RULE LEARNING

Minería de Datos: Preprocesamiento y clasificación

Descripción

- **Temario**:
  - Modelos no lineales.
  - Árboles de Decisión. Multiclasificadores.
  - Descomposición de problemas multiclase.
  - Aprendizaje de Reglas.
  - Máquinas soporte vectorial (SVM).
  - Preprocesamiento de Datos.

- **Bibliografía**:

  - “An Introduction to Statistical Learning with Applications in R”, Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, Springer, 2013.
  - “Introduction to Data Mining”, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Pearson, 2013.
  - “Foundations of Rule Learning”, Johannes Fürnkranz, Dragan Gambergerm Nada Lavrac, Springer, 2012.
  - “Data Preprocessing in Data Mining”. Salvador García, Julián Luengo, Francisco Herrera, Springer, 2015.

- **Relacionado con**: Introducción a la Programación para Ciencia de Datos e Introducción a la Ciencia de Datos

Descripción

- **Temario**:
  - Modelos no lineales.
  - Árboles de Decisión. Multiclasificadores.
  - Descomposición de problemas multiclase.
  - Aprendizaje de Reglas.
  - Máquinas soporte vectorial (SVM).
  - Preprocesamiento de Datos.

- **Bibliografía**:
  - "An Introduction to Statistical Learning with Applications in R", Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, Springer, 2013.
  - "Introduction to Data Mining", Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Pearson, 2013.
  - "Foundations of Rule Learning", Johannes Fürnkranz, Dragan Gambergerm Nada Lavrac, Springer, 2012.
  - "Data Preprocessing in Data Mining". Salvador García, Julián Luengo, Francisco Herrera, Springer, 2015.

- **Relacionado con**: Introducción a la Programación para Ciencia de Datos e Introducción a la Ciencia de Datos

Learning Rules

- If-then rules in logic are a standard representation of knowledge that have proven useful in expert-systems and other AI systems.
- Rules are fairly easy for people to understand and therefore can help provide insight and comprehensible results for human users.
- Methods for automatically inducing rules from data have been shown to build more accurate expert systems than human knowledge engineering for some applications.
- Rule-learning methods have been extended to first-order logic to handle relational (structural) representations.

Rule-Based Classifier

- Classify records by using a collection of "if…then…" rules
- Rule: ($Condition$) $\rightarrow y$
  - where
    - *Condition* is a conjunctions of attributes
    - y is the class label
  - *LHS*: rule antecedent or precondition
  - *RHS*: rule consequent
  - Examples of classification rules:
    - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Evade=No

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Rule Coverage and Accuracy

- A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule

  R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

  R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

  R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

  R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

  R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

- The quality of a classification rule can be evaluated using the measures:
  - **Coverage of a rule**:
    - Fraction of records that satisfy the antecedent of a rule
  - **Accuracy of a rule**:
    - Fraction of records that satisfy both the antecedent and consequent of a rule

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	**Single**	125K	**No**
2	No	Married	100K	**No**
3	No	**Single**	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	**Single**	85K	**Yes**
9	No	Married	75K	**No**
10	No	**Single**	90K	**Yes**

(Status=Single) $\rightarrow$ No

Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

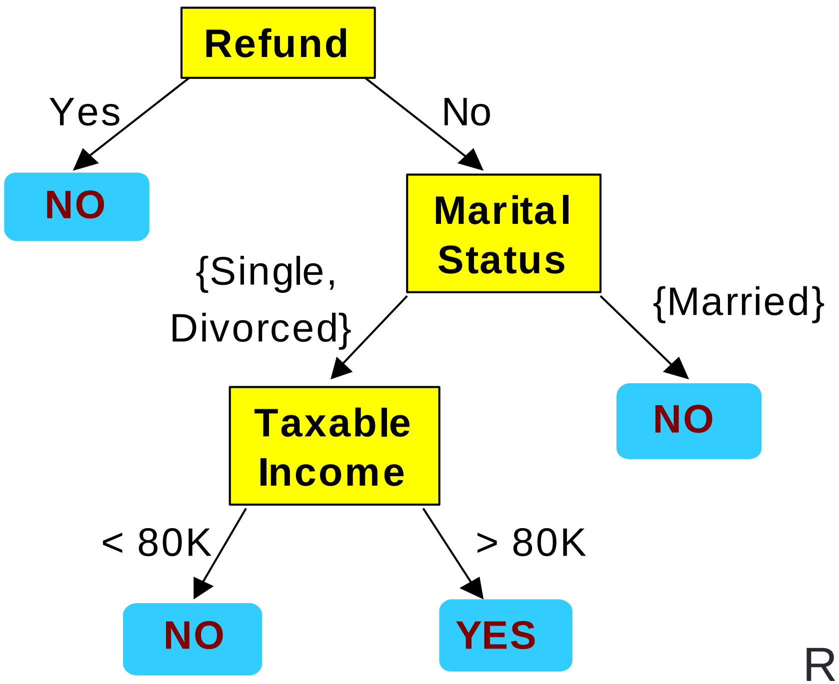
A dogfish shark triggers none of the rules

Characteristics of Rule-Based Classifier

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule

- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

From Decision Trees To Rules



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

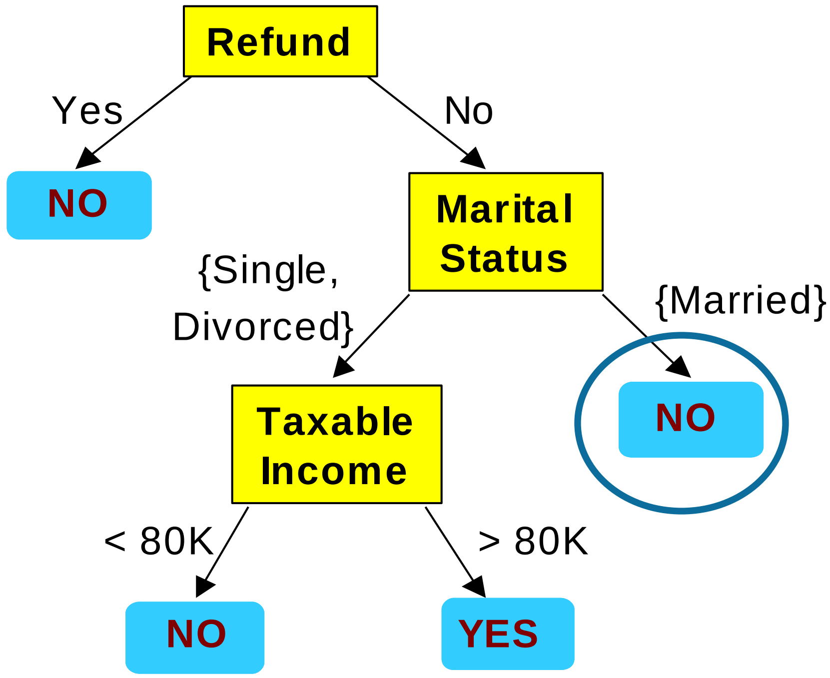
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive

Rule set contains as much information as the tree

Rules Can Be Simplified



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Initial Rule: (Refund=No) $\wedge$ (Status=Married) $\rightarrow$ No

Simplified Rule: (Status=Married) $\rightarrow$ No

Effect of Rule Simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes

- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Rule Ordering Schemes

- Rule-based ordering
  - Individual rules are ranked based on their quality
- Class-based ordering
  - Rules that belong to the same class appear together

Rule-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Class-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Married}) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

Building Classification Rules

- Direct Method:
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, FOIL, …

- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - e.g: C4.5rules

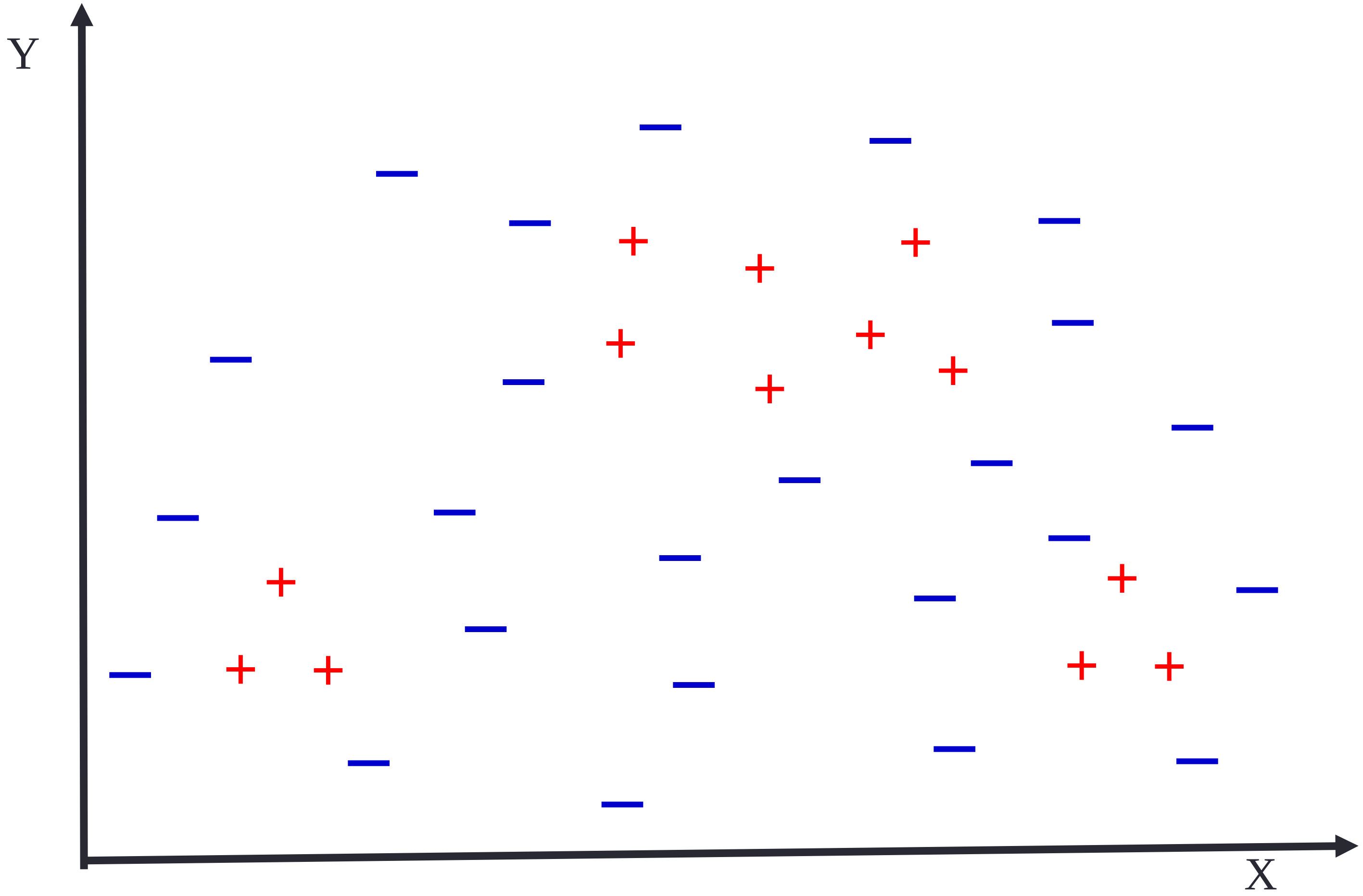
Direct Method: Sequential Covering

- A set of rules is learned one at a time, each time finding a single rule that covers a large number of positive instances without covering any negatives, removing the positives that it covers, and learning additional rules to cover the rest.

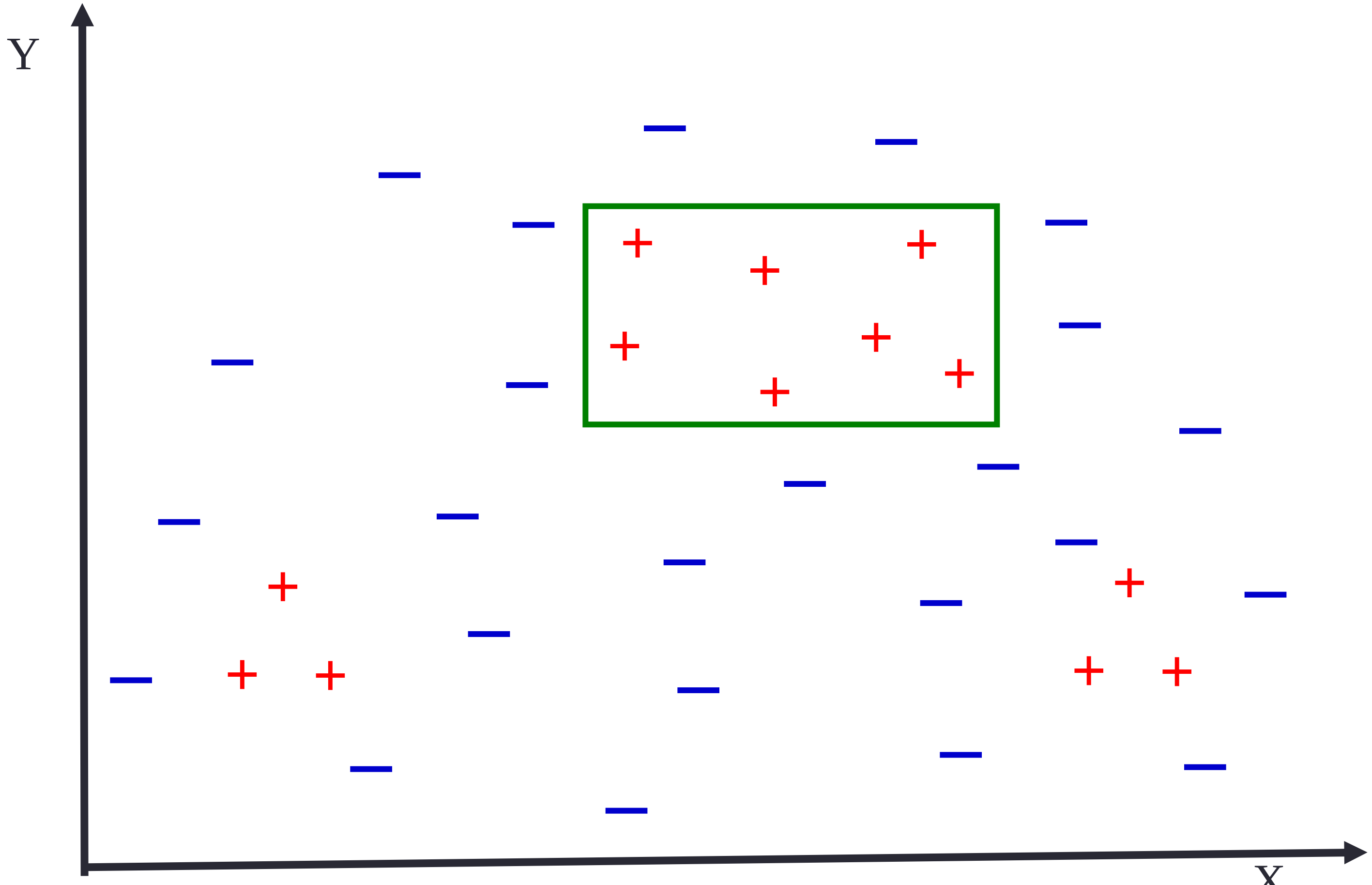
> Let *P* be the set of positive examples
> Until *P* is empty do:
>> Learn a rule *R* that covers a large number of elements of *P* but no negatives.
>> Add *R* to the list of rules.
>> Remove positives covered by *R* from *P*

- This is an instance of the greedy algorithm for minimum set covering and does not guarantee a minimum number of learned rules.
- Minimum set covering is an NP-hard problem and the greedy algorithm is a standard approximation algorithm.
- Methods for learning individual rules vary.

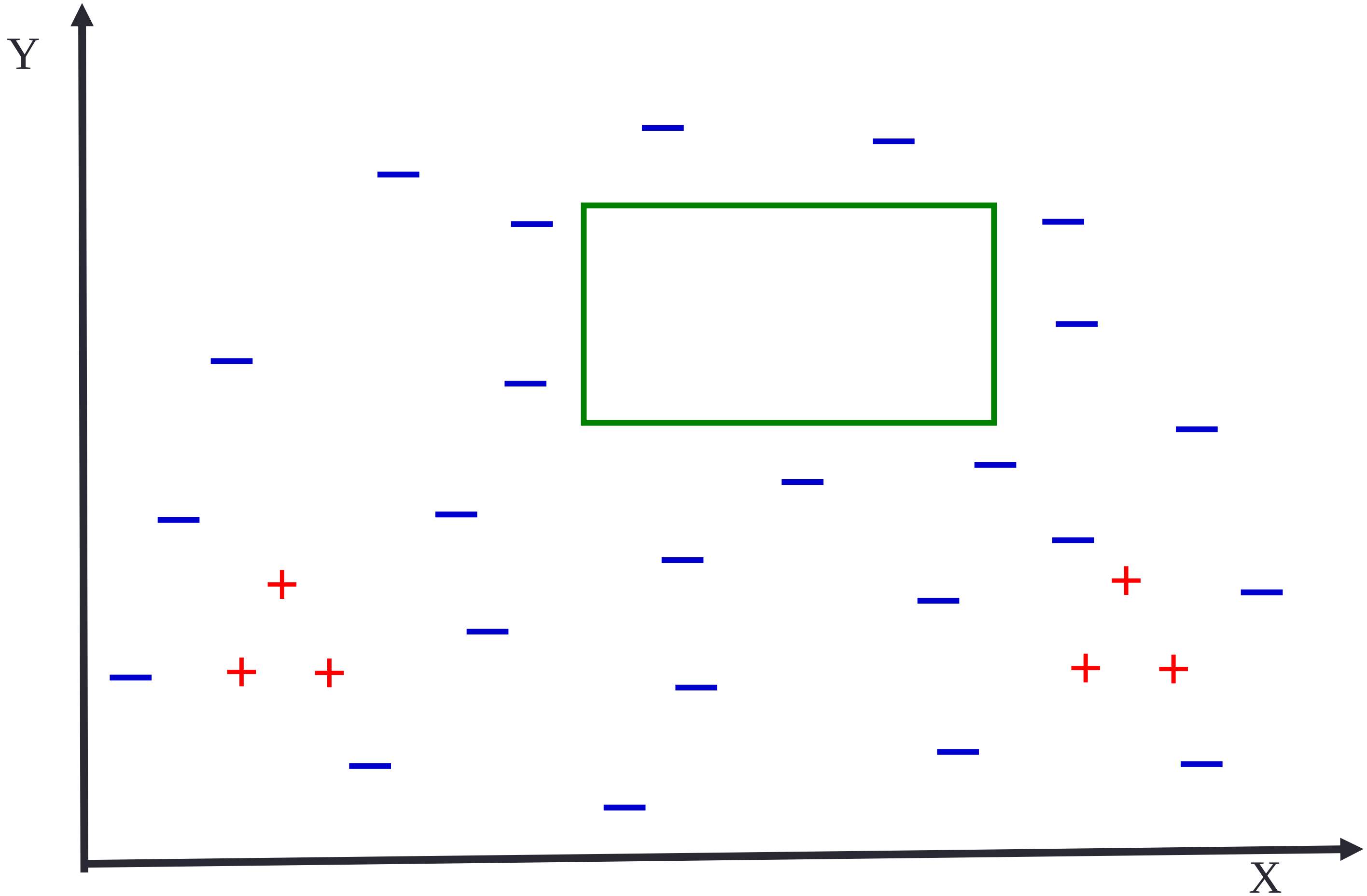
Example of Sequential Covering



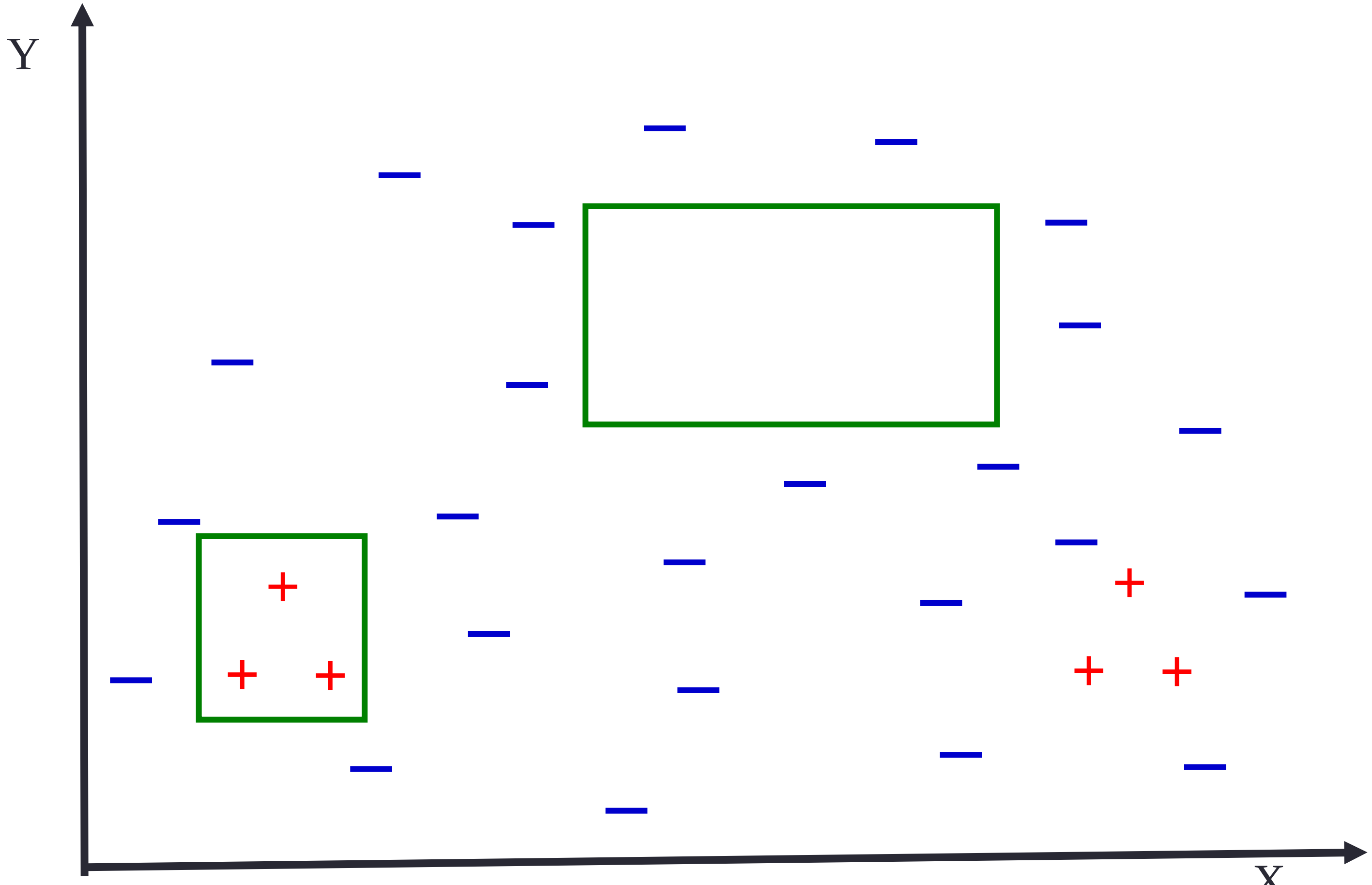
Example of Sequential Covering



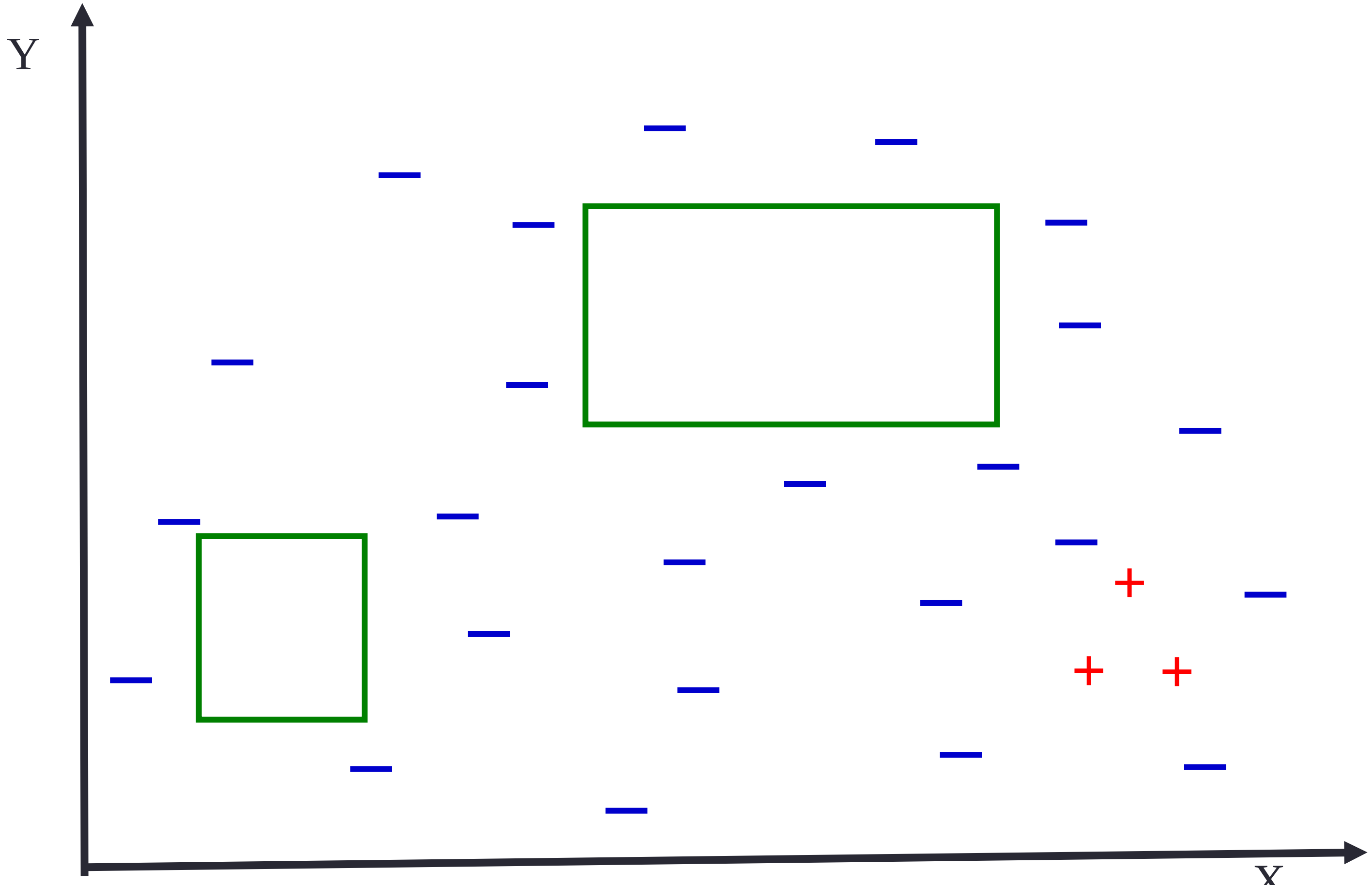
Example of Sequential Covering



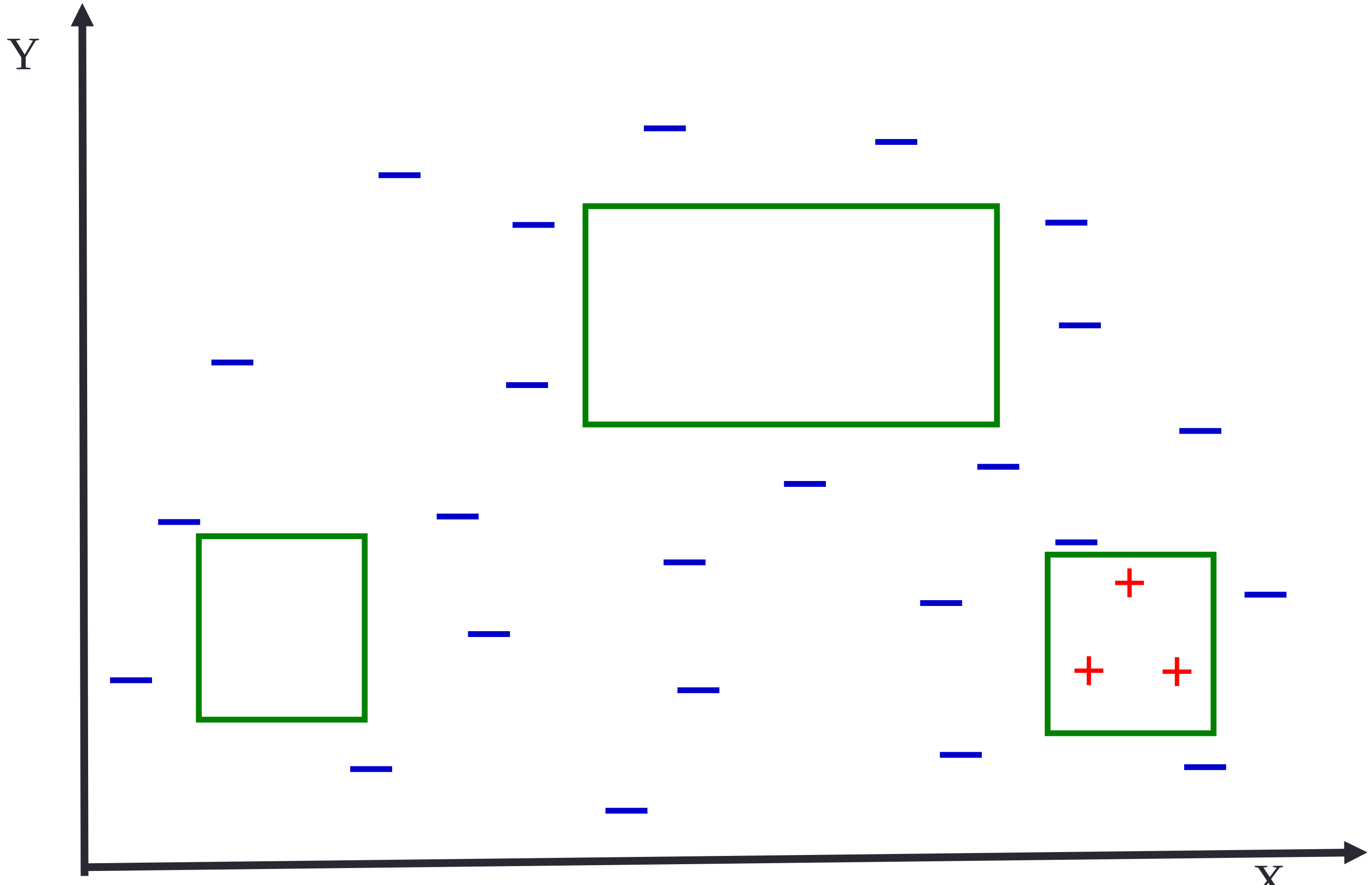
Example of Sequential Covering



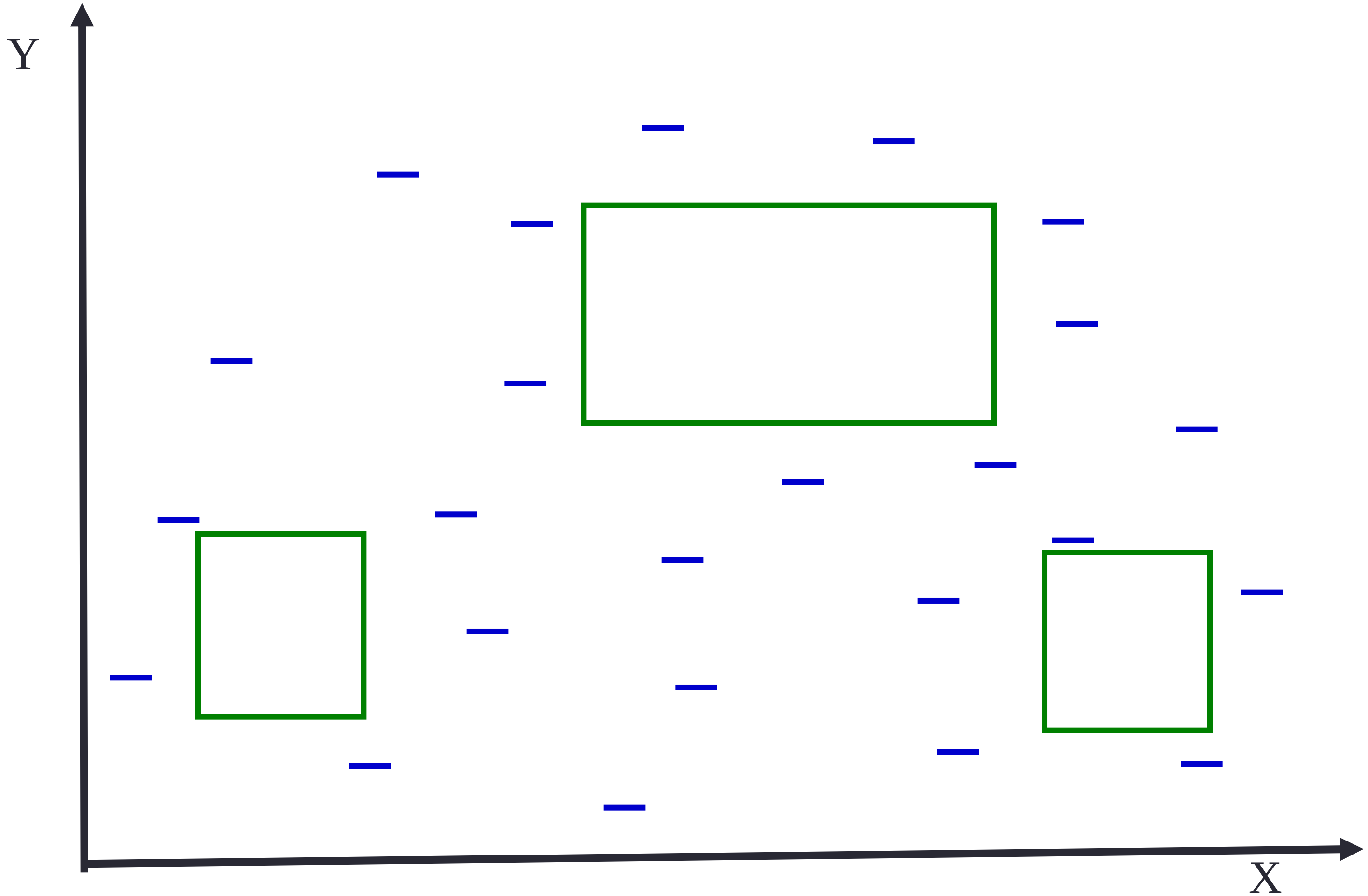
Example of Sequential Covering



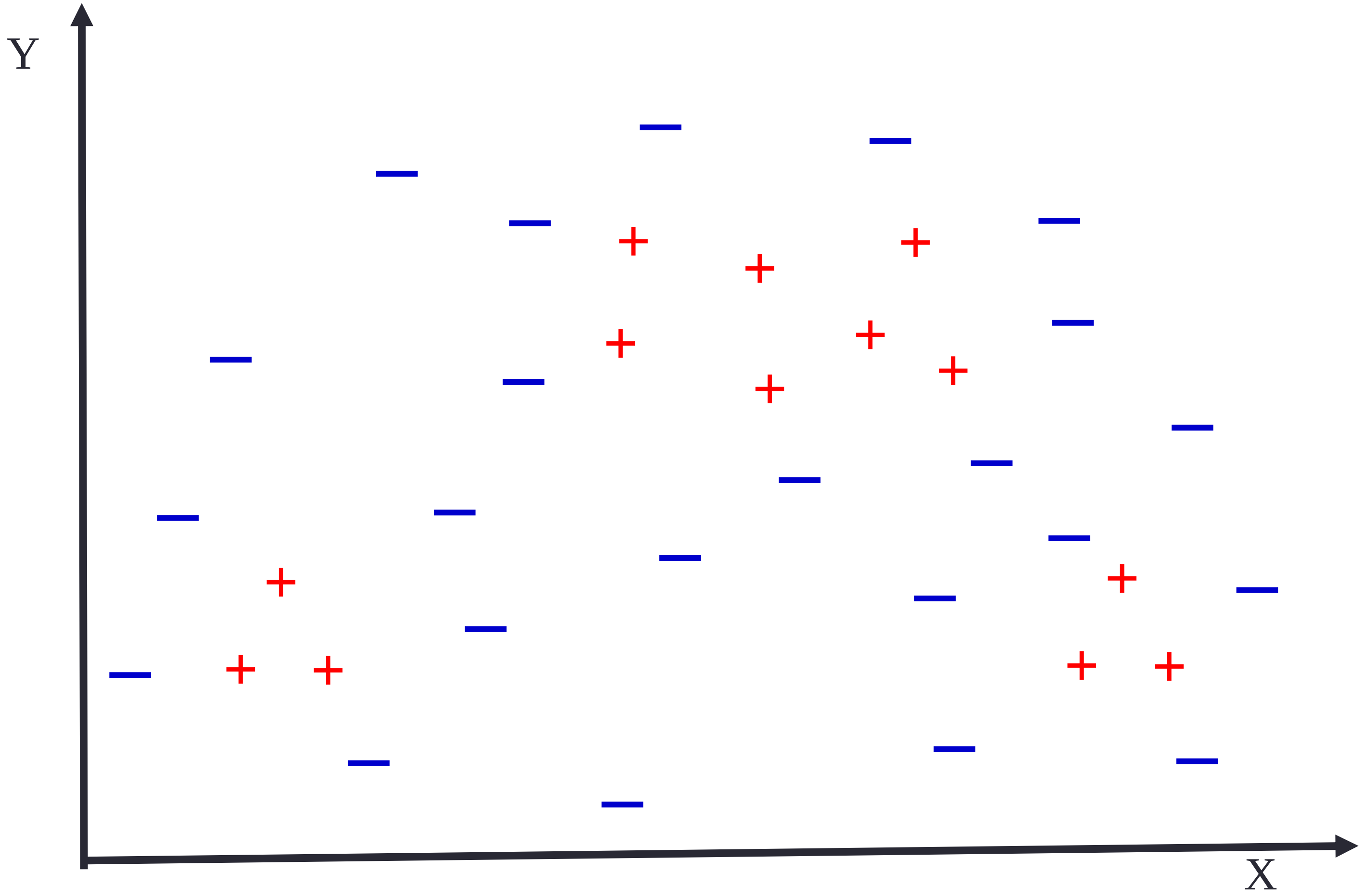
Example of Sequential Covering



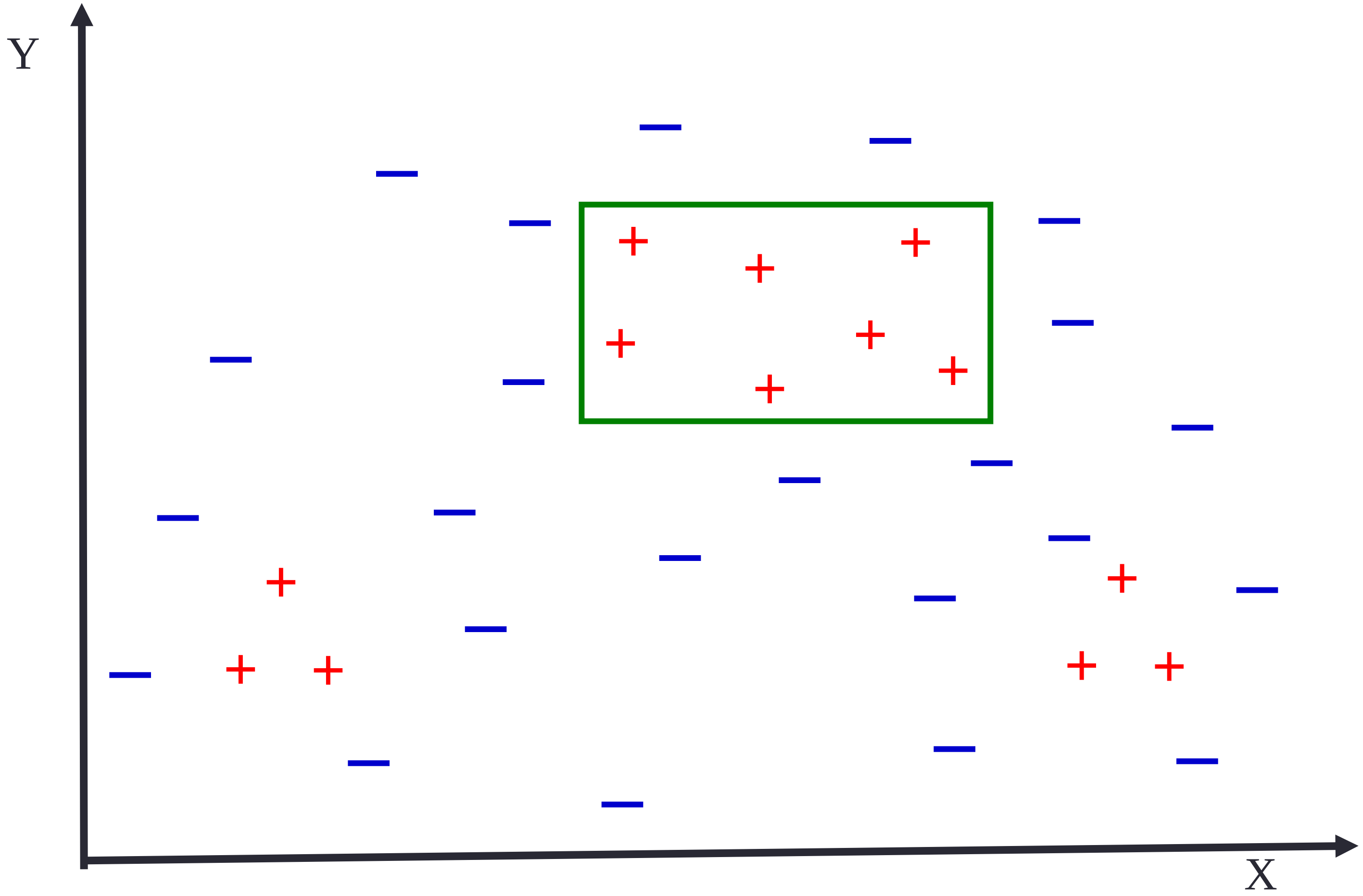
Example of Sequential Covering



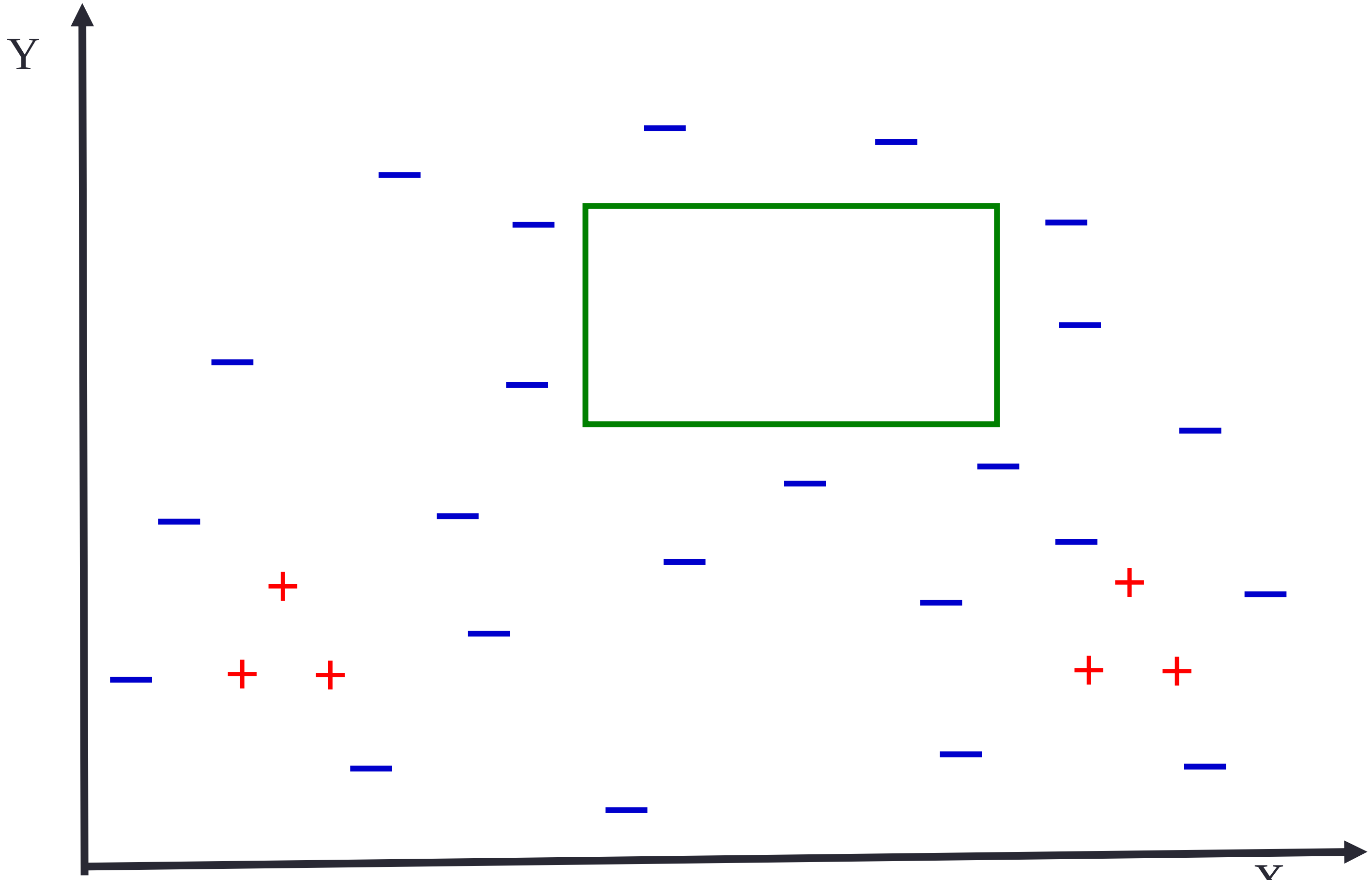
No-optimal Covering Example



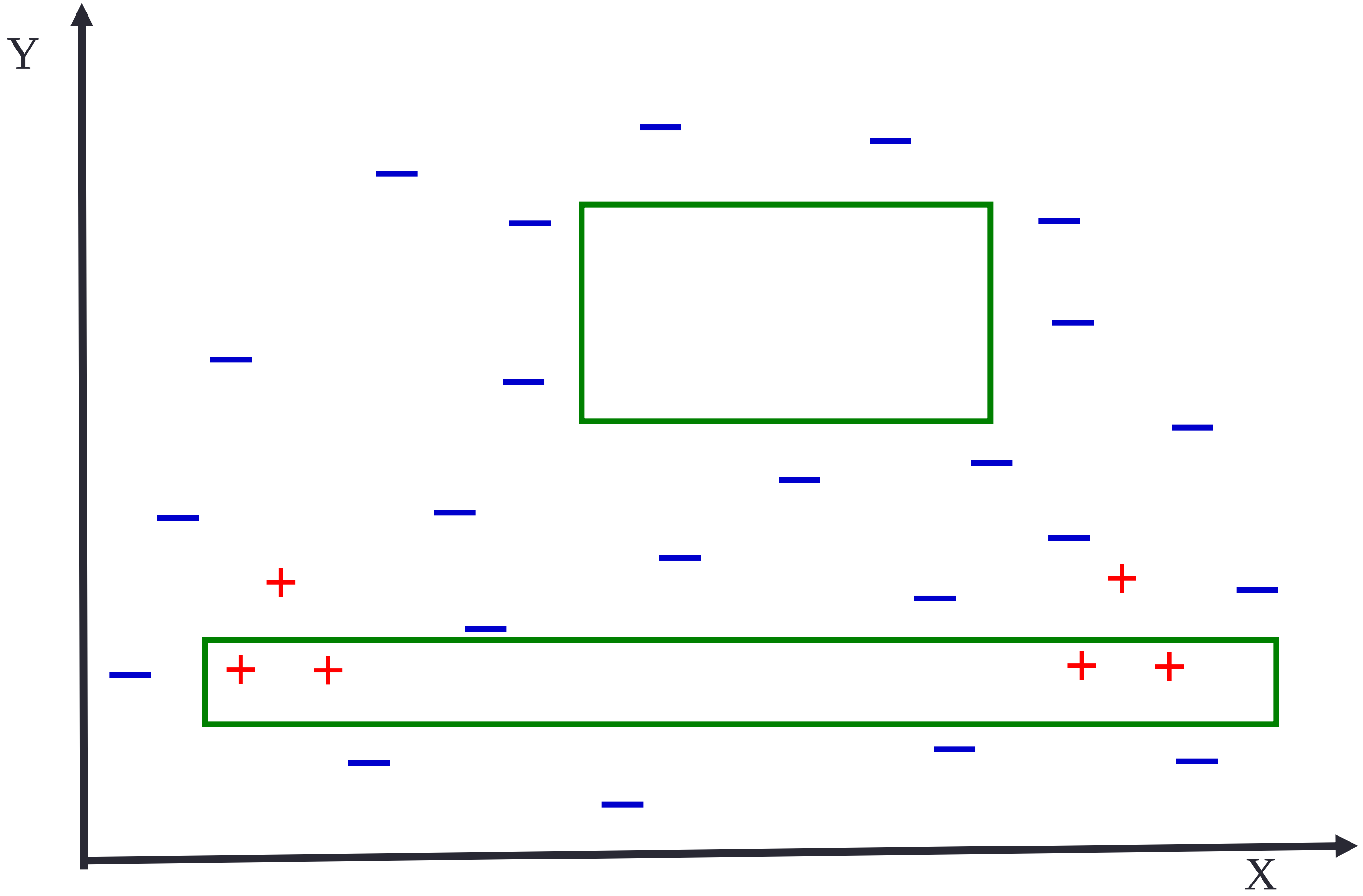
No-optimal Covering Example



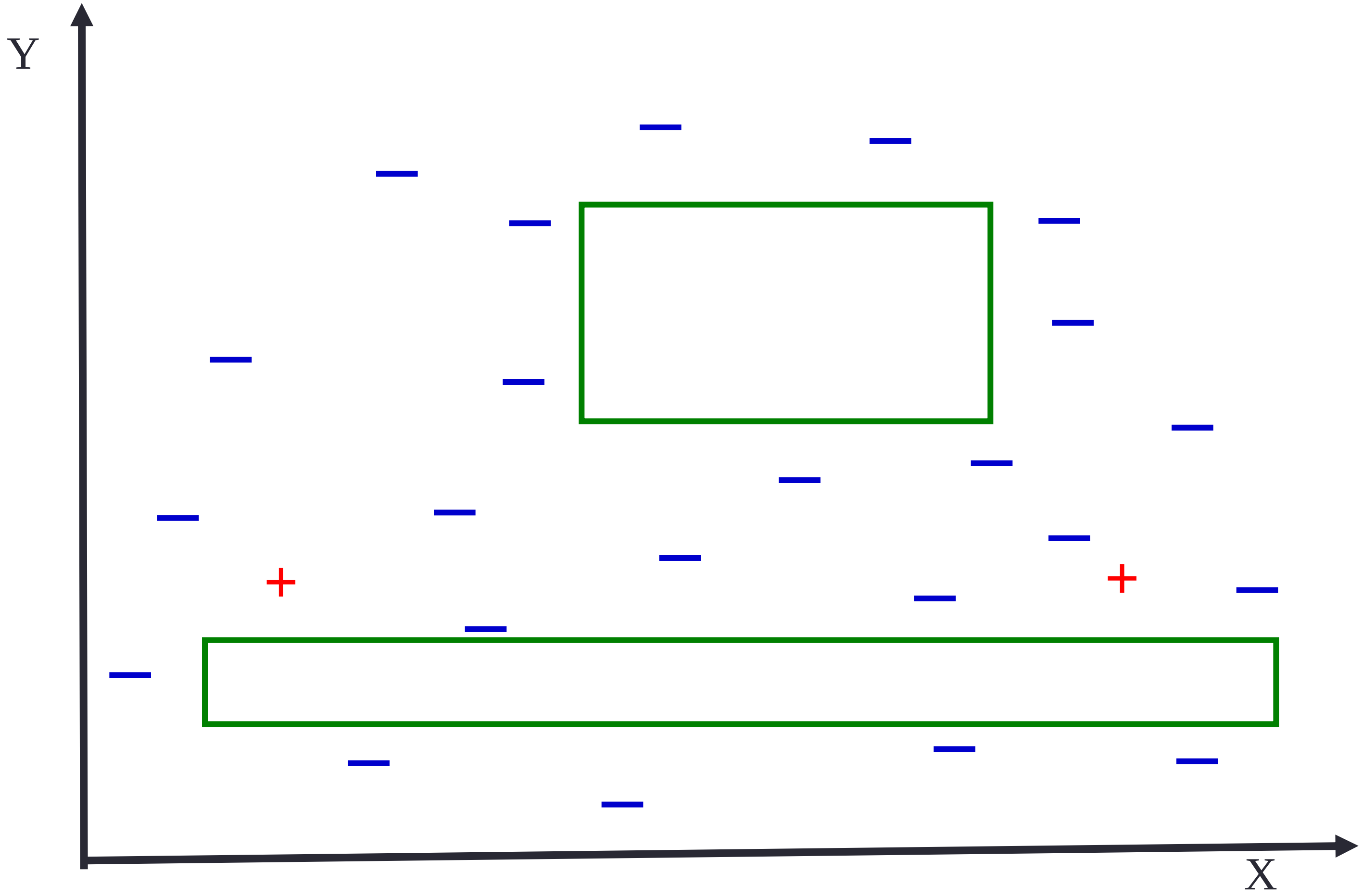
No-optimal Covering Example



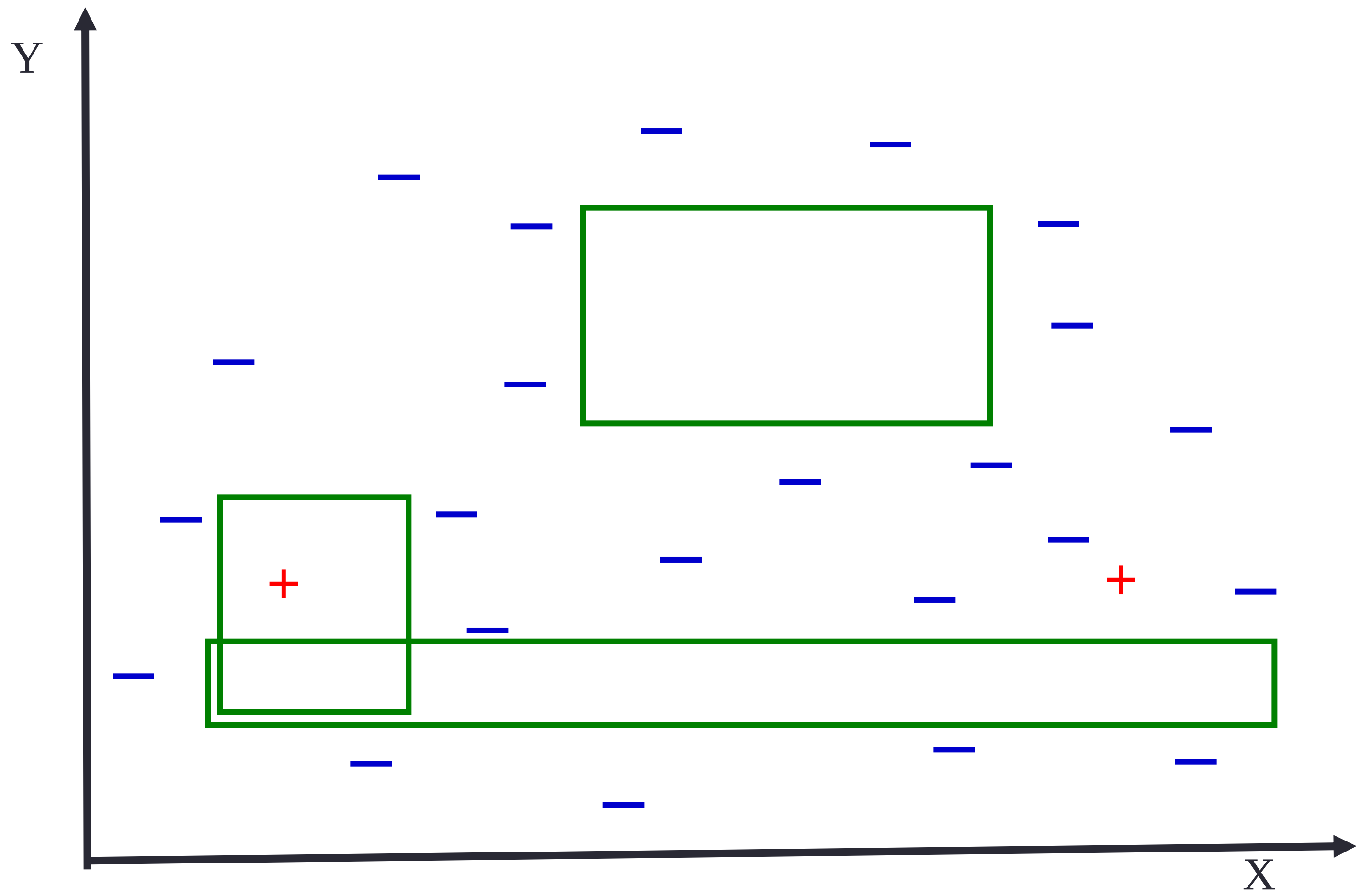
No-optimal Covering Example



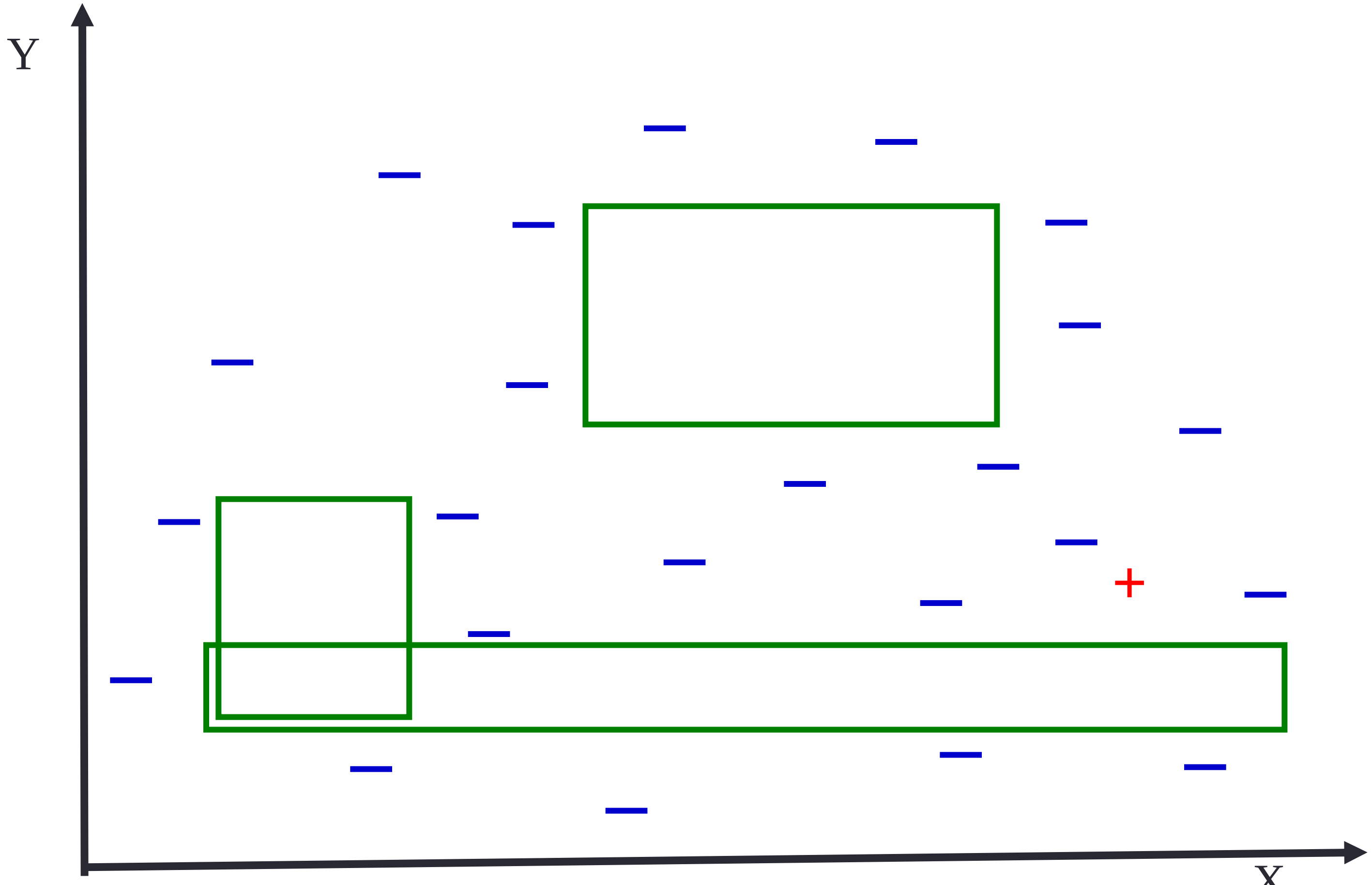
No-optimal Covering Example



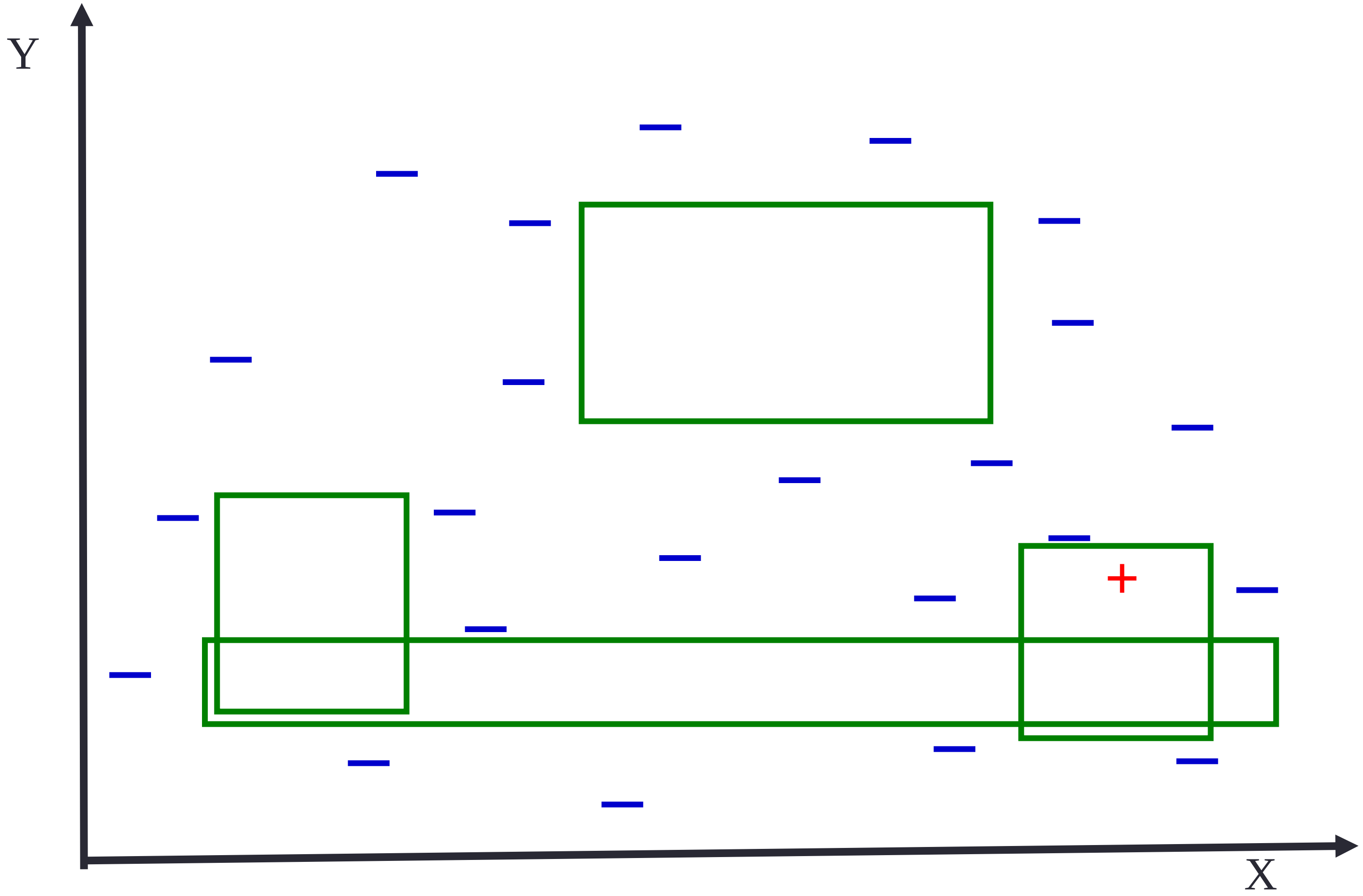
No-optimal Covering Example



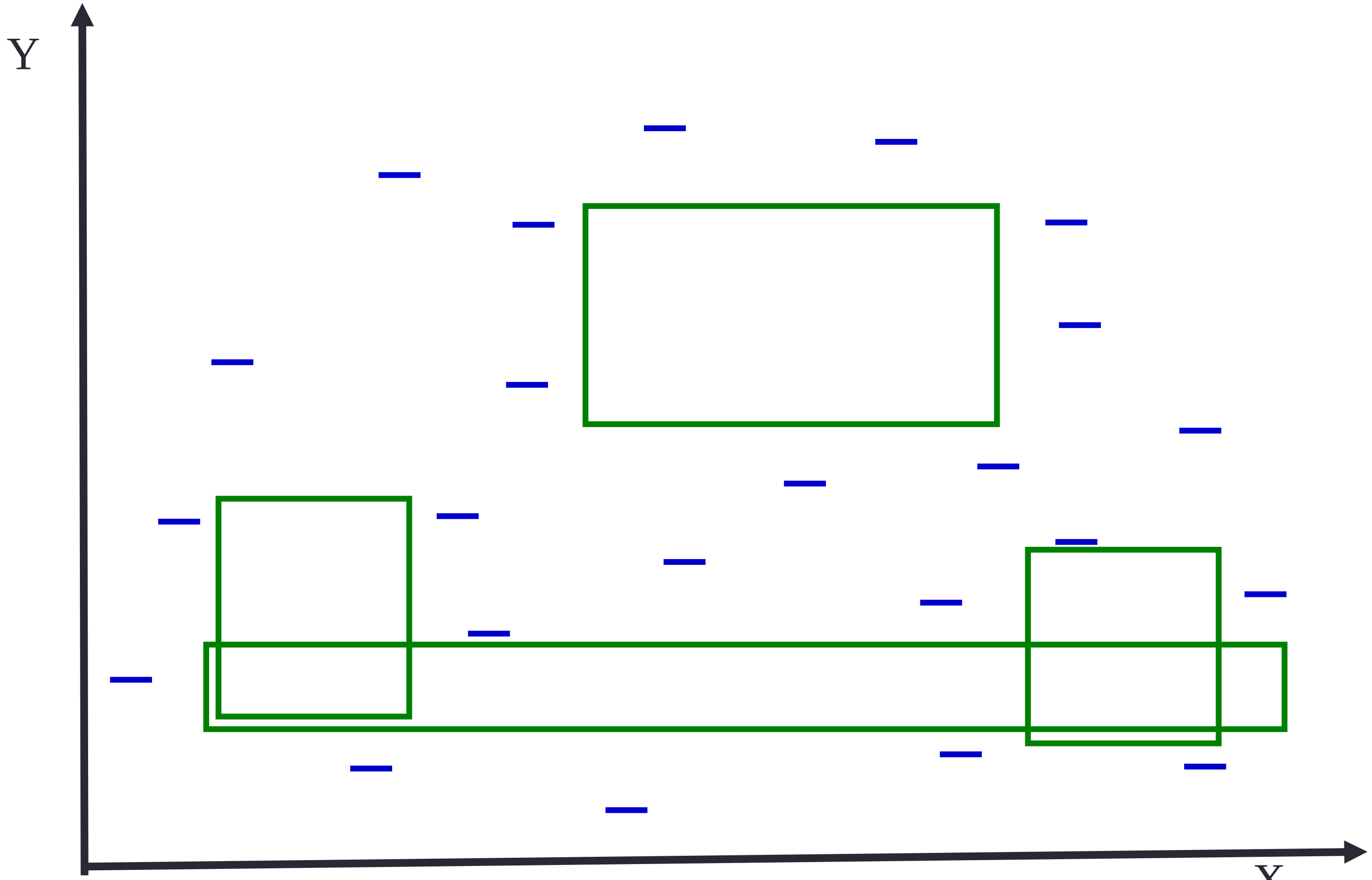
No-optimal Covering Example



No-optimal Covering Example



No-optimal Covering Example

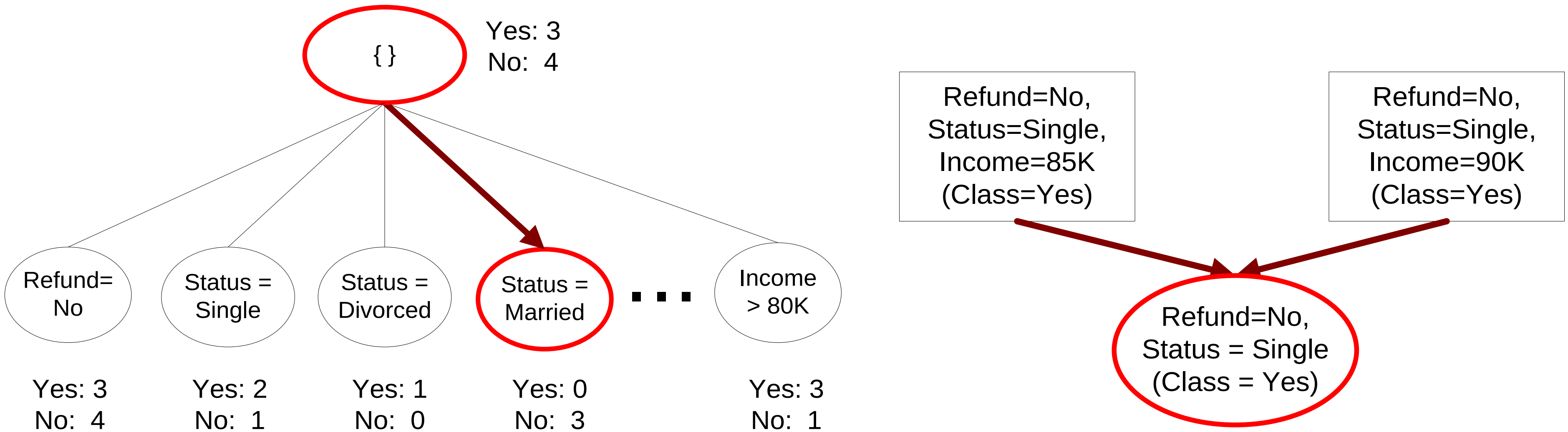


Aspects of Sequential Covering

- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



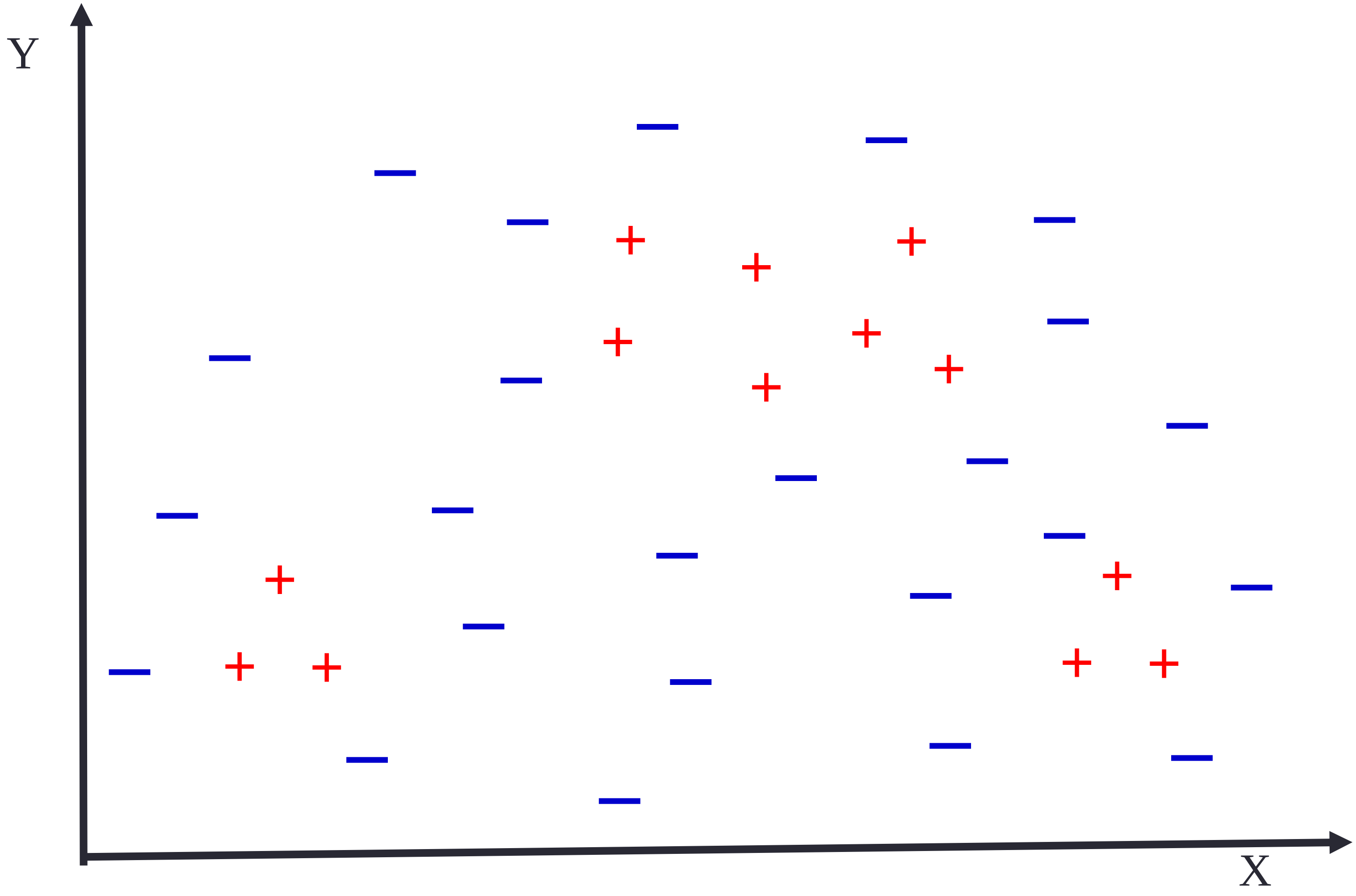
(a) General-to-specific

(b) Specific-to-general

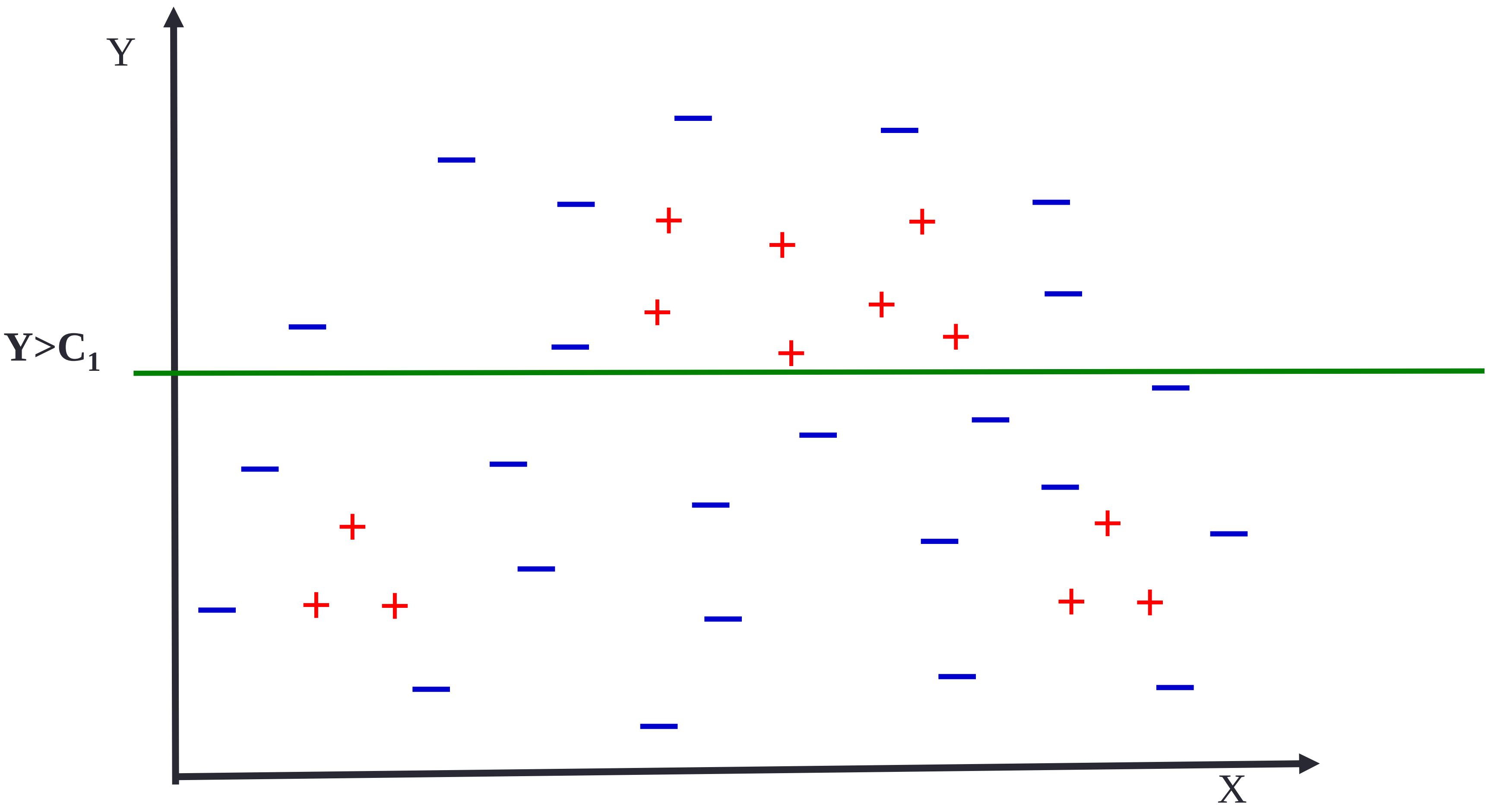
Strategies for Learning a Single Rule

- Top Down (General to Specific):
  - Start with the most-general (empty) rule.
  - Repeatedly add antecedent constraints on features that eliminate negative examples while maintaining as many positives as possible.
  - Stop when only positives are covered.
- Bottom Up (Specific to General)
  - Start with a most-specific rule (e.g. complete instance description of a random instance).
  - Repeatedly remove antecedent constraints in order to cover more positives.
  - Stop when further generalization results in covering negatives.

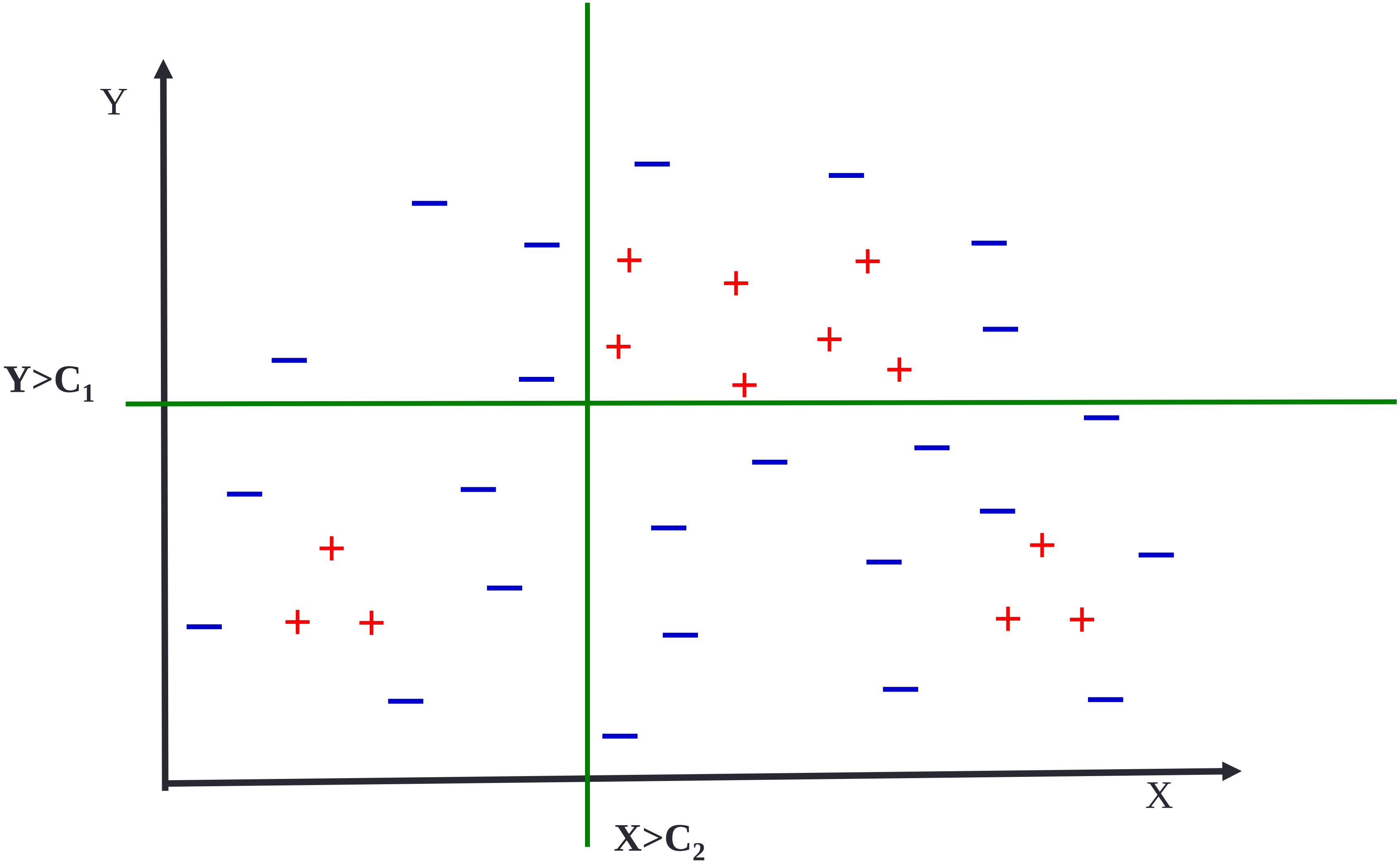
Top-Down Rule Learning Example



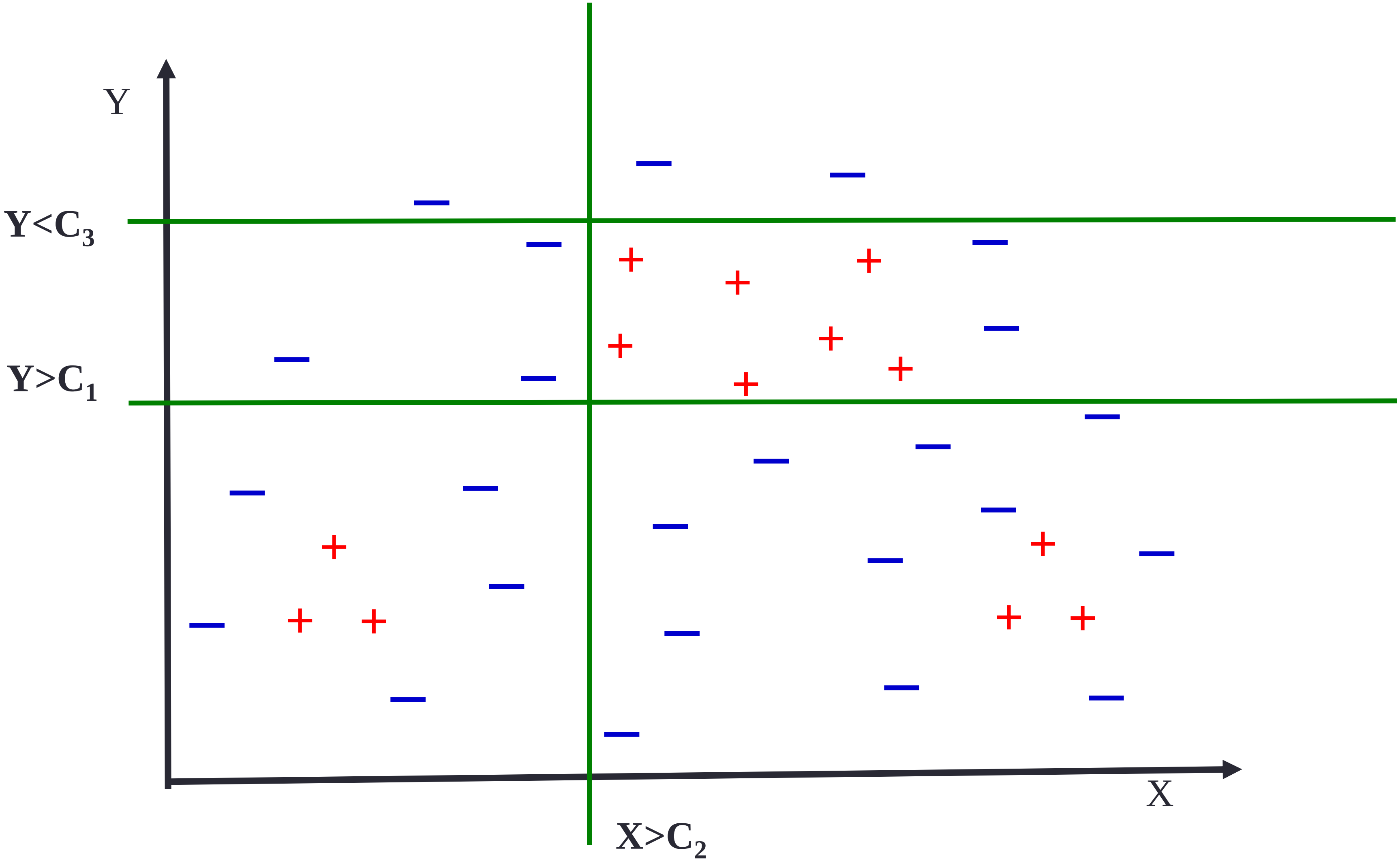
Top-Down Rule Learning Example



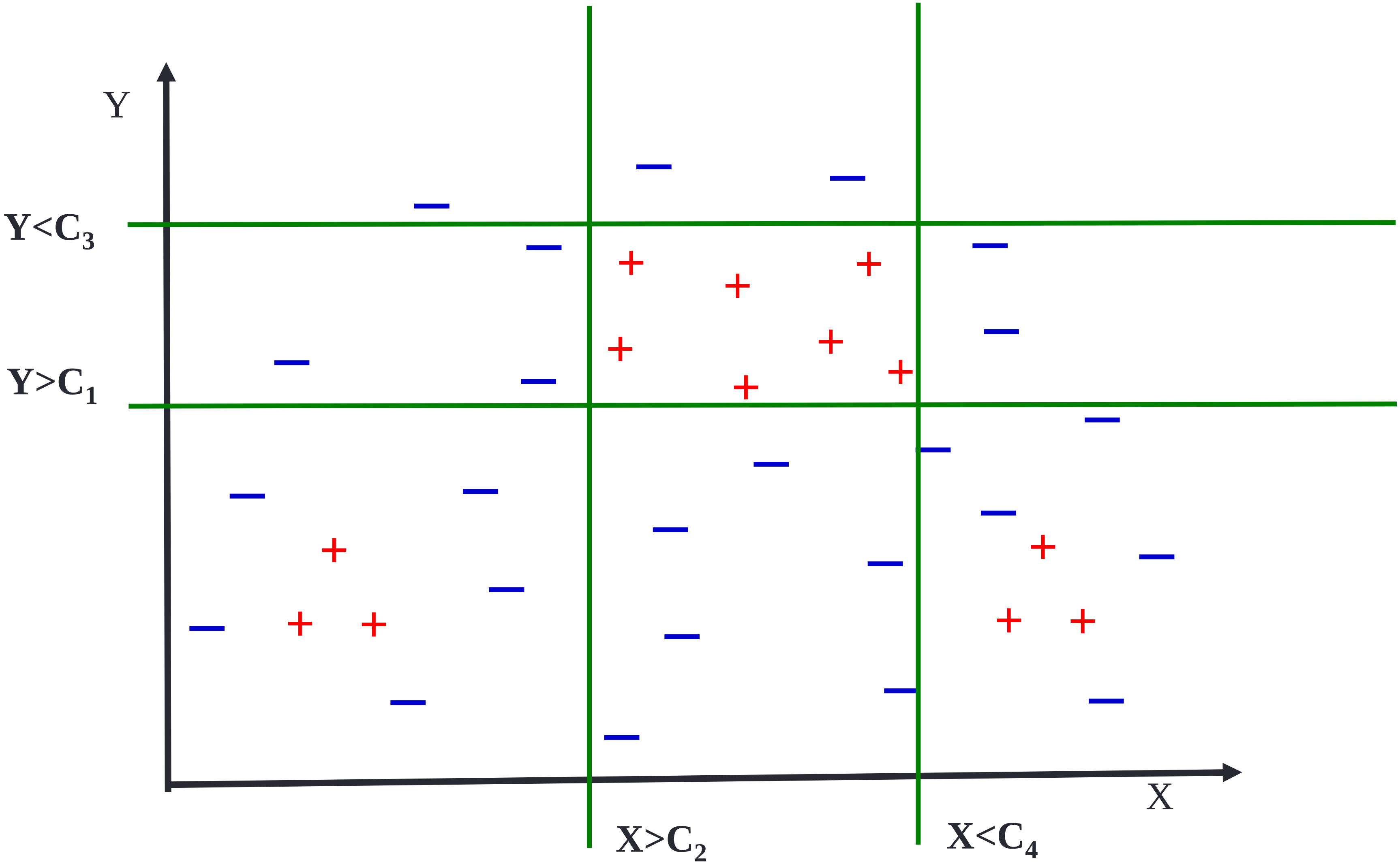
Top-Down Rule Learning Example



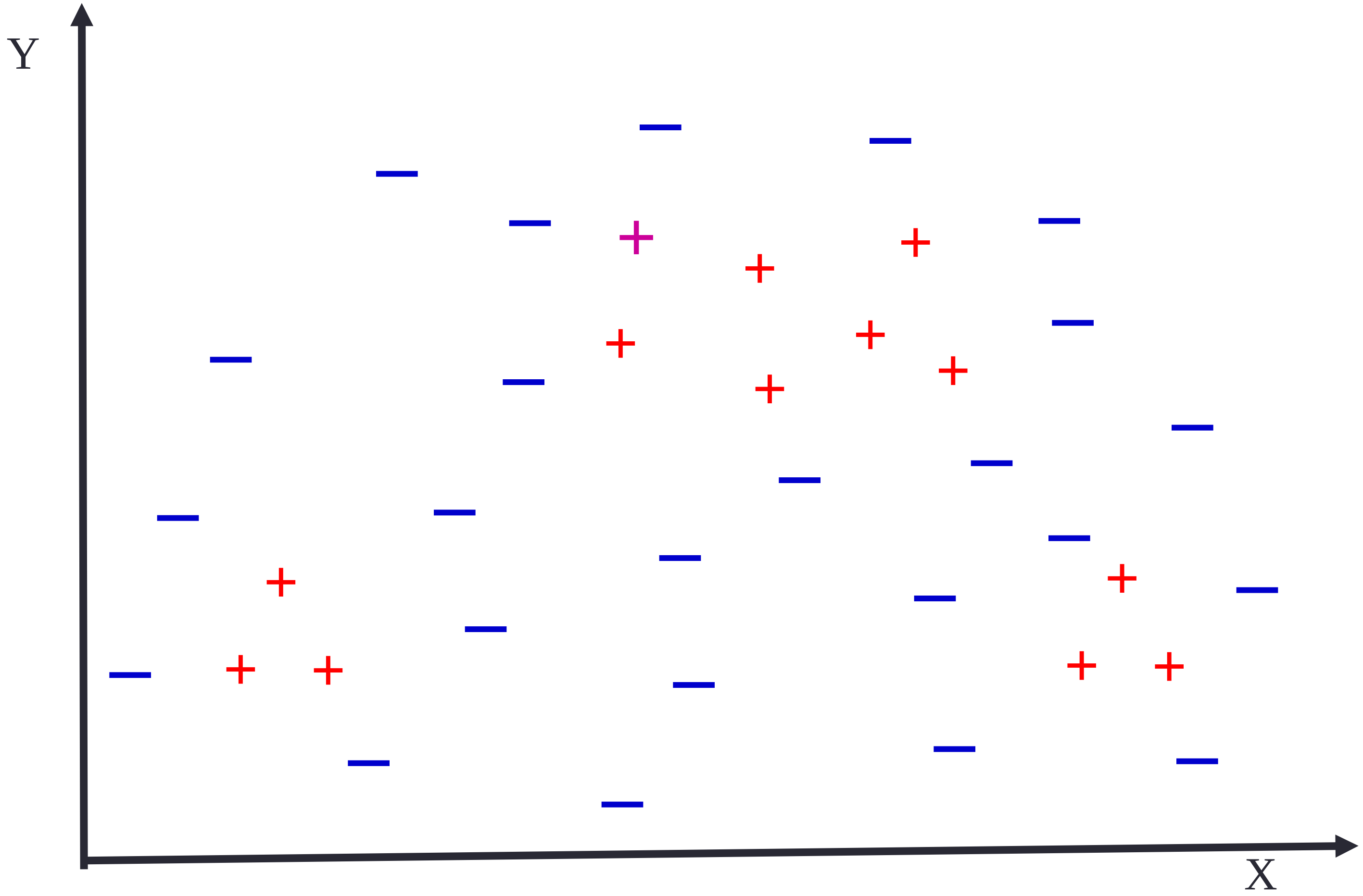
Top-Down Rule Learning Example



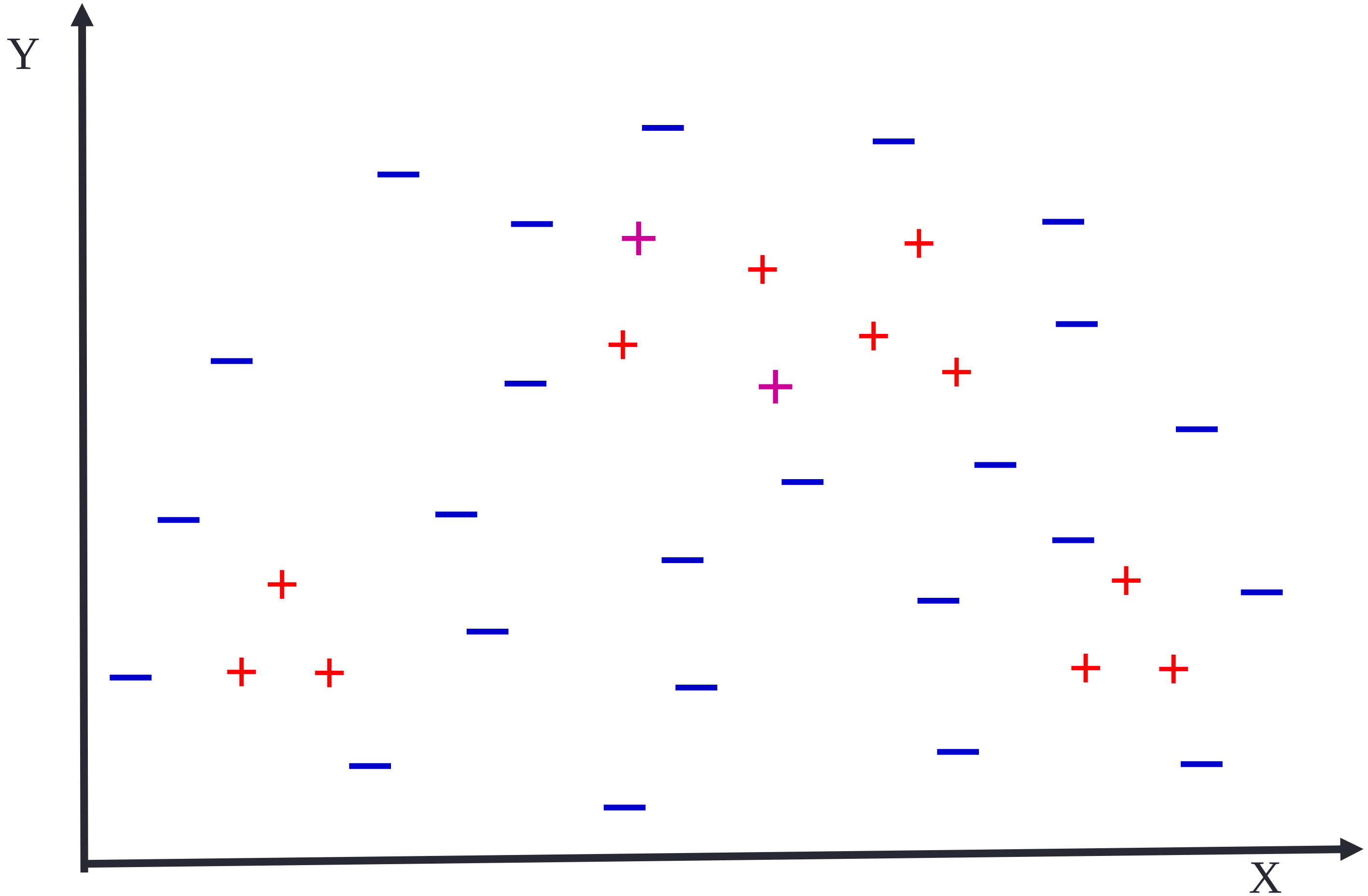
Top-Down Rule Learning Example



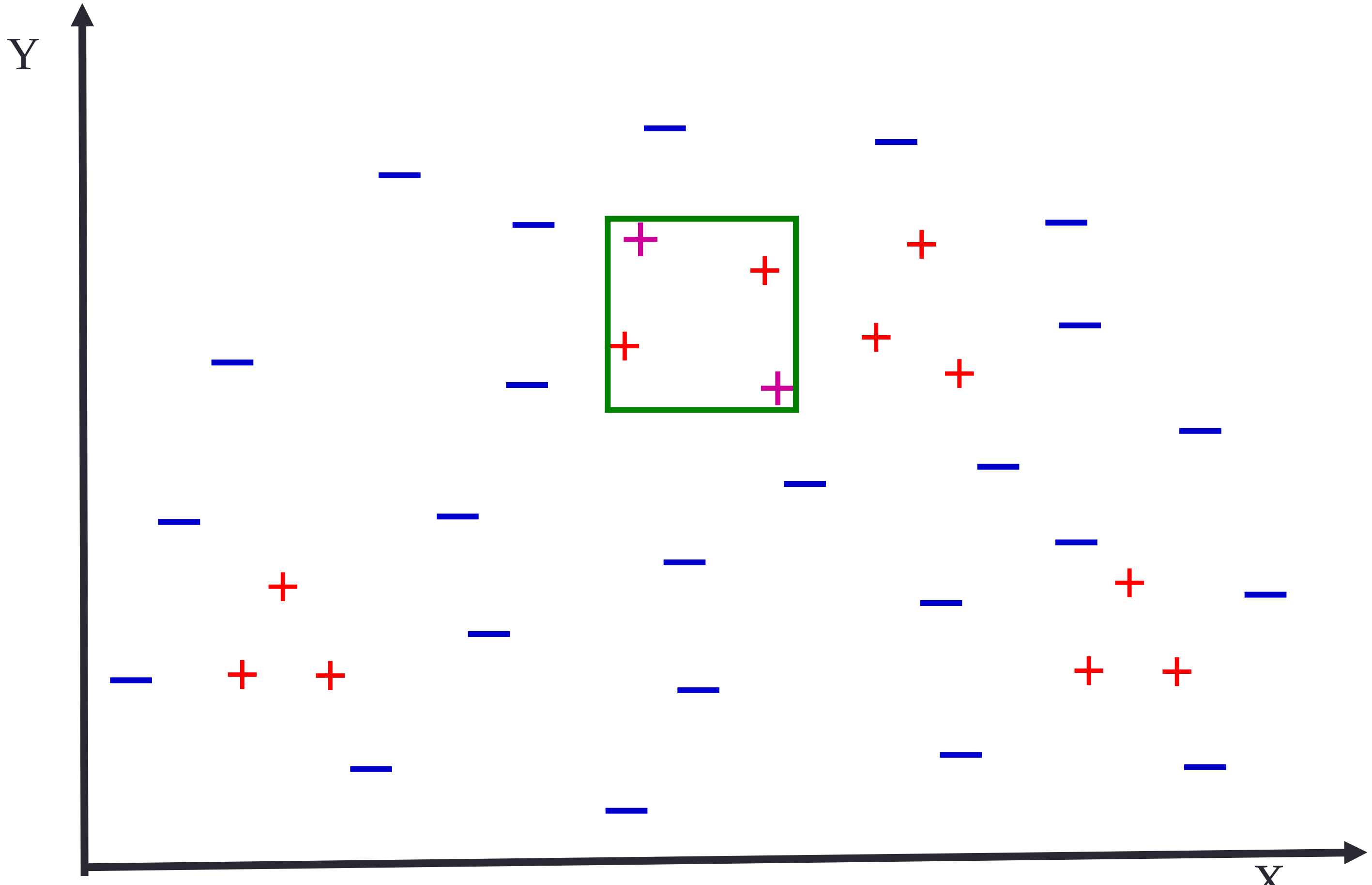
Bottom-Up Rule Learning Example



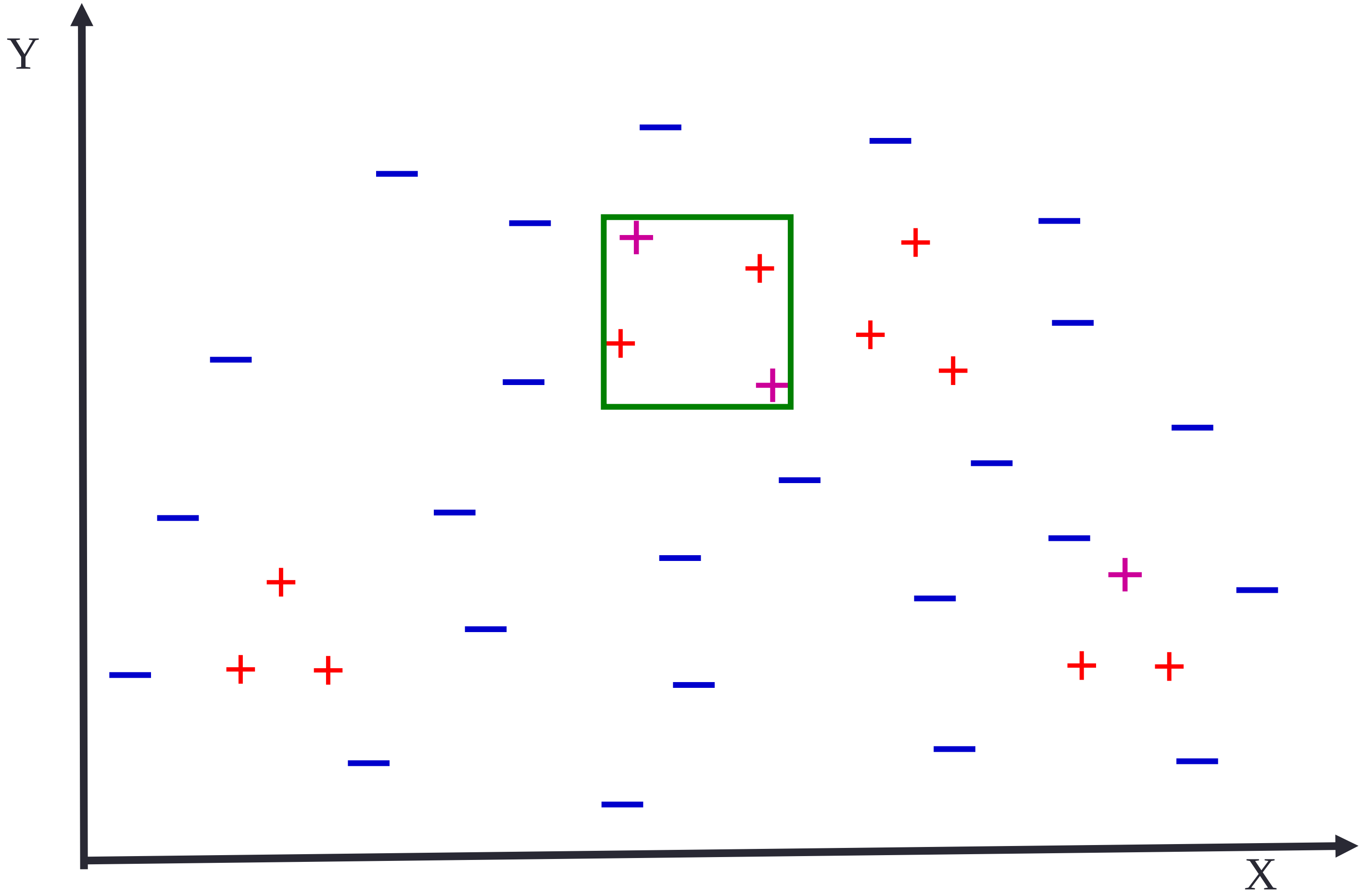
Bottom-Up Rule Learning Example



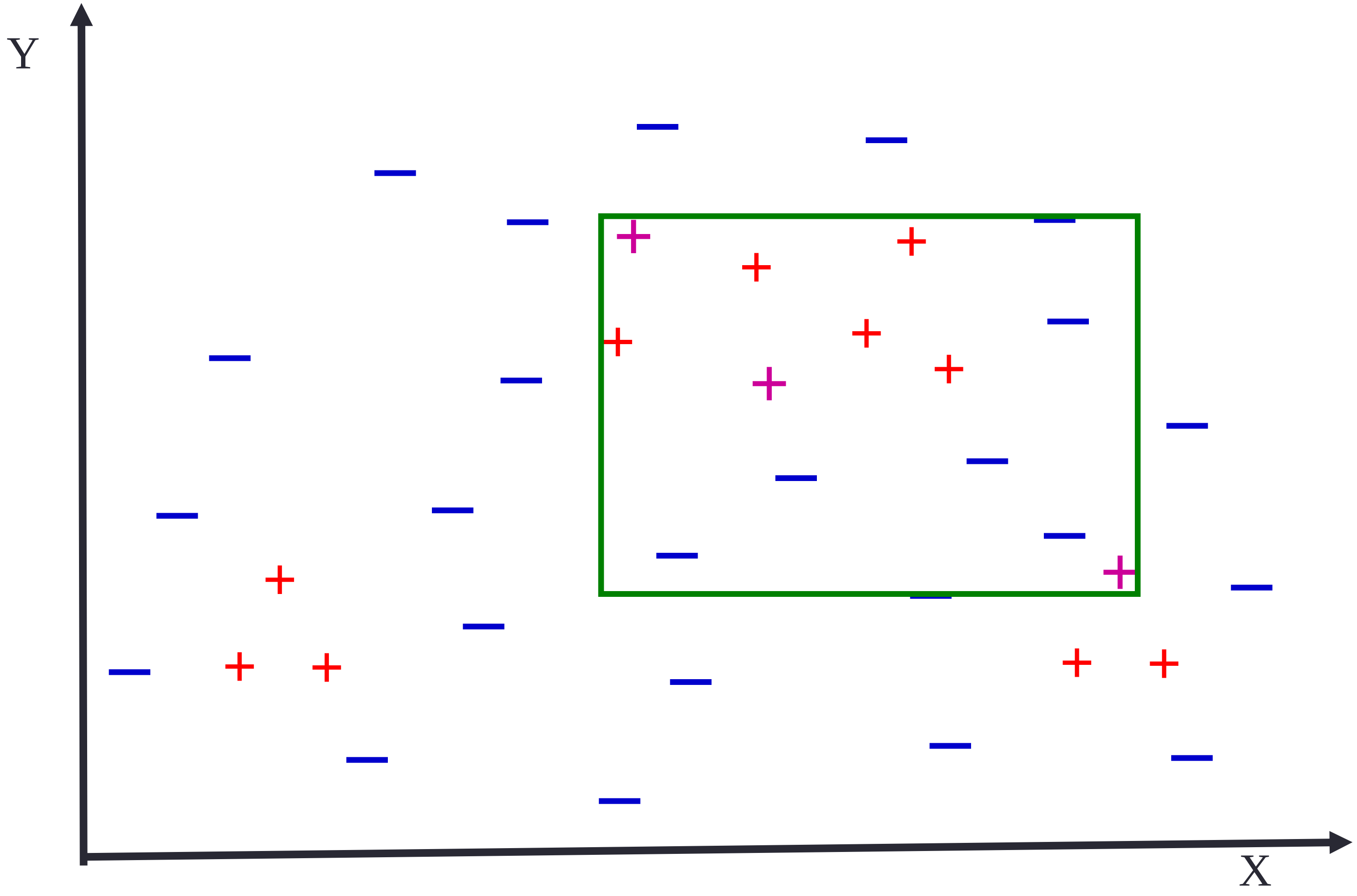
Bottom-Up Rule Learning Example



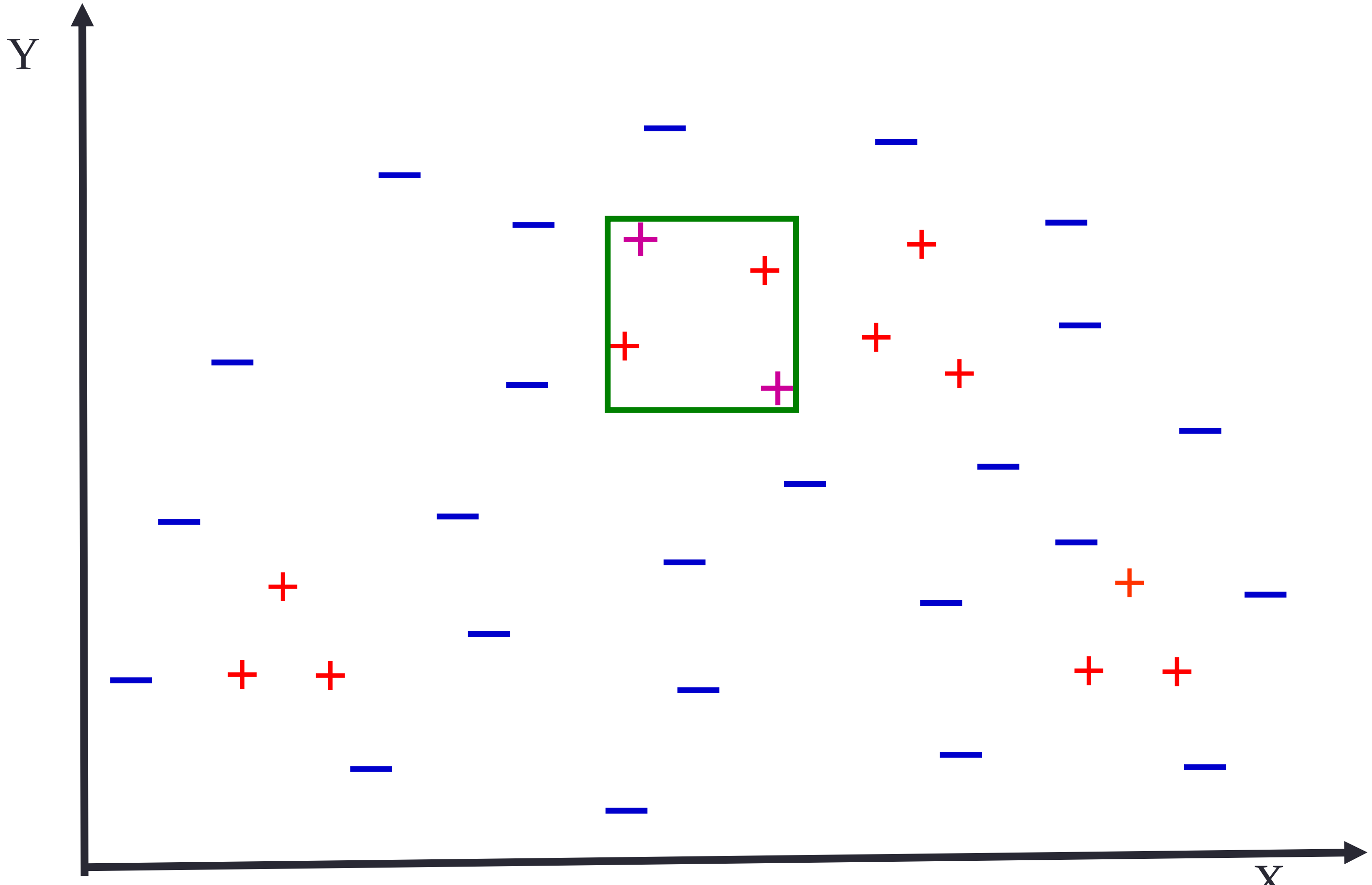
Bottom-Up Rule Learning Example



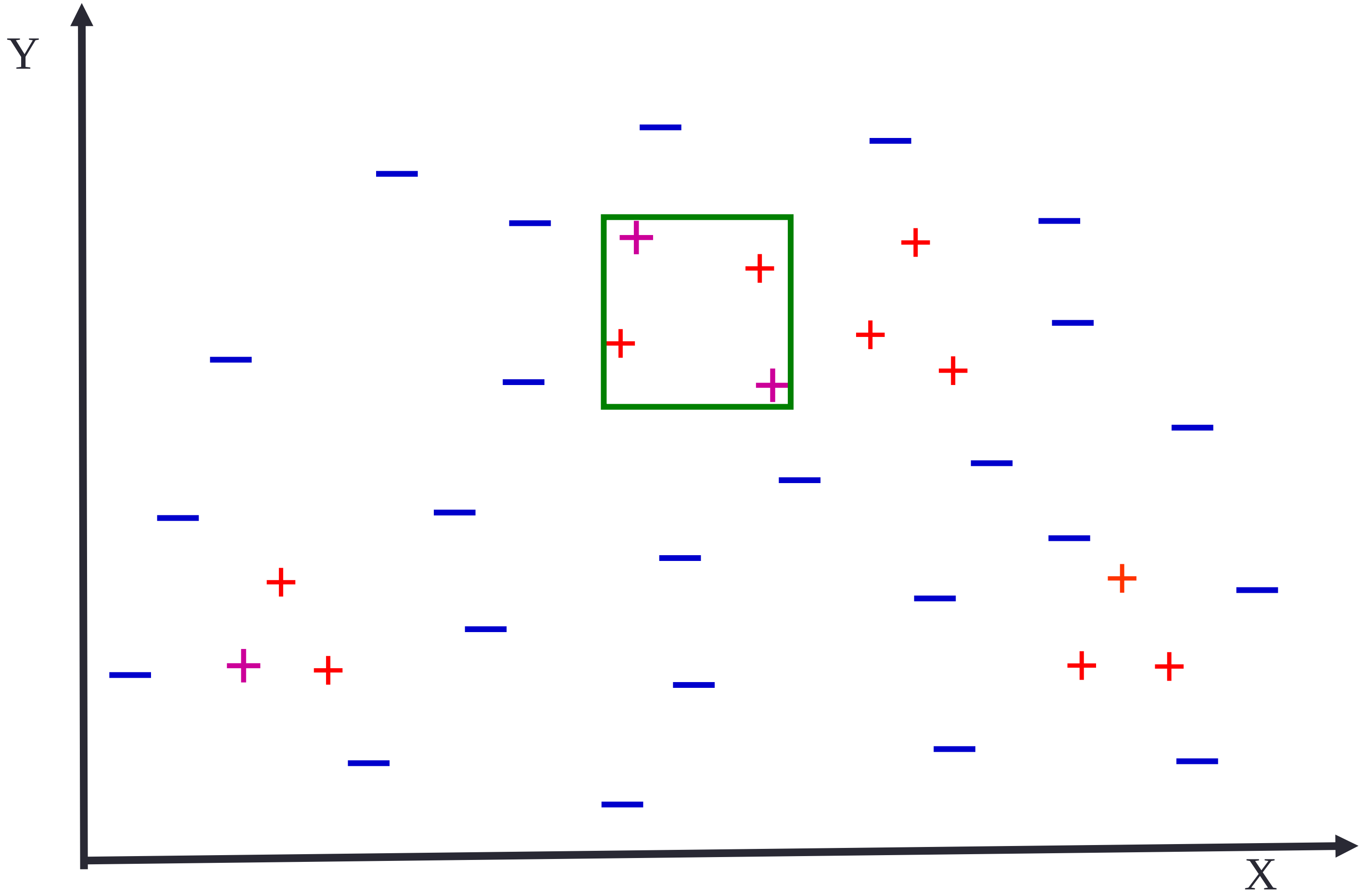
Bottom-Up Rule Learning Example



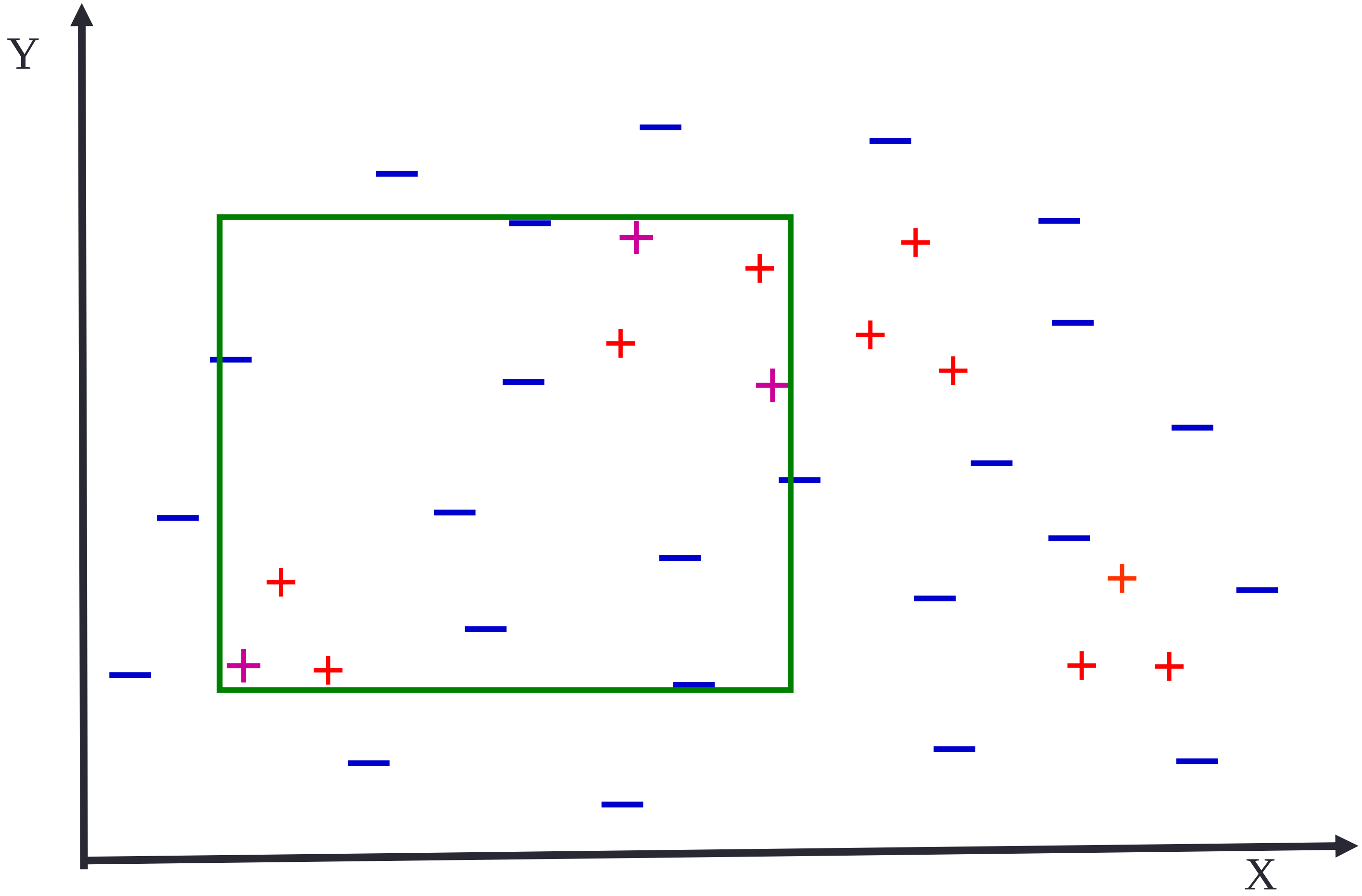
Bottom-Up Rule Learning Example



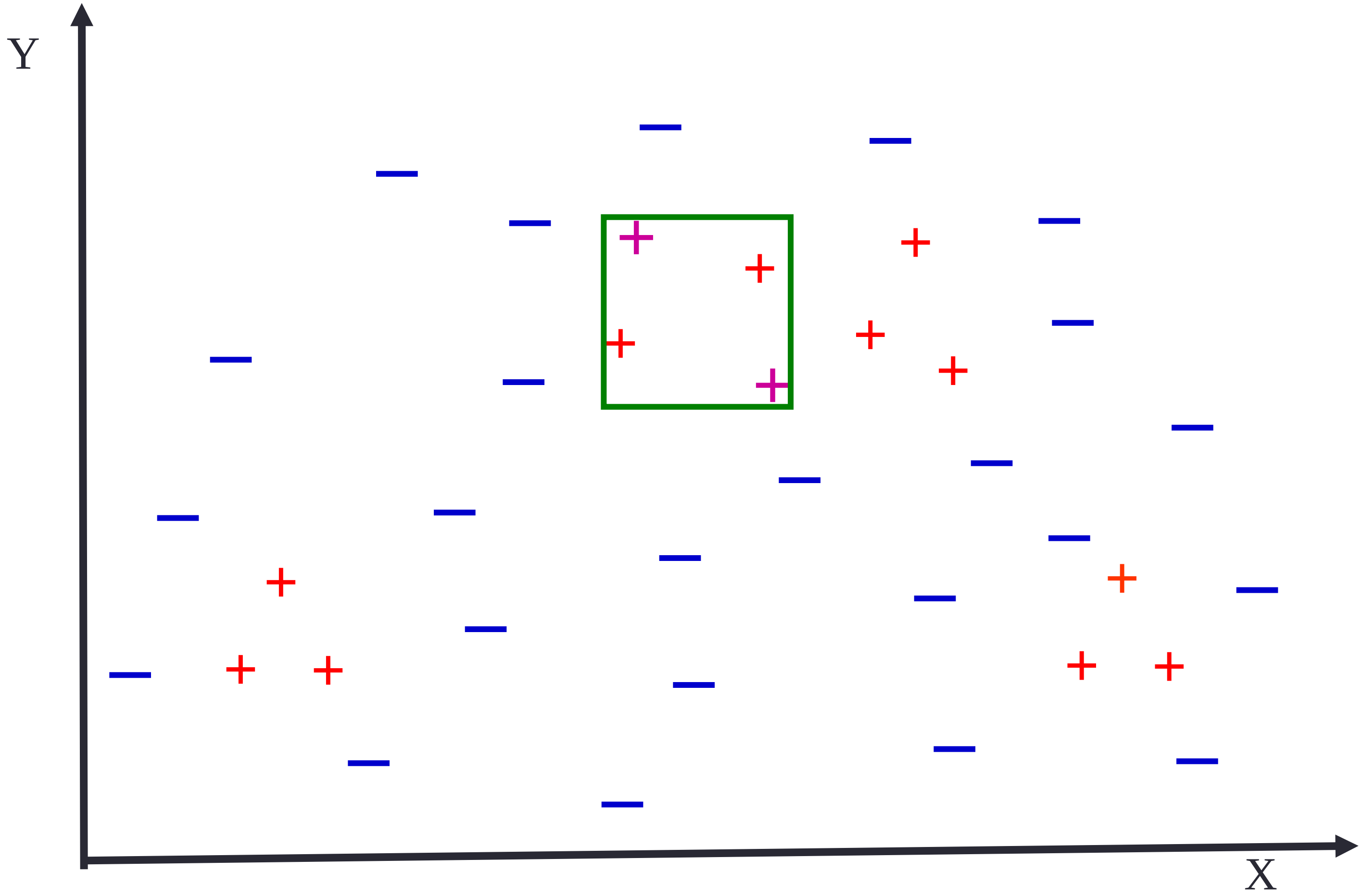
Bottom-Up Rule Learning Example



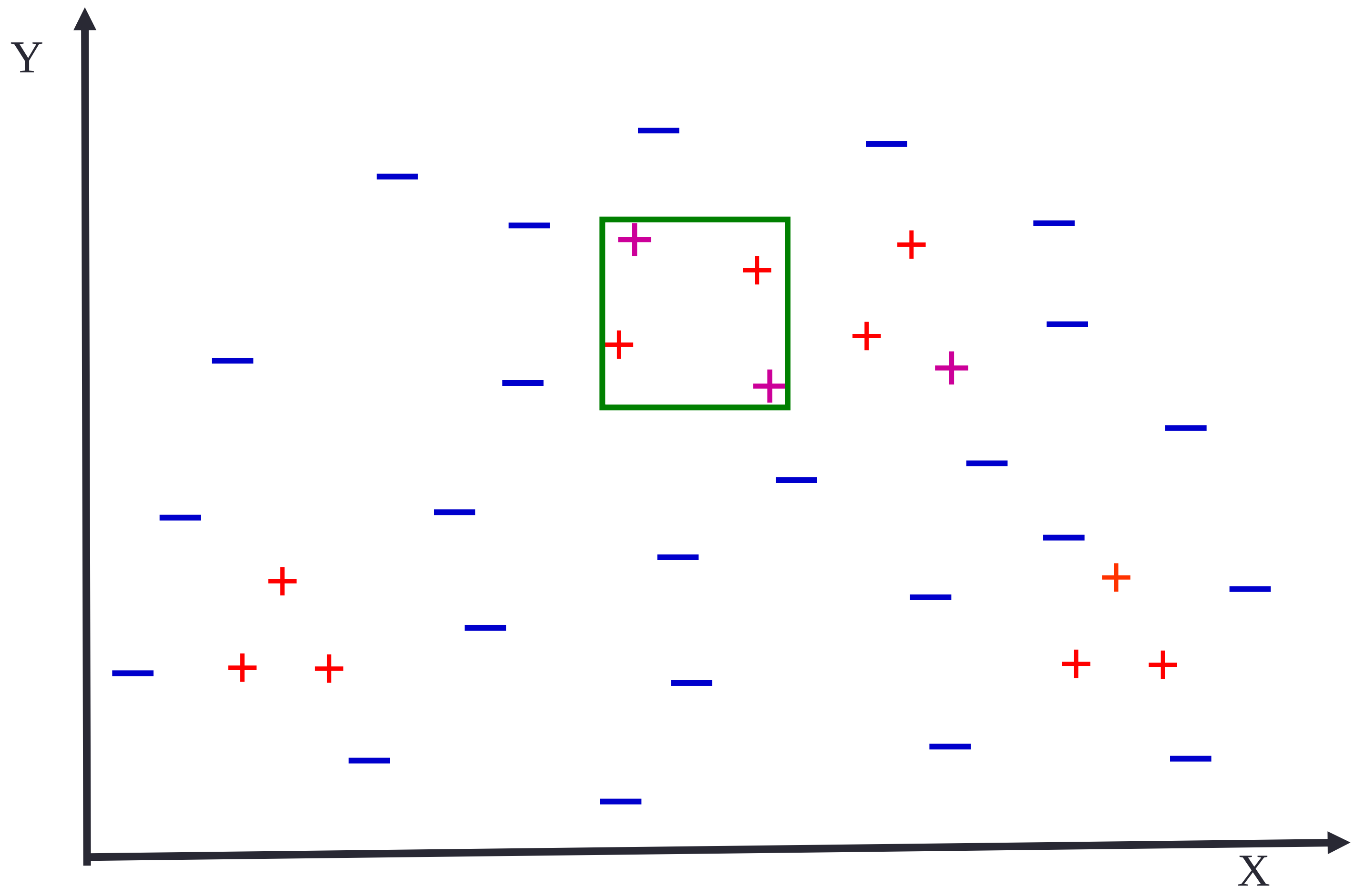
Bottom-Up Rule Learning Example



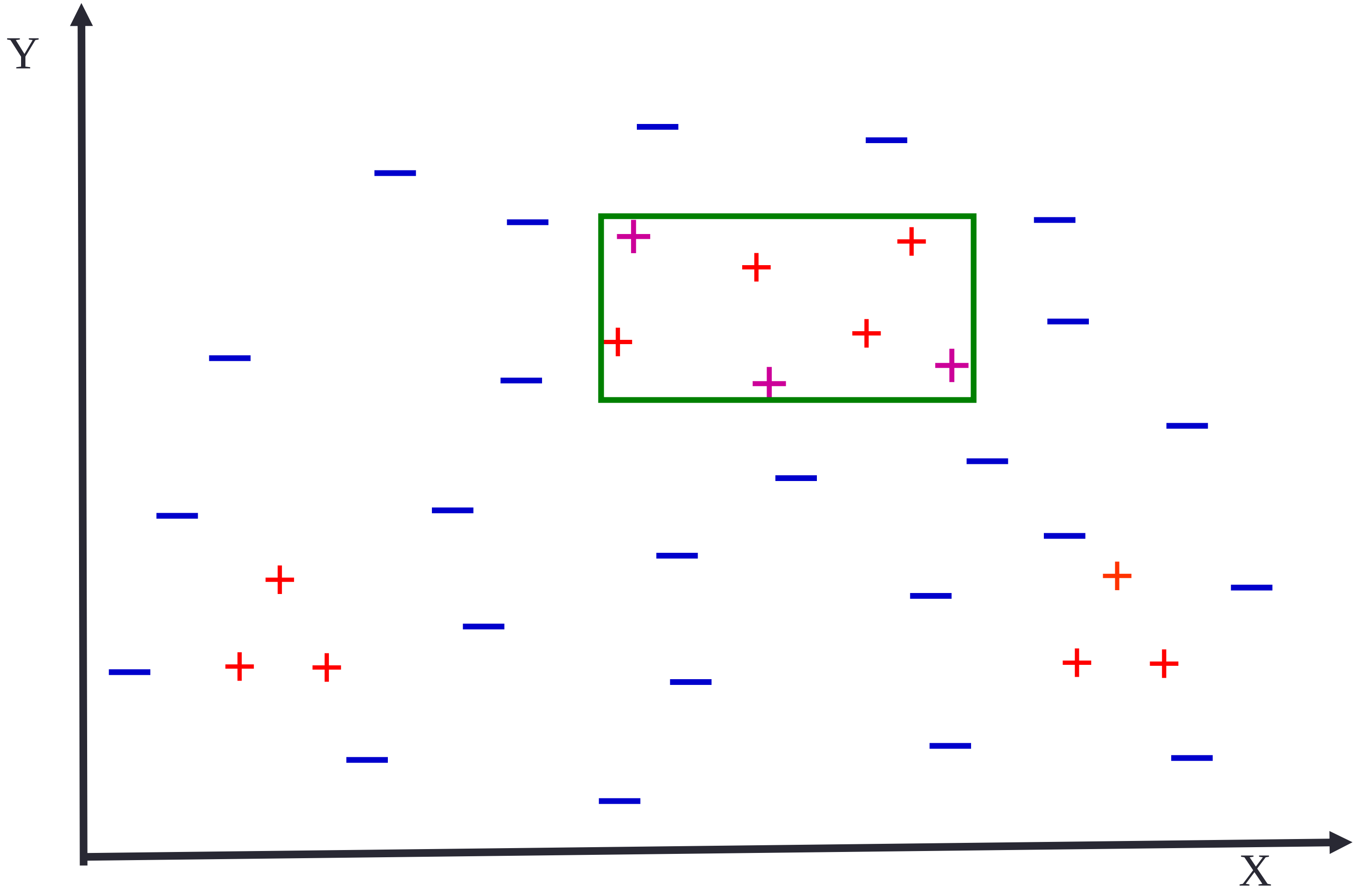
Bottom-Up Rule Learning Example



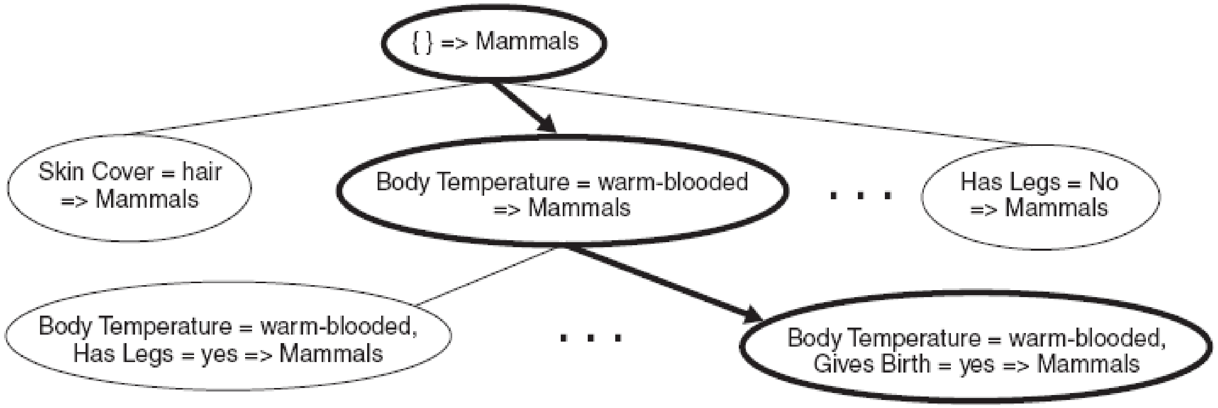
Bottom-Up Rule Learning Example



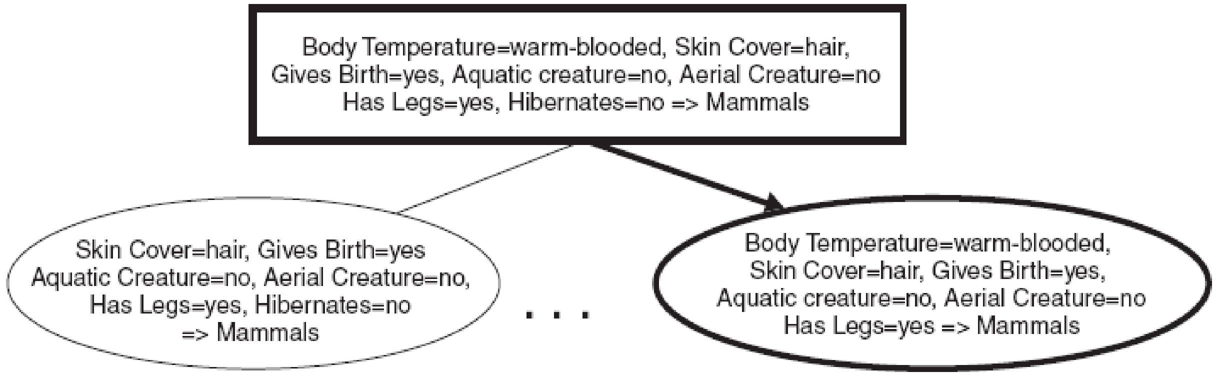
Bottom-Up Rule Learning Example



Strategies for Learning a Single Rule



(a) General-to-specific



(b) Specific-to-general

Strategies for Learning a Single Rule

- Which is better top-down or bottom-up search?
  - Bottom-up is more subject to noise, e.g. the random seeds that are chosen may be noisy.
  - Top-down is wasteful when there are many features which do not even occur in the positive examples (e.g. text categorization).

Rule Growing (Examples)

- CN2 Algorithm:
  - Start from an empty conjunct: {}
  - Add conjuncts that minimizes the entropy measure: {A}, {A,B}, …
  - Determine the rule consequent by taking majority class of instances covered by the rule
- RIPPER Algorithm:
  - Start from an empty rule: {} => class
  - Add conjuncts that maximizes FOIL's information gain measure:
    - 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

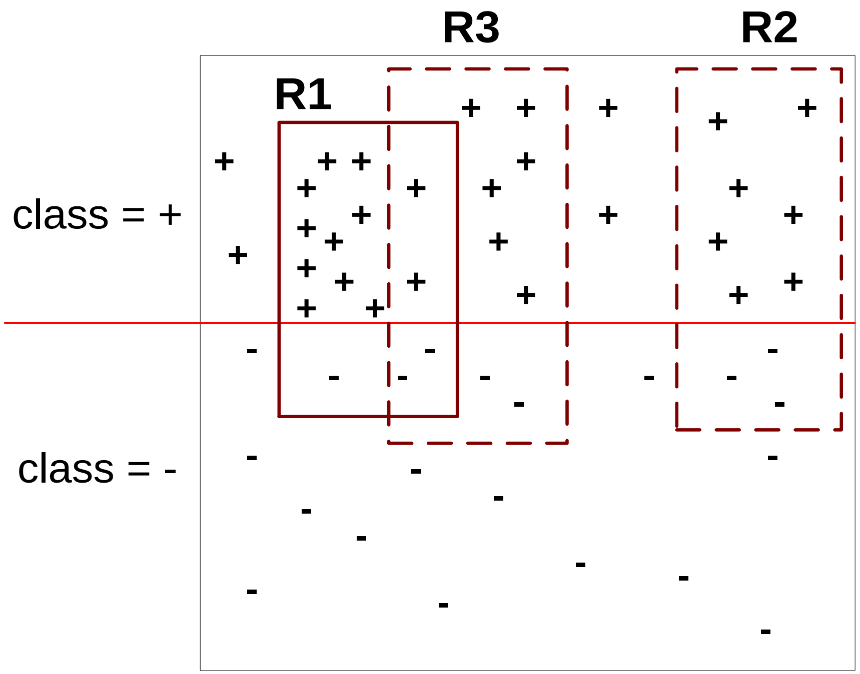
    n0: number of negative instances covered by R0

    p1: number of positive instances covered by R1

    n1: number of negative instances covered by R1

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 Evaluation

- Metrics:
  - Accuracy $= \dfrac{n_c}{n}$
  - Laplace $= \dfrac{n_c + 1}{n + k}$
  - M-estimate $= \dfrac{n_c + kp}{n + k}$

n : Number of examples covered by the rule

n_c *:* Number of positive examples covered by rule

k : Number of classes

p : Prior probability for the positive class

Stopping Criterion and Rule Pruning

- Stopping criterion
  - Compute the gain
  - If gain is not significant, discard the new rule

- Rule Pruning
  - Similar to post-pruning of decision trees
  - Reduced Error Pruning:
    - Remove one of the conjuncts in the rule
    - Compare error rate on validation set before and after pruning
    - If error improves, prune the conjunct

Summary of Direct Method

- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

Learning a Single Rule in FOIL

- Top-down approach originally applied to first-order logic (Quinlan, 1990).
- Basic algorithm for instances with discrete-valued features:

Let A={} (set of rule antecedents)
Let N be the set of negative examples
Let P the current set of uncovered positive examples
Until N is empty do
    For every feature-value pair (literal) (F_i=V_{ij}) calculate
        Gain(F_i=V_{ij}, P, N)
    Pick literal, L, with highest gain.
    Add L to A.
    Remove from N any examples that do not satisfy L.
    Remove from P any examples that do not satisfy L.
Return the rule: $L_1 \wedge L_2 \wedge \ldots \wedge L_n \rightarrow$ Positive

Foil Gain Metric

- Want to achieve two goals
  - Decrease coverage of negative examples
    - Measure increase in percentage of positives covered when literal is added to the rule.
  - Maintain coverage of as many positives as possible.
    - Count number of positives covered.

Define Gain(L, P, N)

Let p be the subset of examples in P that satisfy L.

Let n be the subset of examples in N that satisfy L.

Return:

$$|p| * [\log_2(|p|/(|p|+|n|)) - \log_2(|P|/(|P|+|N|))]$$

Sample Disjunctive Learning Data

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	big	red	circle	positive
3	small	red	triangle	negative
4	big	blue	circle	negative
5	medium	red	circle	negative

Propositional FOIL Trace

New Disjunct:
SIZE=BIG Gain: 0.322
SIZE=MEDIUM Gain: 0.000
SIZE=SMALL Gain: 0.322
COLOR=BLUE Gain: 0.000
COLOR=RED Gain: 0.644
COLOR=GREEN Gain: 0.000
SHAPE=SQUARE Gain: 0.000
SHAPE=TRIANGLE Gain: 0.000
SHAPE=CIRCLE Gain: 0.644
Best feature: COLOR=RED

SIZE=BIG Gain: 1.000
SIZE=MEDIUM Gain: 0.000
SIZE=SMALL Gain: 0.000
SHAPE=SQUARE Gain: 0.000
SHAPE=TRIANGLE Gain: 0.000
SHAPE=CIRCLE Gain: 0.830
Best feature: SIZE=BIG
Learned Disjunct: COLOR=RED & SIZE=BIG

Propositional FOIL Trace

New Disjunct:
SIZE=BIG Gain: 0.000
SIZE=MEDIUM Gain: 0.000
SIZE=SMALL Gain: 1.000
COLOR=BLUE Gain: 0.000
COLOR=RED Gain: 0.415
COLOR=GREEN Gain: 0.000
SHAPE=SQUARE Gain: 0.000
SHAPE=TRIANGLE Gain: 0.000
SHAPE=CIRCLE Gain: 0.415
Best feature: SIZE=SMALL

COLOR=BLUE Gain: 0.000
COLOR=RED Gain: 0.000
COLOR=GREEN Gain: 0.000
SHAPE=SQUARE Gain: 0.000
SHAPE=TRIANGLE Gain: 0.000
SHAPE=CIRCLE Gain: 1.000
Best feature: SHAPE=CIRCLE
Learned Disjunct: SIZE=SMALL & SHAPE=CIRCLE
Final Definition: COLOR=RED & SIZE=BIG v SIZE=SMALL & SHAPE=CIRCLE

Rule Pruning in FOIL

- Prepruning method based on *minimum description length* (MDL) principle.
- Postpruning to eliminate unnecessary complexity due to limitations of greedy algorithm.

For each rule, *R*
  For each antecedent, *A*, of rule
    If deleting *A* from *R* does not cause negatives to become covered
    then delete *A*
  For each rule, *R*
    If deleting *R* does not uncover any positives (since they are redundantly covered by other rules)
    then delete *R*

RIPPER (Repeated Incremental Pruning to Produce Error Reduction)

- 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

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

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.
  - The stopping condition:
    - 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,
    - or when there are no more positive examples.

```
Ripper(Pos,Neg,k)
   RuleSet ← LearnRuleSet(Pos,Neg)
   For k times
      RuleSet ← OptimizeRuleSet(RuleSet,Pos,Neg)
LearnRuleSet(Pos,Neg)
   RuleSet ← ∅
   DL ← DescLen(RuleSet,Pos,Neg)
   Repeat
      Rule ← LearnRule(Pos,Neg)
      Add Rule to RuleSet
      DL' ← DescLen(RuleSet,Pos,Neg)
      If DL'>DL+64
         PruneRuleSet(RuleSet,Pos,Neg)
         Return RuleSet
      If DL'<DL DL ← DL'
         Delete instances covered from Pos and Neg
   Until Pos = ∅
   Return RuleSet
```

DL: description length of the rule base

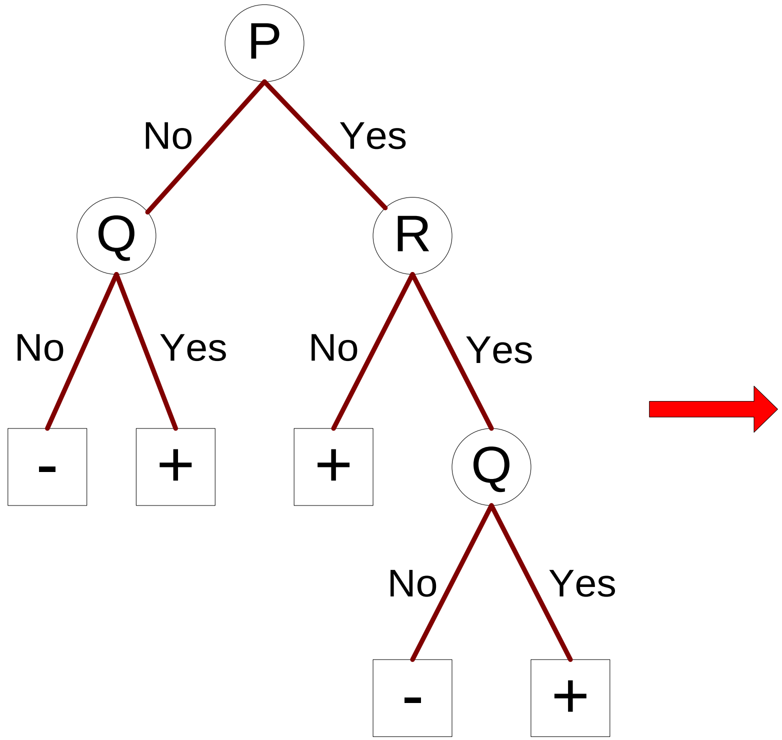
The description length of a rule base
= (the sum of the description lengths of all the rules in the rule base)
+ (the description of the instances not covered by the rule base)

RIPPER: Postprocess the rules

- 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

```
PruneRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet in reverse order
    DL ← DescLen(RuleSet,Pos,Neg)
    DL' ← DescLen(RuleSet-Rule,Pos,Neg)
    IF DL'<DL Delete Rule from RuleSet
  Return RuleSet
OptimizeRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet
      DL0 ← DescLen(RuleSet,Pos,Neg)
      DL1 ← DescLen(RuleSet-Rule+
        ReplaceRule(RuleSet,Pos,Neg),Pos,Neg)
      DL2 ← DescLen(RuleSet-Rule+
        ReviseRule(RuleSet,Rule,Pos,Neg),Pos,Neg)
      If DL1=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
            add ReplaceRule(RuleSet,Pos,Neg)
      Else If DL2=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
            add ReviseRule(RuleSet,Rule,Pos,Neg)
  Return RuleSet
```

Indirect Methods



Rule Set

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

Indirect Method: C4.5rules

- Extract rules from an unpruned decision tree
- For each rule, r: A → y,
  - consider an alternative rule r': A' → y where A' is obtained by removing one of the conjuncts in A
  - Compare the pessimistic error rate for r against all r's
  - Prune if one of the r's has lower pessimistic error rate
  - Repeat until we can no longer improve generalization error

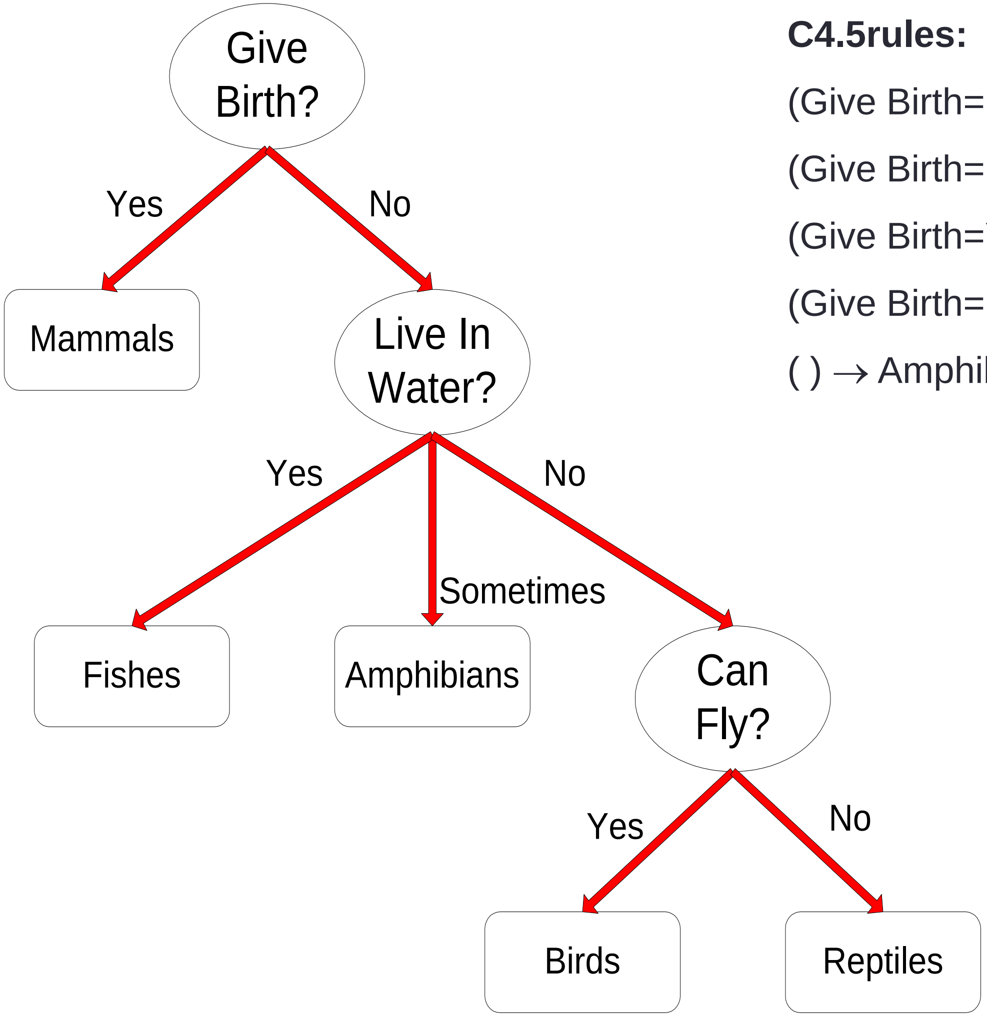
Indirect Method: C4.5rules

- Instead of ordering the rules, order subsets of rules (class ordering)
  - Each subset is a collection of rules with the same rule consequent (class)
  - Compute description length of each subset
    - Description length = L(error) + g L(model)
    - g is a parameter that takes into account the presence of redundant attributes in a rule set
(default value = 0.5)

Example

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds

C4.5 versus C4.5rules versus RIPPER



C4.5rules:

(Give Birth=No, Can Fly=Yes) $\rightarrow$ Birds

(Give Birth=No, Live in Water=Yes) $\rightarrow$ Fishes

(Give Birth=Yes) $\rightarrow$ Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) $\rightarrow$ Reptiles

( ) $\rightarrow$ Amphibians

RIPPER:

(Live in Water=Yes) $\rightarrow$ Fishes

(Have Legs=No) $\rightarrow$ Reptiles

(Give Birth=No, Can Fly=No, Live In Water=No) $\rightarrow$ Reptiles

(Can Fly=Yes,Give Birth=No) $\rightarrow$ Birds

() $\rightarrow$ Mammals

C4.5 versus C4.5rules versus RIPPER

C4.5 and C4.5rules:

			PREDICTED CLASS			
		Amphibians	**Fishes**	**Reptiles**	**Birds**	**Mammals**
ACTUAL	**Amphibians**	2	0	0	0	0
CLASS	**Fishes**	0	2	0	0	1
	Reptiles	1	0	3	0	0
	Birds	1	0	0	3	0
	Mammals	0	0	1	0	6

RIPPER:

			PREDICTED CLASS			
		Amphibians	**Fishes**	**Reptiles**	**Birds**	**Mammals**
ACTUAL	**Amphibians**	0	0	0	0	2
CLASS	**Fishes**	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees