# IF4071 Pembelajaran Mesin: Decision Tree Learning

Sumber utama: Bab 2 Machine Learning (Tom M. Mitchell, 1997)
Materi Kuliah IF6051 Pembelajaran Mesin sem 1 2010/2011 (Dosen: NUM)
Update: Masayu Leylia Khodra
S1-IF ITB

# Overview Konsep Pembelajaran

- Pembelajaran konsep
  - Task: given training examples D, determine a hypothesis h in H such that h(x) = c(x) for all x in D
  - Representasi hipotesis
- Asumsi fundamental inductive learning
- Pencarian hipotesis/version space:
  - Find-S
  - List-then-eliminate
  - Candidate-Elimination

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$$

# Review Latihan

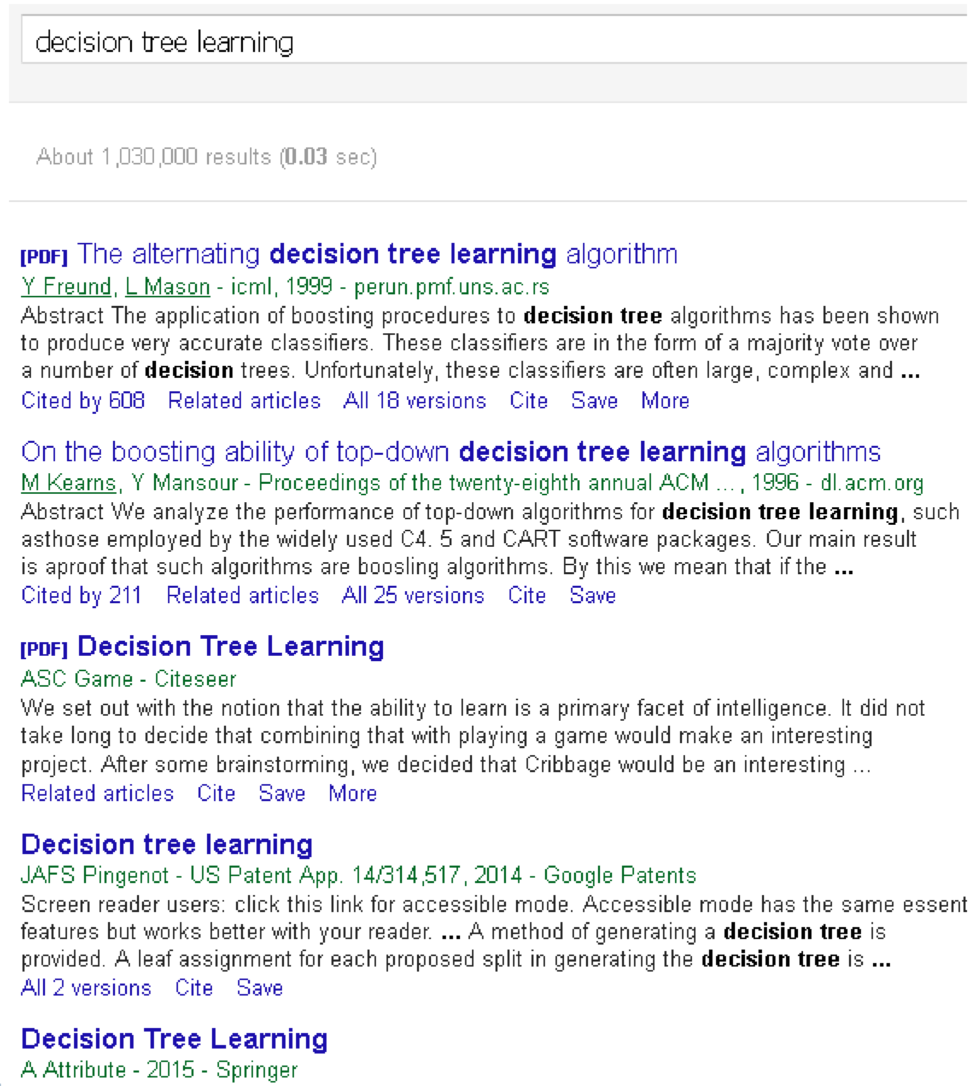
- Find-S: learning ✓; klasifikasi ✓.
- List-then-eliminate:
  - Learning: generate ruang hipotesis ✓, remove hipotesis yang tidak konsisten ✗
    - Contoh: <∅, ∅, ∅> pasti dihapus oleh instance + dan <?,?,?> pasti dihapus oleh instance -.
- Candidate elimination:
  - Learning: penanganan instance + ✓, instance - ✗
    - Contoh: G: {<?,?,F>} ; S: {<T,T,F>}
      Instance <F,T,T,-> ➔ G: {<?,?,F>~~,<T,?,?>~~} ; S: {<T,T,F>}
      <T,?,?> tidak konsisten dgn instances <T,T,T,-> dan <T,T,F,+>
- Klasifikasi LTE-CE ✗ atau tidak selesai
    - Contoh:VS:{<?,?,F>,<?,T,F>}
      Instance <T,F,T> ➔ confidence=1: - ; voting: -
      Instance <T,F,F> ➔ confidence=1: unknown; voting: unknown
      Instance <F,F,F> ➔ confidence=1: unknown; voting: unknown
- Hasil akhir: 1 A, 1 AB, 8 B, 4 BC, 2 C

# Outline

- Representasi: disjungsi dari konjungsi constraint nilai atribut.
- Outputs a single hypothesis (bukan version space)
- Complete hypotesis space of finite discrete-valued function; Incompletely search from search to complex hypotheses.
- Robust to noisy data: statistically-based search choices
  - Noise → overfit problem, more complex tree
- Inductive bias ID3: prefer shortest tree
- BFS-ID3 vs ID3

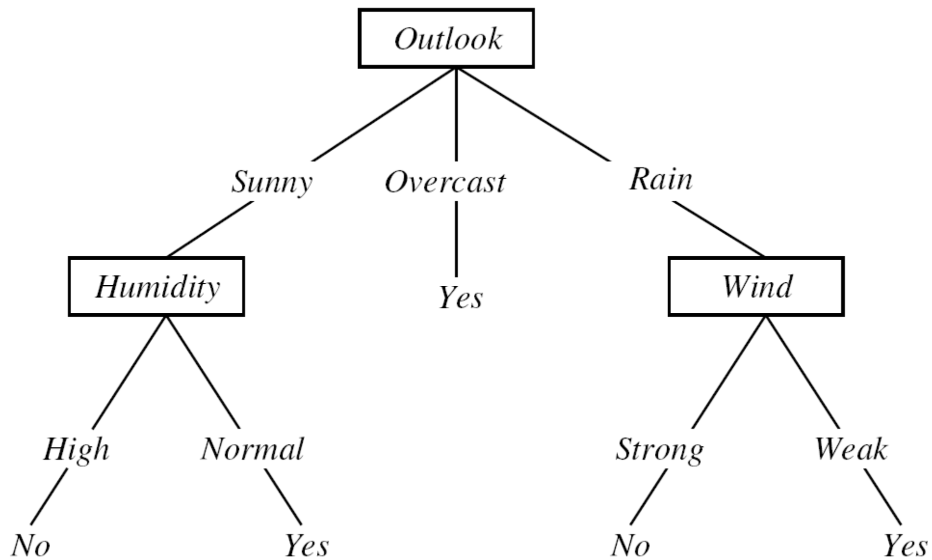
# Why Decision Tree Learning

- Metode pembelajaran induktif yang populer
- Sukses diaplikasikan ke berbagai task (diagnosa medis, kelayakan credit)
- Approximate discrete-valued function
  - Learned function: decision tree ≈ set of if-then rules
- Robust to noisy data
- Capable of learning disjunctive expression



# Decision Tree (DT): Representasi

Decision Tree (DT) for PlayTennis




- Setiap simpul internal mengecek suatu atribut
- Setiap cabang menyatakan nilai atribut
- Setiap daun memberikan hasil klasifikasi
- Disjungsi dari konjungsi constraints pada nilai atribut

(outlook=sunny $\wedge$ humidity=normal) $\vee$ (outlook=overcast) $\vee$ (outlook=rain $\wedge$ wind=weak)

<outlook=sunny, temperature=hot, humidity=high, wind=strong>: No

# Karakteristik Problem yang Cocok dengan Decision Tree Learning (DTL)

- Instances: <attribute=value>*, walaupun dapat juga menangani atribut kontinu
- Persoalan klasifikasi: fungsi target menghasilkan nilai diskrit
- Jika diperlukan deskripsi disjungsi
- Possibly noisy training data
- Possibly missing attribute values

# Decision Tree Learning: Top-down

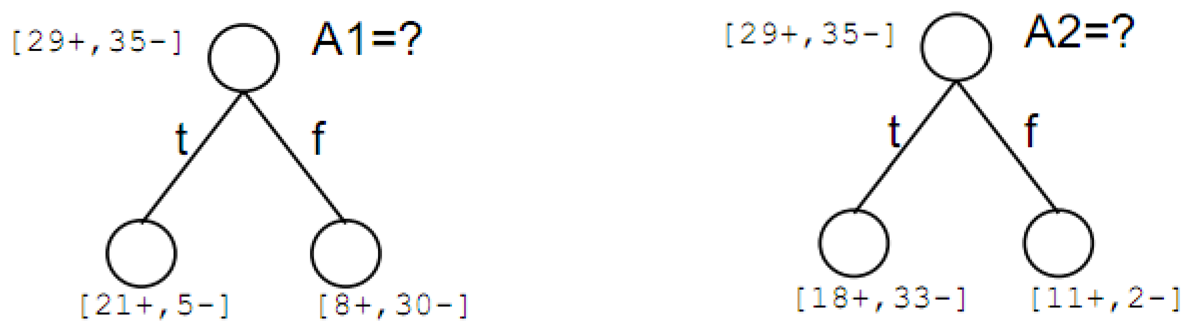
ID3(*Examples*, *Target_attribute*, *Attributes*)

*Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.*

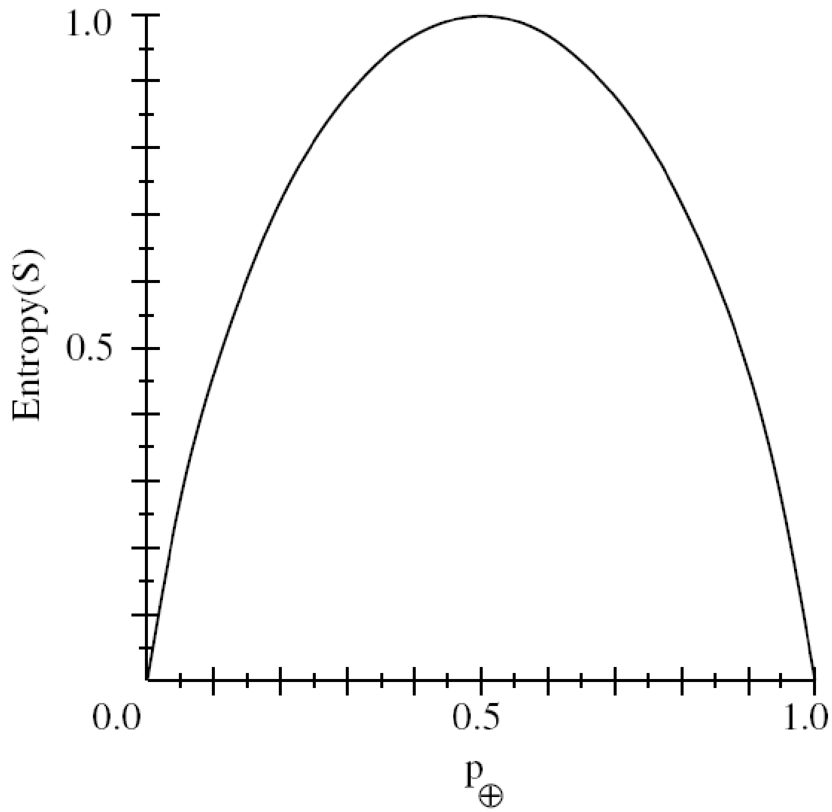
- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, $v_i$, of $A$,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value $v_i$ for $A$
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − $\{A\}$))
- End
- Return *Root*

# Which Attribute is the best classifier ?

- ID3 menggunakan information gain untuk memilih atribut terbaik dari kandidat atribut pada setiap langkahnya ketika membangun DT
- Information gain:
  - mengukur kemampuan suatu atribut untuk memisahkan training data berdasarkan kelas target
  - memerlukan pengukuran impurity dalam training data → entropy



# Fungsi Entropy untuk Boolean Classification



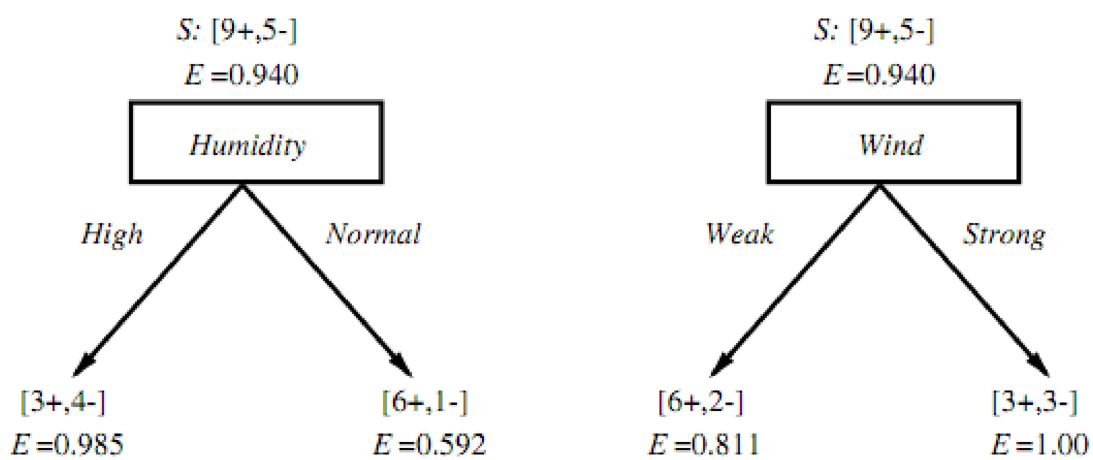

- S: training examples
- $P_+$: proporsi positive examples pada S
- $P_-$: proporsi negative examples pada S
- Entropy mengukur impurity dari S
- Entropy=0: semua examples dalam satu kelas
- Entropy=1: $p_+=p_-$

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

# Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on $A$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$




*Gain (S, Humidity )*
= .940 - (7/14).985 - (7/14).592
= .151

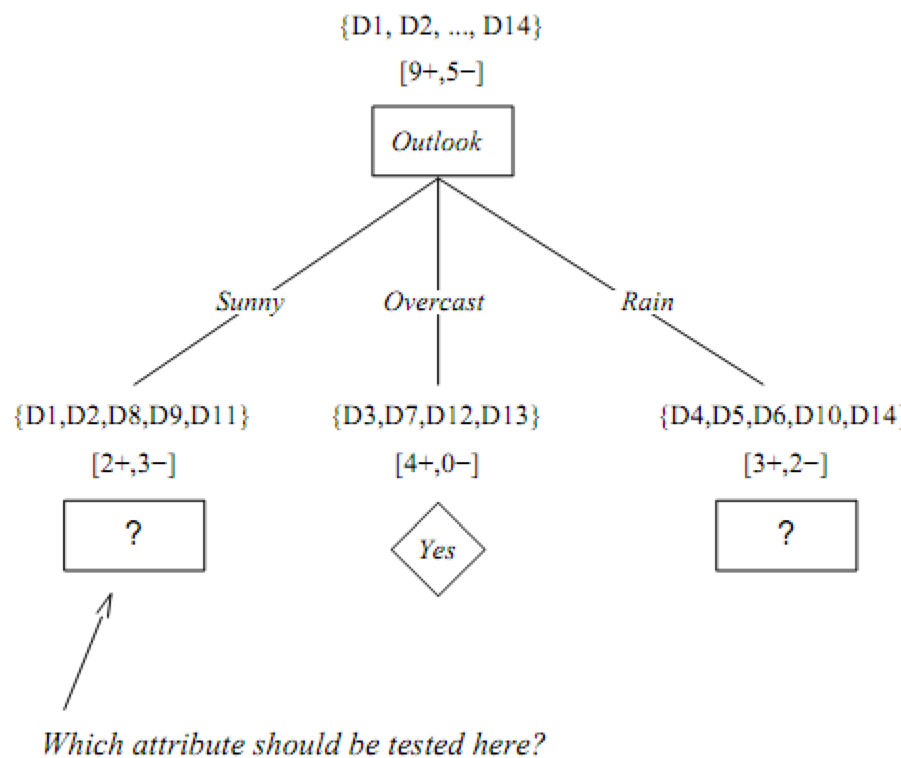
*Gain (S, Wind)*
= .940 - (8/14).811 - (6/14)1.0
= .048

# Training Examples

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# Contoh Pemilihan Atribut

Gain (S, outlook)=0.246
Gain (S, humidity)=0.151
Gain (S,wind)=0.048
Gain (S,temperature)=0.029




$S_{sunny} = \{D1, D2, D8, D9, D11\}$

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

# Hypothesis Space Search by ID3

- Complete space of finite discrete-valued function
- Outputs a single hypothesis (bukan version space)
  - Tidak dapat menentukan jumlah DT alternatif yang konsisten dengan training data
- Greedy: no backtracking, optimum lokal
- Statistically-based search choices
  - +: hasil pencarian tidak sensitif terhadap errors pada individual examples
  - Robust to noisy data

# Inductive Bias in ID3

- Inductive bias: the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances.
- Training examples → n consistent decision trees
- Inductive bias ID3:
  - shorter trees are preferred over longer ones
    - Occam's razor: prefer the shortest hypothesis that fits the data
  - Select trees that place the attributes with highest information gain closest to the root/
- BFS-ID3 → ID3: greedy heuristic (information-gain +hill-climbing strategy)

# Isu dalam DTL

- Overfitting training data
- Continuous-valued attribute
- Alternative measures for selecting attributes
- Handling missing attribute value
- Handling attributes with differing costs

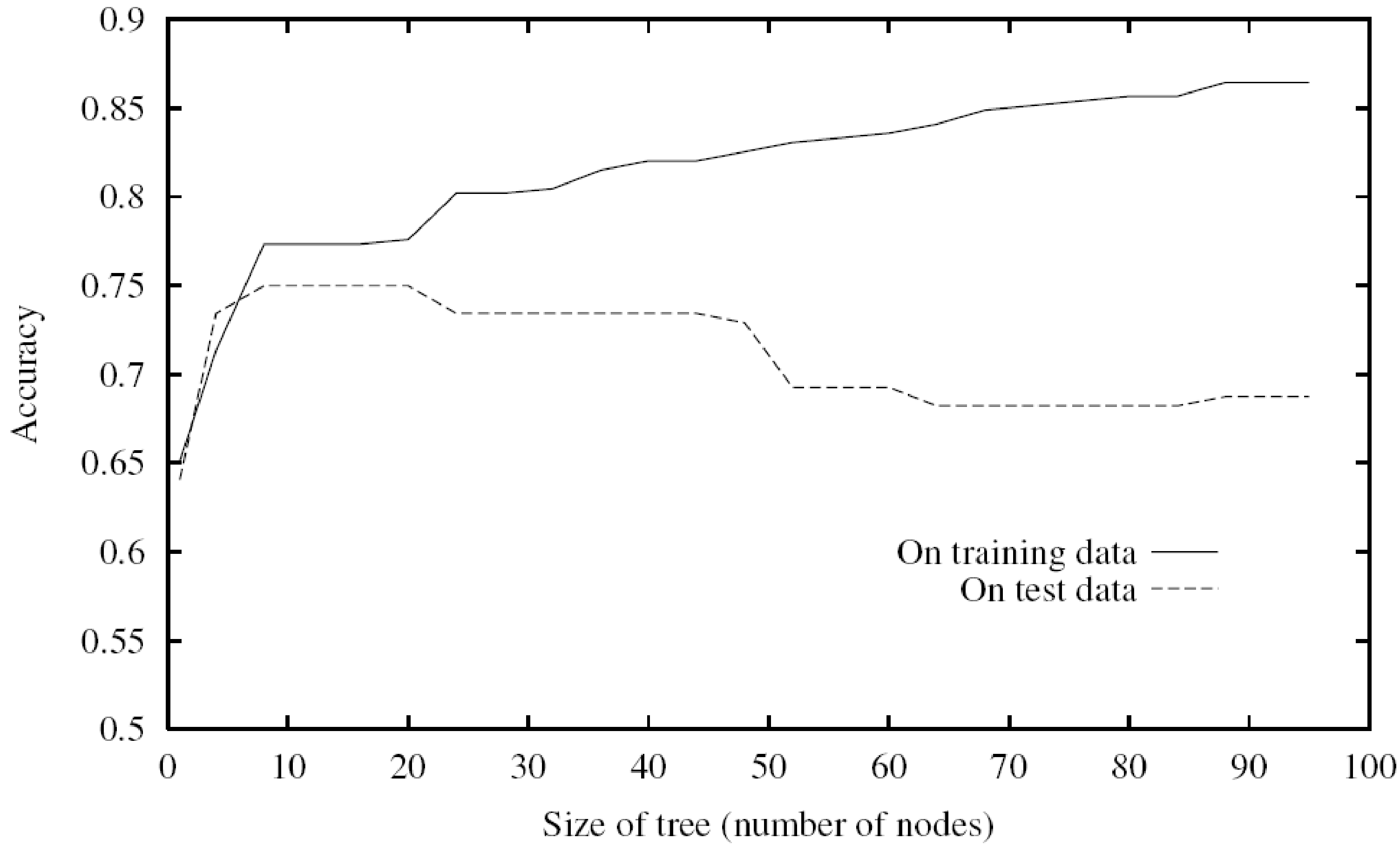
# Overfitting The Data

- Hipotesis overfit training data jika terdapat hipotesis lain yang kurang cocok dgn training data tetapi berkinerja lebih baik pada distribusi data secara keseluruhan
- Definisi formal:

  *Given a hypothesis space H, a hypothesis h∈H is said to overfit the training data if there exists some alternative hypothesis h'∈H, such that h has smaller error than h' over the training examples, but h' has smaller error than h over the entire distribution of instances.*

# Overfitting in Decision Tree Learning

Add new nodes to grow the decision tree, the accuracy increases monotonically.

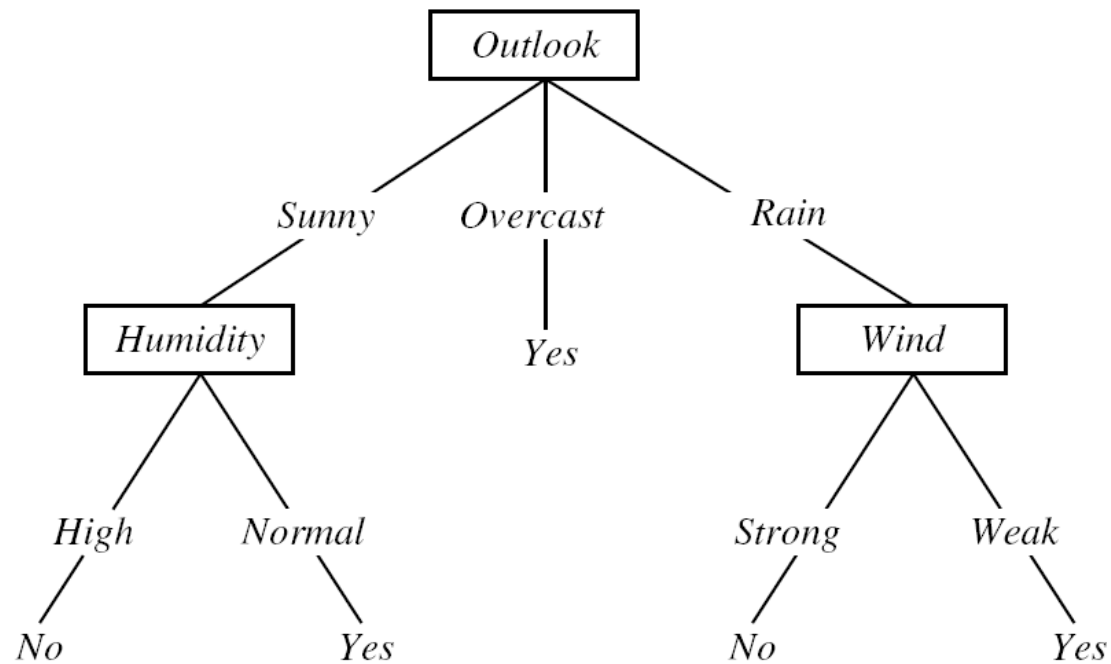


# Overfitting in Decision Trees

Consider adding noisy training example #15:

Sunny; Hot; Normal; Strong; PlayTennis = No

What effect on earlier tree?



# Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution D of data: $error_D(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

# Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Minimum Description Length (MDL):
  minimize size(tree) + size(misclassifications(tree))

# Reduced-Error Pruning

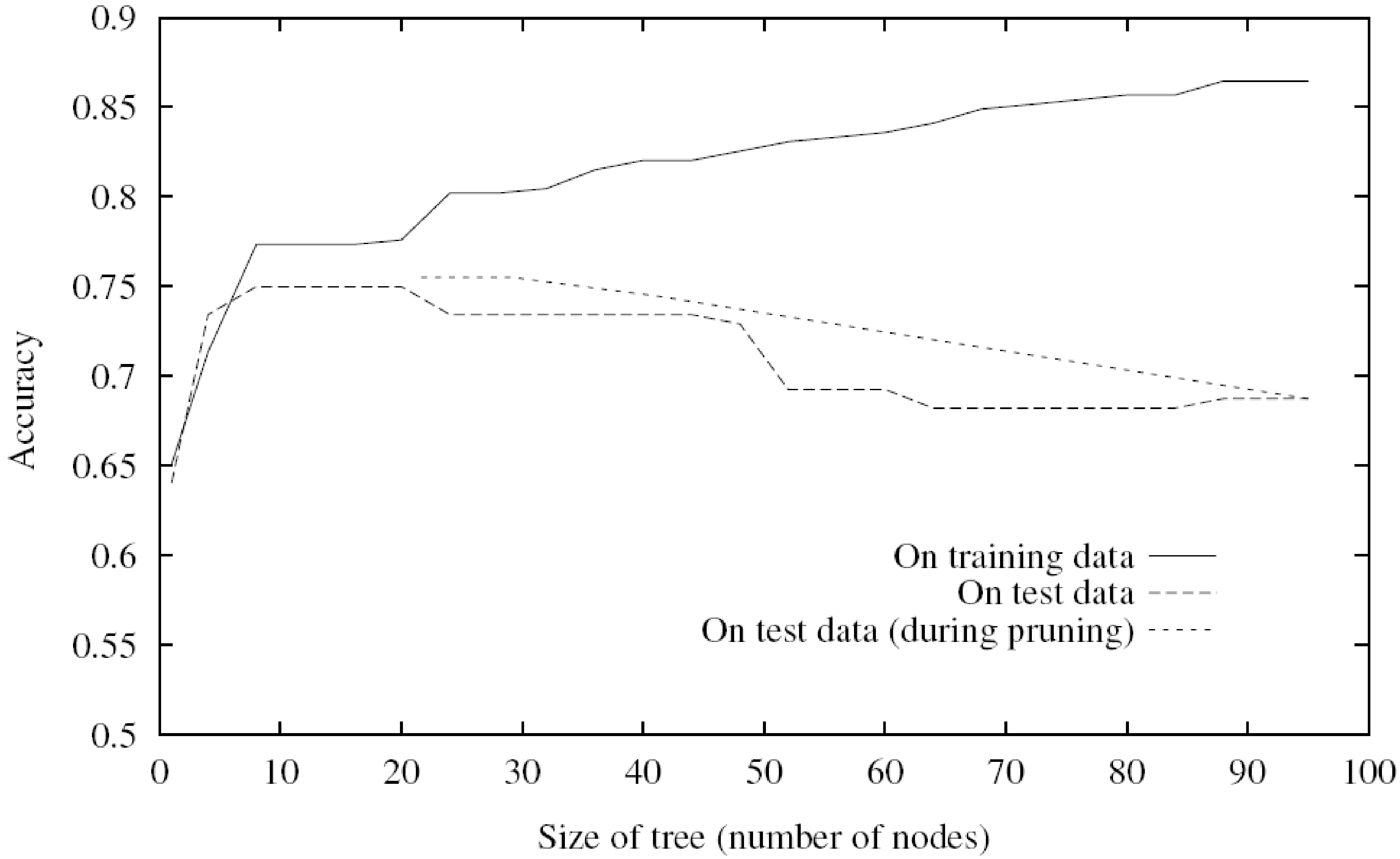
Split data into training and validation set

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

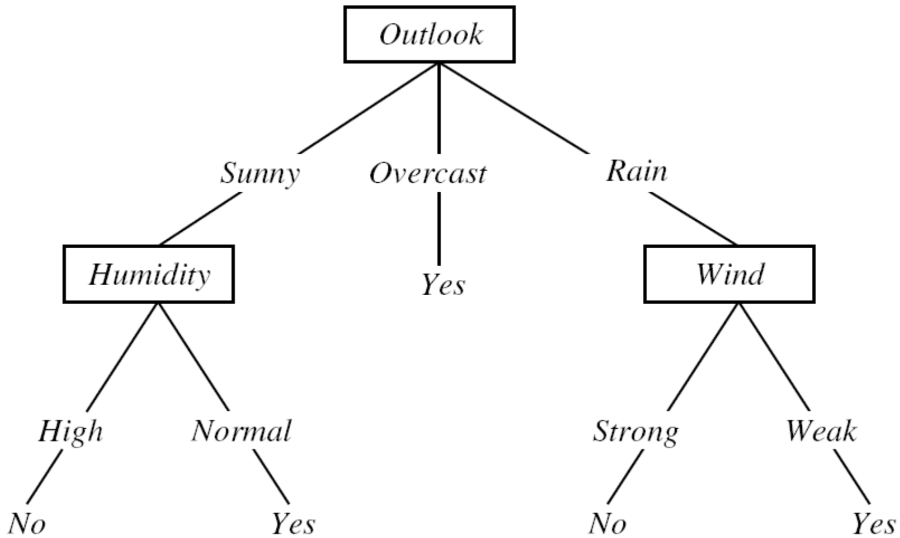
# Effect of Reduced-Error Pruning



# Rule Post-Pruning: C4.5

1. Build decision tree
2. Convert tree to equivalent set of rules
3. Prune (generalize) each rule independently of others
   - Removing any preconditions that result in improving its estimated accuracy
4. Sort final rules into desired sequence for use
   - Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances

# Decision Tree → Decision Rules




- Why ?
  - Distinct path ~ distinct rule: independent pruning
  - No distinction between attribute tests
  - Improves readability

IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

# Pruning

- Rule: IF (outlook=sunny)$\wedge$(humidity=high) THEN No
- Cek akurasi penghapusan (outlook=sunny) atau (humidity=high)
  - Gunakan validation set
  - C4.5: pessimistic estimate (hitung akurasi terhadap training data, hitung standar deviasi)

# Continuous Valued Attributes

- Continuous valued attributes → new discrete valued attribute $A_c$: $A<c$
- Best value for the threshold c ?
  - (48+60)/2 atau (80+90)/2
- Test information gain for each candidate attribute:
  - Best: temperature>54

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

| | | | | | | |
|---|---|---|---|---|---|---|
| *Temperature:* | 40 | 48 | 60 | 72 | 80 | 90 |
| *PlayTennis:* | No | No | Yes | Yes | Yes | No |

## Attributes with Many Values

Problem:

- If attribute has many values, Gain will select it
- Imagine using *Date* = Jun_3_1996 as attribute

One approach: use GainRatio instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where $S_i$ is subset of $S$ for which $A$ has value $v_i$

# Attributes with Costs

Consider

medical diagnosis, BloodTest has cost \$150

robotics, Width_from_1ft has cost 23 sec.

How to learn a consistent tree with low expected cost?

One approach: replace gain by

- Tan and Schlimmer (1990)

$$\frac{Gain^2(S,A)}{Cost(A)}.$$

- Nunez (1988)

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A)+1)^w}$$

where $w \in [0,1]$ determines importance of cost

# Unknown Attribute Values

What if some examples missing values of A?

Use training example anyway, sort through tree

- If node n tests A, assign most common value of A among other examples sorted to node n
- assign most common value of A among other examples with same target value
- assign probability $p_i$ to each possible value $v_i$ of A
  - assign fraction $p_i$ of example to each descendant in tree

Classify new examples in same fashion

# PR 1: maks 2 halaman

- Pelajari source code WEKA untuk ID3 dan C4.5 (J48). Gunakanlah algoritma pada hal 56 Tom Mitchell. Untuk setiap tahapan, carilah persamaan dan perbedaan kedua algoritma berdasarkan source code tersebut dan jelaskanlah perbedaan tersebut.
  - Penentuan atribut terbaik
  - Penanganan label dari cabang setiap nilai atribut
    - Bagaimana jika Examples kosong di cabang tersebut
    - Bagaimana menangani atribut kontinu
  - Penanganan atribut dengan missing values
  - Pruning dan parameter confidence pada J48
- Pengumpulan: Selasa 8 September 2015

# Tugas 1 IF4071: Create classifier baru

Bagian 1: menuliskan kode java untuk mengakses weka,

- mulai dari load data (arrf dan csv)
- remove atribut
- Filter : Resample
- build classifier : NaiveBayes, DT
- testing model given test set,
- 10-fold cross validation, percentage split,
- Save/Load Model,
- using model to classify one unseen data (input data)

- Bagian 2: membuat Classifier baru dengan menurunkan dari Classifier WEKA
  - Implementasi kelas baru pada weka: myID3, myC45
  - Penanganan binary class dan multi class
  - Penanganan atribut diskrit dan kontinu

Test data yang digunakan :

- data binary categorization weather (nominal, kontinu)
- Data multiclass categorization iris

LAPORAN !!! -> Source Code, Hasil Eksekusi terhadap data tes, perbandingan dengan hasil ID3 &J48 weka (pdf)
Rabu 23 September 2015

# THANK YOU