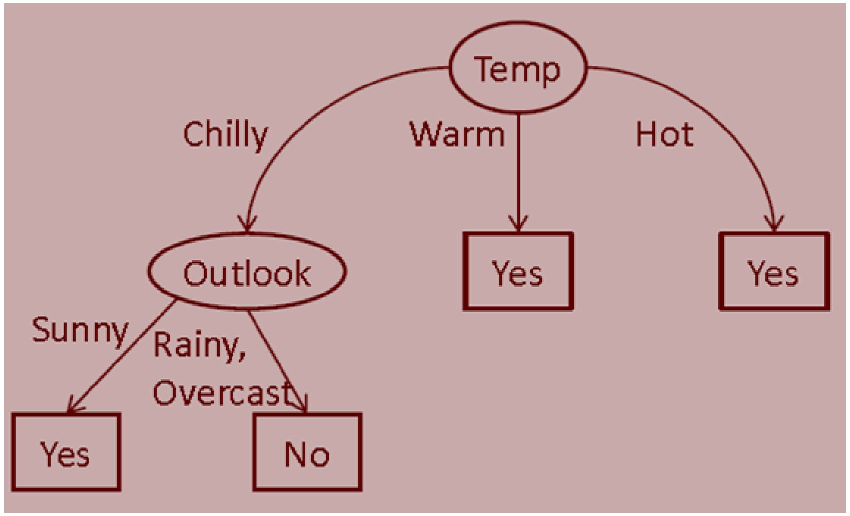
Classification rules

Lecture 07

by *Marina Barsky*

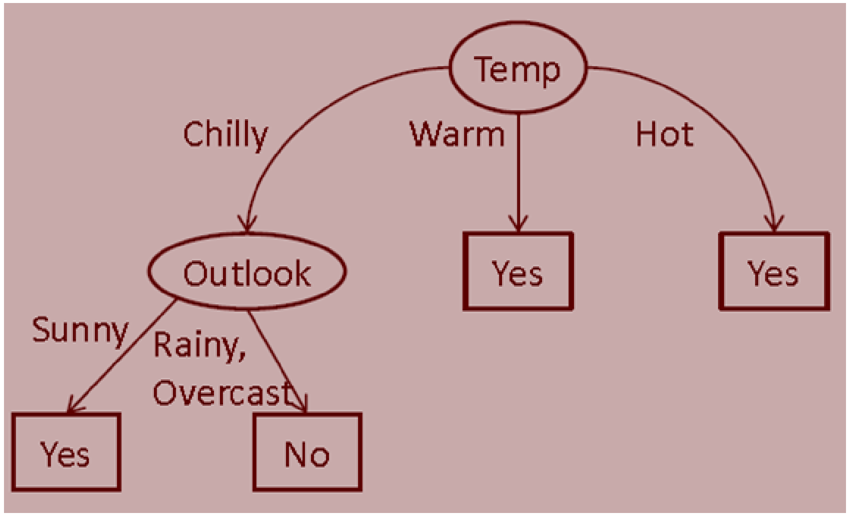
From trees to rules: how?

- How can we produce a set of rules from a decision tree?



Start from the leafs (class labels)

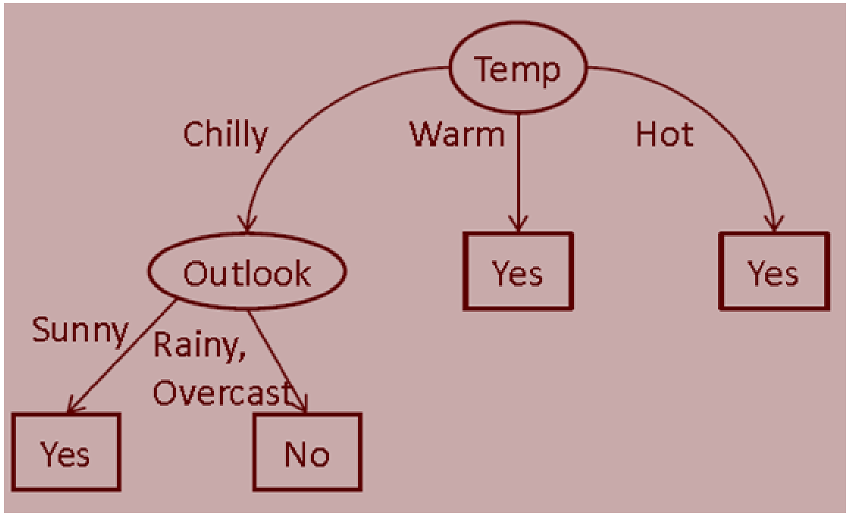
- One rule for each leaf



If Temp = "Warm" **then** play
If Temp = "Hot" **then** play
If Temp = "Chilly" and Outlook="Sunny" **then** play
Default: no play

Rules can be more comprehensive

- The set of rules can be minimized

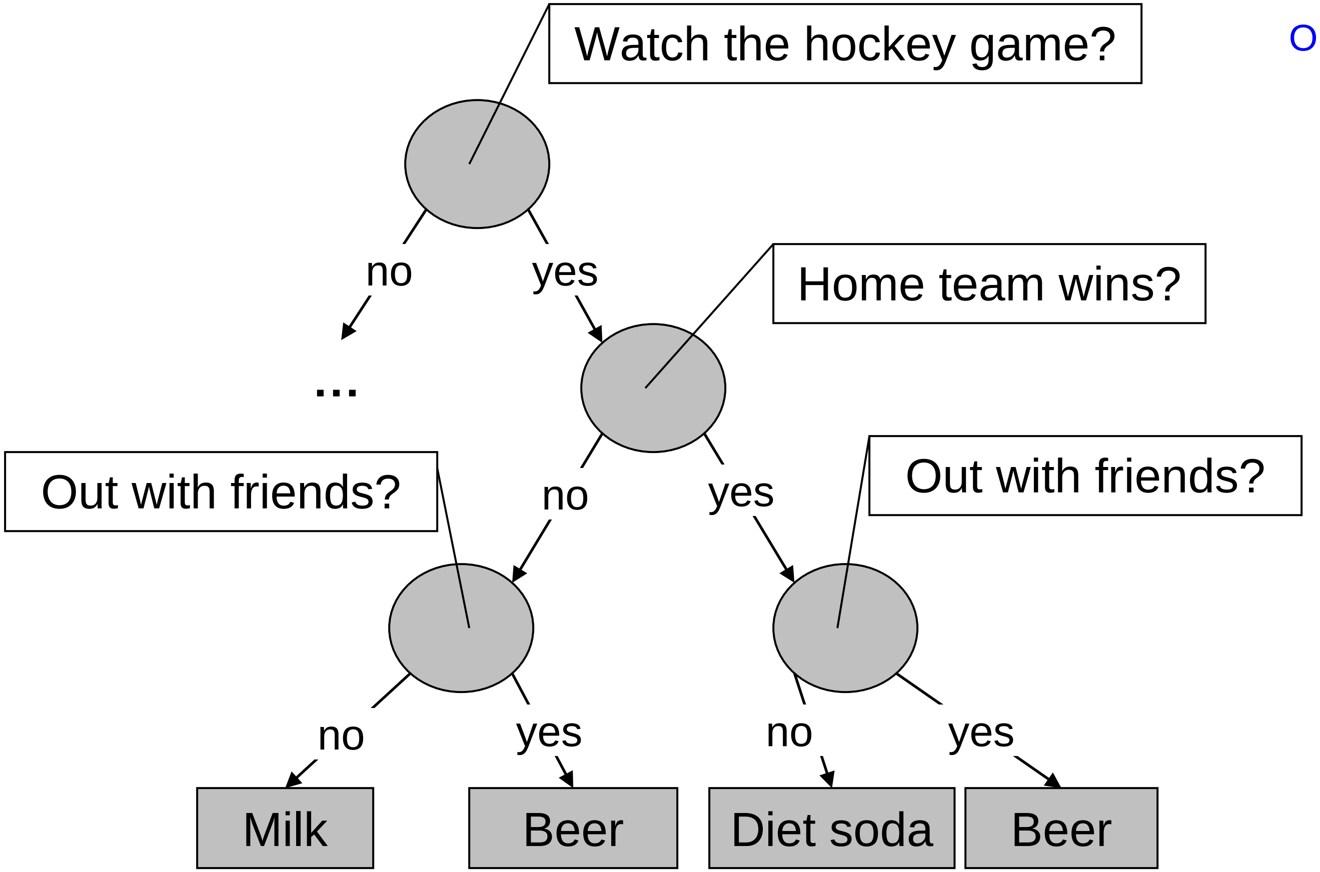


If Temp = “Chilly” and (Outlook=“Rainy” or Outlook = “Overcast”)
then no play

Default: play

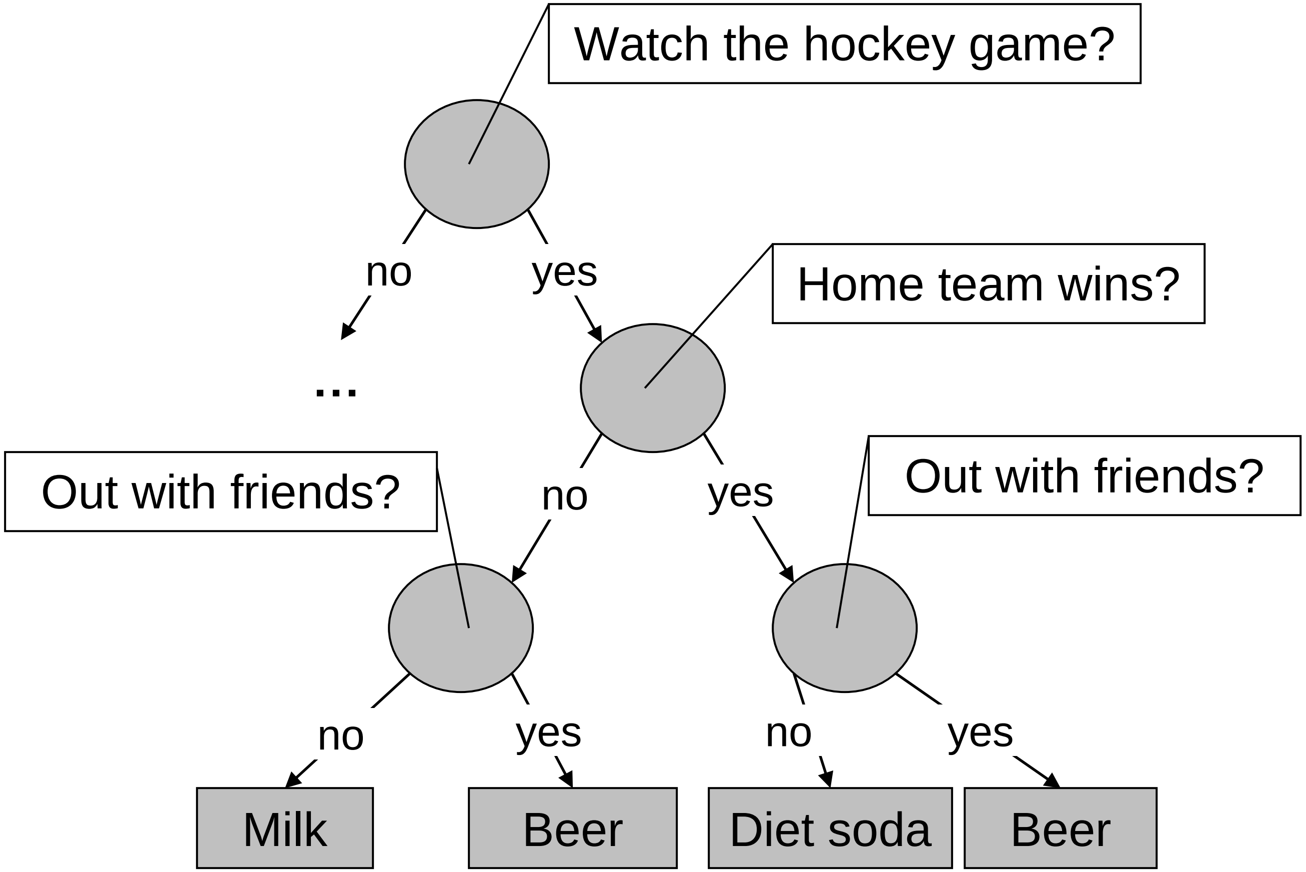
Decision tree – as a collection of rules

One rule for each leaf



If watch the game **and** home team wins **and** sitting at home **then** diet soda
If watch the game **and** home team wins **and** out with friends **then** beer
If watch the game **and** home team loses **and** sitting at home **then** milk
If watch the game **and** home team loses **and** out with friends **then** beer

We can collapse several branches into one rule



If watch the game **and** home team wins **and** sitting at home **then** diet soda
If watch the game **and** home team loses **and** sitting at home **then** milk

If watch the game **and** out with friends **then** beer

Classification rules – bottom-up approach (start from the class)

- Decision tree starts with attribute values (top-down approach)
- Classification rules start with the class label (bottom-up)

?

(*Condition*) → ***class label*** ◀ We start here

- *LHS*: rule *antecedent* or condition
- *RHS*: rule *consequent*

Example: animal classification

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

Animal classification rules

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) ∧ (Can Fly = yes) → Birds

R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

Rule *coverage*

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

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → 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 quality: coverage and accuracy

- **Coverage** of a rule:
  - Fraction of records that satisfy the condition of a rule (over all records)
- **Accuracy** of a rule:
  - Fraction of records that satisfy both the condition and the class (over those that satisfy only the condition)

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) → No

Coverage = 40%, Accuracy = 50%

Using rules for classification

- Rules are ranked according to their quality (e.g. accuracy and coverage)
- An ordered rule set is known as a **decision list, or decision table**
- 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) ∧ (Can Fly = yes) → Birds

R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

Stop here

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

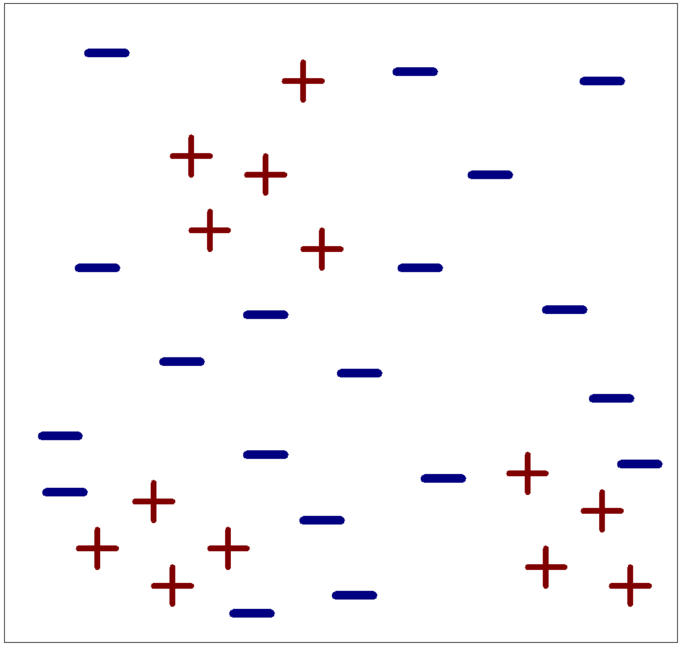
Algorithms for generating the rules

- From decision trees (*divide-and-conquer*)

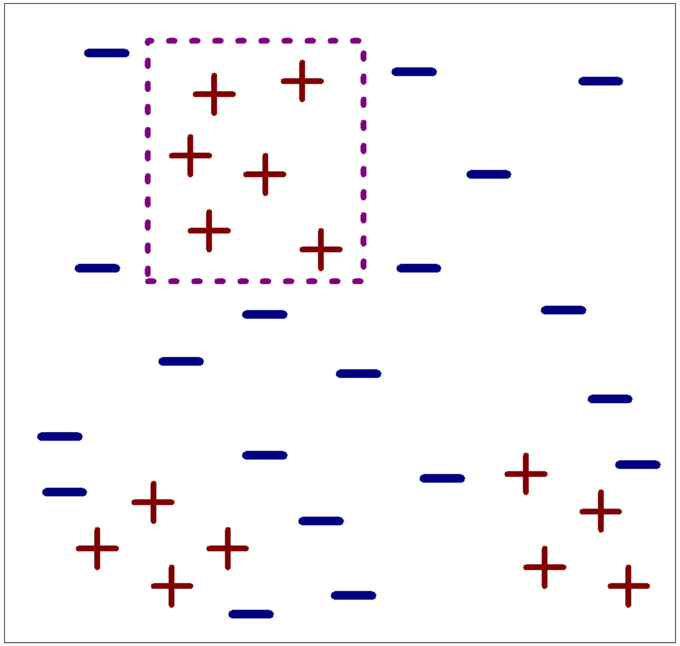
- **Rule covering approach** (*separate and conquer*):
  - At each step – take a class and find a condition which covers most instances in this class
  - The goal - to cover all instances

Building Classification Rules: Sequential Covering

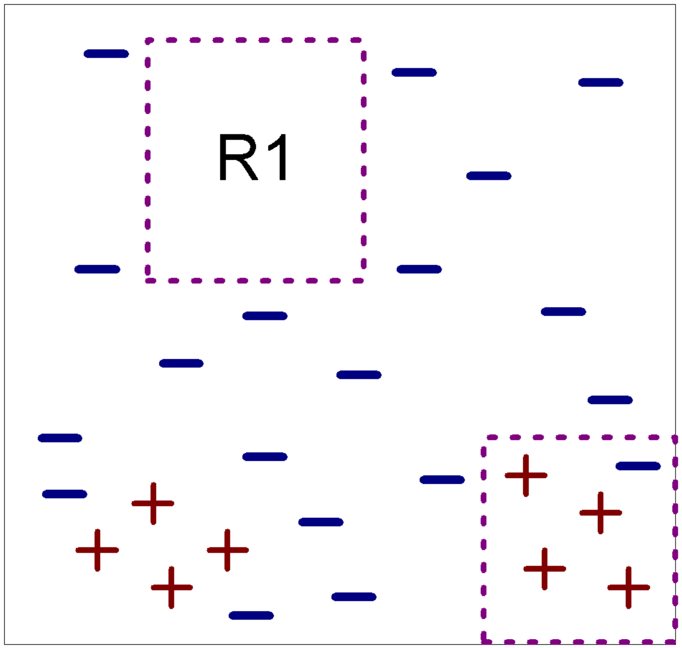
1. Start from an empty rule
2. Grow a rule using some **Learn-One-Rule** function
3. Remove training records **covered** by the rule
4. Repeat Step (2) and (3) until stopping criterion is met



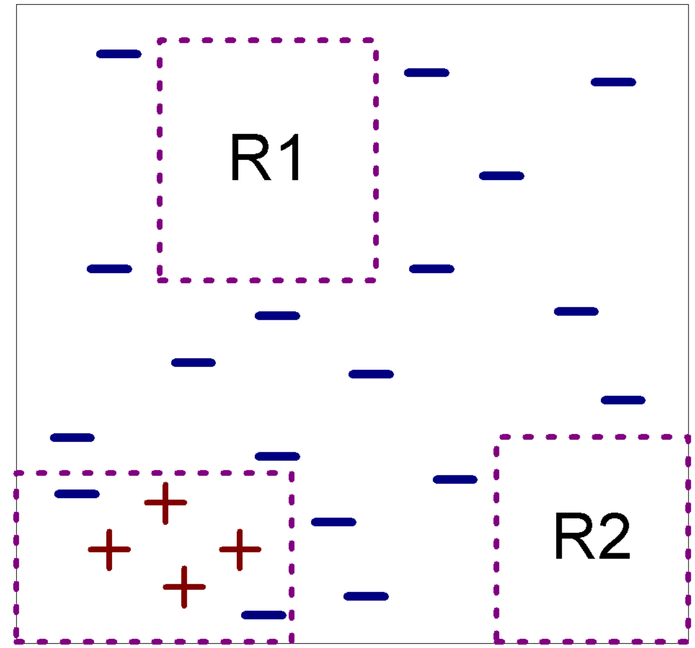
(i) Original Data



(ii) Step 1



(iii) Step 2



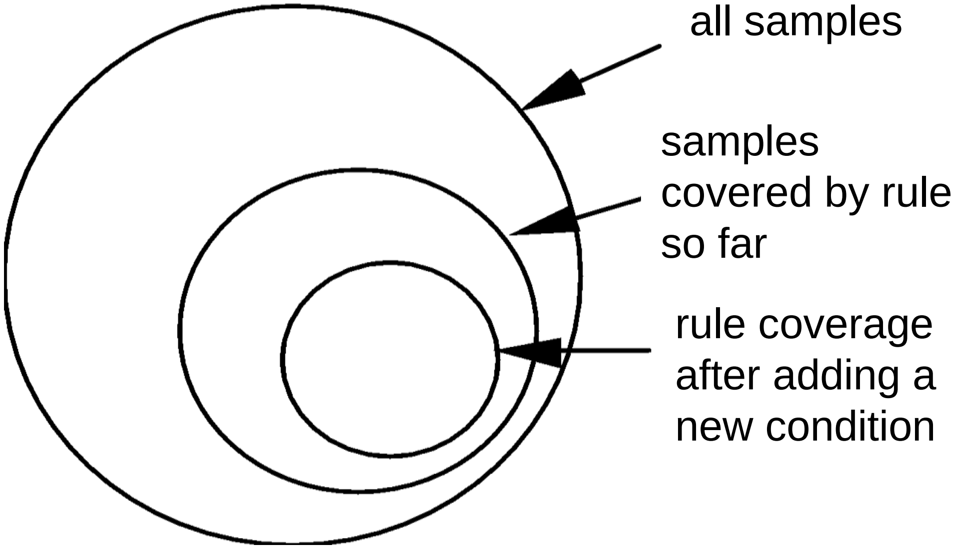
(iv) Step 3

This approach is called a **covering** approach because at each stage a rule is identified that covers some of the instances

A simple covering algorithm idea

- Generate a rule by adding tests that maximize rule's accuracy
  - Similar to situation in decision trees: problem of selecting an attribute to split on
  - But: decision tree inducer maximizes overall purity, for a rule it is important to have purity for only a selected class

Each new test (improving accuracy) reduces rule's coverage.



Rule learning example:
Weather dataset

Outlook	Temp	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

If ? **Then** Yes

Outlook	**Temp**	**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

Try each attribute - estimate accuracy:

If outlook=sunny then yes: 2/5
If outlook=overcast then yes: 4/4
If outlook=rainy then yes: 3/5

If temp=cool then yes: 3/4
If temp=mild then yes: 4/6
If temp=hot then yes: 2/4

If humidity=normal then yes: 6/7
If humidity=high then yes: 4/7

If windy=true then yes: 4/6
If windy=false then yes: 5/8

If ? **Then** no

Outlook	Temp	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

If outlook=sunny then no: 3/5
If outlook=overcast then no: 0/4
If outlook=rainy then no: 2/5

If temp=cool then no: 1/4
If temp=mild then no: 2/6
If temp=hot then no: 2/4

If humidity=normal then no: 1/7
If humidity=high then no: 3/7

If windy=true then no: 2/6
If windy=false then no: 3/8

R1: if outlook=overcast then yes: 4/4

Outlook	Temp	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

Remove instances covered by R1

Rules so far:
R1: if outlook=overcast → yes

Continue with the remaining subset

Outlook	Temp	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

If ? **Then** Yes

If outlook=sunny then yes: 2/5
If outlook=rainy then yes: 3/5

If temp=cool then yes: 2/3
If temp=mild then yes: 3/5
If temp=hot then yes:0/2

If humidity=normal then yes: 4/5
If humidity=high then yes: 1/5

If windy=true then yes: 1/4
If windy=false then yes: 4/6

Rules so far:
R1: if outlook=overcast → yes

Continue with the remaining subset

Outlook	Temp	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

If ? **Then** No

If outlook=sunny then no: 3/5
If outlook=rainy then no: 2/5

If temp=cool then no: 1/3
If temp=mild then no: 2/5
If temp=hot then no: 2/2

Let's assume that the coverage should be at least 3

If humidity=normal then no: 1/5
If humidity=high then no: 4/5

If windy=true then no: 3/4
If windy=false then no: 2/6

We can choose between:
If humidity=high then no: 4/5
If humidity=normal then yes: 4/5

Both have the same accuracy and coverage

Rules so far:
R1: if outlook=overcast → yes

R2: If humidity=normal AND ? then Yes

We want to make 100% accuracy

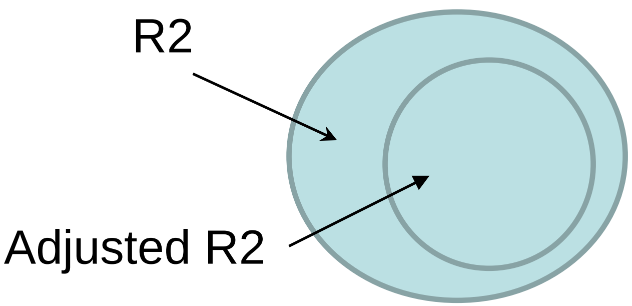
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

If humidity= normal **and** ?
Then Yes

If outlook=sunny then yes: 2/2
If outlook=rainy then yes: 2/3

If temp=cool then yes: 2/3
If temp=mild then yes: 2/2

If windy=true then yes: 1/2
If windy=false then yes: 3/3



Rules so far:
R1: if outlook=overcast → yes

R2: If humidity=normal AND windy=False then Yes: 3/3

Outlook	Temp	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

Remove instances covered by R2

Rules so far:
R1: if outlook=overcast → yes
R2: if humidity=normal and windy=False → yes

Continue with the remaining subset

Outlook	Temp	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

If ? Then no

We do not consider *Yes* rules anymore. Why?

If outlook=sunny then no: 3/4
If outlook=rainy then no: 2/3

If temp=cool then no: 1/1
If temp=mild then no: 2/4
If temp=hot then no: 2/2

If humidity=normal then no: 1/2
If humidity=high then no: 4/5

If windy=true then no: 3/4
If windy=false then no: 2/3

Rules so far:
R1: if outlook=overcast → yes
R2: if humidity=normal and windy=False → yes

R3: if humidity=high and outlook=sunny then No: 3/3

Outlook	Temp	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

Remove instances covered by R3

Rules so far:
R1: if outlook=overcast → yes
R2: if humidity=normal and windy=False → yes
R3: if humidity=high and outlook=sunny → no

R3: if humidity=high and outlook=sunny then No: 3/3

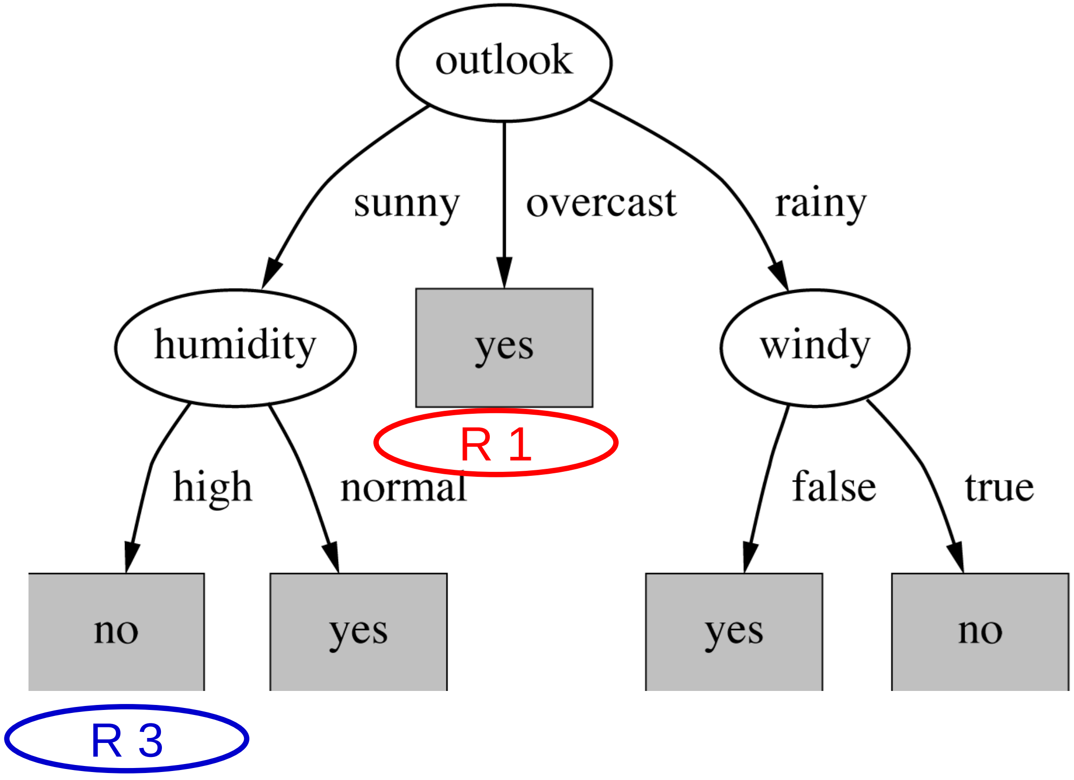
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Because all the remaining rules have coverage < 3, we do not consider them - 4 records are assigned to a default class

Rules so far:
R1: if outlook=overcast → yes
R2: if humidity=normal and windy=False → yes
R3: if humidity=high and outlook=sunny → no

Each rule corresponds to some path in the decision tree

R1: if outlook=overcast → yes
R2: if humidity=normal and windy=False → yes
R3: if humidity=high and outlook=sunny → no



Difference between decision trees and rules

Rules are more readable than decision trees

Decision trees describe the **general concept** extracted from the data, while each rule represents **a nugget of knowledge**

Trees contain predictions for **all class variables**, while each rule predicts only **one class value**

Pseudocode for *PRISM* algorithm

Original paper

Initialize E to all records
Until E is not empty do:
    E = **learn-one-rule** (E)

Algorithm **learn-one-rule** (set E):

For each class C

Initialize E_C to all instances with class label C

Create a rule R with an empty LHS that predicts class C

For each attribute A_i and each attr. value v_j,

check accuracy of rule: "if $A_i = v_j$ then C"

Accuracy:
total with LHS and class C/all with LHS

Start with condition $A_k = v_m$ which maximizes the accuracy of R

Until R is perfect (or there are no more attributes to use) do

For each attribute A_i not mentioned in R, and each attr. value v_j,

consider adding the condition $A_i = v_j$ to the LHS of R

Select condition $A_k = v_m$ to maximize the accuracy of R

(break ties by choosing the condition with larger coverage)

Remove the instances covered by R from E

Return remaining instances

Coverage:
all with LHS/total size of E

Separate and conquer

- Methods like PRISM (for dealing with one class) are ***separate-and-conquer*** algorithms:
  - First, a rule is identified
  - Then, all instances covered by the rule are separated out
  - Finally, the remaining instances are “conquered”

- Difference to divide-and-conquer methods:
  - Subset covered by rule doesn’t need to be explored any further

Full step-by-step example: contact lenses data

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended Lenses
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

Setting up rule's consequent

- Rule we seek:

```
If ?
    then recommendation = hard
```

- Possible tests:

Age = Young	2/8
Age = Pre-presbyopic	1/8
Age = Presbyopic	1/8
Spectacle prescription = Myope	3/12
Spectacle prescription = Hypermetrope	1/12
Astigmatism = no	0/12
Astigmatism = yes	4/12
Tear production rate = Reduced	0/12
Tear production rate = Normal	4/12

The numbers on the right show the fraction of "correct" instances in the set singled out by that choice.
In this case, correct means that their recommendation is "hard."

Modified rule and resulting data

- Rule with best test added:

```
If astigmatism = yes
    then recommendation = hard
```

- Instances covered by modified rule:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

The rule isn't very accurate, getting only 4 out of 12 that it covers. So, it needs further refinement.

Further refinement

- Current state:

```
If astigmatism = yes
    and ?
  then recommendation = hard
```

- Possible tests:

Age = Young	2/4
Age = Pre-presbyopic	1/4
Age = Presbyopic	1/4
Spectacle prescription = Myope	3/6
Spectacle prescription = Hypermetrope	1/6
Tear production rate = Reduced	0/6
Tear production rate = Normal	4/6

Modified rule and resulting data

- Rule with best test added:

```
If astigmatism = yes
     and tear production rate =
normal
   then recommendation = hard
```

- Instances covered by modified rule:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Should we stop here? Perhaps. But let’s say we are going for exact rules, no matter how complex they become.
So, let’s refine further.

Further refinement

- Current state:

```
If astigmatism = yes
     and tear production rate = normal
     and ?
   then recommendation = hard
```

- Possible tests:

Age = Young	2/2
Age = Pre-presbyopic	1/2
Age = Presbyopic	1/2
Spectacle prescription = Myope	3/3
Spectacle prescription = Hypermetrope	1/3

- Tie between the first and the fourth test
  - We choose the one with greater coverage

The result

- Final rule:

```
If astigmatism = yes
   and tear production rate = normal
   and spectacle prescription = myope
   then recommendation = hard
```

- Second rule for recommending “hard lenses”:
  (built from instances not covered by first rule)

```
If age = young and astigmatism = yes
   and tear production rate = normal
   then recommendation = hard
```

- These two rules cover all “hard lenses”:
  - Process is repeated with other two classes

Rule learners

1. PRISM – as we learned. Only nominal attributes (Cendrowska)
2. RIDOR - **RI**pple-**DO**wn **R**ule learner (Gaines and Compton)
3. PART (Eibe and Witten)
4. JRip - Repeated Incremental Pruning to Produce Error Reduction (William W. Cohen)
5. Decision table (Kohavi)