Rules Models

J. Savoy
University of Neuchâtel

Ian H. Witten, Eibe Frank, M.A: Hall: Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann

A. Rajaraman, D. Ullman: Mining of Massive Datasets. Cambridge University Press

P. Flach: Machine Learning Cambridge University Press, 2012

Rule Models

- Represent the knowledge in the form of a set of rules
- The body of the rule indicates the conditions
  - Logical formula with connectors $\cup$ $\cap$ $\neg$
  - Constraints can be imposed in some methods (e.g., only conjunctive form)
  - May have literals (atomic expression) or more complex expressions

Rule Models

- Follow a general pattern
  IF (left) THEN (right)
  left = left-hand side = LHS = antecedent
  right = right-hand side = RHS = conclusion

- The head indicates the conclusion / class / label
- The body indicates the conditions
- The set of rules maybe ordered or unordered
- Example
  - <u>If</u> (length > 2) $\cap$ (dangerous=*yes*) <u>then</u> shark

Overview

- **Ordered Rule Lists**
- Unordered Rule Lists

Ordered Rule List

- Represent the instance set in the form of a set of rules
- The *order* of the rules is *important*
- We want a *small* set of rules
- Each rule must cover the largest possible subset of the training set
- The principle
  Build a conjunctive rule by adding a literal that improves its homogeneity
  - stop when some homogeneity criterion is satisfied
    or when encounter only simple cases

Ordered Rule List

- Homogeneity criterion (or purity criterion)?
- Proportion of positives examples covered by the literal (assuming a binary classification)
  Can be represented by a probability estimate
- Other purity measures are possible
  but tend to produce the same result
- The main idea:
  - Select the purest literal, form a rule
  - From the training set, ignore examples covered by the proposed rule
  - Continue until some stopping criterion is reached

Example

- Discriminate between two classes
- Attributes / values
  - Length = 3 / 4 / 5 — the body length
  - Gills = True / False — present or absent
  - Beak = True / False — present or absent
  - Teeth = Many / Few — how many teeth
- Species
  - Dolphin (or not)

Example

Set of examples (training set)

Length	Gills	Beak	Teeth	Dolphin
3	no	yes	many	yes
4	no	yes	many	yes
3	no	yes	few	yes
5	no	yes	many	yes
5	no	yes	few	yes
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

Ordered Rule List

- Starting point: 5 pos, 5 neg
- List of possible literal

length = *3*	2 pos, 0 neg
gills = *yes*	0 pos, 4 neg
beak = *yes*	5 pos, 3 neg
beak = *no*	0 pos, 2 neg
teeth = few	2 pos, 1 neg
teeth = many	3 pos, 4 neg

  …
- How many literal can we have? 3 x 2 x 2 x 2 = 24
- Which one is the purest?
  As measure: min(p, 1-p) or ½ - |p – ½|
  with p = probability of the pos. category

Ordered Rule List

- Starting point: 5 pos, 5 neg
- List of possible literal

length = *3*	2 pos, 0 neg: p = 2/2, q=0/2
gills = *yes*	0 pos, 4 neg: p = 0/4, q=4/4
beak = *yes*	5 pos, 3 neg: p = 5/8, q=3/8
beak = *no*	0 pos, 2 neg: p = 0/2, q=2/2
teeth = many	3 pos, 4 neg: p = 3/7, q=4/7

…

- Not clear!

IF (gills = *yes*) THEN class ⊗
IF (beak = *no*) THEN class ⊗
IF (length = *3*) THEN class ⊕

Ordered Rule List

- We select the rule covering the largest number of observations

  R1: IF (gills = *yes*) THEN class ⊗
- The remaining training set

Length	Gills	Beak	Teeth	Dolphin
3	no	yes	many	yes
4	no	yes	many	yes
3	no	yes	few	yes
5	no	yes	many	yes
5	no	yes	few	yes
4	no	yes	few	no

Ordered Rule List

- We select the rule covering the largest number of observations
  R2: IF (teeth = *many*) THEN class $\oplus$
- The remaining training set

Length	Gills	Beak	Teeth	Dolphin
3	no	yes	few	yes
5	no	yes	few	yes
4	no	yes	few	no

Ordered Rule List

- We select the rule covering the largest number of observations
  R3: IF (length = *4*) THEN class ⊗
- The remaining training set

Length	Gills	Beak	Teeth	Dolphin
3	no	yes	few	yes
5	no	yes	few	yes

all instances belong to the same class

R4: ELSE class ⊕

Ordered Rule List

- We have found the ordered rule list

  R1: IF (gills = *yes*) THEN class ⊗
  R2: IF (teeth = *many*) THEN class ⊕
  R3: IF (length = *4*) THEN class ⊗
  R4: ELSE class ⊕

- More formally, we have two algorithms
  - one to generate the list
  - one to generate one rule
- This is *separate-and-conquer* approach

Ordered Rule List

`LearnRuleList(D)` – learn an ordered list of rules

input: labeled training data D
output: rule list R

```
1. R ← ∅                # list is empty
2. while (D ≠ ∅) do     # set not empty
3.     r ← LearnRule(D)
4.     append r to the end of R
5.     D ← D \ {x ∈ D | x is covered by r}
6. end
7. return R    # return the rule list
```

Ordered Rule List

LearnRule(D) – learn one rule

input: labeled training data D; output: rule *r*

```
b ← true
L ← set of available literals
while not Homogeneous(D) do
    l ← BestLiteral(D, L)
    b ← b ∩ l
    D ← D \ {x ∈ D | x is covered by b}
    L ← L \ {l' ∈ L | l' uses same feature as l}
end
c ← Label(D)
r ← if b then class=c
return r
```

Ordered Rule List

- With our example we have found the ordered rule list

  R1: IF (gills = *yes*) THEN class ⊗
  R2: IF (teeth = *many*) THEN class ⊕
  R3: IF (length = *4*) THEN class ⊗
  R4: ELSE class ⊕

- Can we form a single rule?
- Can we form an unordered rule list?

Ordered Rule List

- In one rule
  IF (gills = *yes*) THEN class ⊗
    ELSE IF (teeth = *many*) THEN class ⊕
      ELSE IF (length = *4*) THEN class ⊗
        ELSE class ⊕
- The unordered rule list
  R1: IF (gills=*yes*) THEN class ⊗
  R2': IF (gills=*no*) ∩ (teeth=*many*) THEN class ⊕
  R3': IF (gills=*no*) ∩ (teeth=*few*) ∩ (length=*4*) THEN class ⊗
  R4': IF (gills=*no*) ∩ (teeth=*few*) ∩ (length≠*4*) THEN class ⊕

Length	Gills	Beak	Teeth	Dolphin
3	yes	yes	many	?

Ordered Rule List

- Many variants do exist
- Instead of being limited to literal, we can consider two literals together, e.g. $(p \cap q)$
- Relationships with decision tree

Overview

- Ordered Rule Lists
- **Unordered Rule Lists**

Unordered Rule List

- The main idea is to learn one class at the time
- Instead of min(p, 1-p), we can maximize p (empirical probability of the given class)
- Instead of being limited to literal, we can consider two literals together, e.g. $(p \cap q)$
- Relationships with decision tree

Example

Our training set

Length	Gills	Beak	Teeth	Dolphin
3	no	yes	many	yes
4	no	yes	many	yes
3	no	yes	few	yes
5	no	yes	many	yes
5	no	yes	few	yes
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

Unordered Rule List

- We start with the positive class
- We can begin with

  IF (length = 3) THEN class = $\oplus$
- We then remove the two examples covered by the rule

Length	Gills	Beak	Teeth	Dolphin
4	no	yes	many	yes
5	no	yes	many	yes
5	no	yes	few	yes
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

Unordered Rule List

- We continue with ?

Length	Gills	Beak	Teeth	Dolphin
4	no	yes	many	yes
5	no	yes	many	yes
5	no	yes	few	yes
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

- No single literal…

Unordered Rule List

- Thus we continue with
  IF (gills = *no*) ∩ (length = 5) THEN class = ⊕
- We remove two more examples…

Length	Gills	Beak	Teeth	Dolphin
4	no	yes	many	yes
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

- We finish with
  IF (gills = *no*) ∩ (teeth = *many*) THEN class = ⊕

Unordered Rule List

- And for the rest?

Length	Gills	Beak	Teeth	Dolphin
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

- Default rule: ELSE class = ⊗
- Introduces a bias in favor of the second class
- Can we have a better description of the second class?

Unordered Rule List

- Instead of simply adding
  ELSE class = no
- Start the learning with the second class (no) but with the remaining examples

Length	Gills	Beak	Teeth	Dolphin
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

Unordered Rule List

- First?
- IF (gills = *yes*) THEN class = ⊗

Length	Gills	Beak	Teeth	Dolphin
5	yes	yes	many	no
4	yes	yes	many	no
5	yes	no	many	no
4	yes	no	many	no
4	no	yes	few	no

- Then?
  IF (length = 4) ∩ (teeth = *few*) THEN class = ⊗

Unordered Rule List

- The final set of rules is

R1: IF (length = 3) THEN class = $\oplus$

R2: IF (gills = *no*) $\cap$ (length = 5) THEN class = $\oplus$

R3: IF (gills = *no*) $\cap$ (teeth = *many*) THEN class = $\oplus$

R4: IF (gills = *yes*) THEN class = $\otimes$

R5: IF (length = 4) $\cap$ (teeth = *few*) THEN class = $\otimes$

Length	Gills	Beak	Teeth	Dolphin
3	yes	yes	many	?

Unordered Rule List

- More formally
- 1. We need to iterate over all possible classes
- 2. For each class, we determine the set of needed rules
- But, in this second step, we remove covered examples and remove the used literal(s).

Unordered Rule List

LearnRuleSet(D) – learn an unordered list of rules

input: labeled training data D;   output: rule set R

```
R ← ∅
for every class ci do
    Di ← D    # select the subset of class ci
    while Di contains example of class ci do
        r ← LearnRuleForClass(Di, ci)
        R ← R ∪ {r}
        Di ← Di \ {x ∈ ci | x is covered by r}
    end
end
return R
```

Unordered Rule List

LearnRuleForClass(D, c_i) – learn one rule for a class

input: labeled training data D, class c_i;   output: rule *r*

```
b ← true
L ← set of available literals
while not Homogeneous(D) do
    l ← BestLiteral(D, L, c_i)
    b ← b ∩ l
    D ← D \ {x ∈ D | x is covered by b}
    L ← L \ {l' ∈ L | l' uses same feature as l}
end
r ← if b then class = c_i
return r
```

Unordered Rule List

- The suggested algorithm is myopia…
  we have formed
  R1: IF (length = 3) THEN class = ⊕
  but a better choice could be
  R1': IF (gills = *no*) ∩ (teeth = *many*) THEN class = ⊕

- We can apply a beam search to improve the overall quality of the rules
- And with a new observation?

Length	Gills	Beak	Teeth	Dolphin
4	yes	yes	many	?

Example: Weather Problem

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Example: Weather Problem

R1: IF (outlook = overcast) THEN class = yes

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
rainy	mild	high	true	no

Example: Weather Problem

R1: IF (outlook = overcast) THEN class = yes
R2: IF (temperature = hot) THEN class = not

Outlook	Temperature	Humidity	Windy	Play
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
rainy	mild	high	true	no

Example: Weather Problem

R1: IF (outlook = overcast) THEN class = yes
R2: IF (temperature = hot) THEN class = no
R3: IF (outlook = rainy) $\cap$ (windy = false) THEN class = yes

Outlook	Temperature	Humidity	Windy	Play
rainy	cool	normal	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes
rainy	mild	high	true	no

Example: Weather Problem

R1: IF (outlook = overcast) THEN class = yes
R2: IF (temperature = hot) THEN class = no
R3: IF (outlook = rainy) $\cap$ (windy = false) THEN class = yes
R4: IF (humidity=normal) $\cap$ (outlook=sunny) THEN class = yes
R5: ELSE class = no

Outlook	Temperature	Humidity	Windy	Play
rainy	cool	normal	true	no
sunny	mild	high	false	no
rainy	mild	high	true	no

Conclusion

- Generating a set of rules is a simple approach
- But could be time comsuming $O(n) = 2^n$

- Called also *concept learning*
- May work well in some contexts, and poorly in others

- Assume that all attribute-value pairs can be used to predict the correct classifiaction
- The decision can be easily explained to the final user