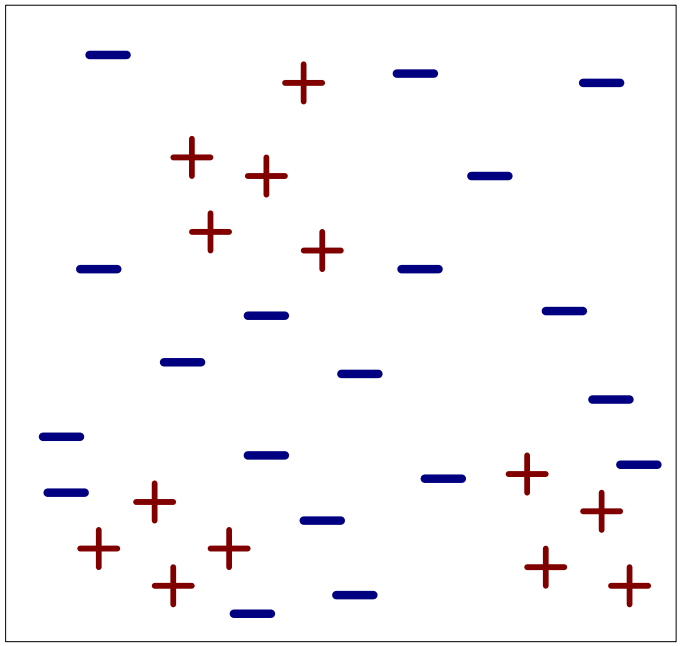
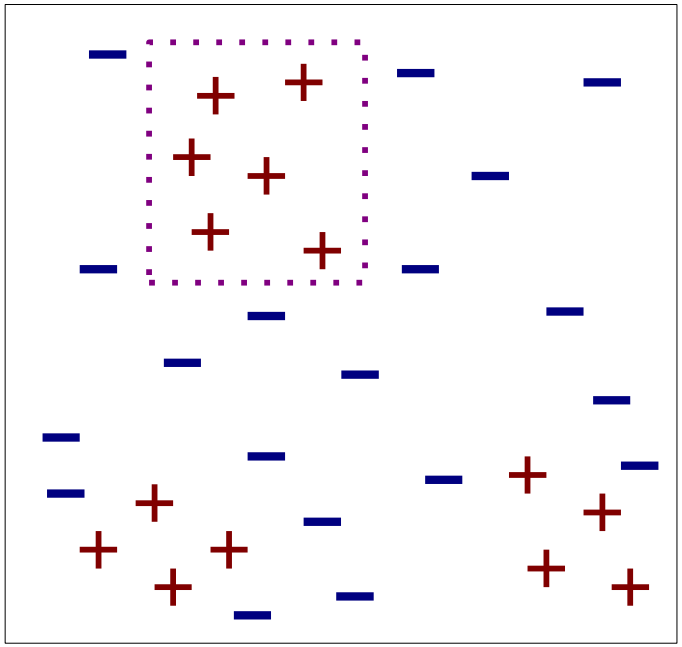
Direct Method: Sequential Covering

1. Start from an empty rule
2. Grow a rule using the Learn-One-Rule function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Example of Sequential Covering

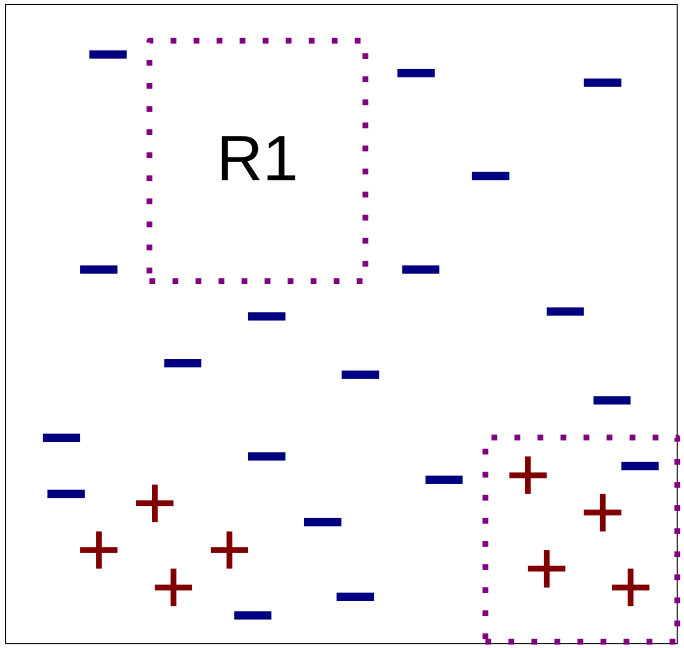


(i) Original Data

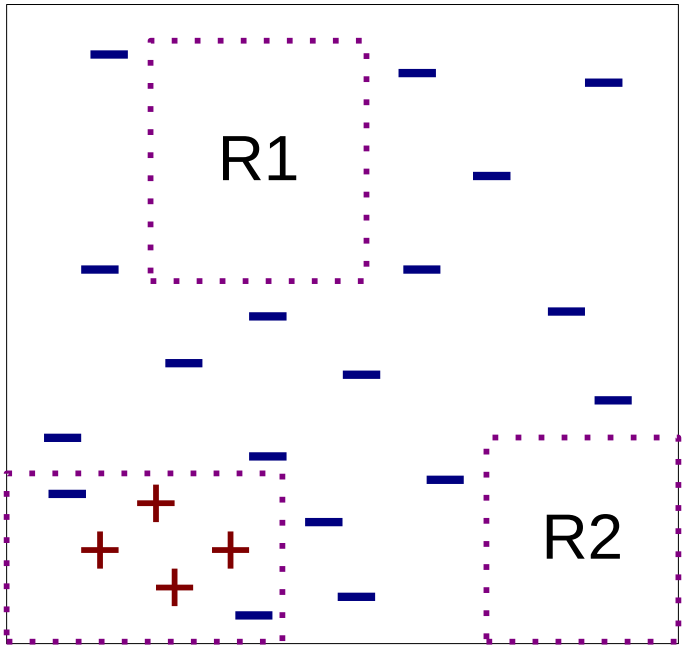


(ii) Step 1

Example of Sequential Covering…



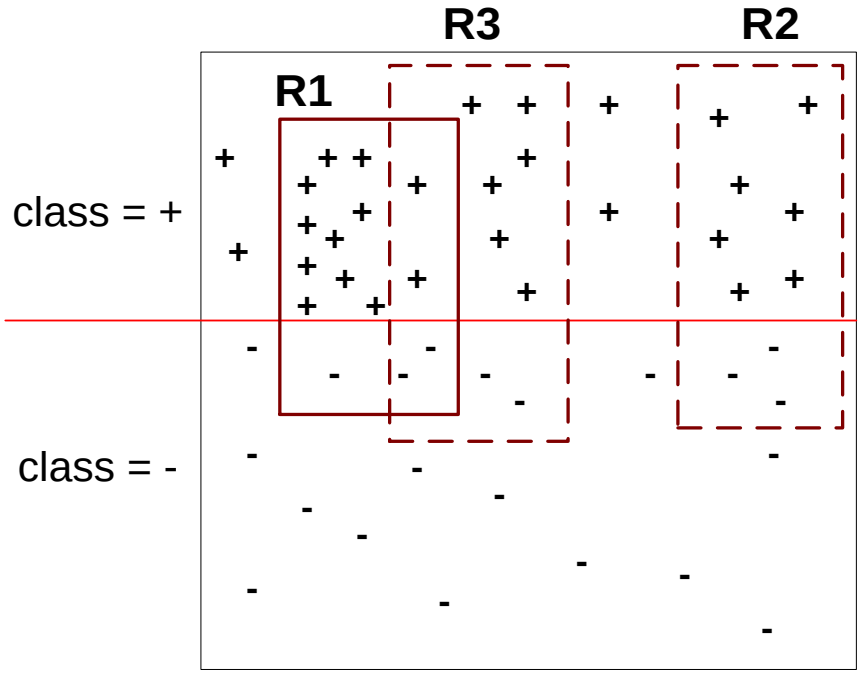
(iii) Step 2



(iv) Step 3

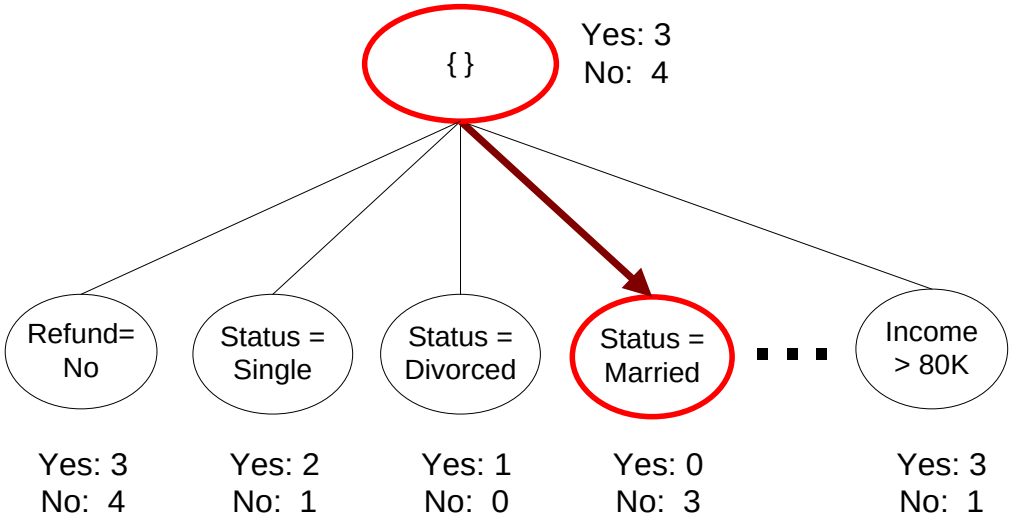
Instance Elimination

- Why do we need to eliminate instances?
  - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
  - Ensure that the next rule is different
- Why do we remove negative instances?
  - Prevent underestimating accuracy of rule
  - Compare rules R2 and R3 in the diagram

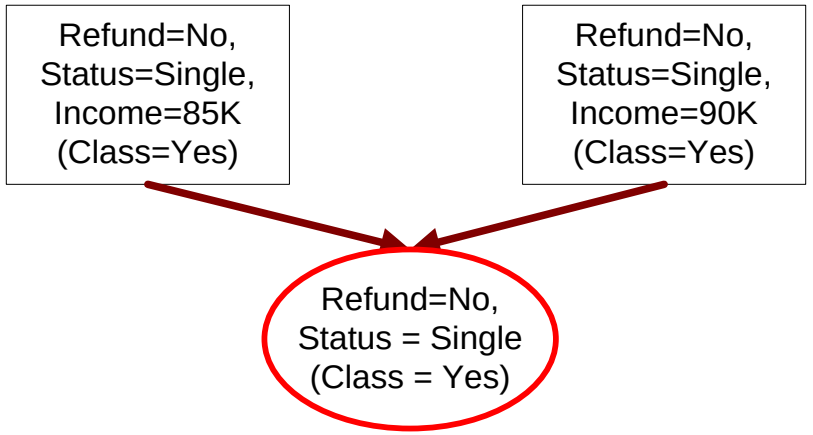


Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

Rule Evaluation

- Foil's Information Gain

FOIL: First Order Inductive Learner – an early rule-based learning algorithm

  - R0: {} => class (initial rule)
  - R1: {A} => class (rule after adding conjunct)
  - Gain(R0, R1) = t [log (p1/(p1+n1)) – log (p0/(p0 + n0))]
  - where t: number of positive instances covered by both R0 and R1

    p0: number of positive instances covered by R0

    n0: number of negative instances covered by R0

    p1: number of positive instances covered by R1

    n1: number of negative instances covered by R1

Direct Method: RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn rules for positive class
  - Negative class will be default class
- For multi-class problem
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class

Direct Method: RIPPER

- Growing a rule:
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL's information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning: $v = (p-n)/(p+n)$
    - p: number of positive examples covered by the rule in the validation set
    - n: number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes v

Direct Method: RIPPER

- Building a Rule Set:
  - Use sequential covering algorithm
    - Finds the best rule that covers the current set of positive examples
    - Eliminate both positive and negative examples covered by the rule
  - Each time a rule is added to the rule set, compute the new description length
    - Stop adding new rules when the new description length is d bits longer than the smallest description length obtained so far

Direct Method: RIPPER

- Optimize the rule set:
  - For each rule *r* in the rule set ***R***
    - Consider 2 alternative rules:
      - Replacement rule (r*): grow new rule from scratch
      - Revised rule(r′): add conjuncts to extend the rule *r*
    - Compare the rule set for *r* against the rule set for r* and r′
    - Choose rule set that minimizes MDL principle
  - Repeat rule generation and rule optimization for the remaining positive examples