# Inductive Rule Learning in a Nutshell

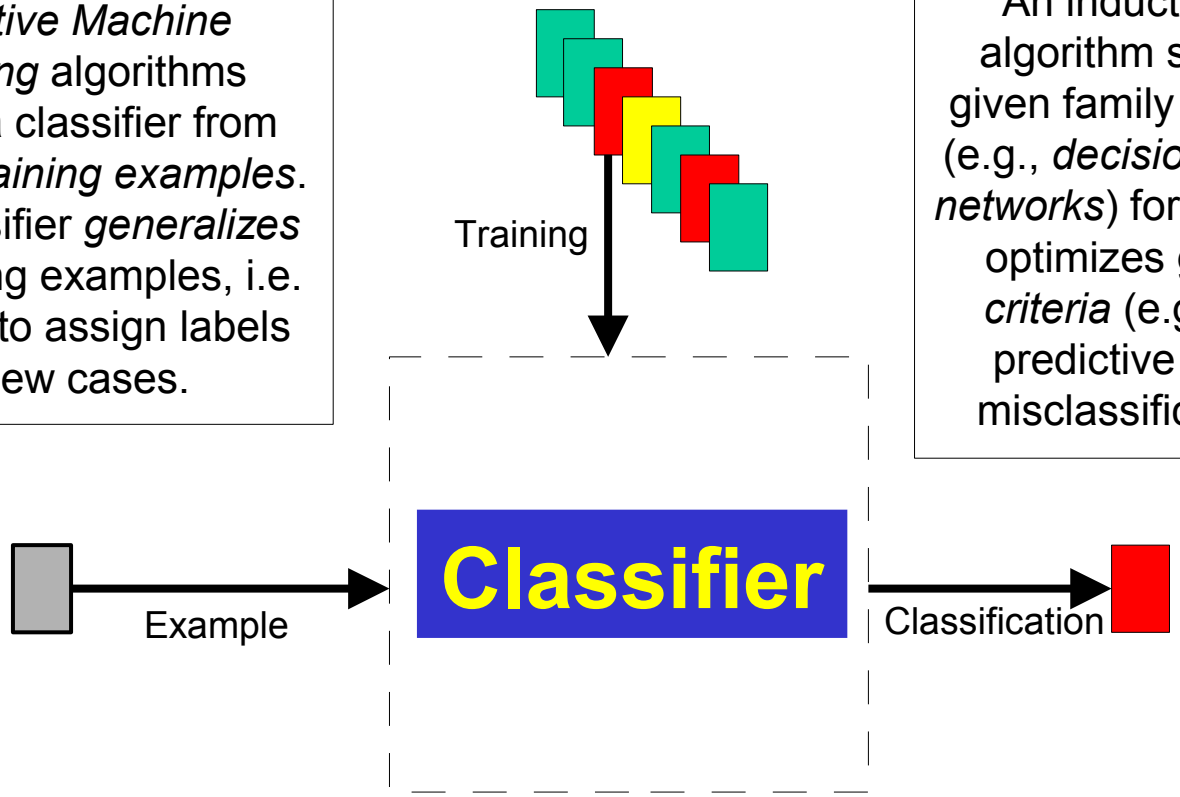
**Johannes Fürnkranz**

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduction
  - Learning Rule Sets
  - Terminology
  - Coverage Spaces
- Separate-and-Conquer Rule Learning
  - Covering algorithm
  - Top-Down Hill-Climbing
  - Rule Learning Heuristics
  - Overfitting and Pruning

# 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).

# 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

# Conjunctive Rule

**if** $(\text{att}_i = \text{val}_{iI})$ **and** $(\text{att}_j = \text{val}_{jJ})$ **then** +

Body of the rule (IF-part)

- contains a conjunction of conditions
- a condition typically consists of comparison of attribute values

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 task

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

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

# A Simple Solution

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

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

→ it must be part of the version space

- but it does not generalize to new examples!

# A Better Solution

IF Outlook = overcast THEN yes

IF Humidity = normal AND Outlook = sunny THEN yes

IF Outlook = rainy AND Windy = false THEN yes

# Separate-and-Conquer Rule Learning

- Learn a set of rules, one by one

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

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

# Relaxing Completeness and Consistency

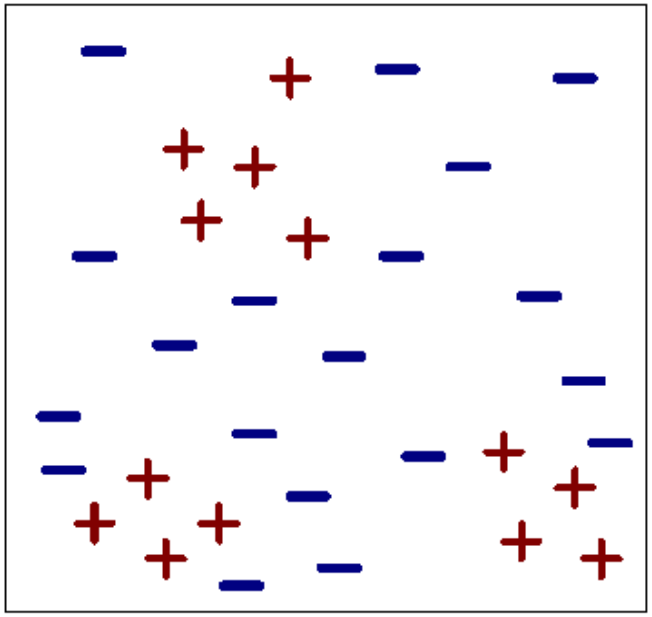
- So far we have always required a learner to learn a complete and consistent theory
  - e.g., one rule that covers all positive and no negative examples
- This is not always a good idea (→ overfitting)
- Example:

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

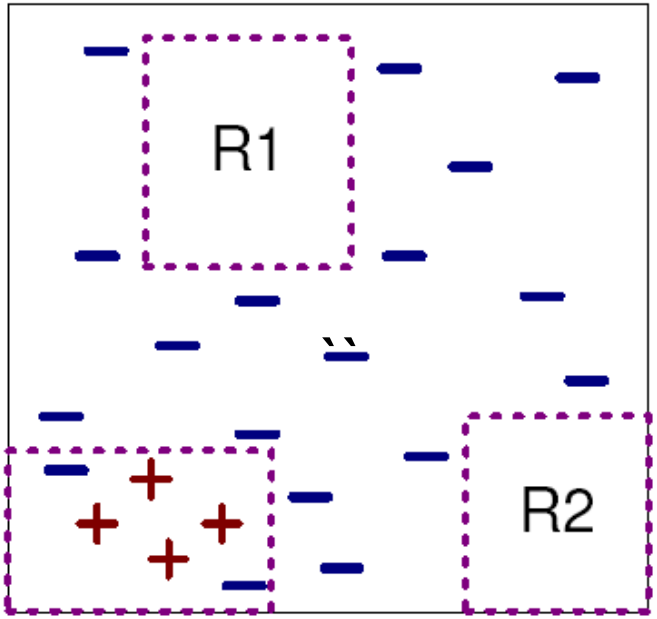
    → Theory A is complete and consistent on the training set
  - **Theory B** consists of a single rule, covering 99 positive and 1 negative example

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

# Separate-and-Conquer Rule Learning



(i) Original Data




(iv) Step 3

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

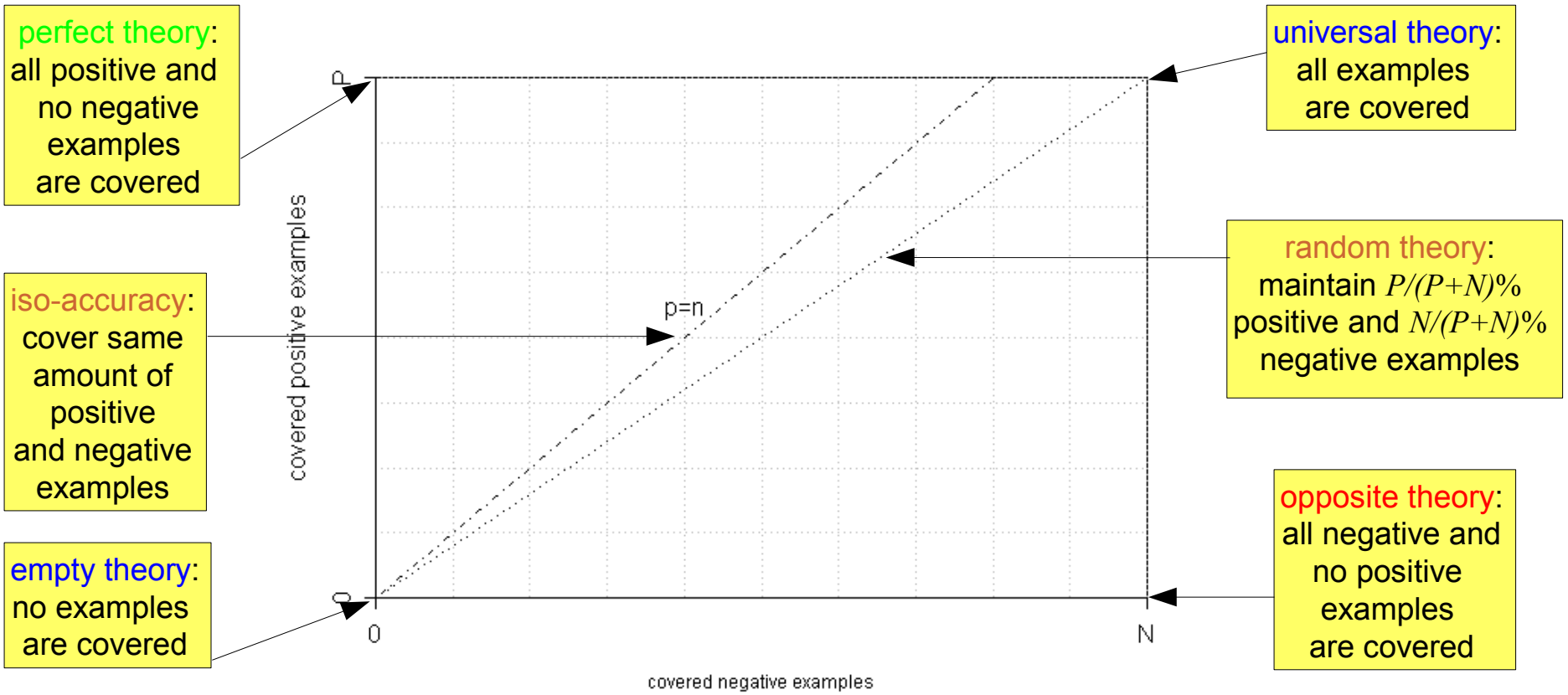
# Terminology

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

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

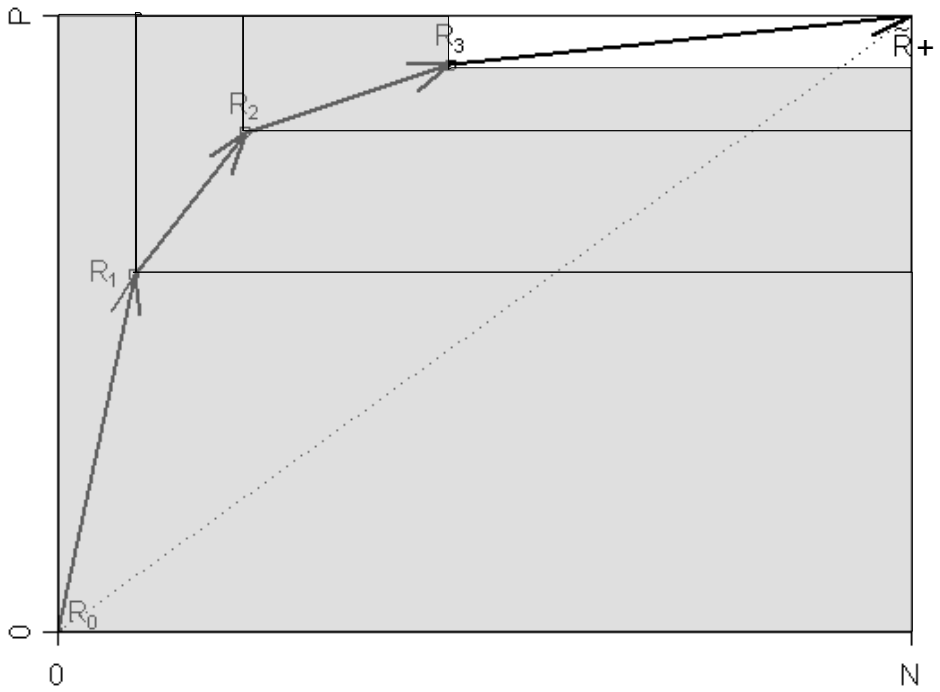
# Coverage Spaces

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



# Covering Strategy

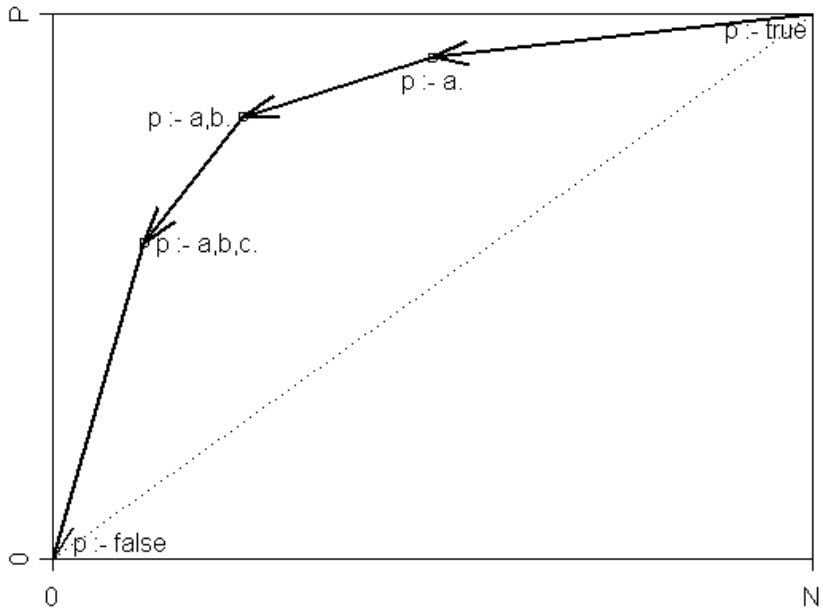
- Covering or Separate-and-Conquer rule learning learning algorithms learn one rule at a time
- This corresponds to a path in coverage space:
  - The empty theory $R_0$ (no rules) corresponds to (0,0)
  - Adding one rule never decreases $p$ or $n$ because adding a rule covers *more* examples (generalization)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



# Top-Down Hill-Climbing

- successively extends a rule by adding conditions

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




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

# Rule Learning Heuristics

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

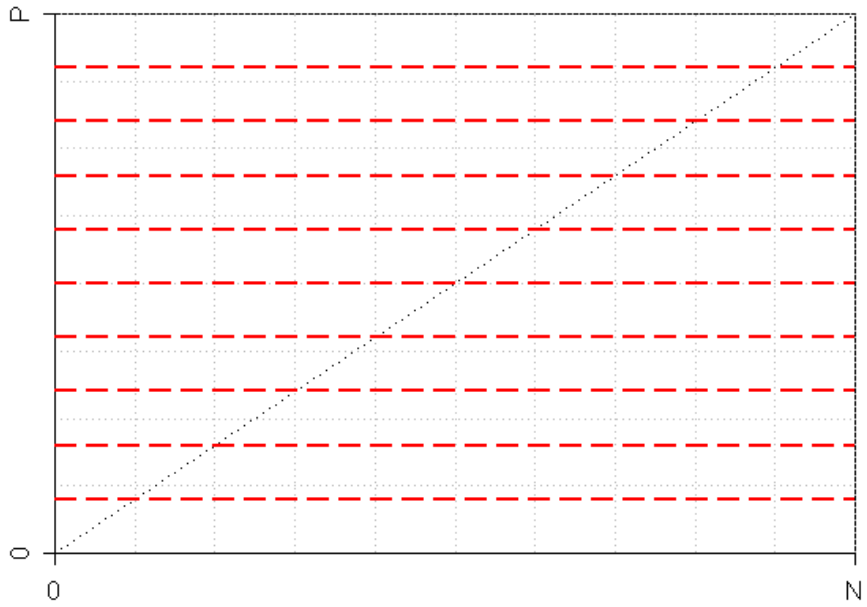
# Example

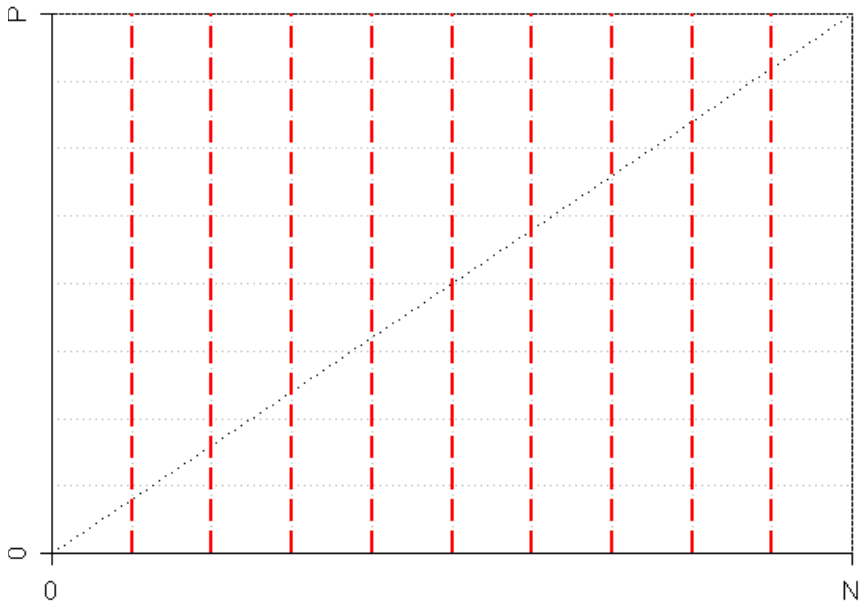
| Condition | | p | n | Precision | Laplace | p-n |
|---|---|---|---|---|---|---|
| Temperature = | Hot | 2 | 2 | 0.5000 | 0.5000 | 0 |
| | Mild | 3 | 1 | 0.7500 | 0.6667 | 2 |
| | Cold | 4 | 2 | 0.6667 | 0.6250 | 2 |
| Outlook = | Sunny | 2 | 3 | 0.4000 | 0.4286 | -1 |
| | Overcast | 4 | 0 | 1.0000 | 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)

# 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 $h_n = -n$



# Top-Down Hill-Climbing

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

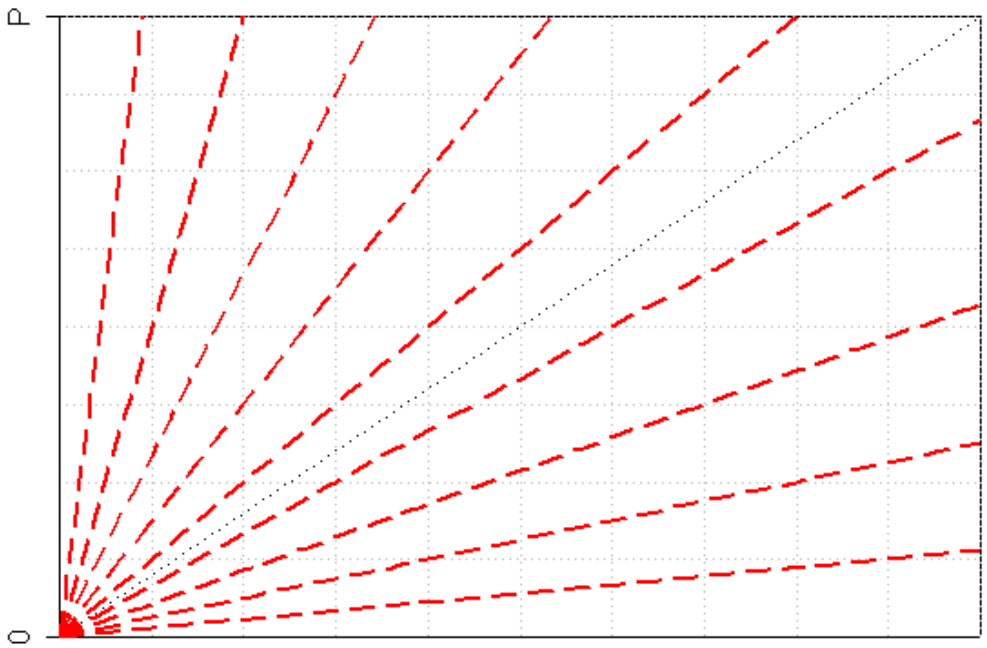
1. Start with the universal rule R that covers all examples
2. Evaluate all possible ways to add a condition to R
3. Choose the best one (according to some heuristic)
4. If R is satisfactory, return it
5. Else goto 2.

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

# Precision (Confidence)

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

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

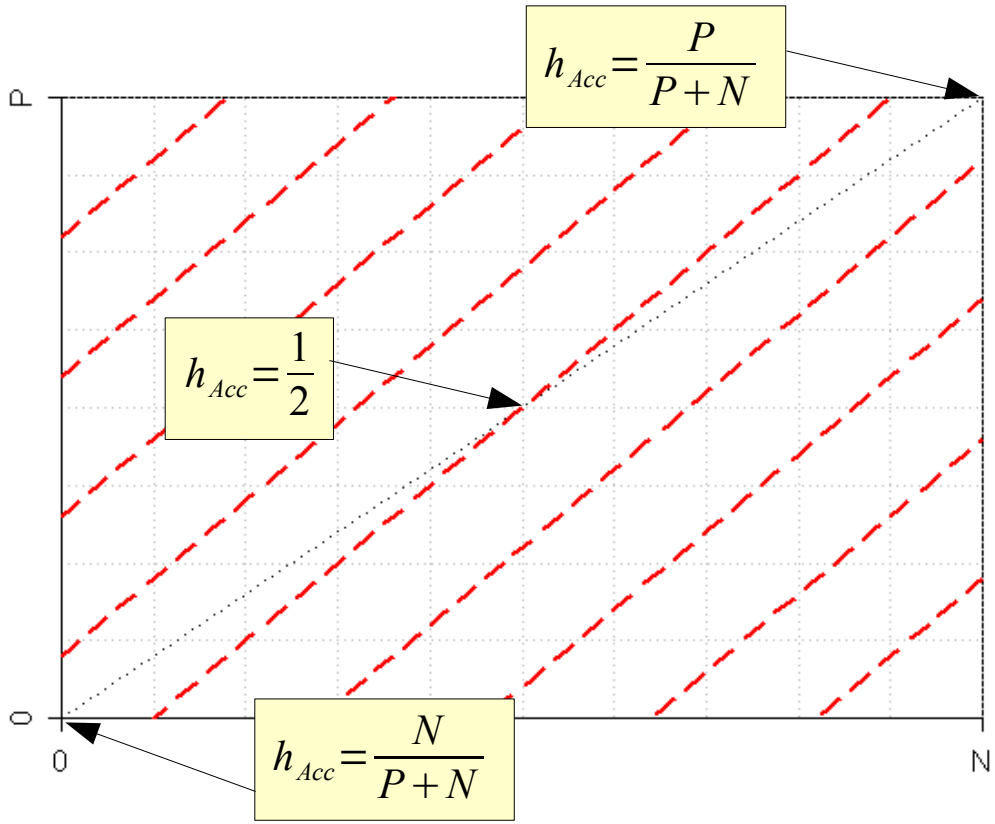


# Accuracy

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

Why are they equivalent?

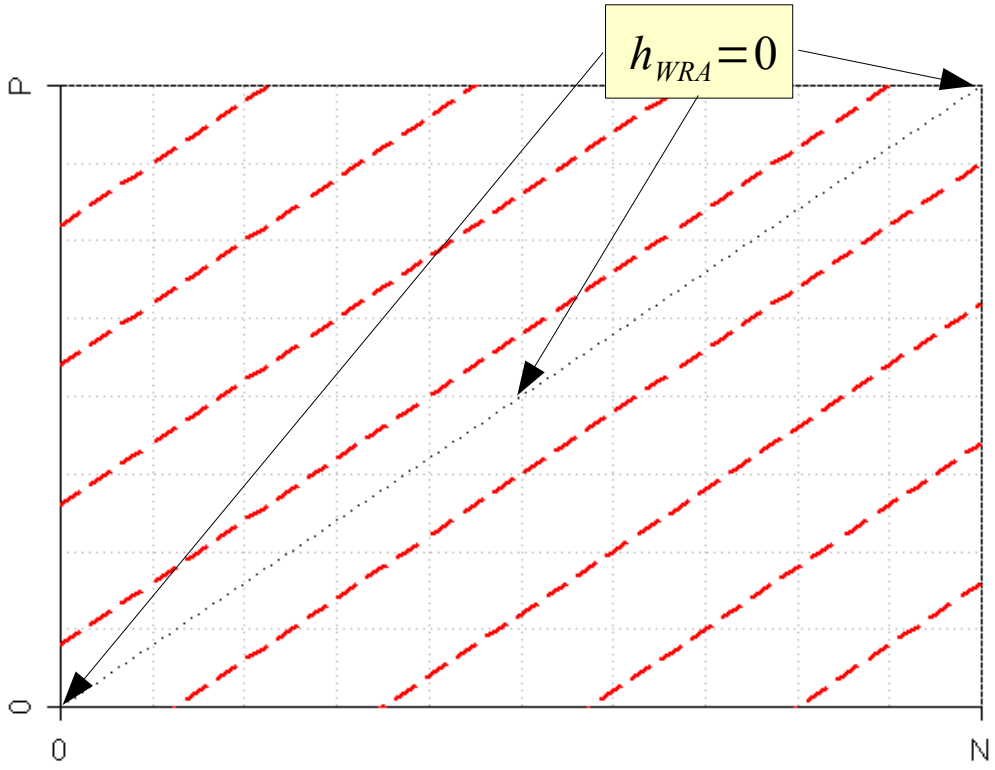
- *basic idea:*
  percentage of correct classifications (*covered positives* plus *uncovered negatives*)
- *effects:*
  - isometrics are parallel to 45º line
  - covering one positive example is as good as not covering one negative example



# Weighted Relative Accuracy

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

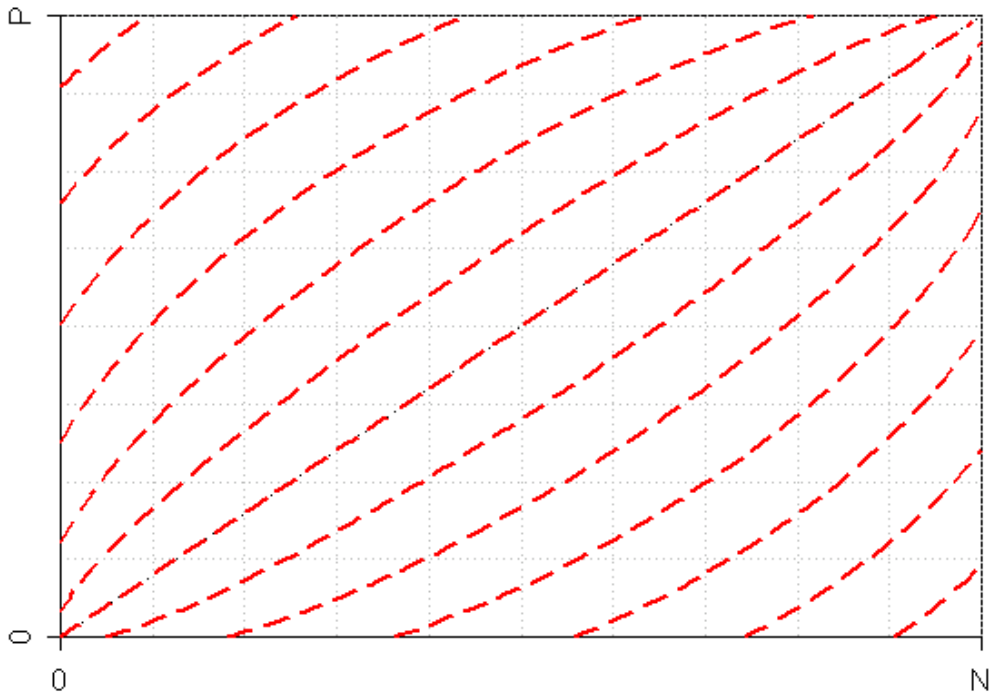
- *basic idea:*
  normalize accuracy with the class distribution
- *effects:*
  - isometrics are parallel to diagonal
  - covering $x\%$ of the positive examples is considered to be as good as not covering $x\%$ of the negative examples



# 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 $\chi^2$



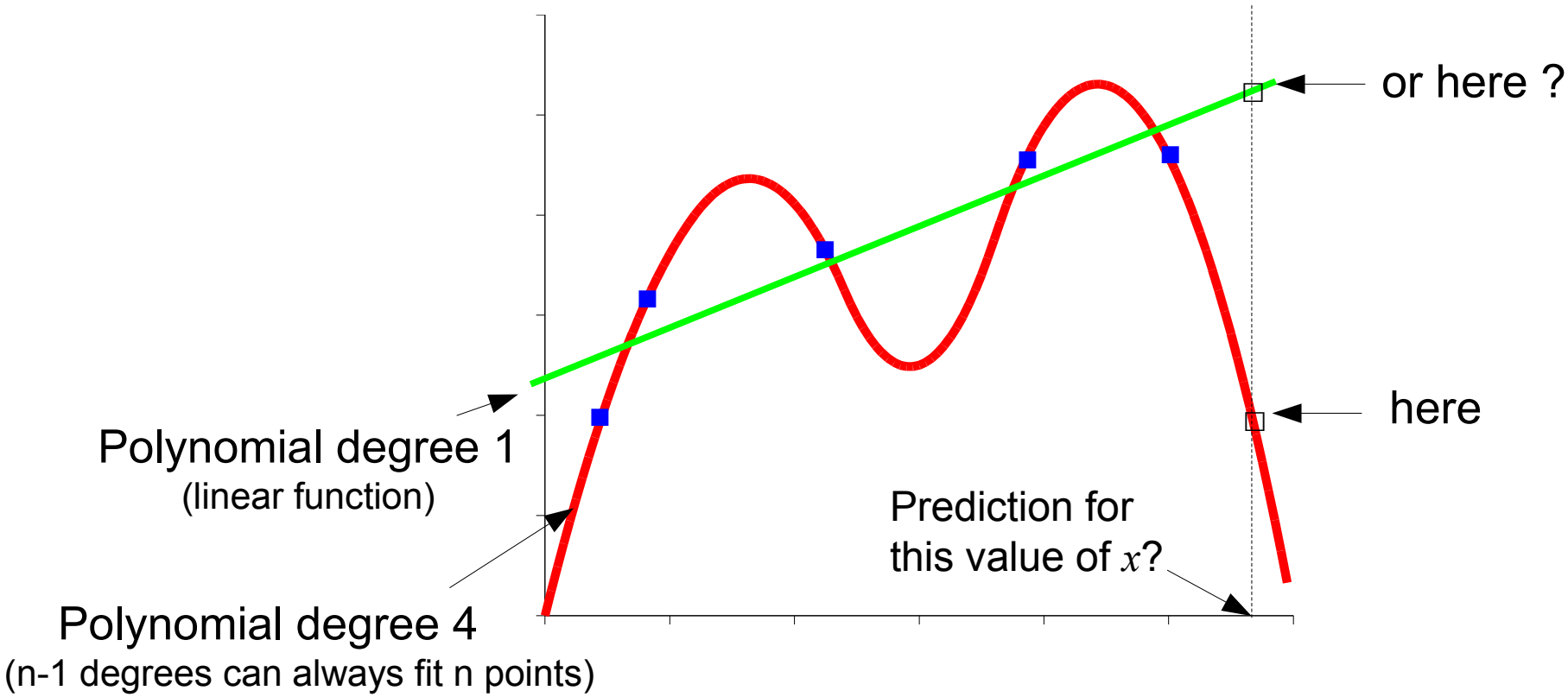
# Overfitting

- 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

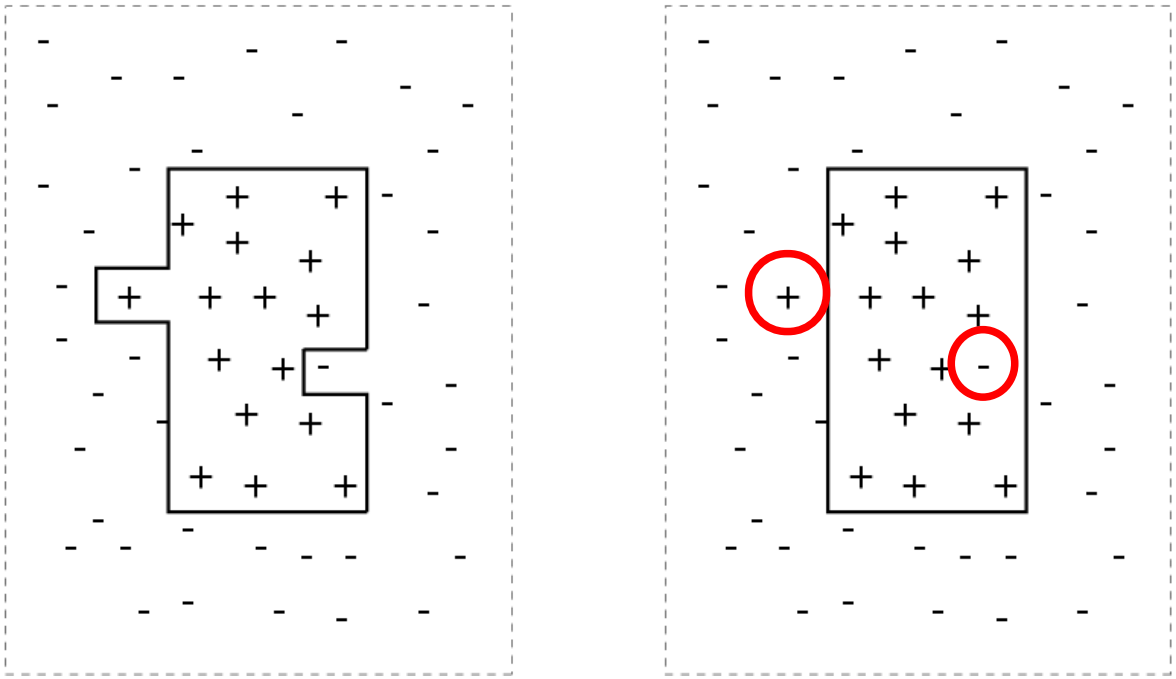
# Overfitting - Illustration



# 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

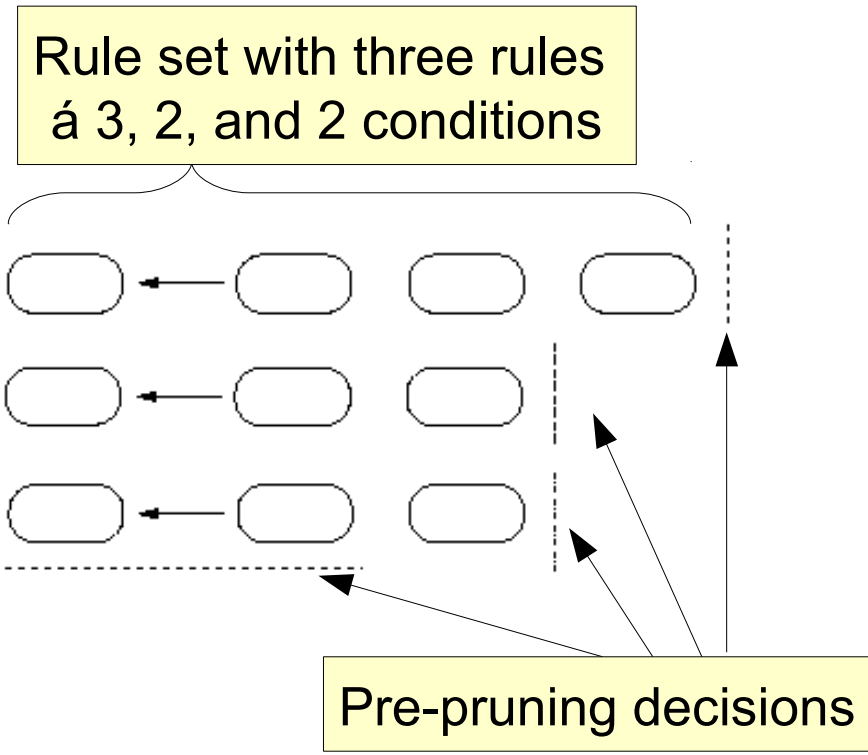
→ many learning algorithms focus on learning simple concepts

  - 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

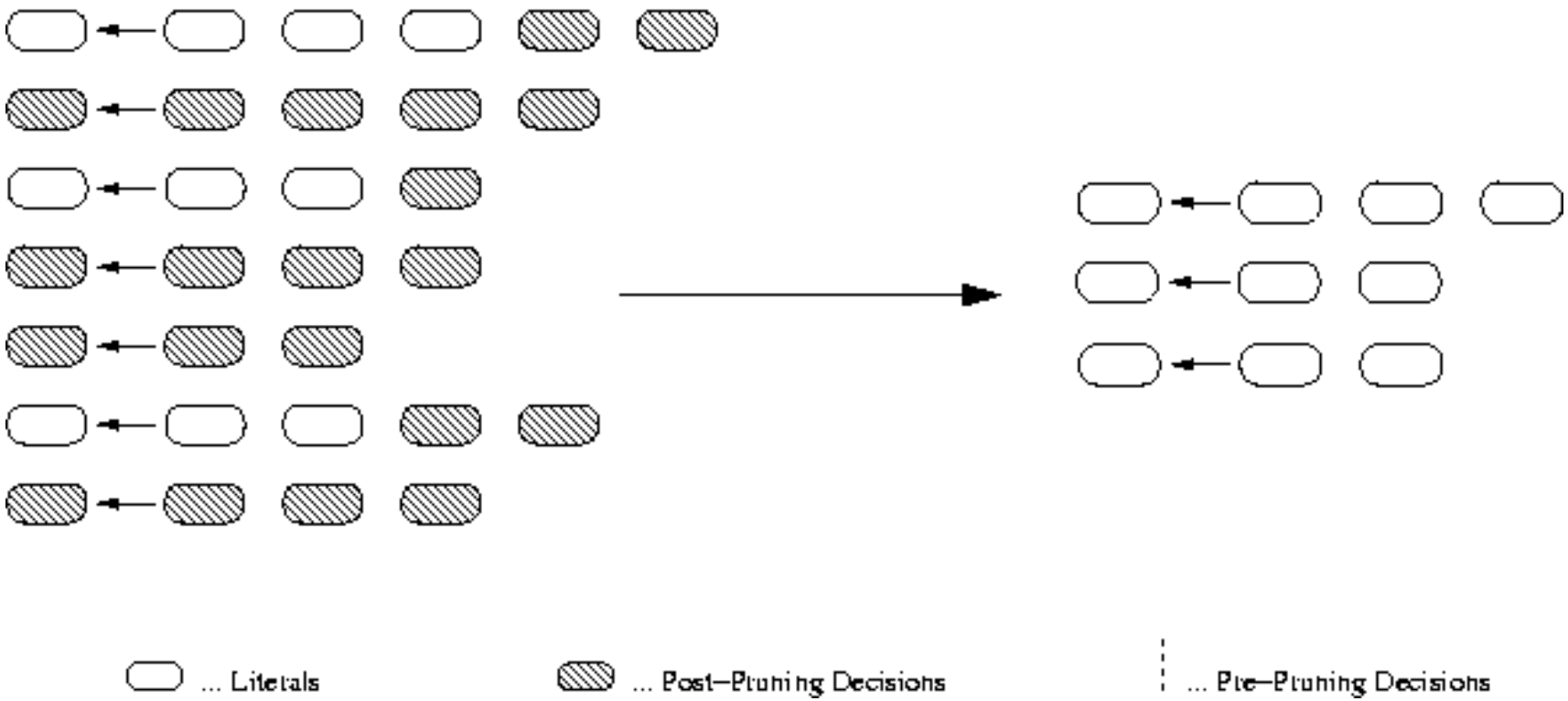
- Pruning: Complex rules will be simplified
  - **Pre-Pruning:**
    - during learning
  - **Post-Pruning:**
    - after learning

# 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



# Post Pruning



# Post-Pruning: Example

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

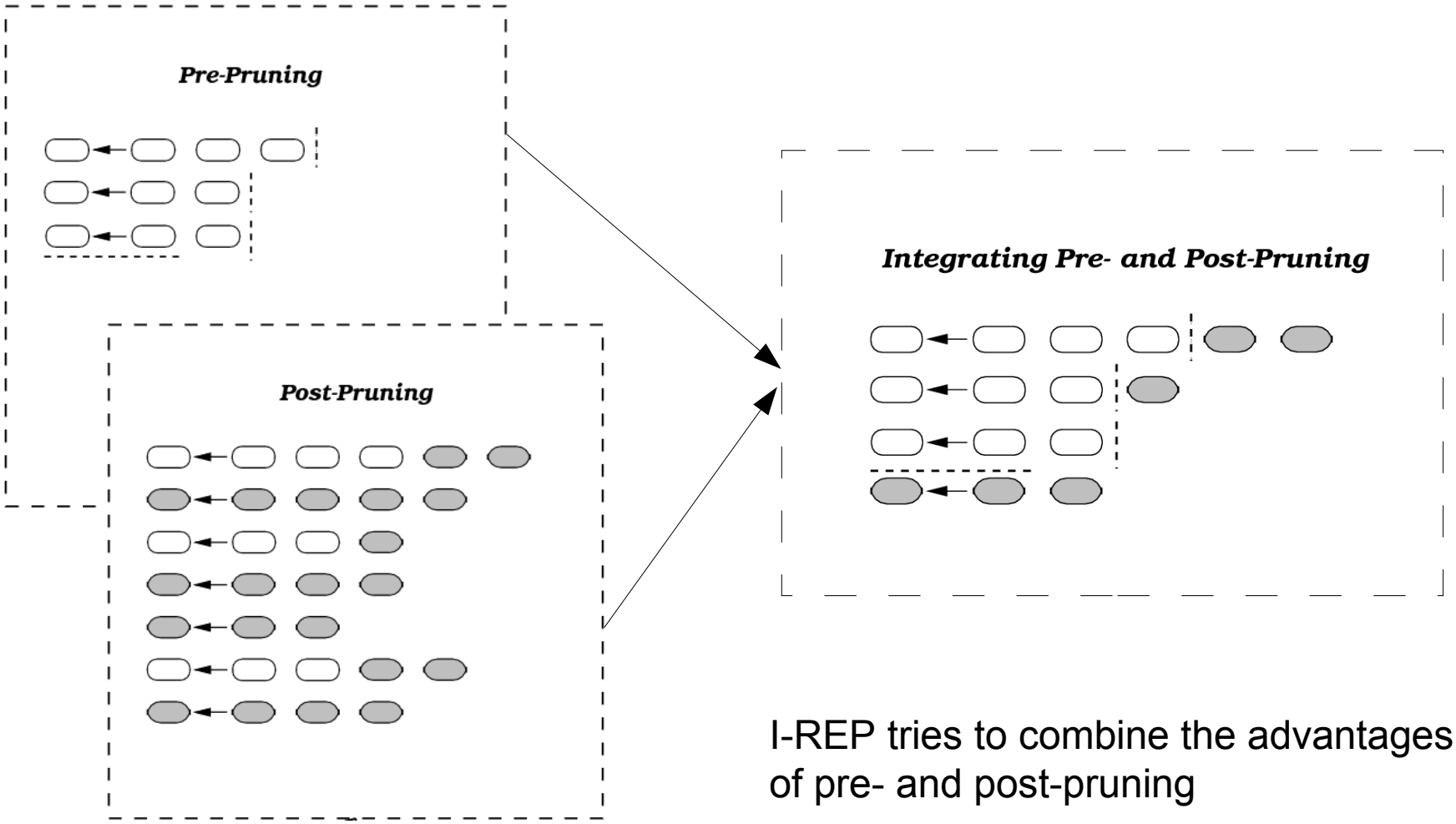
# Post-Pruning: Example

IF H=high AND O=sunny THEN no

IF O=rain AND W=true THEN no

ELSE yes

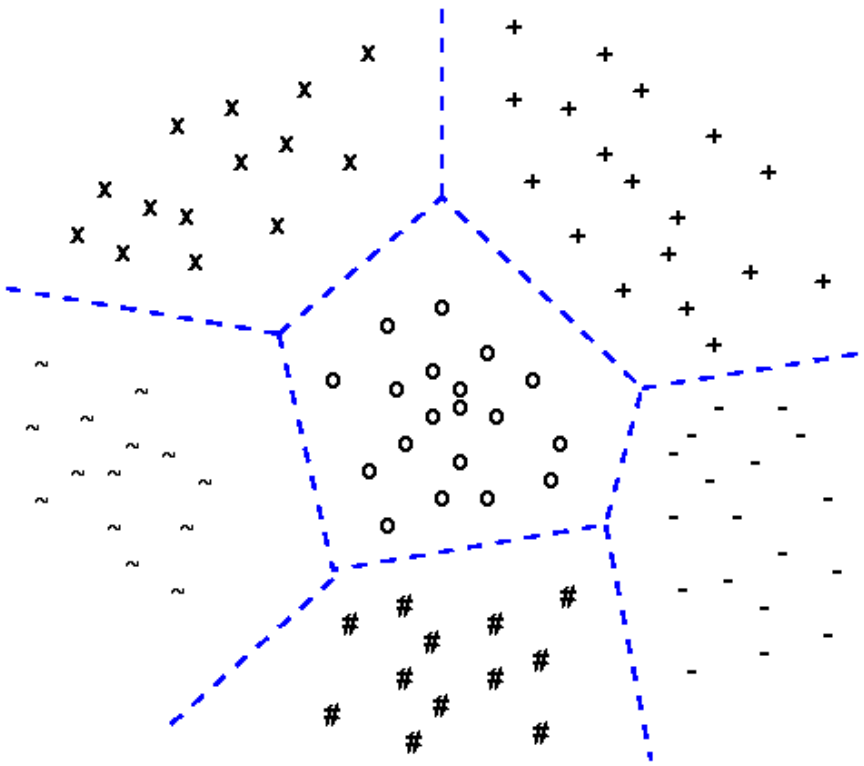
# Incremental Reduced Error Pruning



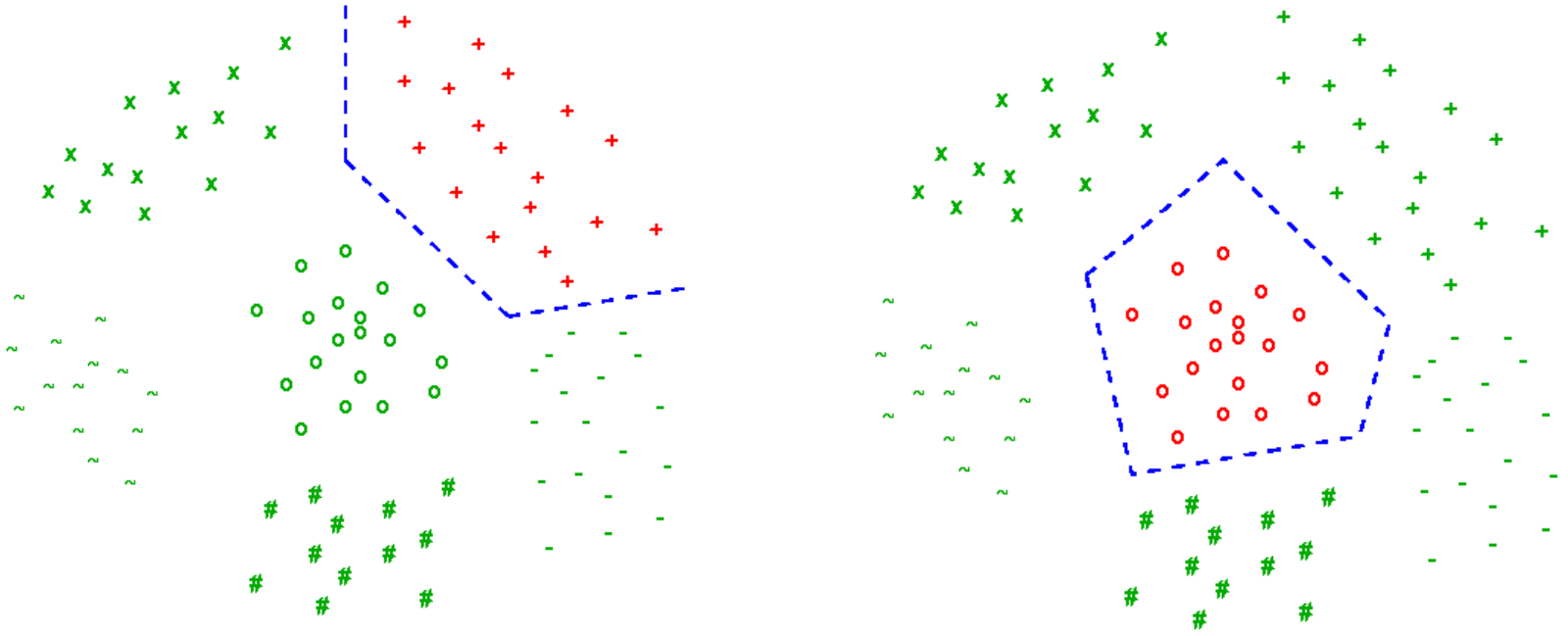

I-REP tries to combine the advantages of pre- and post-pruning

# 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



# One-against-all binarization

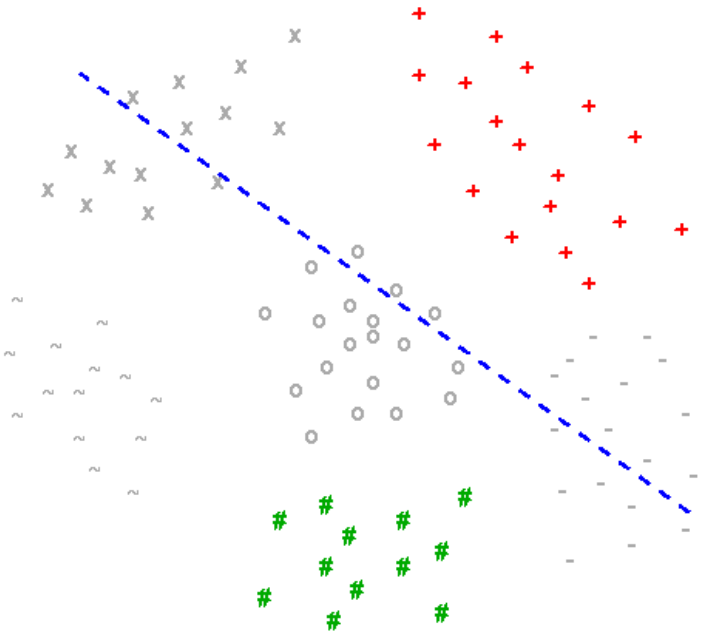


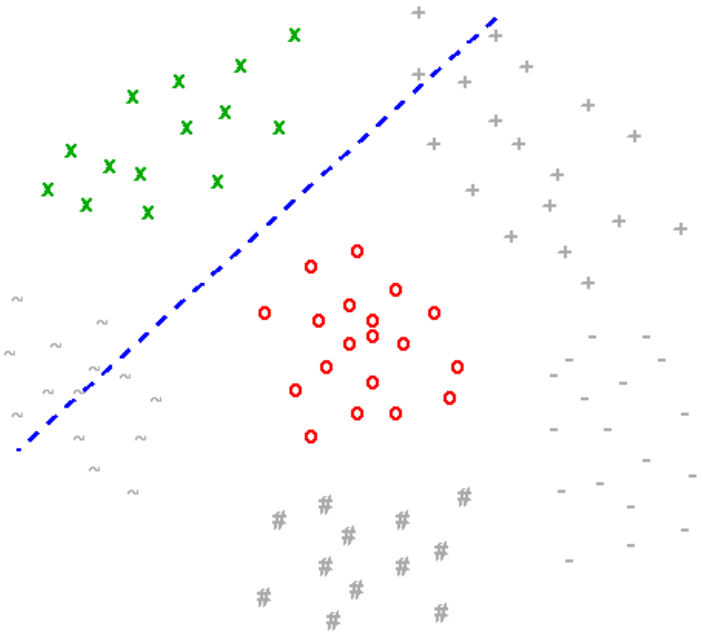
Treat each class as a separate concept:

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

# Pairwise Classification

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





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

# Accuracy

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

# 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