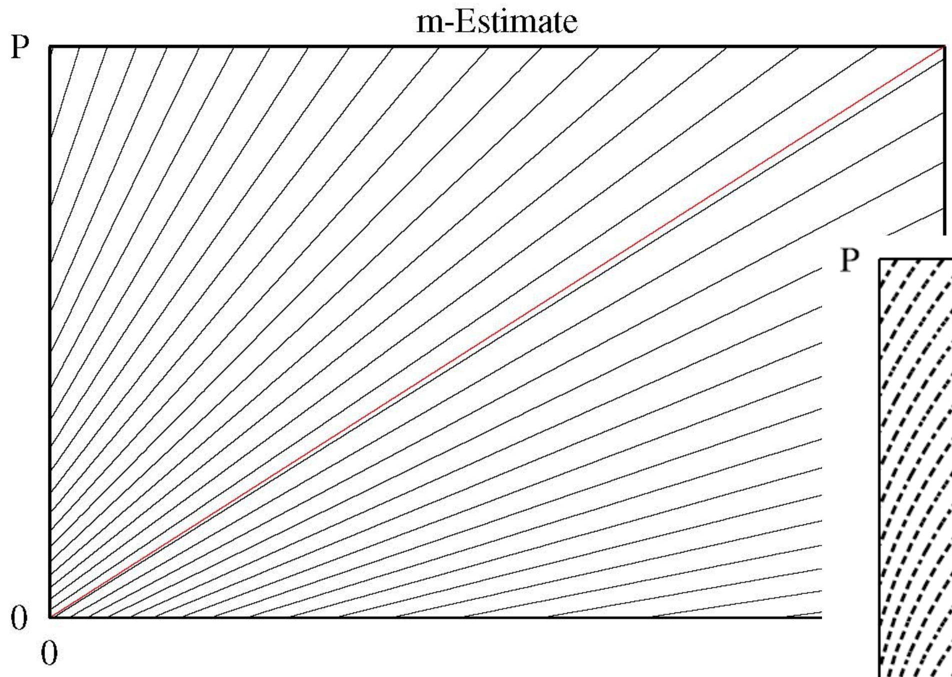
Empirical Comparison of Different Heuristics

	Training Datasets		Independent Datasets	
Heuristic	Accuracy	# Conditions	Accuracy	#Conditions
Ripper (JRip)	84,96	16,93	78,97	12,20
Relative Cost Metric (c =0.342)	85,63	26,11	78,87	25,30
m-Estimate (m = 22.466)	85,87	48,26	78,67	46,33
Correlation	83,68	37,48	77,54	47,33
Laplace	82,28	91,81	76,87	117,00
Precision	82,36	101,63	76,22	128,37
Linear Cost Metric (c = 0.437)	82,68	106,30	76,07	122,87
WRA	82,87	14,22	75,82	12,00
Accuracy	82,24	85,93	75,65	99,13

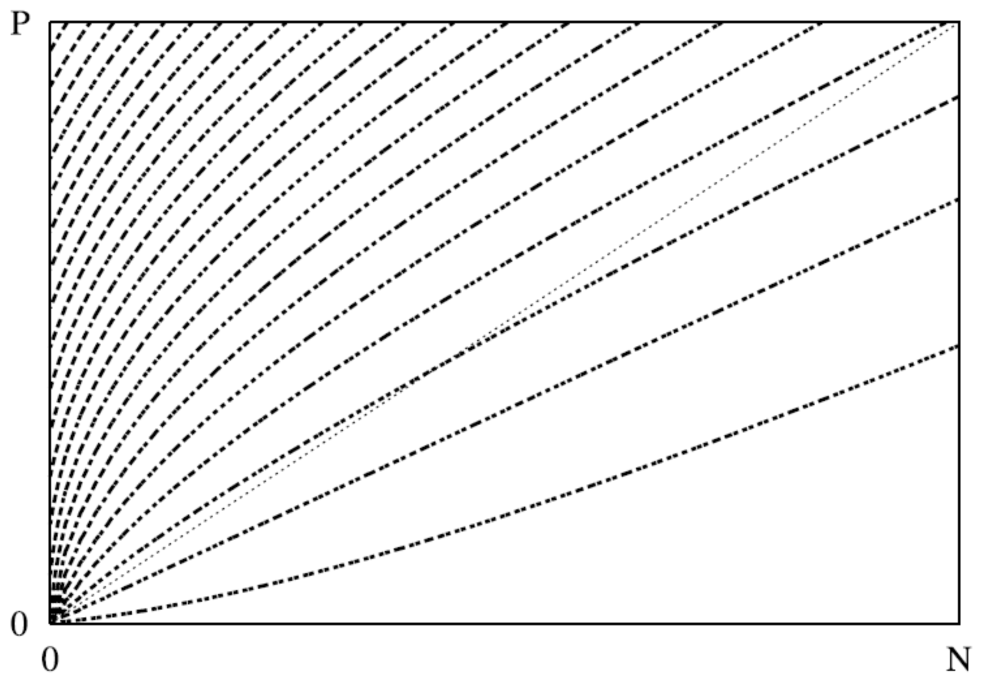
- Ripper is best, but uses pruning (the others don't)
- the optimized parameters for the m-estimate and the relative cost metric perform better than all other heuristics
  - also on the 30 datasets on which they were not optimized
- some heuristics clearly overfit (bad performance with large rules)
- WRA over-generalizes (bad performance with small rules)

Best Parameter Settings

for m-estimate: $m = 22.5$

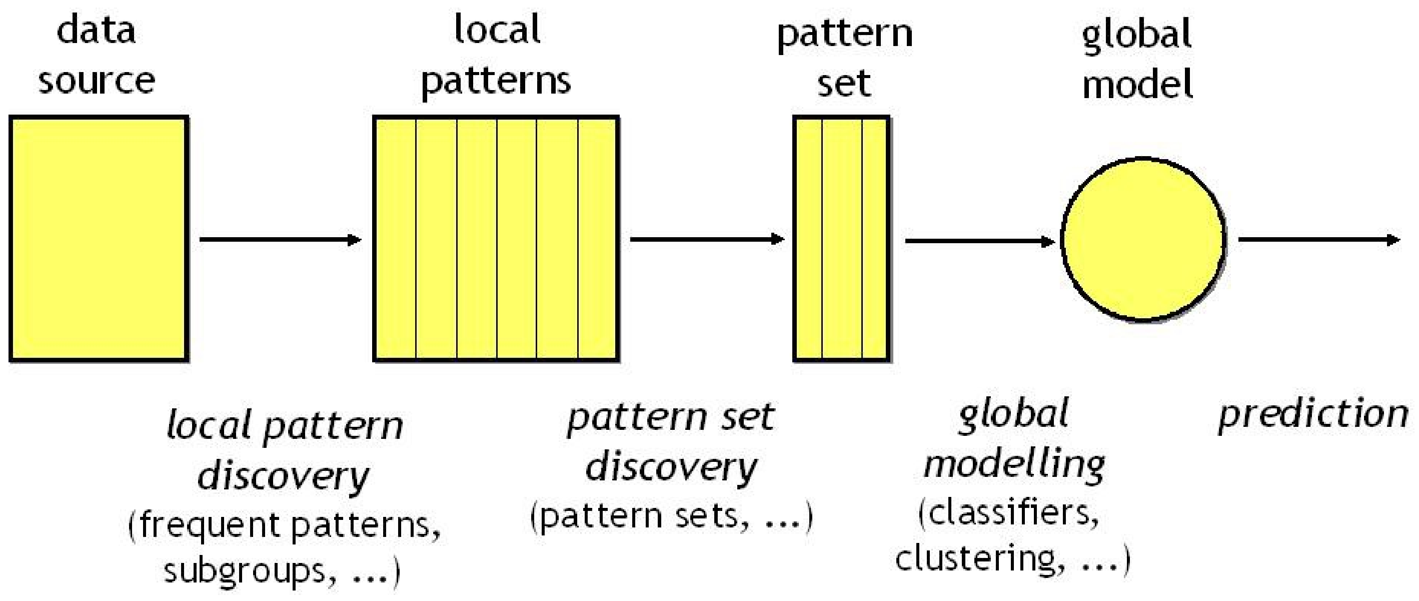


meta-learned with a NN



LeGo Approach to Rule Learning

- General framework for aggregating local patterns to global models
  - key idea: use frequently occurring patterns are features

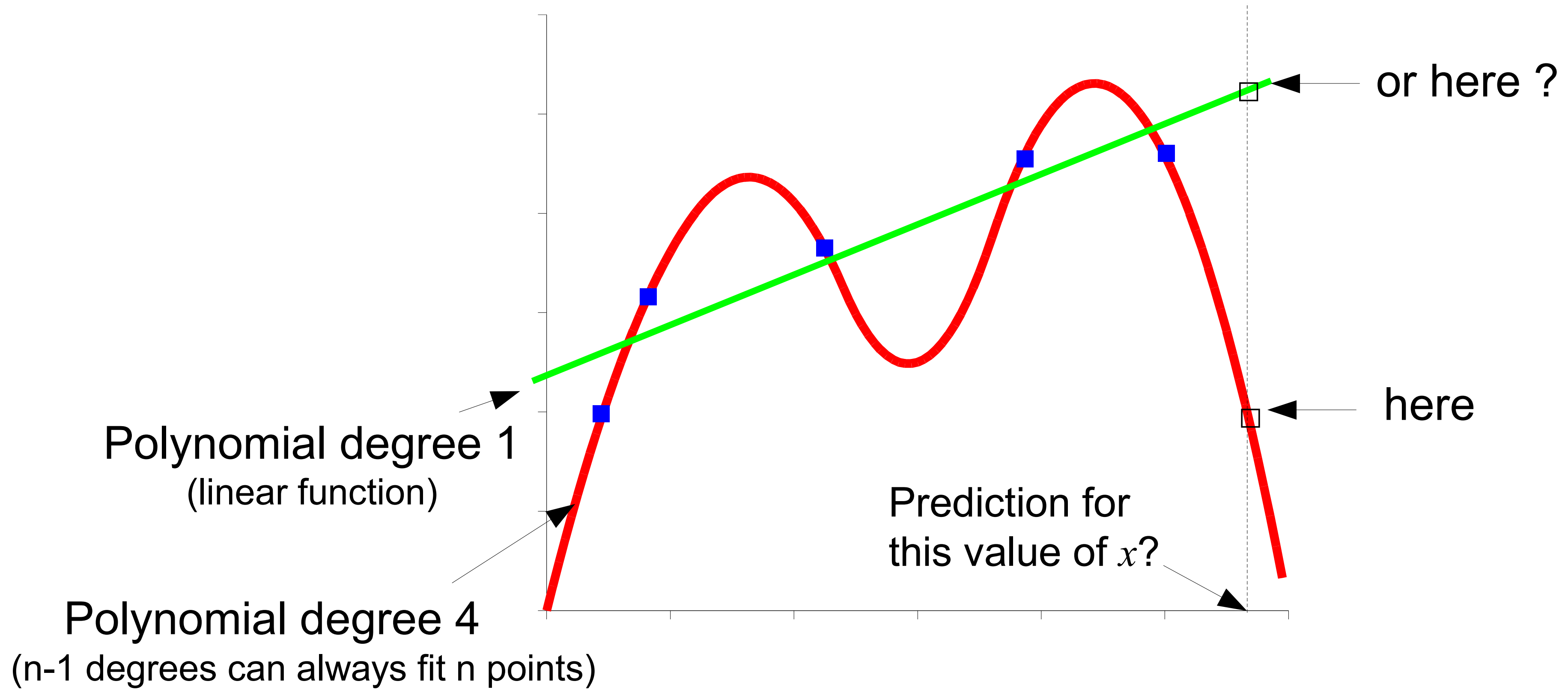


Overfitting

- 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

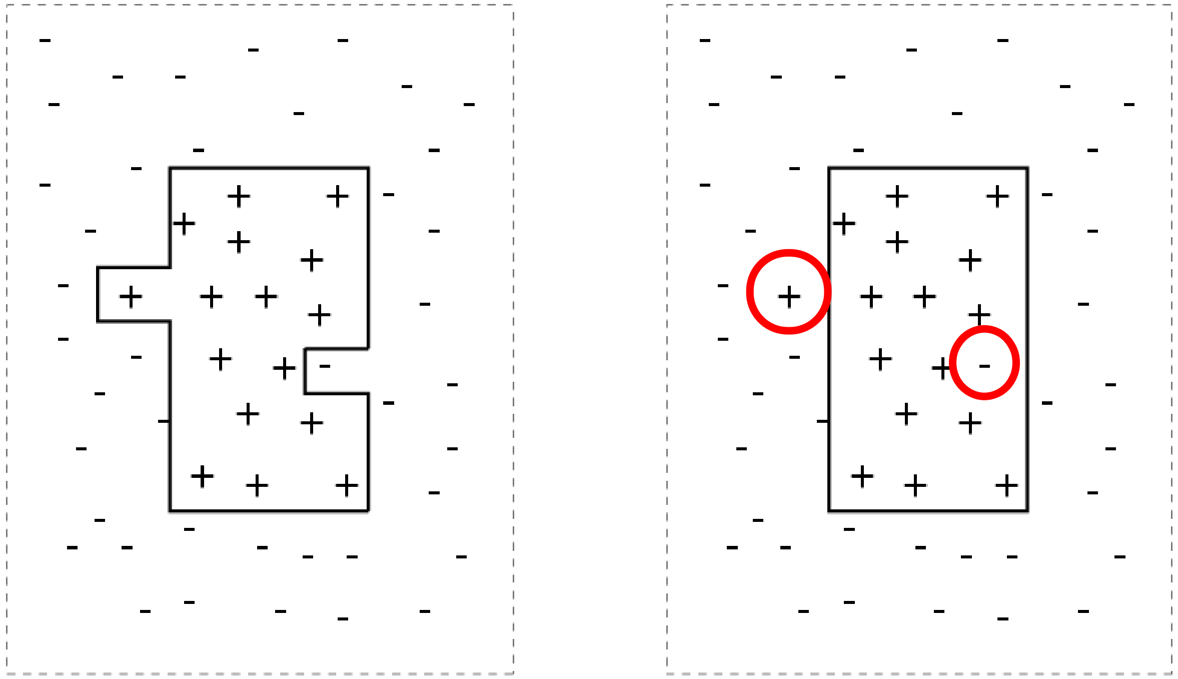


Overfitting Avoidance

- A perfect fit to the data is not always a good idea
  - data could be imprecise
    - e.g., random noise
  - the hypothesis space may be inadequate
    - a perfect fit to the data might not even be possible
    - or it may be possible but with bad generalization properties (e.g., generating one rule for each training example)

- Thus it is often a good idea to avoid a perfect fit of the data
  - fitting polynomials so that
    - not all points are exactly on the curve
  - learning concepts so that
    - not all positive examples have to be covered by the theory
    - some negative examples may be covered by the theory

Overfitting Avoidance



- learning concepts so that
  - not all positive examples have to be covered by the theory
  - some negative examples may be covered by the theory

Complexity of Concepts

- For simpler concepts there is less danger that they are able to overfit the data
  - for a polynomial of degree n one can choose $n+1$ parameters in order to fit the data points

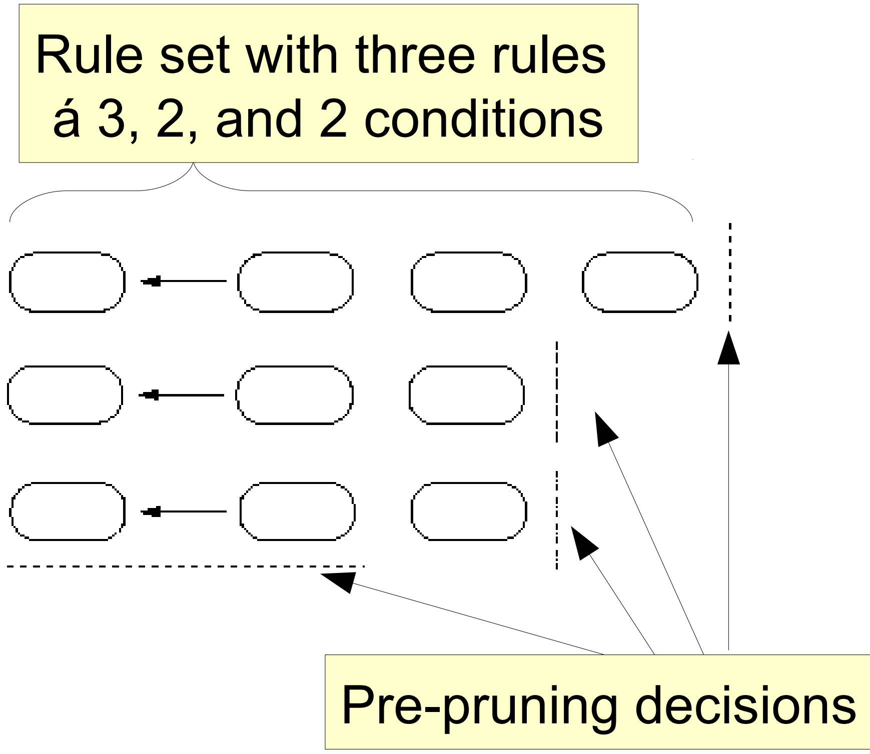
→ many learning algorithms focus on learning simple concepts

  - a short rule that covers many positive examples (but possibly also a few negatives) is often better than a long rule that covers only a few positive examples

- Pruning: Complex rules will be simplified
  - **Pre-Pruning:**
    - during learning
  - **Post-Pruning:**
    - after learning

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



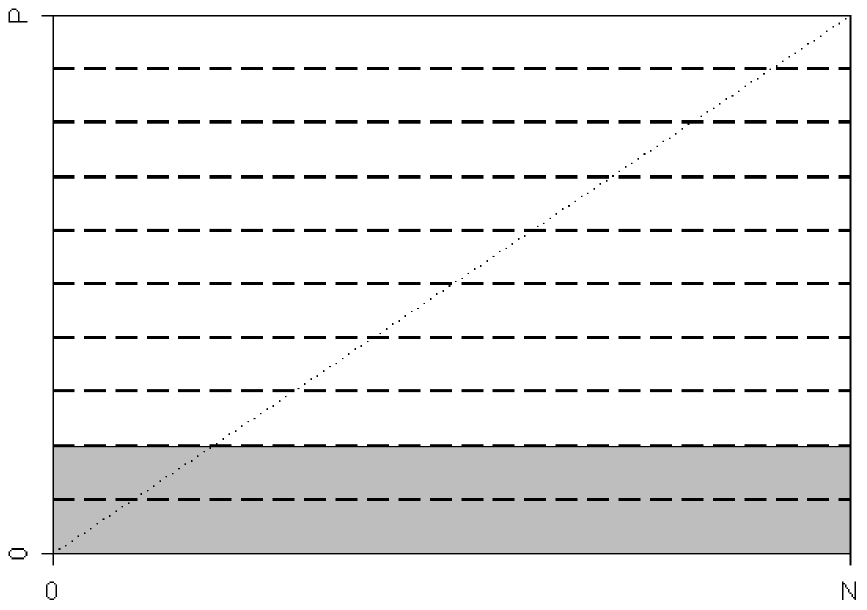
Pre-Pruning Heuristics

1. Thresholding a heuristic value
   - require a certain minimum value of the search heuristic
   - e.g.: Precision > 0.8.
2. 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)
3. 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

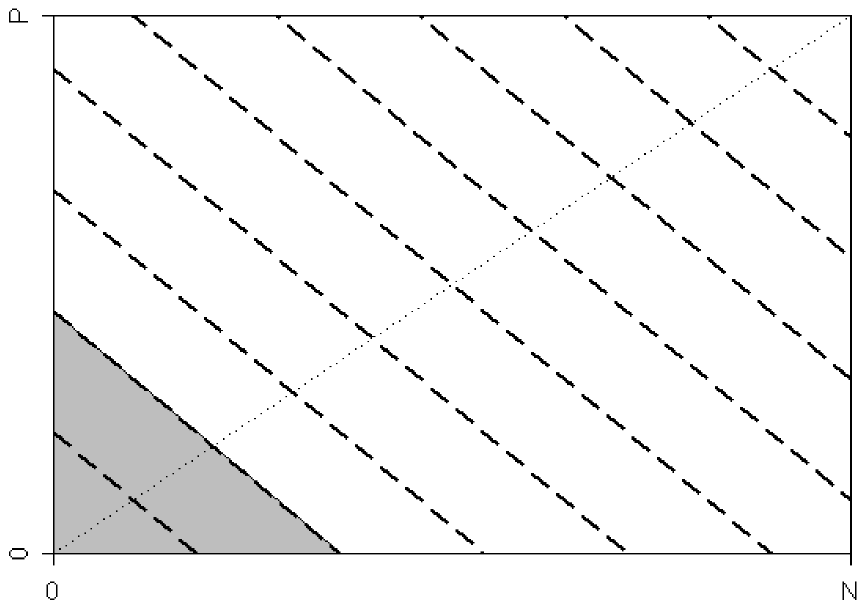
Minimum Coverage Filtering

filter rules that do not cover a minimum number of

positive examples (support)

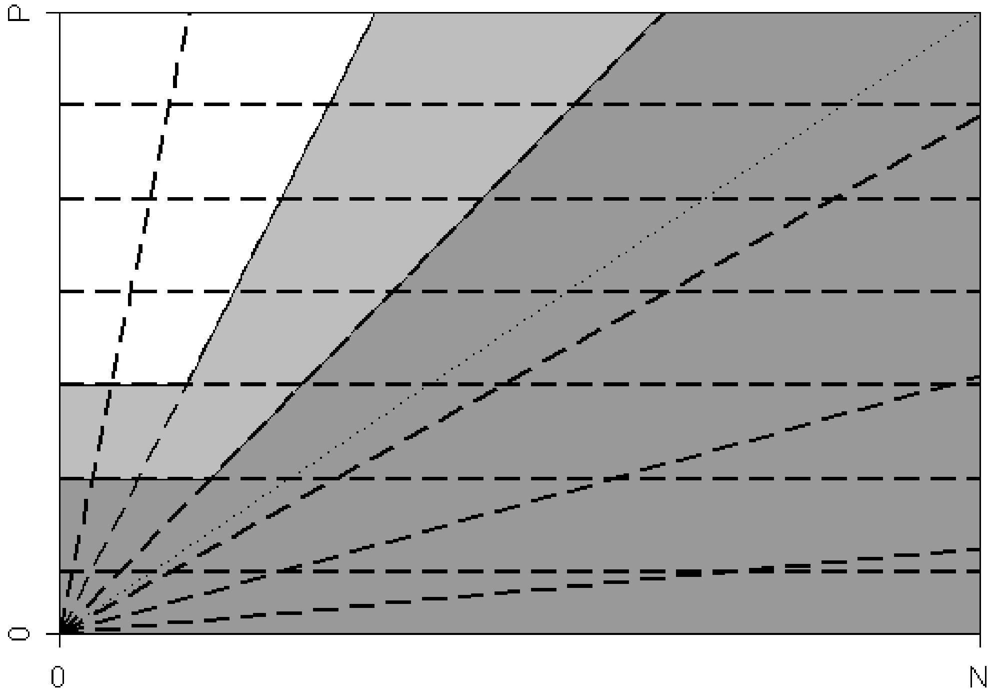


all examples (coverage)



Support/Confidence Filtering

- *basic idea:*
  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



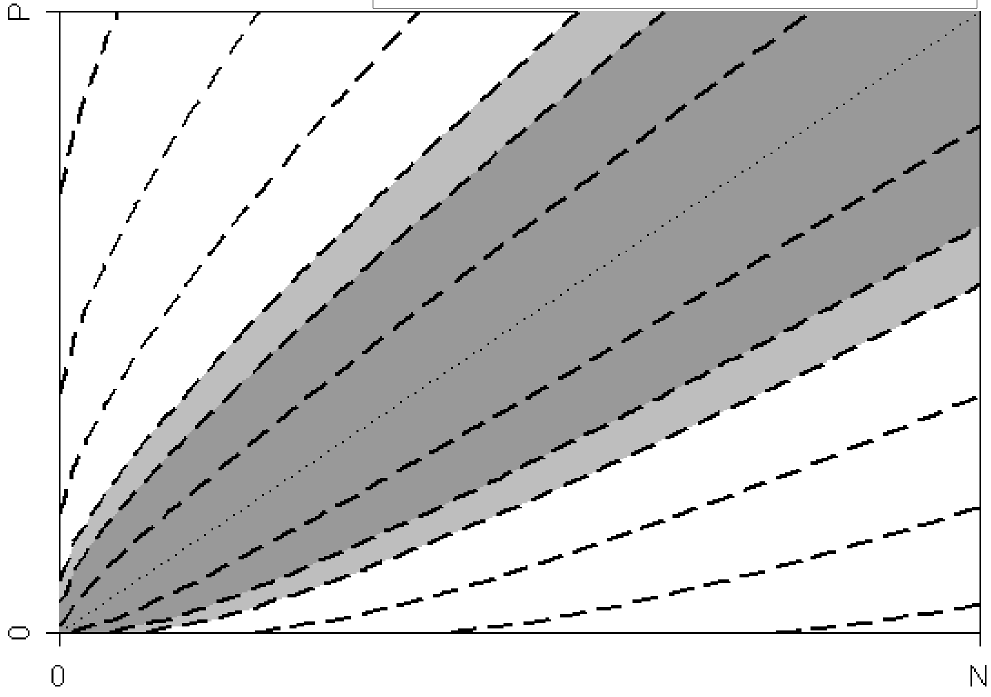
CN2's likelihood ratio statistics

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

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

are the expected number of positive and negative example in the $p+n$ covered examples.

- *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)



MDL-Pruning in Foil

- based on the Minimum Description Length-Principle (MDL)
  - is it more effective to transmit the rule or the covered examples?
    - compute the information contents of the rule (in bits)
    - compute the information contents of the examples (in bits)
    - if the rule needs more bits than the examples it covers, on can directly transmit the examples → no need to further refine the rule
  - Details → (Quinlan, 1990)
- doesn't work all that well
  - if rules have expections (i.e., are inconsistent), the negative examples must be encoded as well
    - they must be transmitted, otherwise the receiver could not reconstruct which examples do not conform to the rule
  - finding a minimal encoding (in the information-theoretic sense) is practically impossible

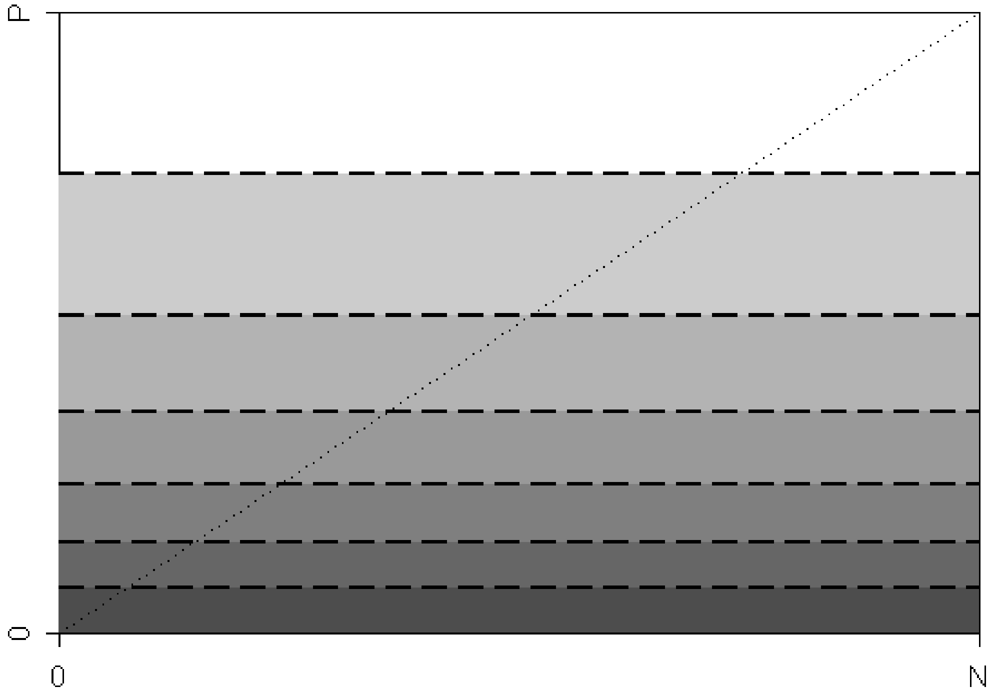
Foil's MDL-based Stopping Criterion

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

costs for transmitting how many examples we have (can be ignored)

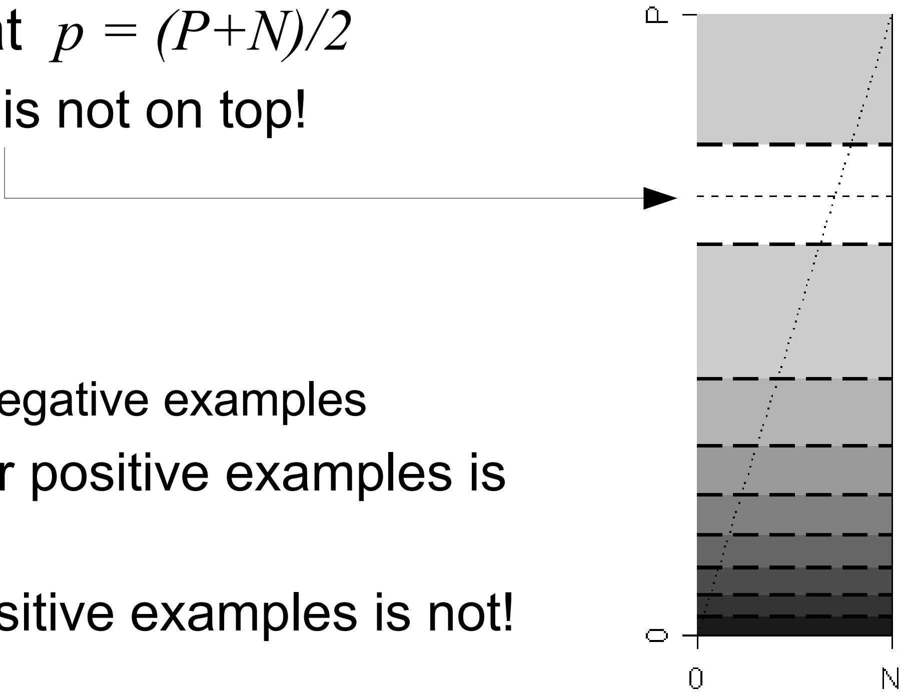
costs for transmitting which of the $P+N$ examples are covered and positive

- *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
  - because function only depends on p



Anomaly of Foil's Stopping criterion

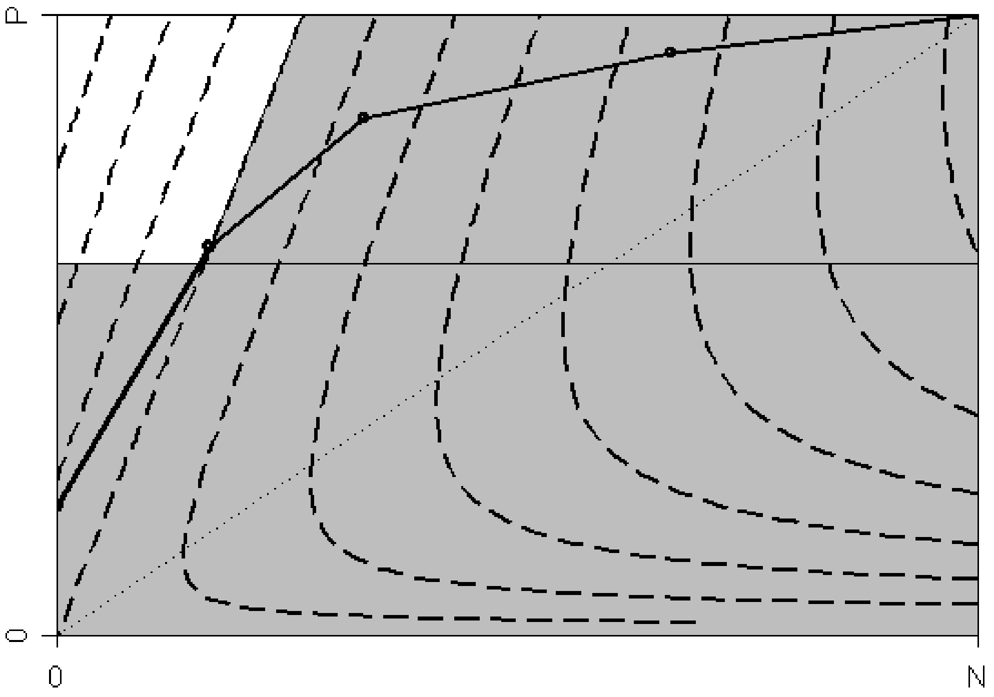
- We have tacitly assumed $N > P$...
- h_{MDL} assumes its maximum at $p = (P+N)/2$
  - thus, for $P > N$, the maximum is not on top!
- there may be rules
  - of equal length
  - covering the same number of negative examples
  - so that the rule covering fewer positive examples is acceptable
  - but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



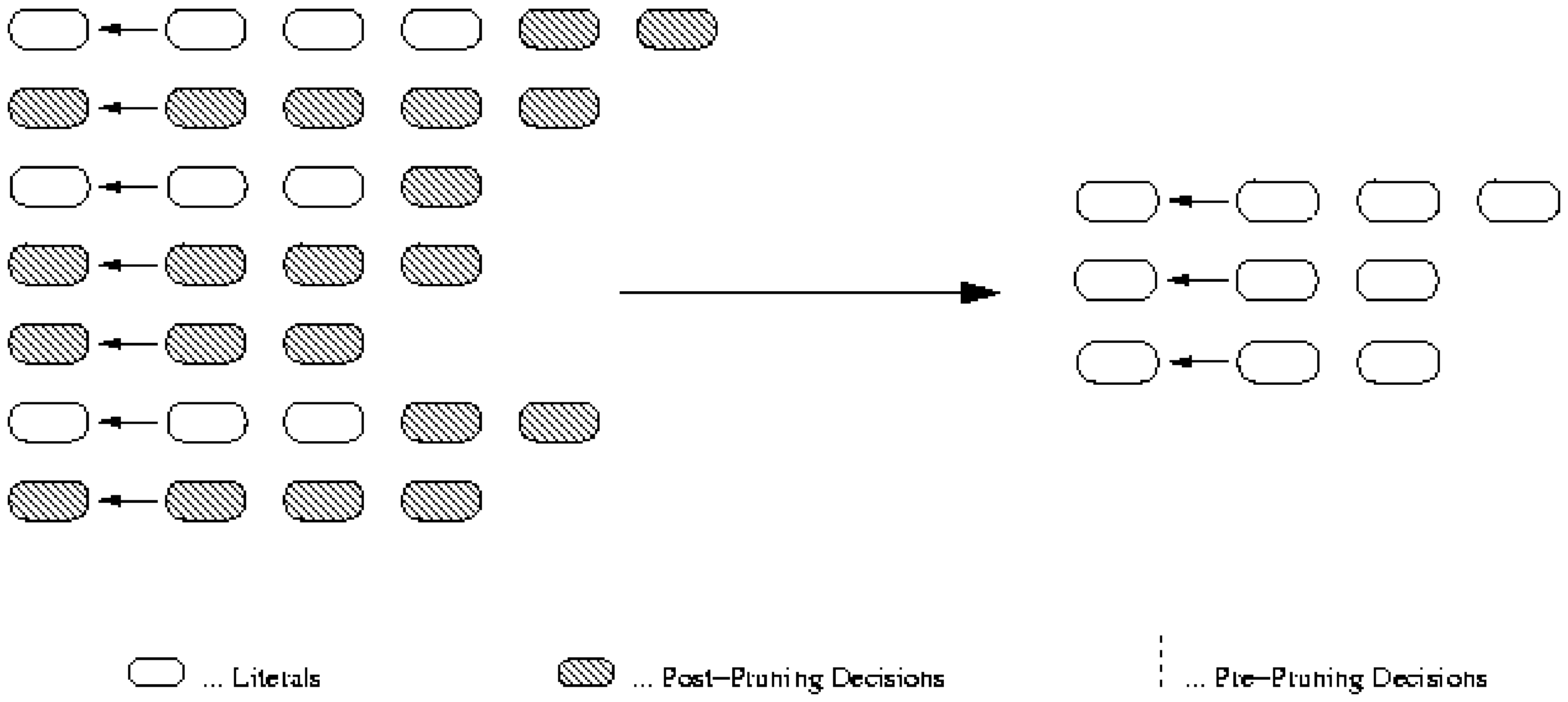
How Foil Works

→ Foil (almost) implements Support/Confidence Filtering

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



Post Pruning



Post-Pruning: Example

```
IF  E=primary     AND S=male   AND M=single   AND C=no  THEN no
IF  E=primary     AND S=male   AND M=single   AND C=yes THEN no
IF  E=primary     AND S=male   AND M=married  AND C=no  THEN yes
IF  E=university  AND S=female AND M=divorced AND C=no  THEN yes
IF  E=university  AND S=female AND M=married  AND C=yes THEN yes
IF  E=secondary   AND S=male   AND M=single   AND C=no  THEN no
IF  E=university  AND S=female AND M=single   AND C=no  THEN yes
IF  E=secondary   AND S=female AND M=divorced AND C=no  THEN yes
IF  E=secondary   AND S=female AND M=single   AND C=yes THEN yes
IF  E=secondary   AND S=male   AND M=married  AND C=yes THEN yes
IF  E=primary     AND S=female AND M=married  AND C=no  THEN yes
IF  E=secondary   AND S=male   AND M=divorced AND C=yes THEN no
IF  E=university  AND S=female AND M=divorced AND C=yes THEN no
IF  E=secondary   AND S=male   AND M=divorced AND C=no  THEN yes
```

Post-Pruning: Example

```
IF   S=male      AND      M=single        THEN  no
IF   M=divorced  AND      C=yes           THEN  no
ELSE                                            yes
```

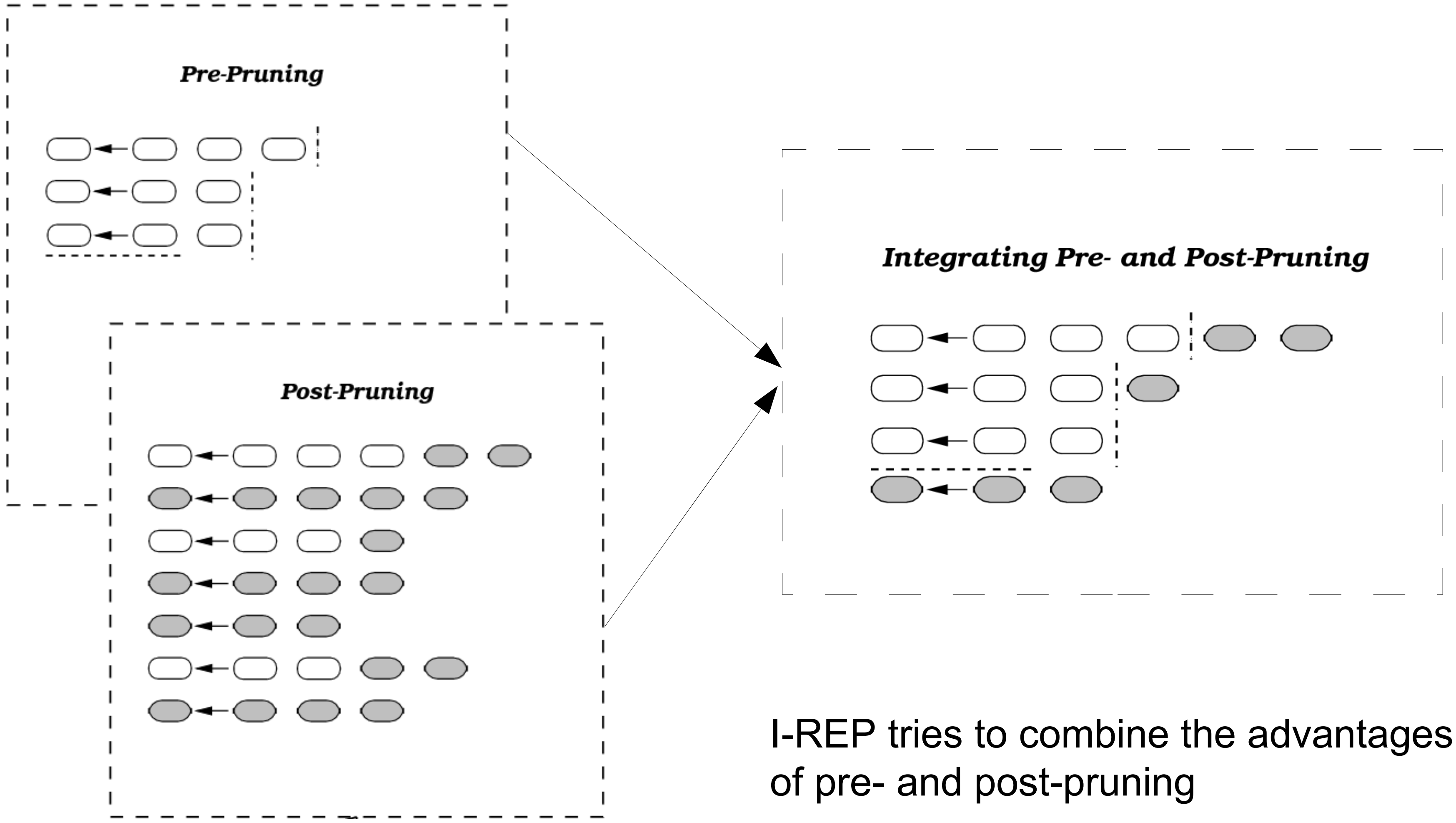
Reduced Error Pruning

- 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

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

Incremental Reduced Error Pruning



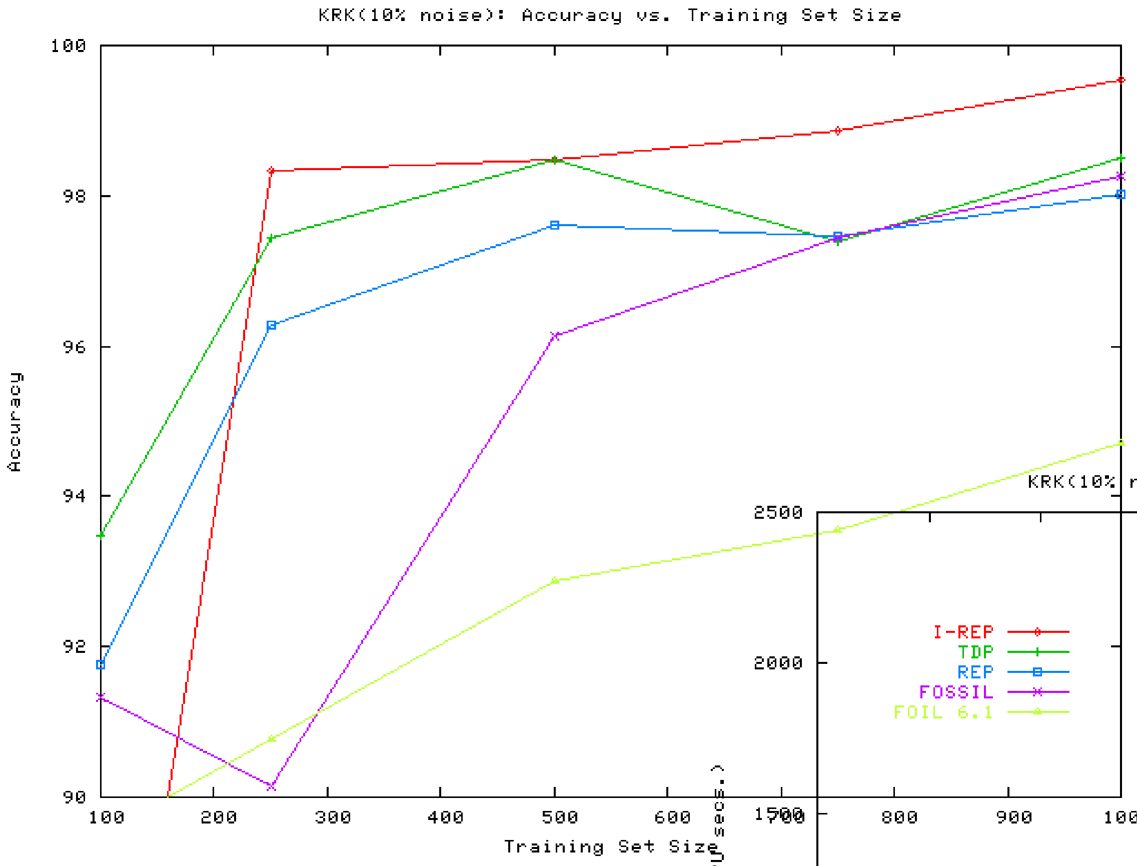
I-REP tries to combine the advantages of pre- and post-pruning

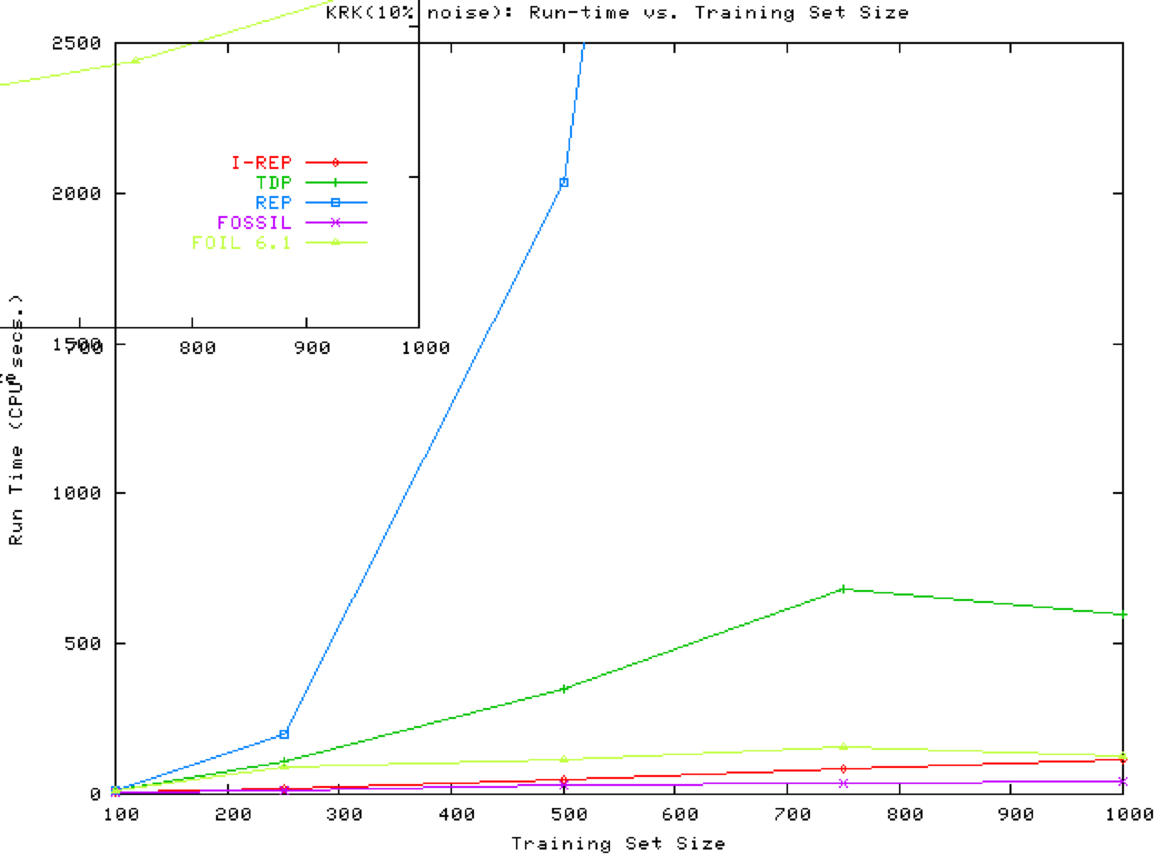
Incremental Reduced Error Pruning

- 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 the rule to the rule set
    - 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 RIPPER rule learner (Cohen, 1995)
  - JRip in Weka





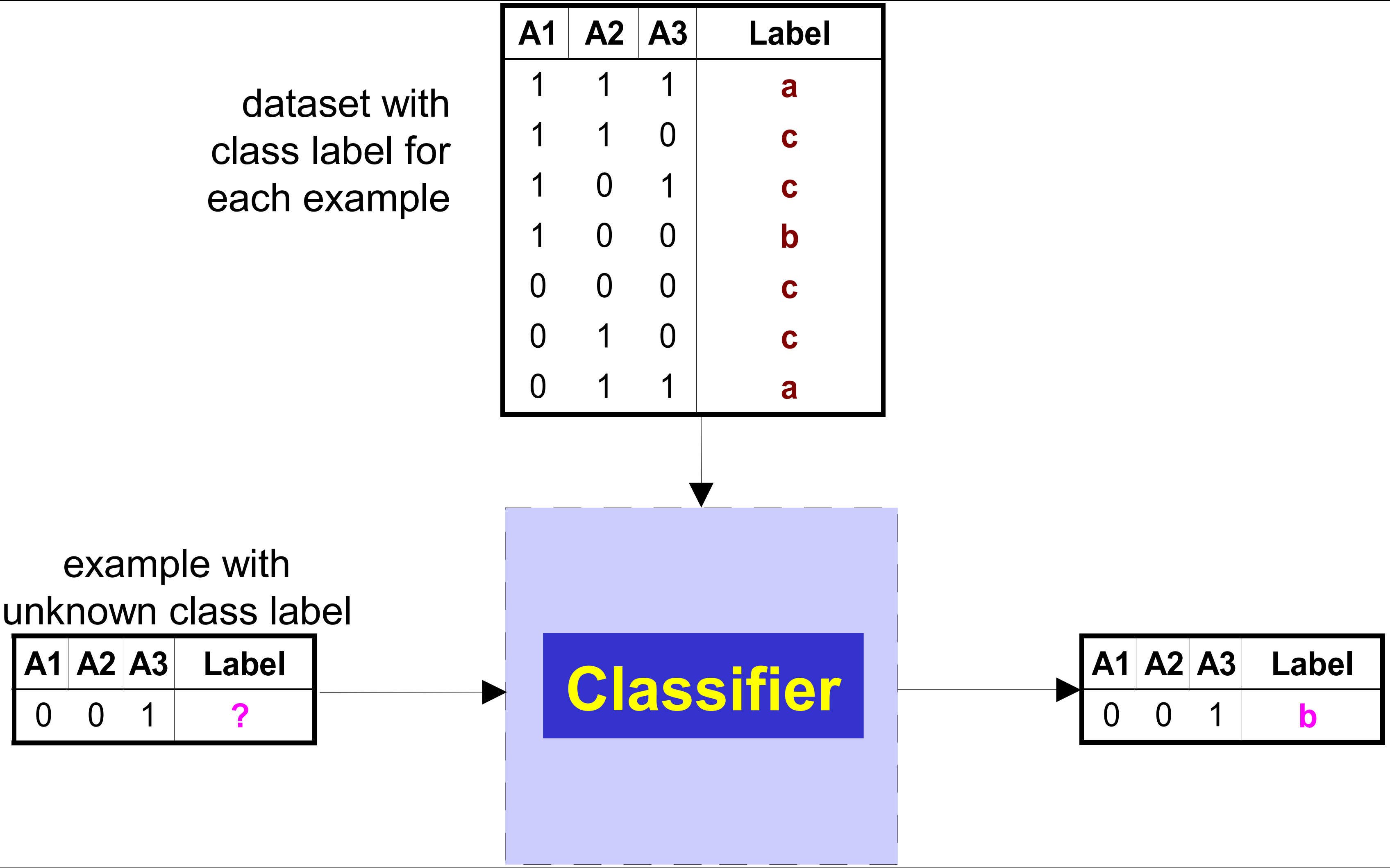
Empirical comparison of accuracy and run-time of various pruning algorithms on a dataset with 10% noise

Multi-Class Classification

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

Property of Interest
(“class variable”)

Learning Multiclass Classifiers



dataset with
class label for
each example

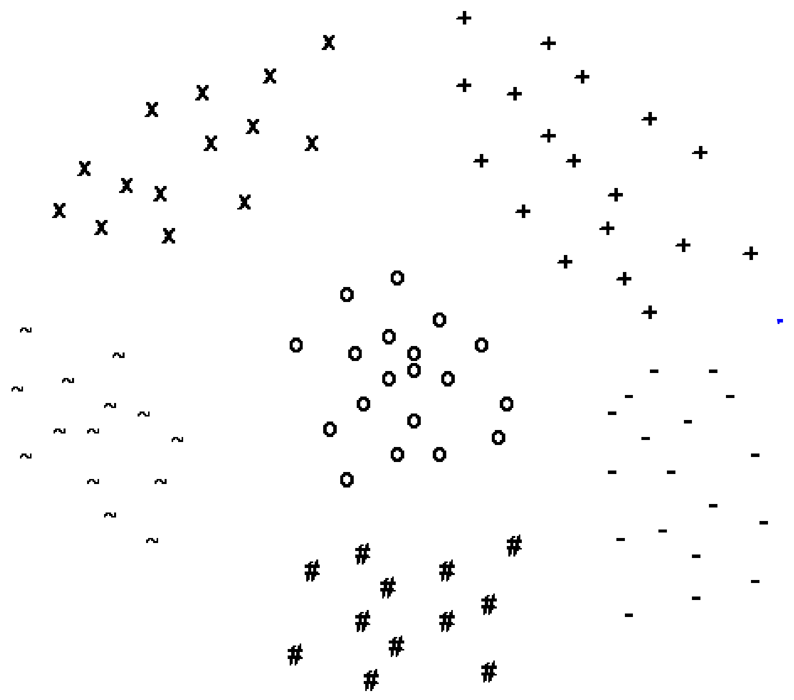
A1	A2	A3	Label
1	1	1	a
1	1	0	c
1	0	1	c
1	0	0	b
0	0	0	c
0	1	0	c
0	1	1	a

example with
unknown class label

A1	A2	A3	Label
0	0	1	?

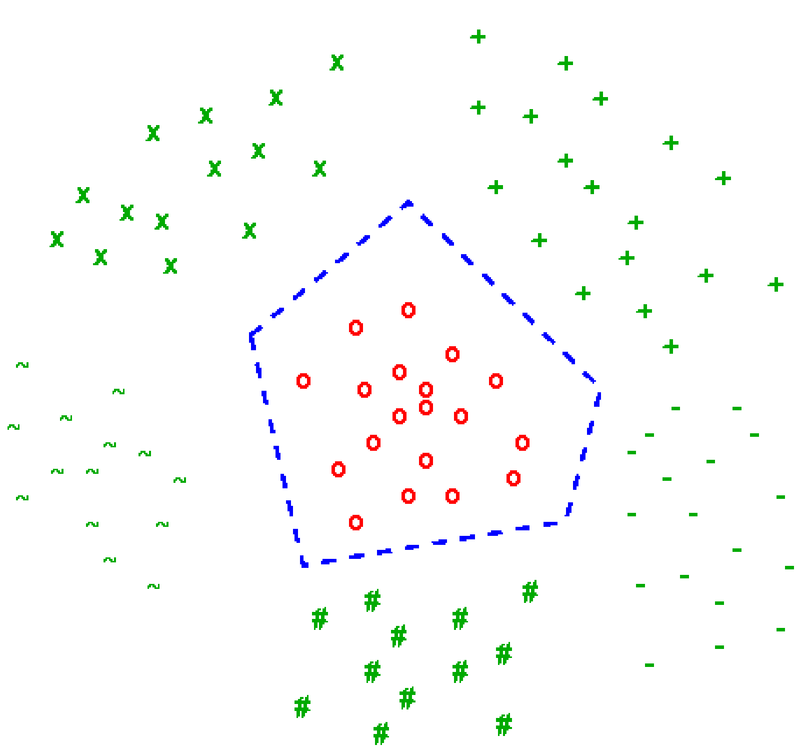
A1	A2	A3	Label
0	0	1	b

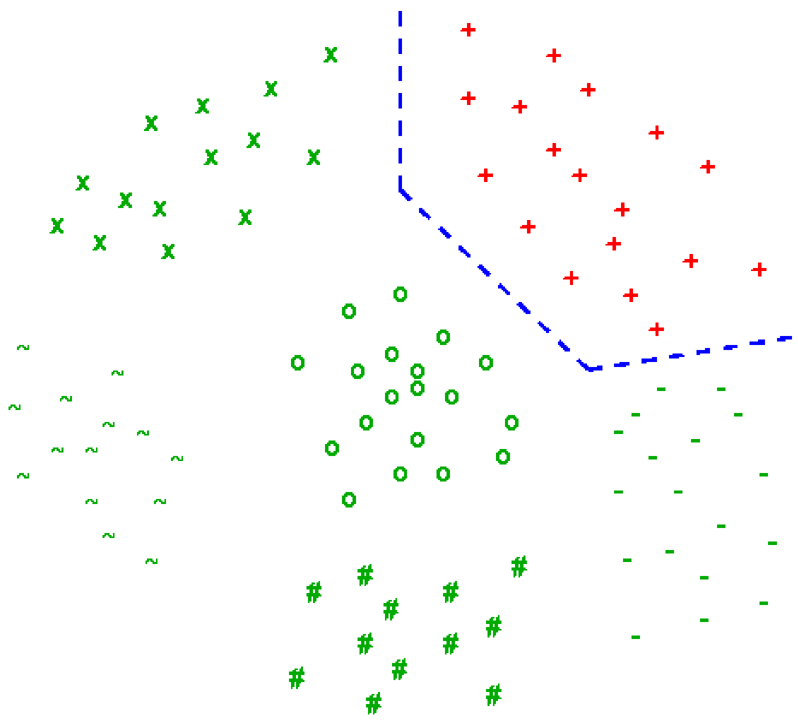
Multi-Class Classification



Can we solve multi-class problems with what we already know (by reducing them to binary problems)?

One-against-all Reduction

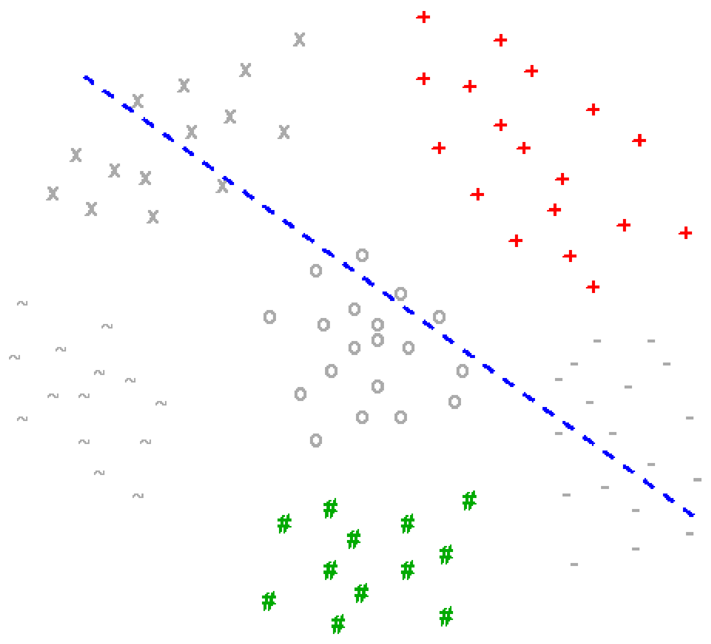


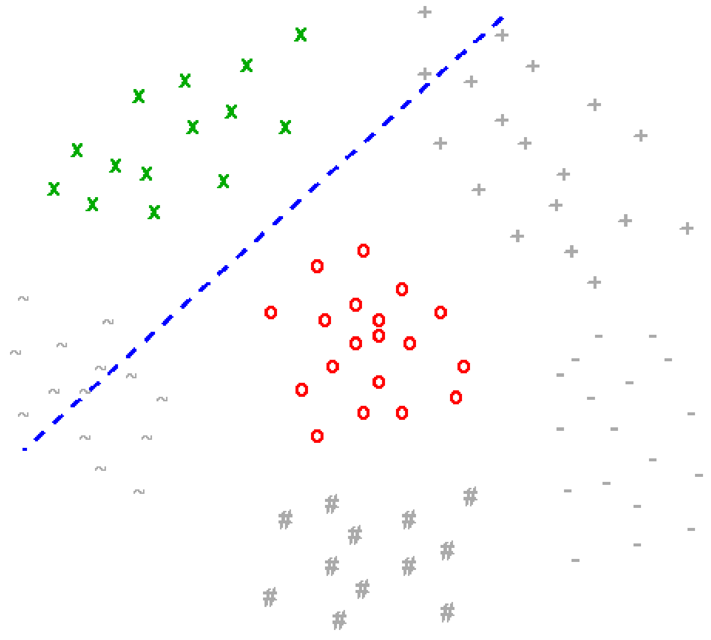


- c binary problems, one for each class
- label examples of class positive, all others negative

Pairwise Reduction

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





- smaller training sets
- simpler decision boundaries
- larger margins

Aggregating Pairwise Predictions

- Aggregate the predictions $P(i \succ j)$ of the binary classifiers into a final ranking by computing a score s_i for each class i
  - **Voting:** count the number of predictions for each class (number of points in a tournament)

$$s_i = \sum_{j \neq i} \delta\{P(i \succ j) > 0.5\} \qquad \delta\{x\} = \begin{cases} 1 & \text{if } x = \text{true} \\ 0 & \text{if } x = \text{false} \end{cases}$$

  - **Weighted Voting:** weight the predictions by their probability

$$s_i = \sum_{j=1}^{c} P(i \succ j)$$

- General Pairwise Coupling problem:
  - Given $P(i \succ j) = P(i \mid i, j)$ for all i, j
  - Find $P(i)$ for all i
  - Can be turned into an (underconstrained) system of linear equations

Pairwise Classification & Ranking Loss

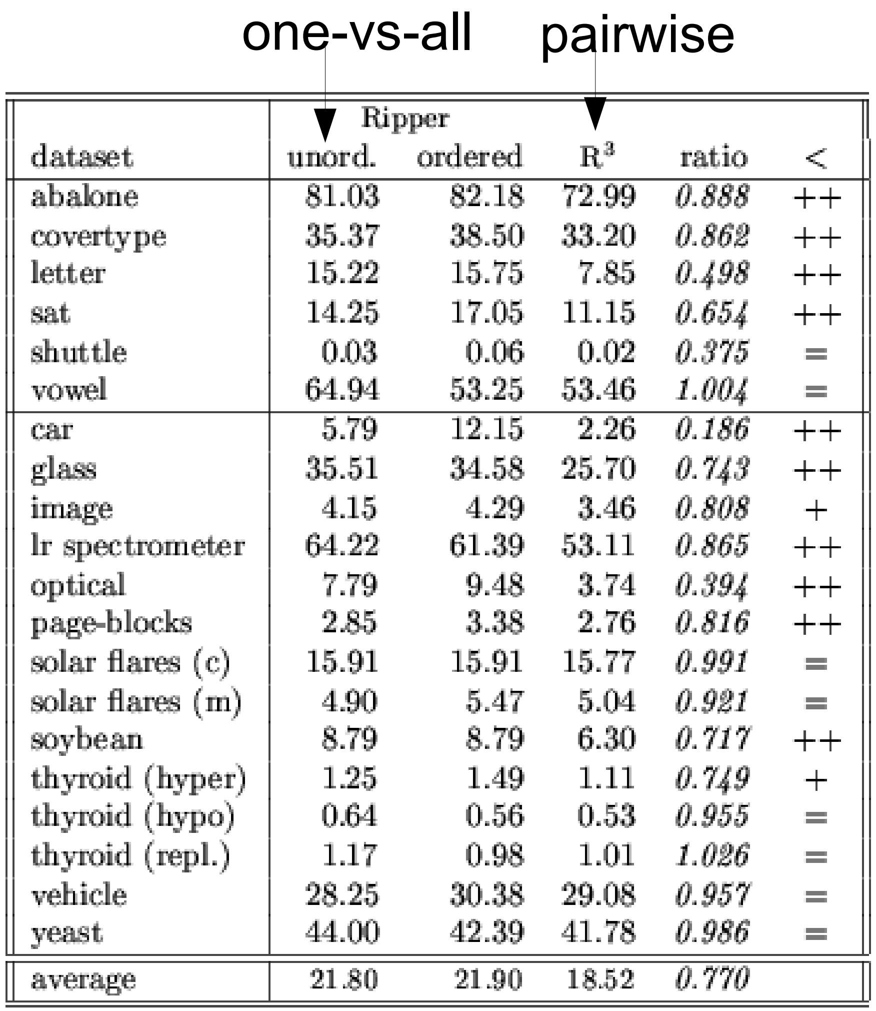
→ Weighted Voting optimizes Spearman Rank Correlation
- assuming that pairwise probabilities are estimated correctly

→ Kendall's Tau can in principle be optimized
- problem is NP-complete, but can be approximated

- Different ways of combining the predictions of the binary classifiers optimize different loss functions
  - without the need for re-training of the binary classifiers!

- However, not all loss functions can be optimized
  - e.g., 0/1 loss for rankings cannot be optimized
  - or in general the probability distribution over the rankings cannot be recovered from pairwise information

Performance of Pairwise Classification


one-vs-all
pairwise

dataset	Ripper unord.	Ripper ordered	R³	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 benchmark data sets with 4 or more classes
  - 10 significantly better ($p > 0.99$, McNemar)
  - 2 significantly better ($p > 0.95$)
  - 8 equal
  - never (significantly) worse

Advantages of Pairwise Decompositions

- Accuracy
  - better than one-against-all (also in independent studies)
- Example Size Reduction
  - subtasks might fit into memory where entire task does not
- Stability
  - simpler boundaries/concepts with possibly larger margins
- Understandability
  - e.g., recommend for ranking criteria in the Analytic Hierachy Process
- Parallelizable
  - each task is independent of all other tasks
- Modularity
  - train binary classifiers once
  - can be used with different combiners
  - for optimizing different loss functions
- Ranking ability
  - provides a ranking of classes for free
- Complexity?

Training Complexity of Pairwise Classification

Lemma: The total number of training examples for all binary classifiers in a pairwise classification ensemble is $(c-1)\cdot n$

Proof:

- each of the n training examples occurs in all binary tasks where its class is paired with one of the other $c-1$ classes

Theorem: For learning algorithms with at least linear complexity, pairwise classification is more efficient than one-against-all.

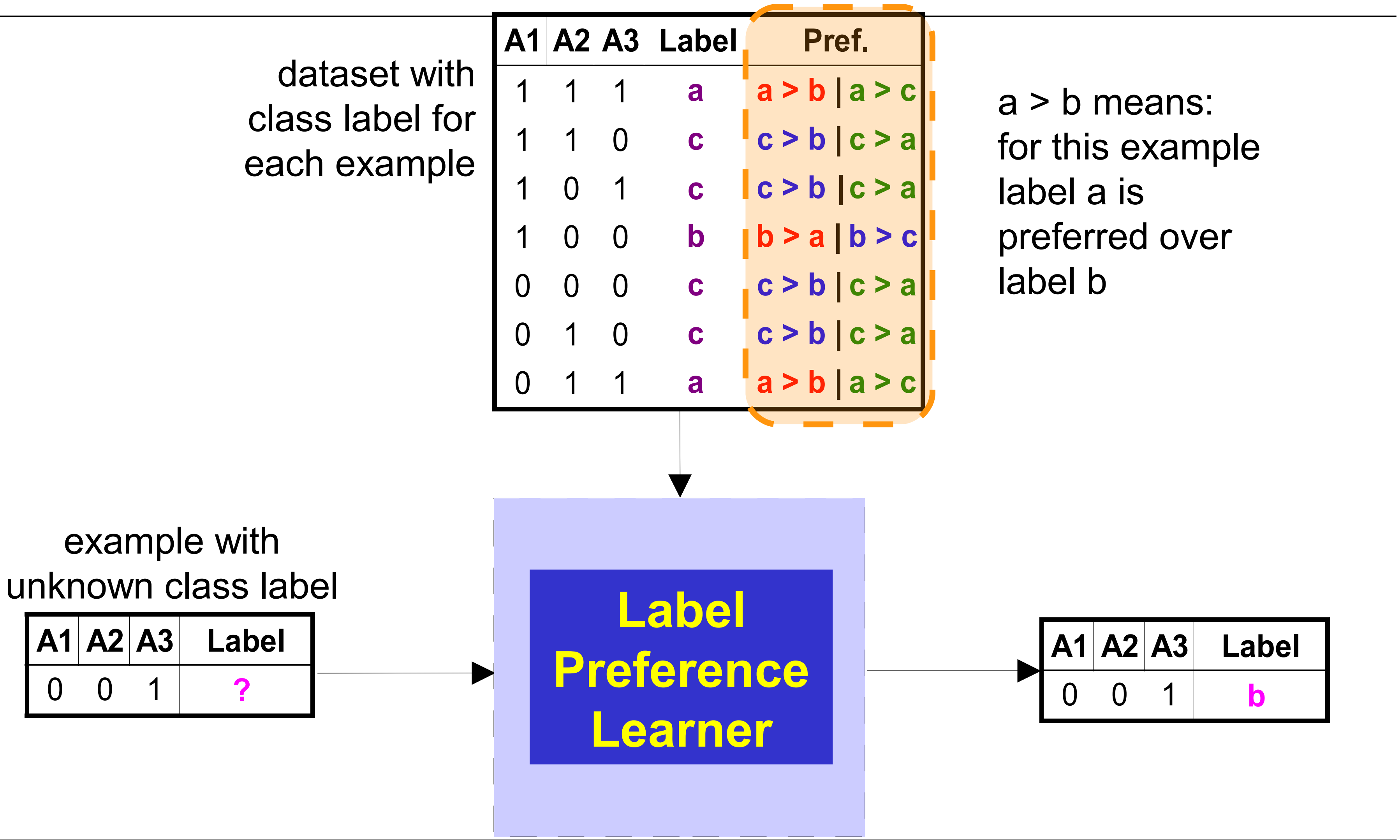
Proof Sketch:

- one-against-all binarization needs a total of $c\cdot n$ examples
- fewer training examples are distributed over more classifiers
- more small training sets are faster to train than few large training sets
- for complexity $f(n) = n^o$ $(o > 1)$: $o>1 \rightarrow \sum n_i^o < \left(\sum n_i\right)^o$

Preference Data

No.	Education	Marital S.	Sex.	Children?	Car Preferences
1	Primary	Single	M	N	Sports > Family
2	Primary	Single	M	Y	Family > Sports, Family > Mini
3	Primary	Married	M	N	Sports > Family > Mini
4	University	Divorced	F	N	Mini > Family
5	University	Married	F	Y	Mini > Sports
6	Secondary	Single	M	N	Sports > Mini > Family
7	University	Single	F	N	Mini > Family, Mini > Sports
8	Secondary	Divorced	F	N	Mini > Sports
9	Secondary	Single	F	Y	Mini > Sports, Family > Sports
10	Secondary	Married	M	Y	Family > Mini
11	Primary	Married	F	N	Mini > Family
12	Secondary	Divorced	M	Y	Family > Sports > Mini
13	University	Divorced	F	Y	Sports > Mini, Family > Mini
14	Secondary	Divorced	M	N	Sports > Mini

Class Information encodes Preferences



dataset with class label for each example

A1	A2	A3	Label	Pref.
1	1	1	a	a > b | a > c
1	1	0	c	c > b | c > a
1	0	1	c	c > b | c > a
1	0	0	b	b > a | b > c
0	0	0	c	c > b | c > a
0	1	0	c	c > b | c > a
0	1	1	a	a > b | a > c

a > b means: for this example label a is preferred over label b

example with unknown class label

A1	A2	A3	Label
0	0	1	?

A1	A2	A3	Label
0	0	1	b

General Label Preference Learning Problem

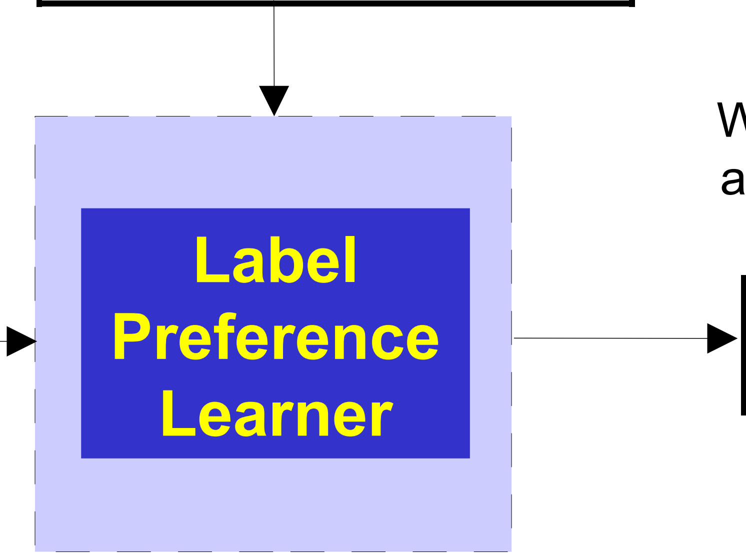
dataset with preferences for each example

A1	A2	A3	Pref.
1	1	1	a > b | b > c
1	1	0	a > b | c > b
1	0	1	b > a
1	0	0	b > a | a > c | c > b
0	0	0	c > a
0	1	0	c > b | c > a
0	1	1	a > c

Each example may have an arbitrary number of preferences

example with unknown preferences

A1	A2	A3	Pref.
0	0	1	?



We typically predict a complete ranking (a total order)

A1	A2	A3	Pref.
0	0	1	b > a > c

Label Ranking

- Preference learning scenario in which
  - contexts are characterized by features
  - no information about the items is given except a unique name (a label)

GIVEN:

- a set of *labels*:
- a set of *contexts*:
- for each training context e_k:
  - a set of *preferences*

$$L = \{\lambda_i | i = 1 \ldots c\}$$

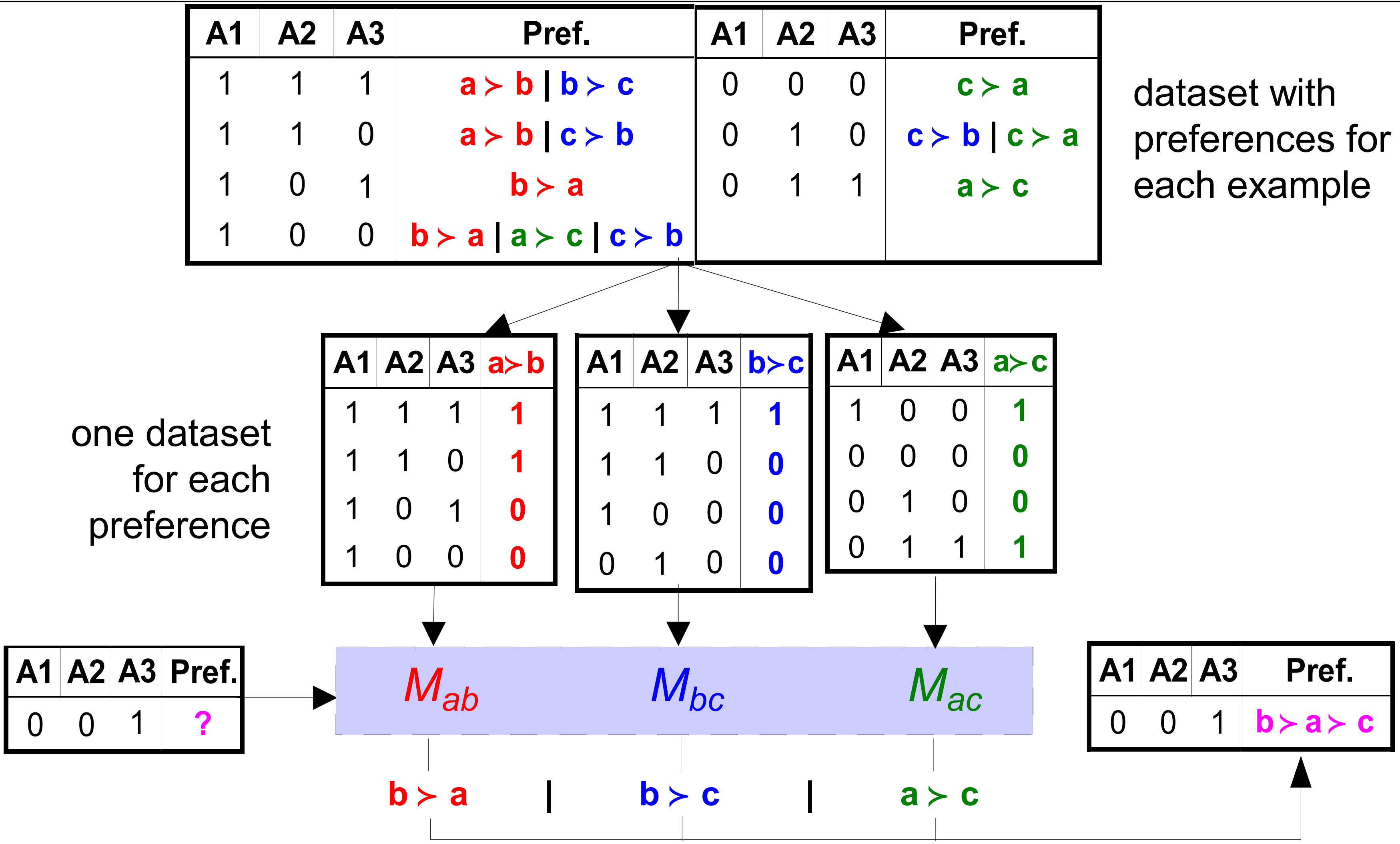
$$E = \{e_k | k = 1 \ldots n\}$$

$$P_k = \{\lambda_i \succ_k \lambda_j\} \subseteq L \, x \, L$$

FIND:

- a label ranking function that orders the labels for any given context

Pairwise Preference Learning



dataset with preferences for each example

A1	A2	A3	Pref.
1	1	1	$a \succ b \mid b \succ c$
1	1	0	$a \succ b \mid c \succ b$
1	0	1	$b \succ a$
1	0	0	$b \succ a \mid a \succ c \mid c \succ b$
0	0	0	$c \succ a$
0	1	0	$c \succ b \mid c \succ a$
0	1	1	$a \succ c$

one dataset for each preference

A1	A2	A3	$a \succ b$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0

A1	A2	A3	$b \succ c$
1	1	1	1
1	1	0	0
1	0	0	0
0	1	0	0

A1	A2	A3	$a \succ c$
1	0	0	1
0	0	0	0
0	1	0	0
0	1	1	1

A1	A2	A3	Pref.
0	0	1	?

A1	A2	A3	Pref.
0	0	1	$b \succ a \succ c$

A Multilabel Database

No.	Education	Marital S.	Sex.	Children?	Quality	Tabloid	Fashion	Sports
1	Primary	Single	M	N	0	0	0	0
2	Primary	Single	M	Y	0	0	0	1
3	Primary	Married	M	N	0	0	0	0
4	University	Divorced	F	N	1	1	1	0
5	University	Married	F	Y	1	0	1	0
6	Secondary	Single	M	N	0	1	0	0
7	University	Single	F	N	1	1	0	0
8	Secondary	Divorced	F	N	1	0	0	1
9	Secondary	Single	F	Y	0	1	1	0
10	Secondary	Married	M	Y	1	1	0	1
11	Primary	Married	F	N	1	0	0	0
12	Secondary	Divorced	M	Y	0	1	0	0
13	University	Divorced	F	Y	0	1	1	0
14	Secondary	Divorced	M	N	1	0	0	1

Multi-Label Classification

Multilabel Classification:

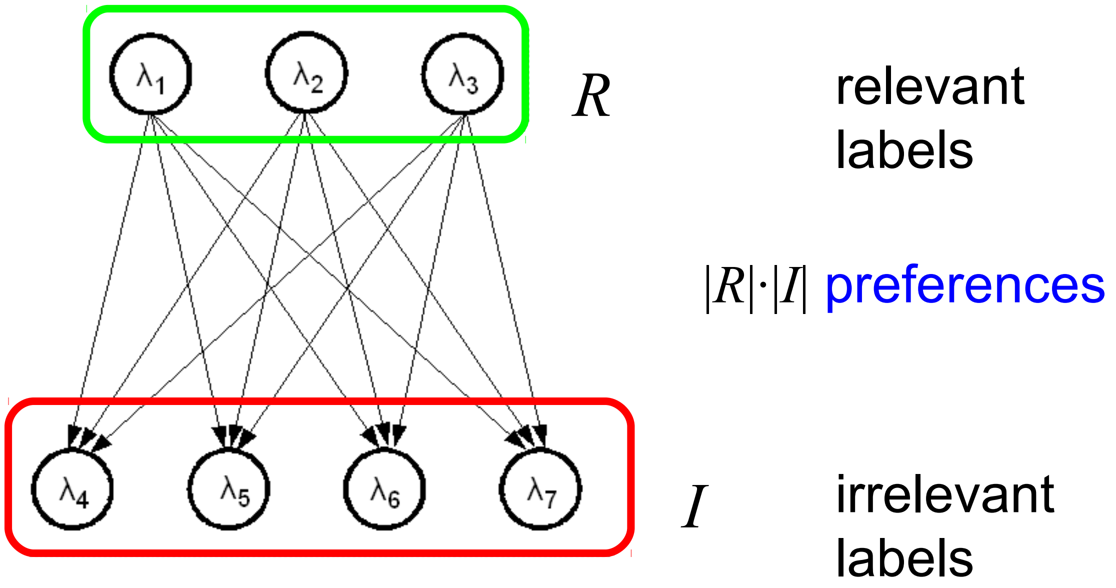
- each context is associated with multiple labels
  - e.g., keyword assignments to texts

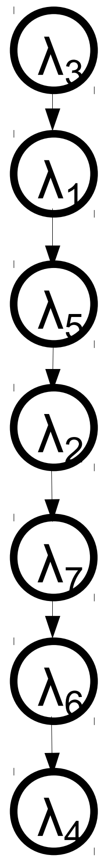
- Relevant labels R for an example
  - those that should be assigned to the example
- Irrelevant labels $I = L \setminus R$ for an example
  - those that should not be assigned to the examples

- Simple solution:
  - Predict each label independently (Binary Relevance / one-vs-all)
- Key Challenge:
  - The prediction tasks are not independent!

Pairwise Multi-Label Ranking

- Tranformation of Multi-Label Classification problems into preference learning problems is straight-forward





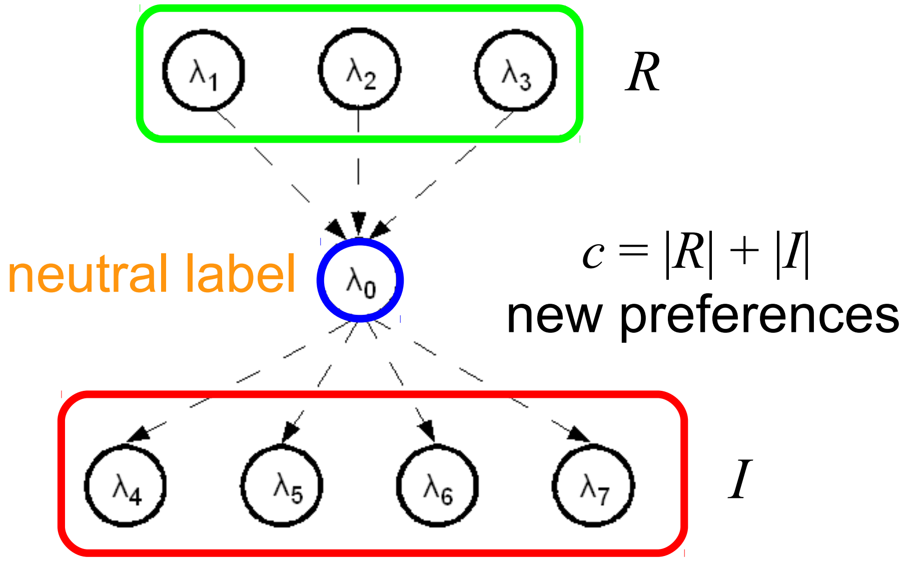
- at prediction time, the pairwise ensemble predicts a label ranking

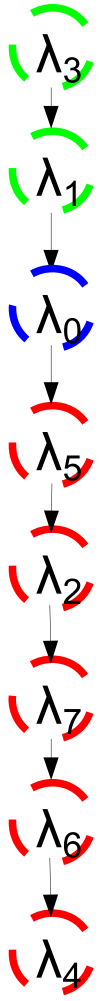
Problem:

- Where to draw boundary between relevant and irrelevant labels?

Calibrated Multi-Label PC

- Key idea:
  - introduce a neutral label into the preference scheme
  - the neutral label is
    - less relevant than all relevant classes
    - more relevant than all irrelevant classes





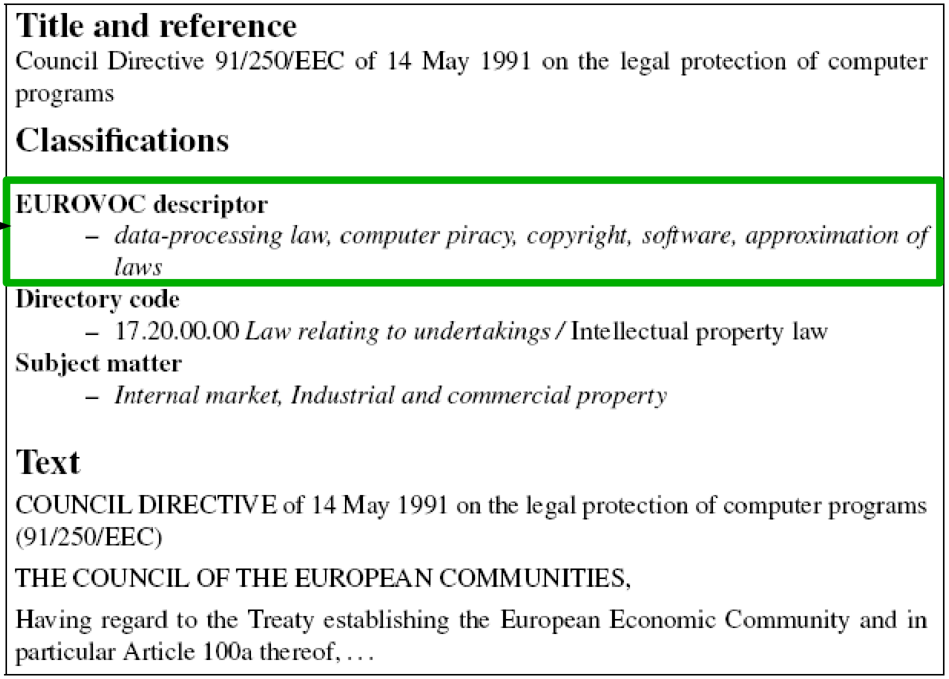
- at prediction time, all labels that are ranked above the neutral label are predicted to be relevant

EuroVOC Classification of EC Legal Texts

Eur-Lex database

- ≈ 20,000 documents
- ≈ 4,000 labels
- ≈ 5 labels per document

- Pairwise modeling approach learns ≈8,000,000 perceptrons
  - memory-efficient dual representation necessary



Title and reference

Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs

Classifications

EUROVOC descriptor
- *data-processing law, computer piracy, copyright, software, approximation of laws*

Directory code
- 17.20.00.00 *Law relating to undertakings* / Intellectual property law

Subject matter
- *Internal market, Industrial and commercial property*

Text

COUNCIL DIRECTIVE of 14 May 1991 on the legal protection of computer programs (91/250/EEC)

THE COUNCIL OF THE EUROPEAN COMMUNITIES,

Having regard to the Treaty establishing the European Economic Community and in particular Article 100a thereof, . . .

Results:

- average precision of pairwise method is almost 50%

  → on average, the 5 relevant labels can be found within the first 10 labels of the ranking of all 4000 labels
- one-against-all methods (BR and MMP) had an avg. precision < 30%

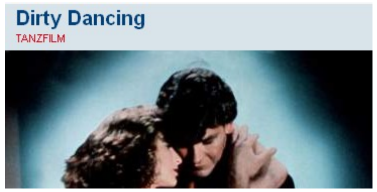
Current Work:
Graded Multilabel Classification

- Relevance of multiple labels is assessed on an ordered scale

Spaß	Action	Erotik	Spannung	Anspruch
			●●●	●

Spaß	Action	Erotik	Spannung	Anspruch
●●●	●●		●	

Spaß	Action	Erotik	Spannung	Anspruch
	●●●	●	●●●	



Spaß	Action	Erotik	Spannung	Gefühl
	●	●		●●●

Open Problems

Multilabel Rule Learning

- The key challenge in multi-label classification is to model the dependencies between the labels
  - much of current research in this area is devoted to this topic
- Rules can make these dependencies explicit and exploit them in the learning phase
  - regular rule: `university, female` → `quality, fashion`
  - label dependency: `fashion` ≠ `sports`
  - mixed rule: `university, tabloid` → `quality`

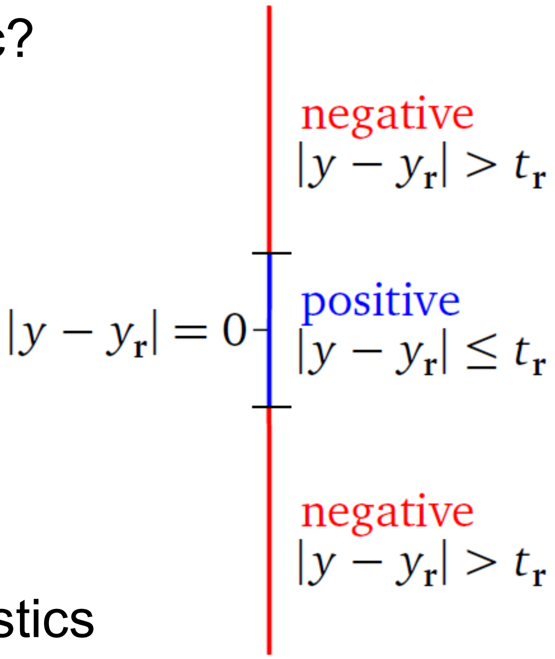
Regression

No	Education	Marital S.	Sex.	Children?	Income
1	Primary	Single	M	N	20,000
2	Primary	Single	M	Y	23,000
3	Primary	Married	M	N	25,000
4	University	Divorced	F	N	50,000
5	University	Married	F	Y	60,000
6	Secondary	Single	M	N	45,000
7	University	Single	F	N	80,000
8	Secondary	Divorced	F	N	55,000
9	Secondary	Single	F	Y	30,000
10	Secondary	Married	M	Y	75,000
11	Primary	Married	F	N	35,000
12	Secondary	Divorced	M	Y	70,000
13	University	Divorced	F	Y	65,000
14	Secondary	Divorced	M	N	38,000

Numeric Target Variable

Rule-Based Regression

- Regression trees are quite successful
- Work on directly learning regression rules was not yet able to match that performance
  - Main Problem: How to define a good heuristic?
- Transformation approach:
  - Reduce regression to classification
  - use the idea of ε-insensitive loss functions proposed for SVMS:
  - all examples in an ε-environment of the value predicted in the rule head are considered to be positive, all others negative
  - rules can then be learned using regular heuristics for classification rules



Open Problems

Rule-Based Regression

- Efficient method for learning rules with linear models instead of constant predictions
  - methods from decision-tree learning can be adapted
- Development of good rule learning heuristics for regression data
  - direct learning instead of mapping to classification

Summary (1)

- Rules can be learned via top-down hill-climbing
  - add one condition at a time until the rule covers no more negative exs.
- Heuristics are needed for guiding the search
  - can be visualize through isometrics in coverage space
- Rule Sets can be learned one rule at a time
  - using the covering or separate-and conquer strategy
- Overfitting is a serious problem for all machine learning algorithms
  - too close a fit to the training data may result in bad generalizations
- Pruning can be used to fight overfitting
  - Pre-pruning and post-pruning can be efficiently integrated
- Multi-class problems can be addressed by multiple rule sets
  - one-against-all classification or pairwise classification

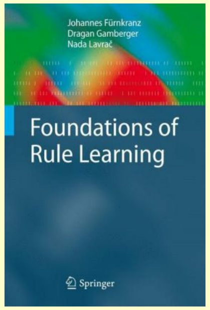
Summary (2)

- Problem decomposition as a powerful tool
  - Try to understand simple problems first
  - Build solutions for complex problems on well-understood solutions for simpler problems

subgroup discovery → concept learning → multi-class classification → → preference learning → multi-label classification

- motivated by research in rule learning but applied to other fields
  - e.g., work on preference learning has (so far) used little rule learning
- Preference Learning is a general framework for decomposing complex machine learning problems into simpler problems
  - multi-label classification, graded ML classification, ordered classification, hierarchical classification, ranking...

Thanks for your attention!



J. Fürnkranz, D. Gamberger, N. Lavrac,
Foundations of Rule Learning
Springer-Verlag, 2012.

J. Fürnkranz, E. Hüllermeier (eds.)
Preference Learning
Springer-Verlag, 2011.