Data Mining and Machine Learning

Learning Rule Sets

- Introduction
  - Learning Rule Sets
- Separate-and-Conquer Rule Learning
  - Covering algorithm
- Overfitting and Pruning
- Multi-Class Problems

Learning Rule Sets

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

A Sample 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")

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

→ it must be part of the version space

- but it does not generalize to new examples!

The Need for a Bias

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

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

A Better Solution


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

Recap: Subgroup Discovery

- Abstract algorithm for learning a single rule:

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

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

Key idea of Covering algorithms

Properties of Subgroup Discovery algorithms:

- Consistency can always be maximized
  - a rule that covers no negative examples can always be found
- Completeness can not necessarily be ensured
  - Many concepts can only be formulated with multiple rules

Learning strategy:

- Try to learn a rule that is as consistent as possible
- Fix completeness by repeating this step until each (positive) training example is covered by at least one rule

Relaxing Completeness and Consistency

- So far we have defined correctness on training data as consistency + completeness
  - → we aim for a rule that covers all positive and no negative examples
- This is not always a good idea (→ overfitting)
- Example:

  Training set with 200 examples, 100 positive and 100 negative
  - **Rule Set A** consists of 100 complex rules, each covering a single positive example and no negatives

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

    → B is incomplete and inconsistent on the training set
- Which one will generalize better to unseen examples?

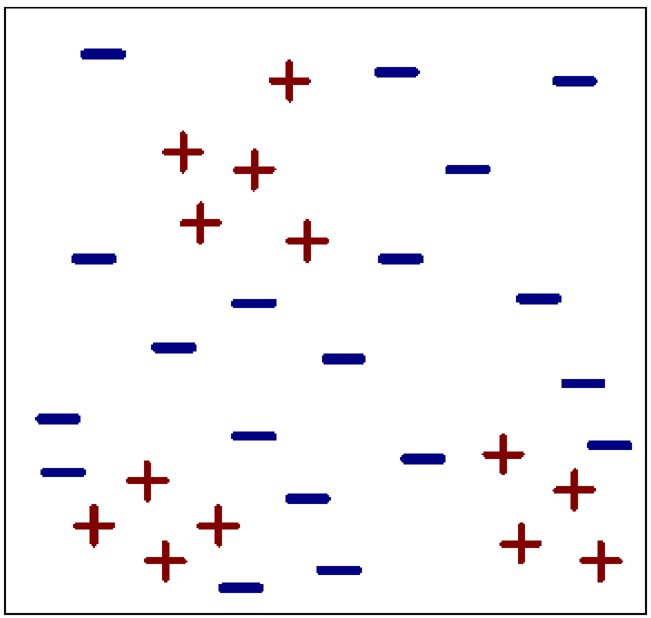
Separate-and-Conquer Rule Learning

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

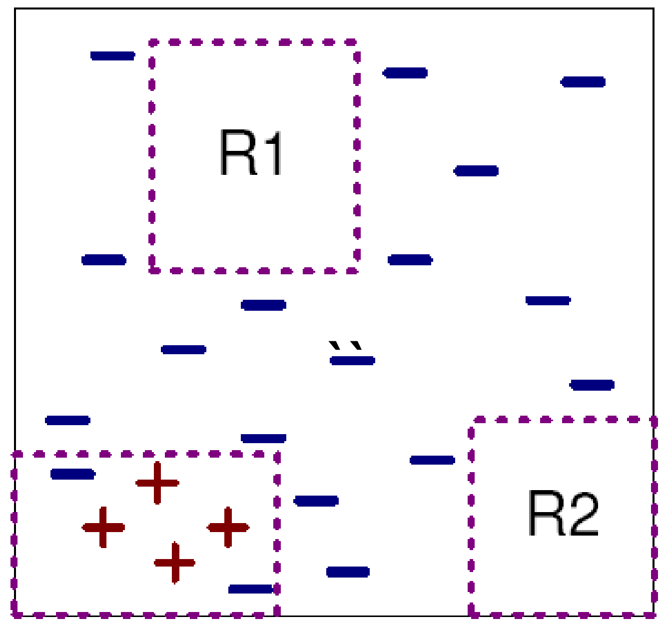
1. Start with an empty theory T and training set E
2. Learn a single (*consistent*) rule R from E and add it to T
3. If T is satisfactory (*complete*), return T
4. Else:
   - *Separate:* Remove examples explained by R from E
   - *Conquer:* goto 2.

- 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

Separate-and-Conquer Rule Learning



(i) Original Data

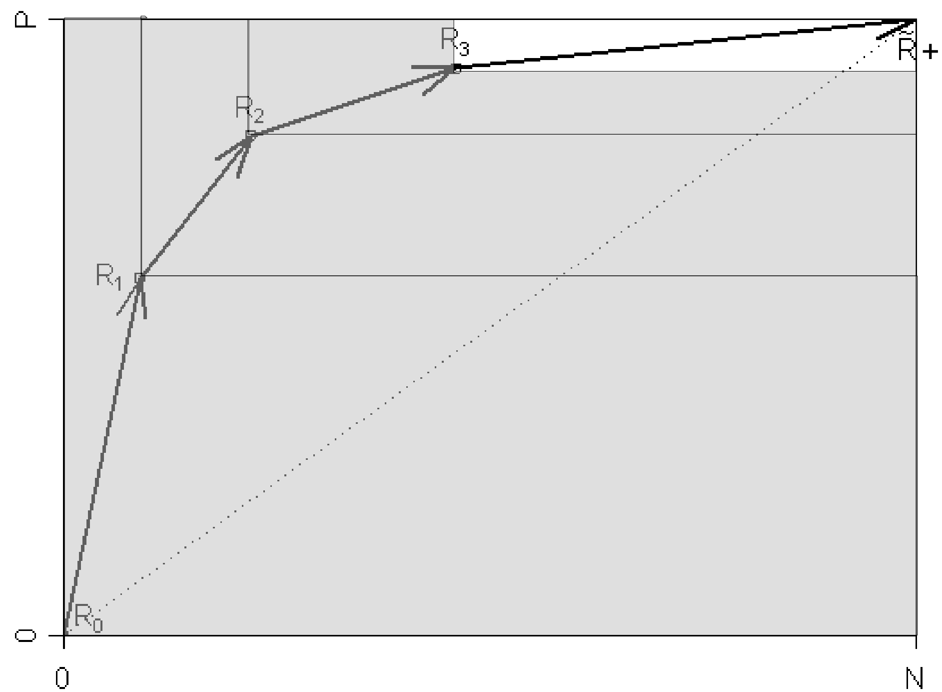


(iv) Step 3

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

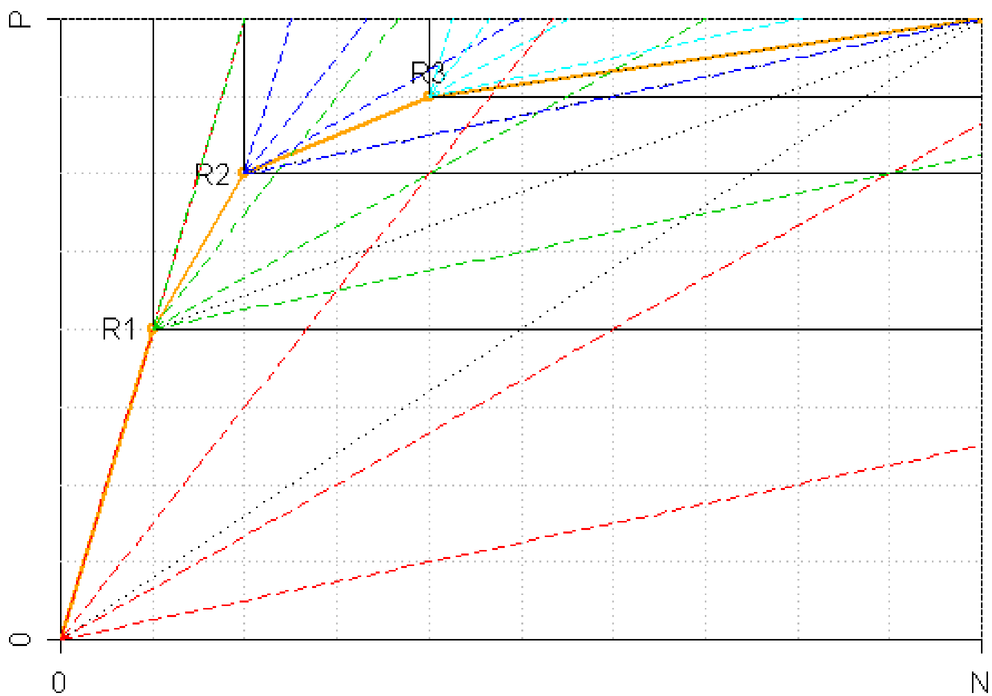
Covering Strategy

- 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)



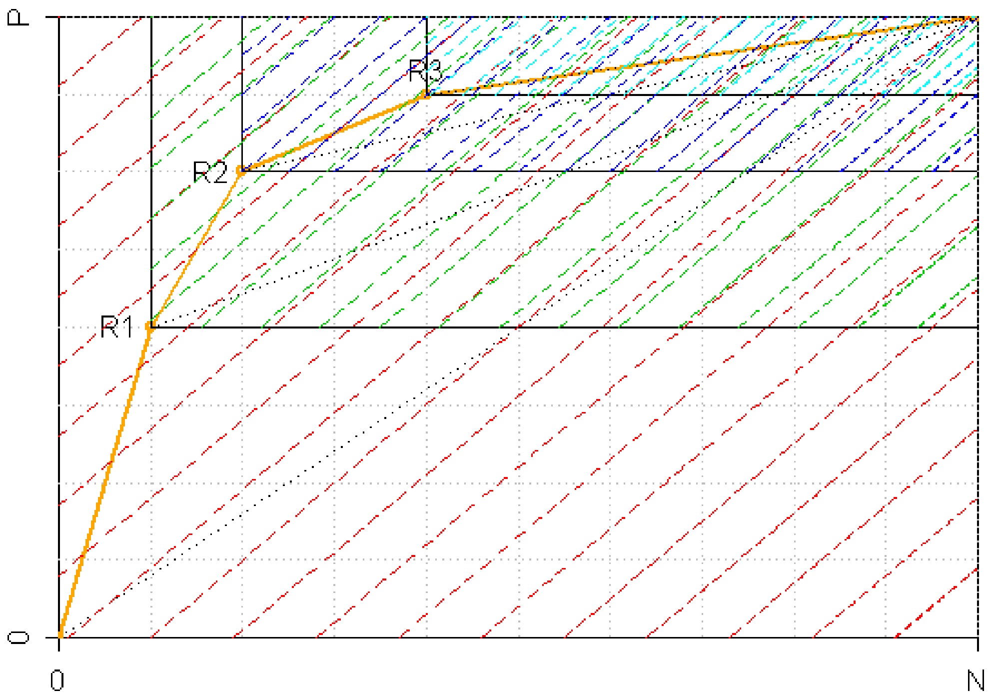
Rule Selection with Precision

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



Rule Selection with 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

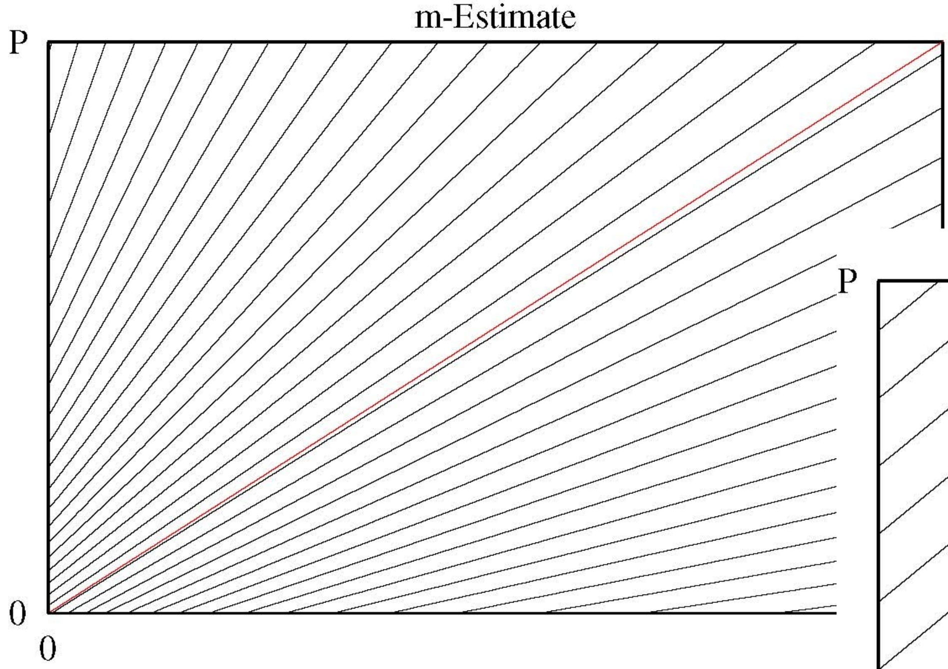


Which Heuristic is Best?

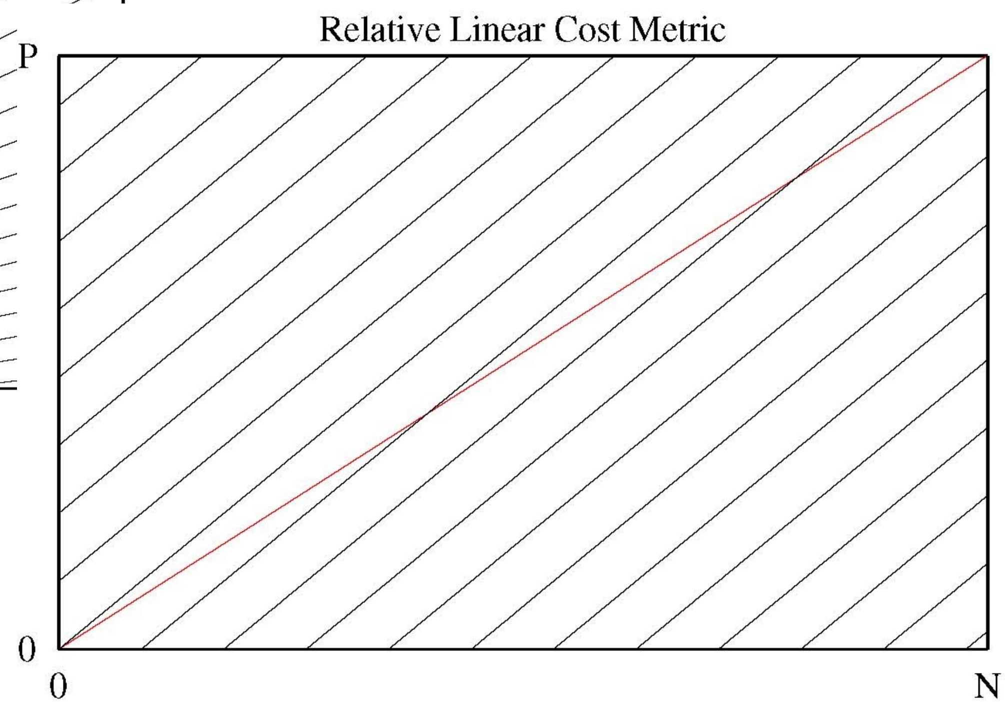
- There have been many proposals for different heuristics
  - and many different justifications for these proposals
  - some measures perform better on some datasets, others on other datasets
- Large-Scale Empirical Comparison:
  - 27 training datasets
    - on which parameters of the heuristics were tuned)
  - 30 independent datasets
    - which were not seen during optimization
  - Goals:
    - see which heuristics perform best
    - determine good parameter values for parametrized functions

Best Parameter Settings

for m-estimate: $m = 22.5$



for relative cost metric: $c = 0.342$



Empirical Comparison of Different Heuristics

Heuristic	Training Datasets Accuracy	Training Datasets # Conditions	Independent Datasets Accuracy	Independent Datasets #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

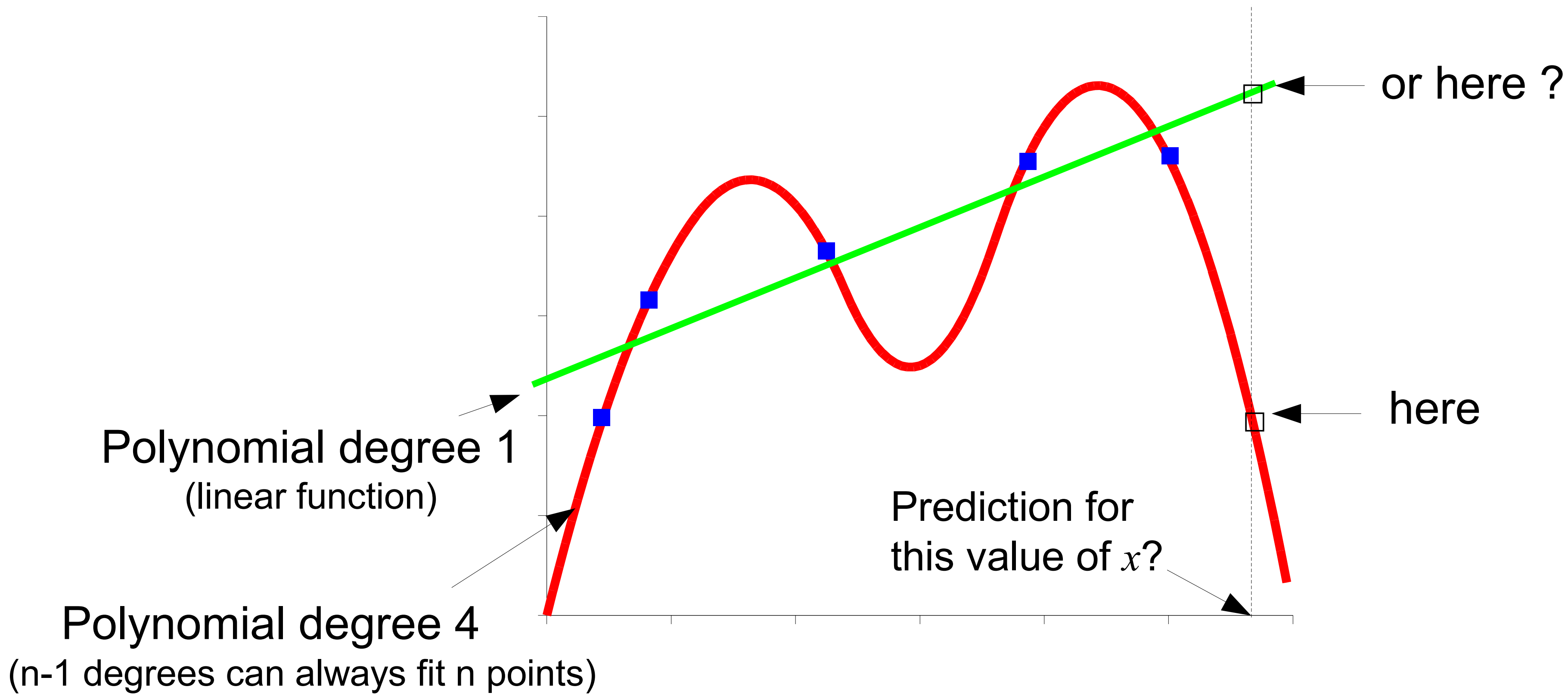
- Ripper is best, but uses pruning (the others don't)
- the optimized parameters for the m-estimate and the relative cost metric perform better than all other heuristics
  - also on the 30 datasets on which they were not optimized
- some heuristics clearly overfit (bad performance with large rules)
-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

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

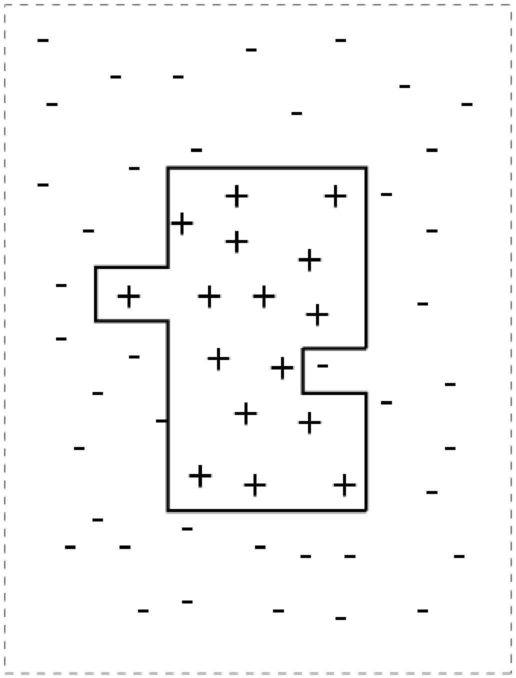


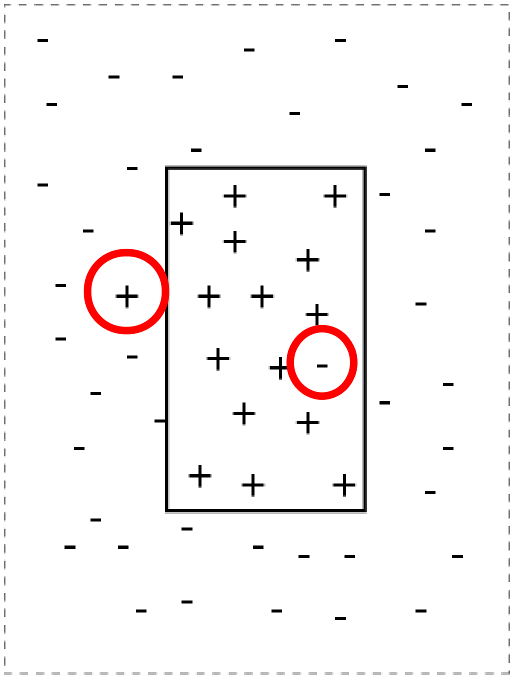
Overfitting Avoidance

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

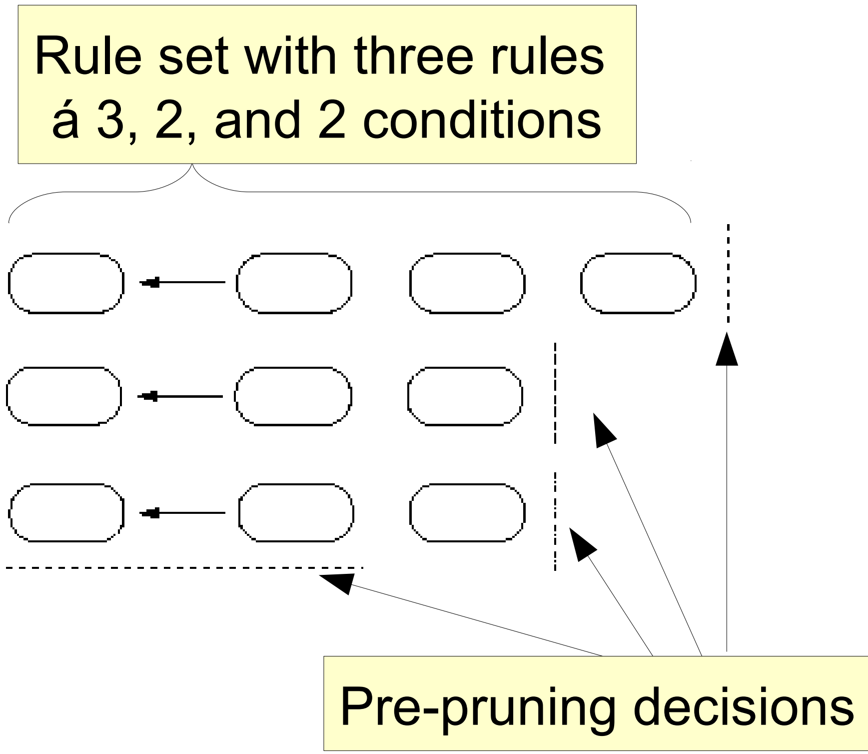
- For simpler concepts there is less danger that they are able to overfit the data
  - for a polynomial of degree n one can choose $n+1$ parameters in order to fit the data points

→ 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

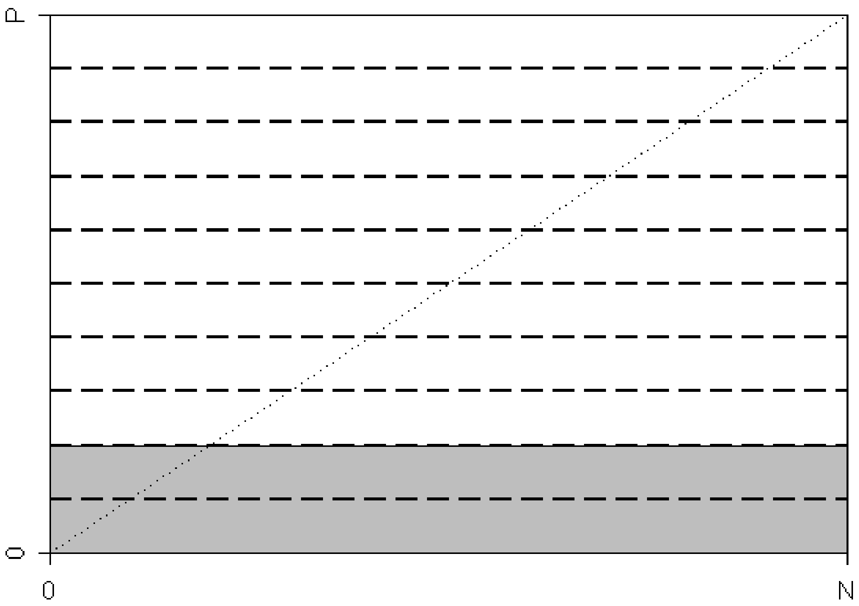
TECHNISCHE UNIVERSITÄT DARMSTADT

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

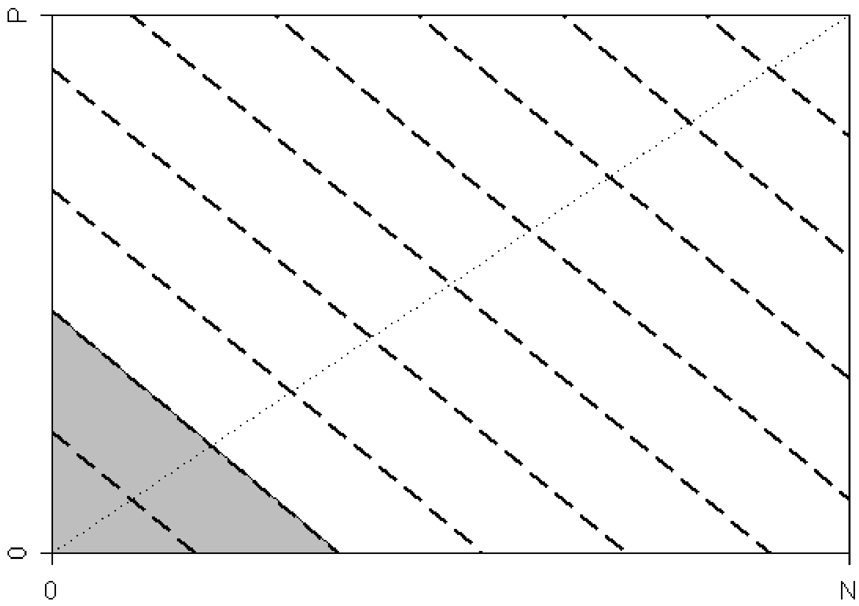
Minimum Coverage Filtering

filter rules that do not cover a minimum number of

positive examples (support)

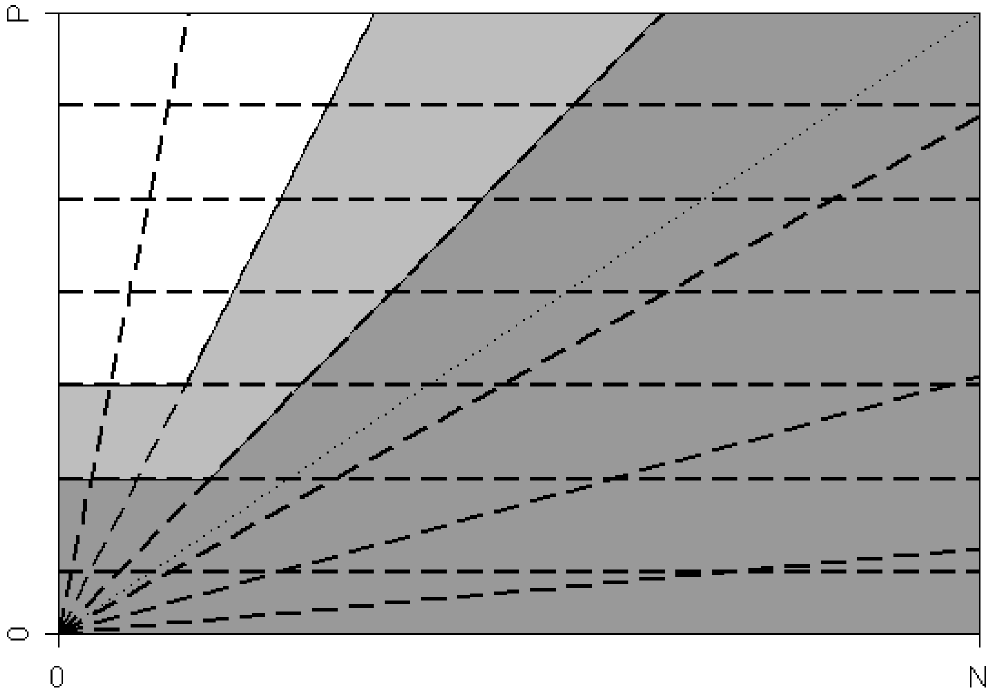


all examples (coverage)



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



→ we will return to support/confidence in the context of association rule learning algorithms!

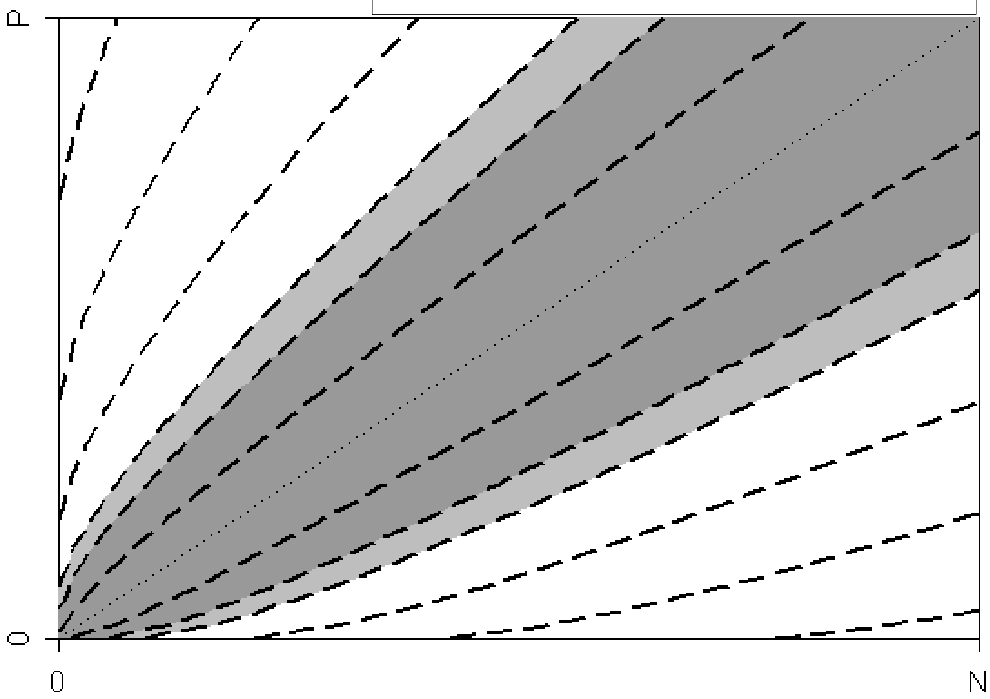
CN2's likelihood ratio statistics

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

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

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

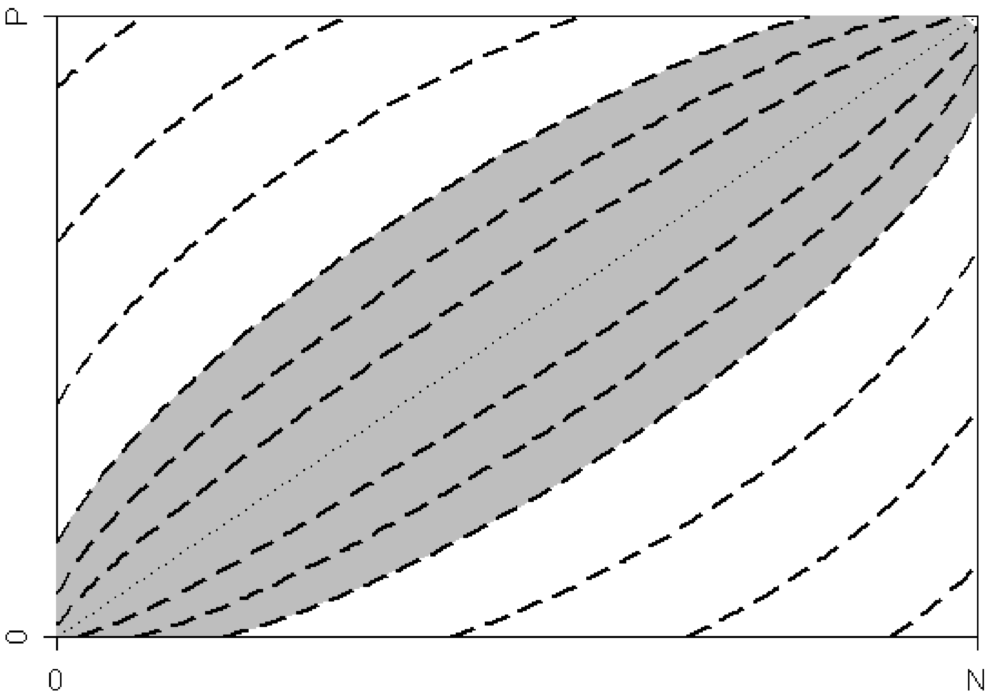
- *basic idea:* measure significant deviation from prior probability distribution
- *effects:*
  - non-linear isometrics
    - similar to m-estimate
    - but prefer rules near the edges
  - distributed χ^2
  - significance levels 95% (dark) and 99% (light grey)



Correlation

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

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



MDL-Pruning in Foil

- 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

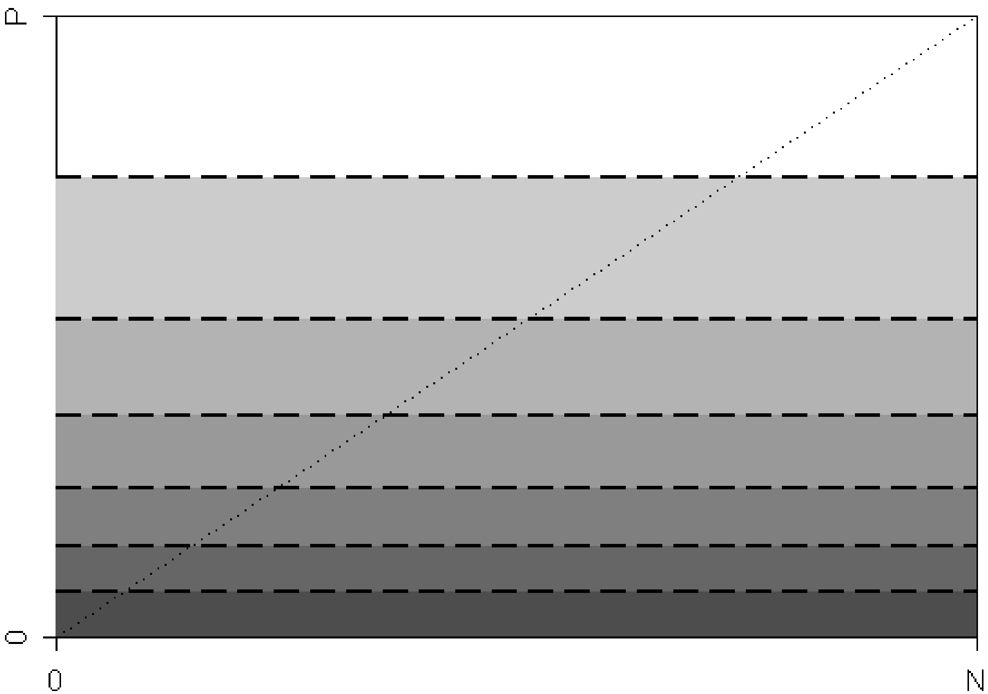
Foil's MDL-based Stopping Criterion

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

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

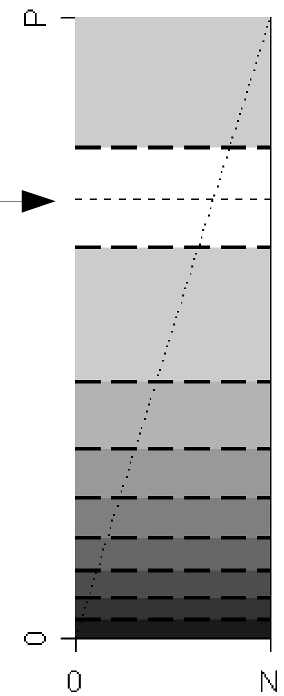
costs for transmitting which of the $P+N$ examples are covered and positive

- *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!

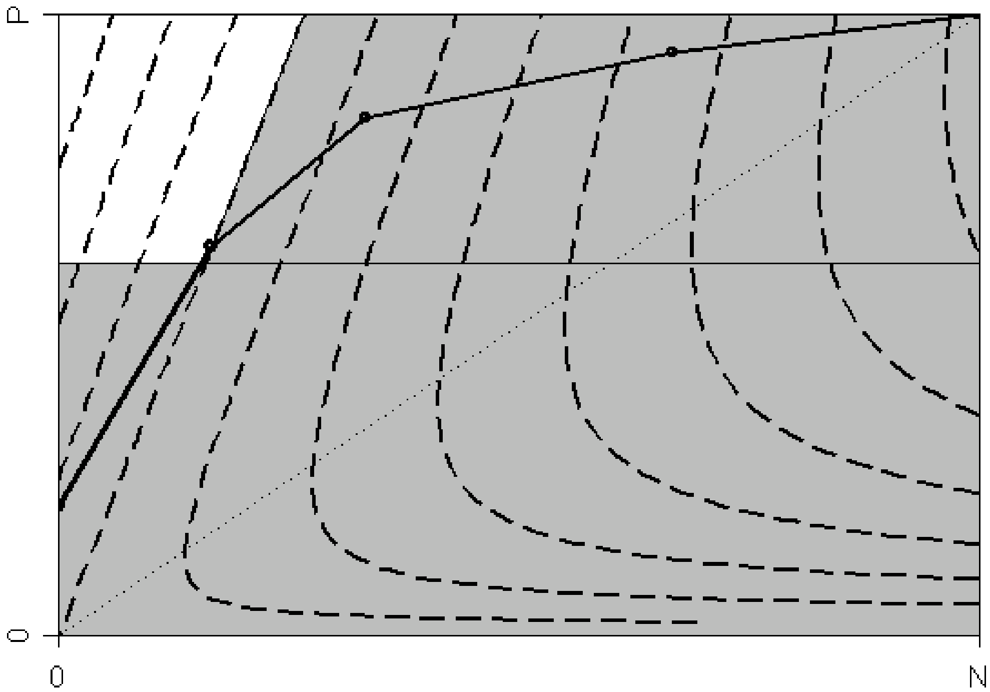


How Foil Works

TECHNISCHE UNIVERSITÄT DARMSTADT

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

- 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



Pre-Pruning Systems

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

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
```

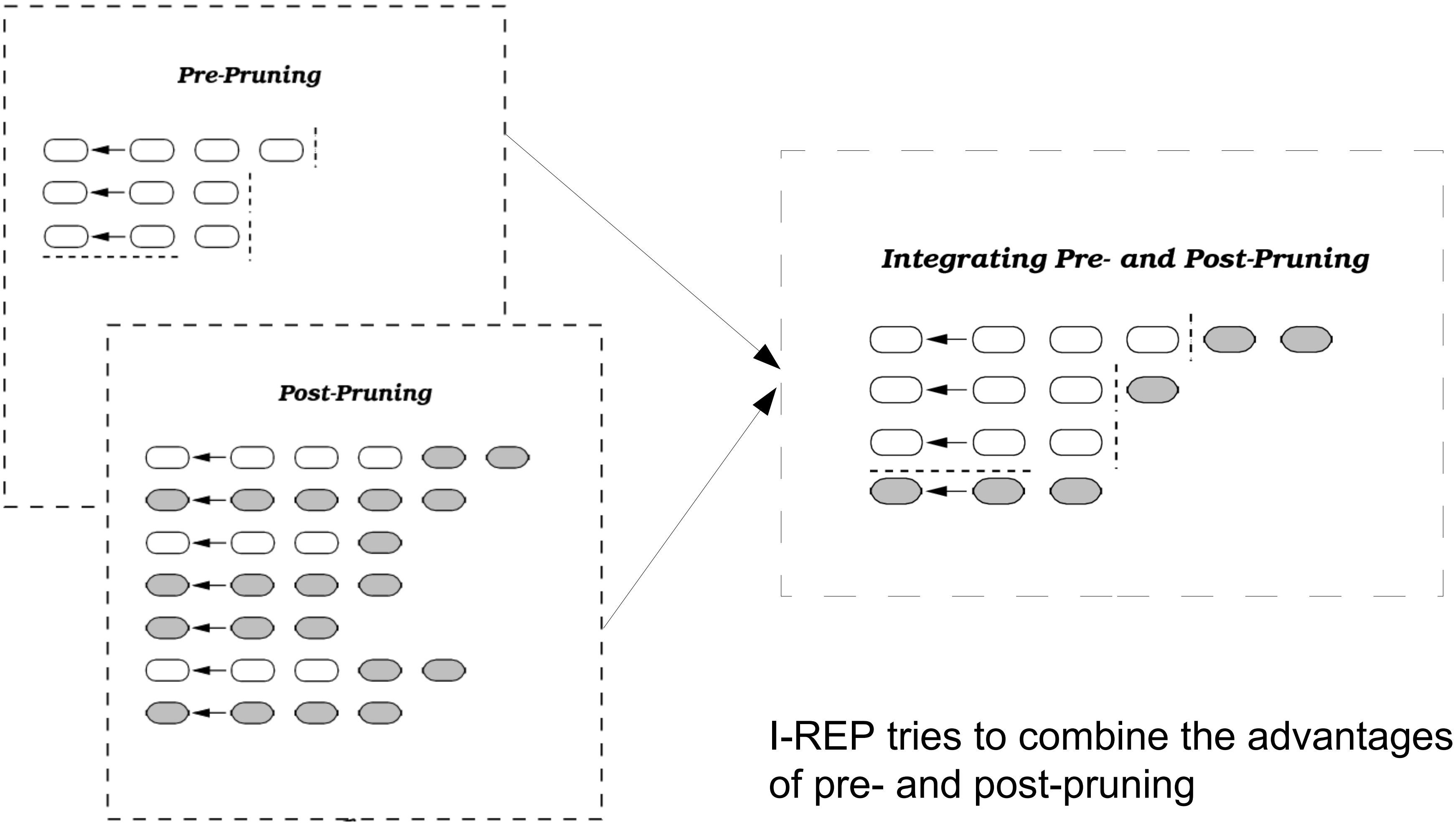
Reduced Error Pruning

- 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 in the growing set
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



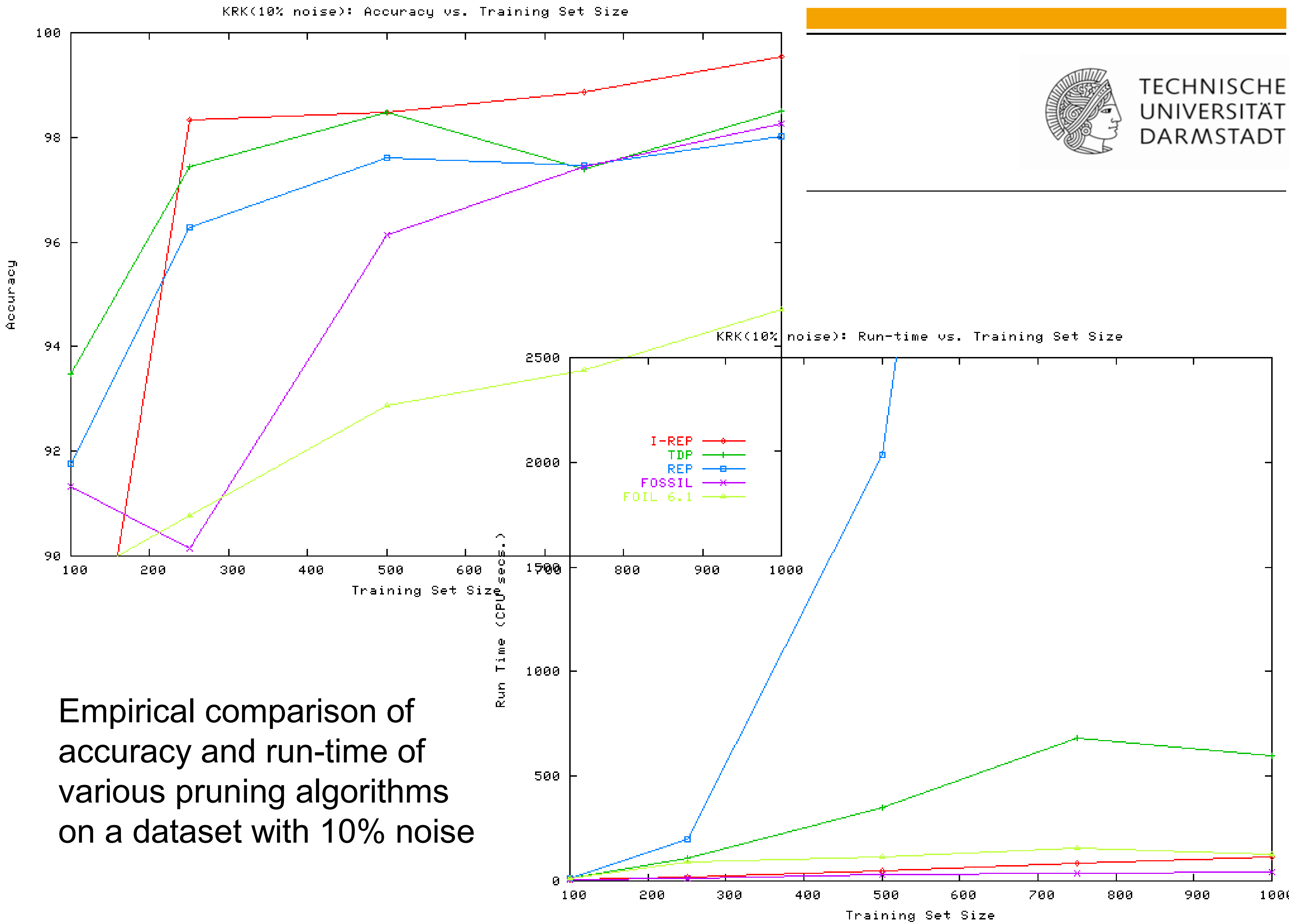
I-REP tries to combine the advantages of pre- and post-pruning

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

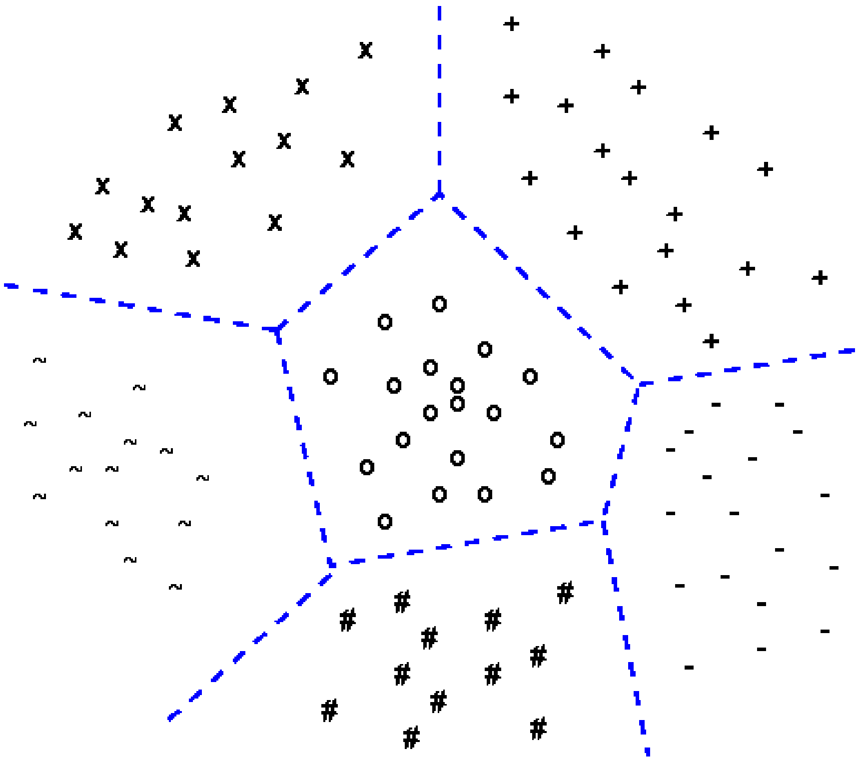
Multi-Class Classification

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")

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

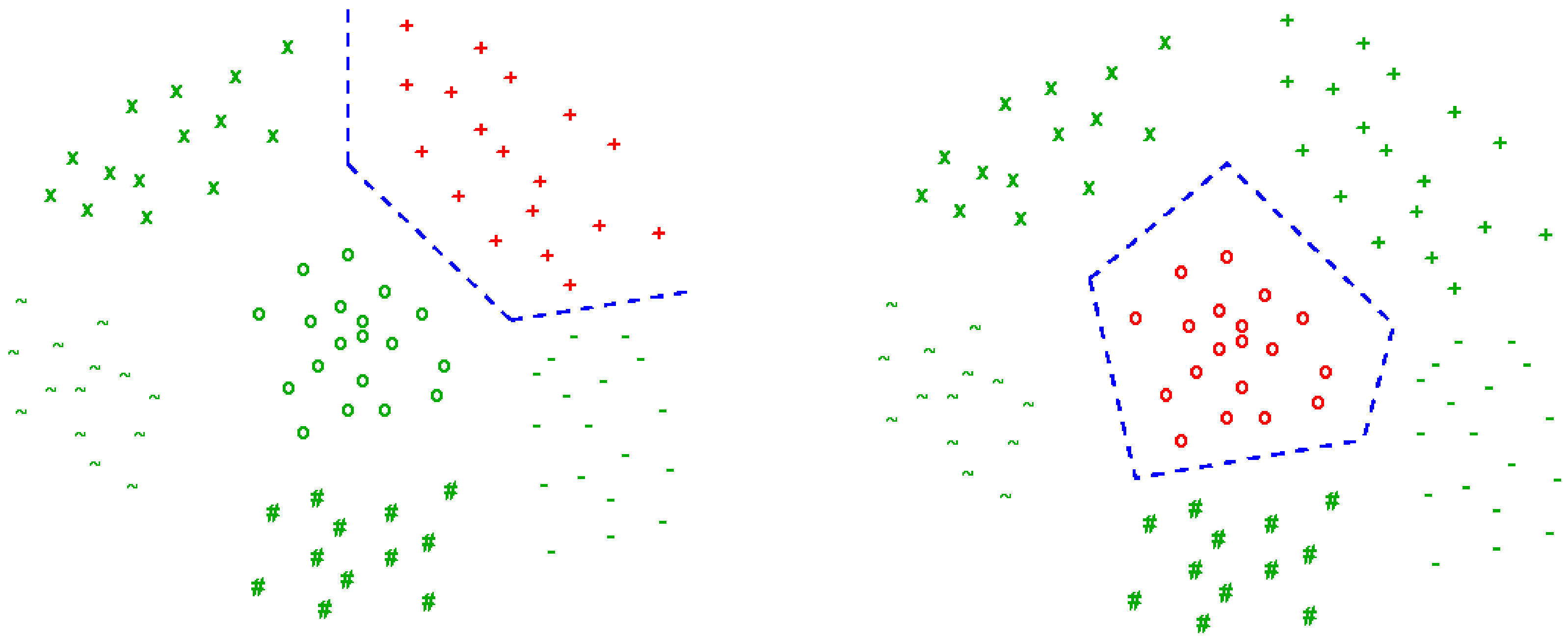


Class Binarization for Rule Learning

- 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
  - Another variant in Ripper sorts the classes first and learns first against rest - remove first - repeat
- Pairwise Classification / one-vs-one
  - Learn one rule-set for each *pair* of classes
- Error Correcting Output Codes (Dietterich & Bakiri, 1995)
  - generalized by (Allwein, Schapire, & Singer, JMLR 2000)

  → Ensemble Methods

One-against-all binarization



Treat each class as a separate concept:

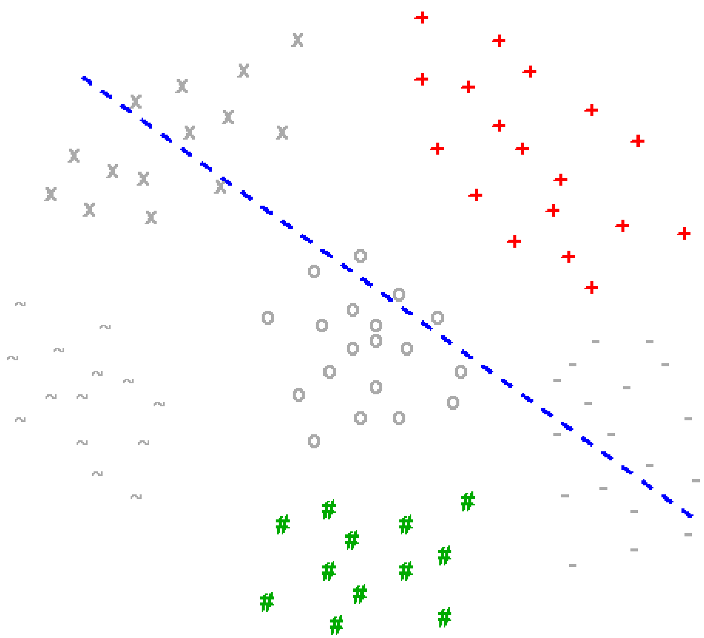
- c binary problems, one for each class
- label examples of one class positive, all others negative

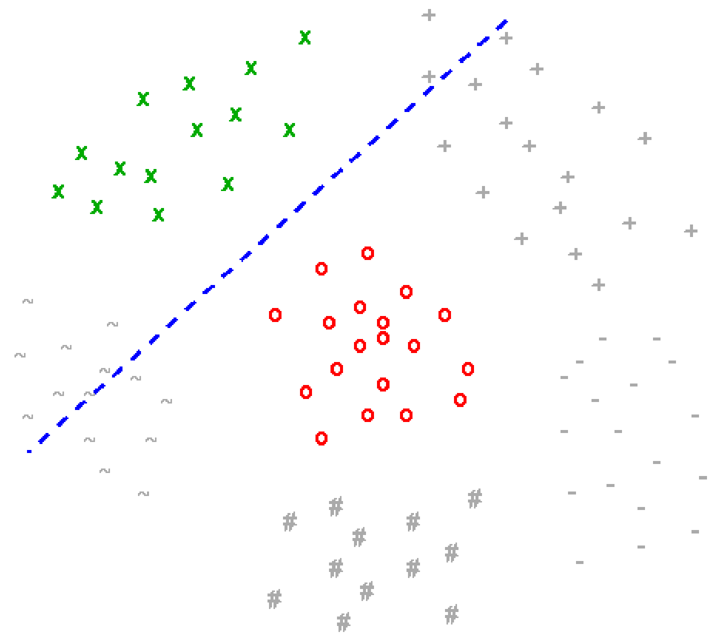
Prediction

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

Pairwise Classification

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





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

Prediction

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

Accuracy

TECHNISCHE UNIVERSITÄT DARMSTADT

one-vs-all pairwise

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

Advantages of the Pairwise Approach

- Accuracy
  - better than one-against-all (also in independent studies)
  - improvement appr. on par with 10 boosting iterations
- Example Size Reduction
  - subtasks might fit into memory where entire task does not
- Stability
  - simpler boundaries/concepts with possibly larger margins
- Understandability
  - similar to pairwise ranking as recommended by Pyle (1999)
- Parallelizable
  - each task is independent of all other tasks
- Modularity
  - train binary classifiers once
  - can be used with different combiners
- Ranking ability
  - provides a ranking of classes for free
- Complexity?
  - we have to learn a quadratic number of theories...
  - but with fewer examples

Training Complexity of PC

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 $\mathrm{f}(n) = n^o$ $(o > 1)$: $o > 1 \rightarrow \sum n_i^o < (\sum n_i)^o$

Preference Data

TECHNISCHE UNIVERSITÄT DARMSTADT

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

<table>
<tr><th>A1</th><th>A2</th><th>A3</th><th>Label</th><th>Pref.</th></tr>
<tr><td>1</td><td>1</td><td>1</td><td>a</td><td>a > b | a > c</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>c</td><td>c > b | c > a</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>c</td><td>c > b | c > a</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>b</td><td>b > a | b > c</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>c</td><td>c > b | c > a</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>c</td><td>c > b | c > a</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>a</td><td>a > b | a > c</td></tr>
</table>

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	?

Label Preference Learner

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	?

Label Preference Learner

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*: $L = \{\lambda_i | i = 1 \ldots c\}$
- a set of *contexts*: $E = \{e_k | k = 1 \ldots n\}$
- for each training context e_k:
  - a set of *preferences* $P_k = \{\lambda_i \succ_k \lambda_j\} \subseteq L \, x \, L$

FIND:

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

Pairwise Preference Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

dataset with preferences for each example

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	?

M_{ab} M_{bc} M_{ac}

b $\succ$ a | b $\succ$ c | a $\succ$ c

A 1	A 2	A 3	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:

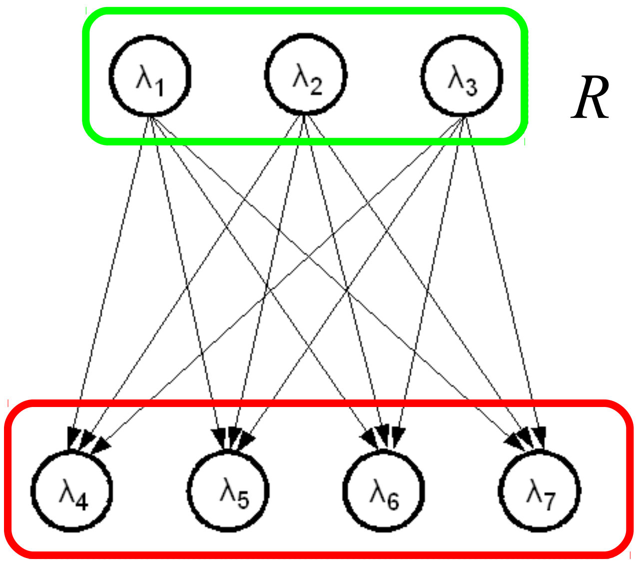
- each context is associated with multiple labels
  - e.g., keyword assignments to texts

- 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

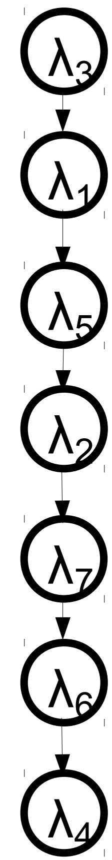
- Tranformation of Multi-Label Classification problems into preference learning problems is straight-forward



R relevant labels

$|R|\cdot|I|$ preferences

I irrelevant labels



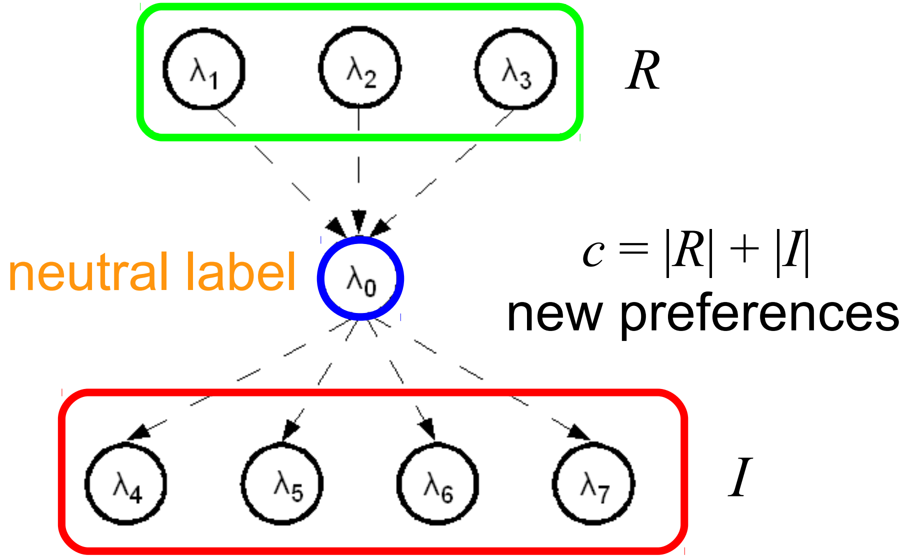
- at prediction time, the pairwise ensemble predicts a label ranking

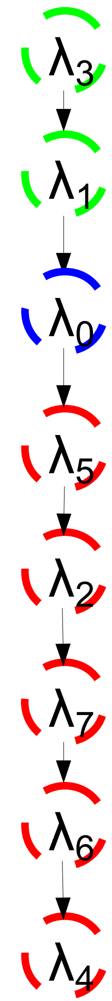
Problem:

- 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
  - can also be reduced to pairwise comparisons

Current Work

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**

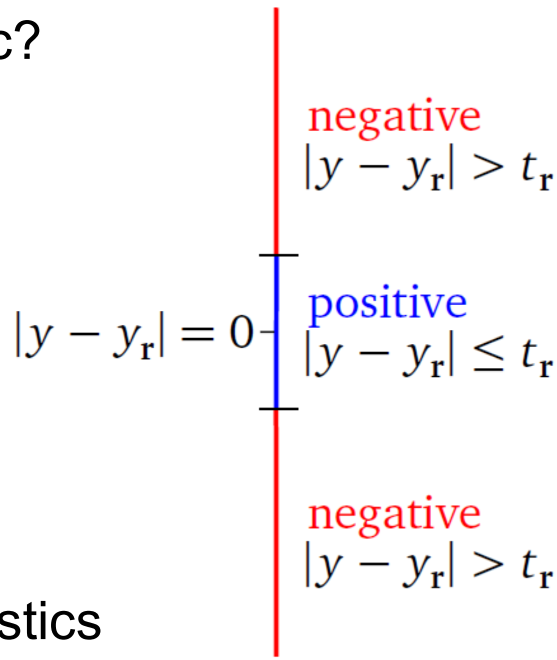
Regression

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

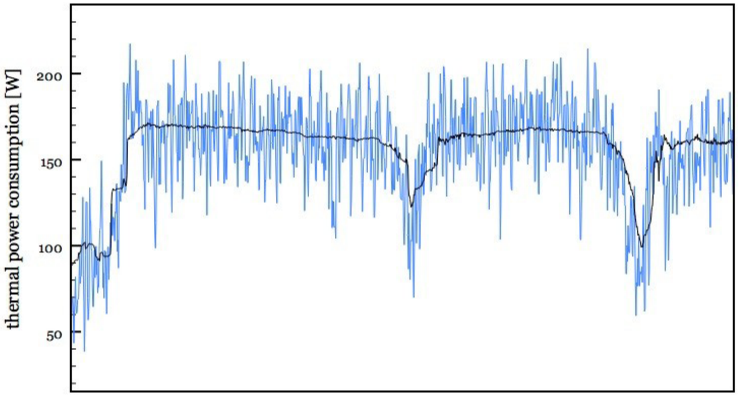
Rule-Based Regression

- 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



Application Example: Venus Express Power Consumption

- Goal
  - Learn a model of the energy consumption of the heating system of the Venus express
- Approach
  - Information about the consumption is available in hindsight
    - can be used to train a model
  - Best results obtained with ensembles of regression trees
    - local differences cannot be modeled
    - but trends can be captured well
- Partner
  - ESA / ESOC
  - University of Cordoba



Summary

- 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