Machine Intelligence

Lecture 7: Learning - Introduction and decision trees

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- **Machine learning**
- Planning
- Reinforcement learning
- Multi-agent systems

Supervised Learning

So far

The general pattern so far:

Problem/Domain description	State Space Problem	Variables, Constraints	Probabilistic Model
Inference Algorithms	Search (A^*,...)	Arc Consistency, Variable Elimination	Variable Elimination
Solutions	Goal states, plans, diagnoses, predictions,...		

- Problem/Domain description and algorithms designed by human
- Agent will always act the same in the same situation

So far

The general pattern so far:

Problem/Domain description	State Space Problem	Variables, Constraints	Probabilistic Model
Inference Algorithms	Search (A^*,...)	Arc Consistency, Variable Elimination	Variable Elimination
Solutions	Goal states, plans, diagnoses, predictions,...		

- Problem/Domain description and algorithms designed by human
- Agent will always act the same in the same situation

Objective of (machine) **learning**:

- Agent can *learn by experience*: improve performance over time
- Agent (program) can be automatically constructed from *examples* (rather than designed by expert)

Tasks and Model Types

The models constructed by machine learning algorithms are used for several kinds of tasks:

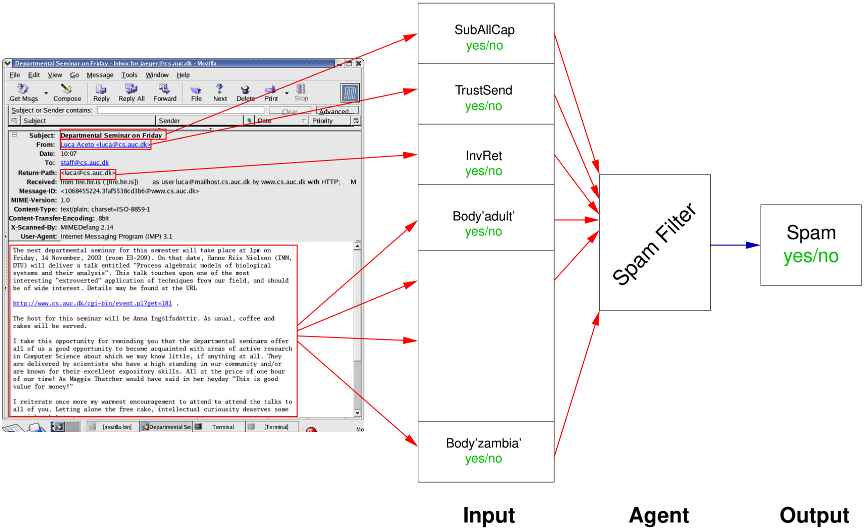
Predictive tasks/models

- Task: predict some (unobserved) target or class variable based on observed values of (predictor) attributes
  - Regression, if target is continuous
  - Classification, if target is discrete
- Examples: Spam filtering, Character recognition, ...
- Model types e.g.: Decision trees, k nearest neighbors, Neural networks, Naive Bayes,...

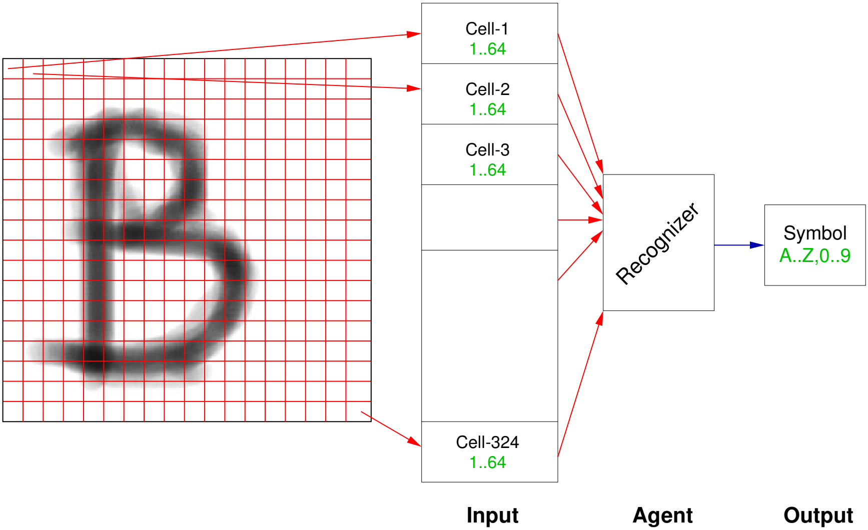
Descriptive tasks/models

- Task: Clustering: identify coherent subgroups in data
- Examples: Recommender systems,
- Model types e.g.: k-means, hierarchical clustering, Self-organizing maps, probabilistic clustering,...

Example: Spam filter



Example: Character Recognition



Experience, Data

In order to show a certain behavior, or perform certain tasks, an agent needs to

- make (the right) decisions based on possible observations

Experience from which the right behavior can be learned consists of

- **Examples** or **Cases** of inputs that are **labeled** with the correct decisions (outputs).

This is also called **Labeled data**.

Data

We assume that data (experience) consists of an attribute-value table:

	Input Features (Attributes, Predictor Variables)				**Target Feature** (Class variable)
SubAllCap	TrustSend	InvRet	...	B'zambia'	Spam
y	n	n	...	n	y
n	n	n	...	n	n
n	y	n	...	n	y
n	n	n	...	n	n
...	...	...	...	...	...

- Columns correspond to attributes given by a name and a state space (attributes are basically the same as (chance) variables).
- Rows correspond to examples (also called cases or instances): observations of joint occurrences of values of the attributes.
- In prediction problems, there is a distinguished target attribute. When the target attribute is discrete, it is usually called the class variable. The attributes used for prediction are then called predictor attributes.

In table above: *Spam* is class variable for the prediction problem: predict whether a mail is spam, given characteristics of the mail.

Continuous Attributes

current temp	*pressure change (24h)*	*rain tomorrow*	*temp tomorrow*
16.8	-8.5	yes	15.3
21.7	2.1	no	22.5
19.5	-1.4	no	20.4
8.4	0.5	yes	7.2
...	...	...	...

- Classification problem: predict whether it rains tomorrow, given current temperature and pressure change (*rain tomorrow* is class variable).
- Regression problem: predict temperature tomorrow, given current temperature and pressure change (*temperature tomorrow* is target attribute).
- Clustering problem: identify groups of observations representing similar weather patterns (e.g. stable winter high pressure situations).

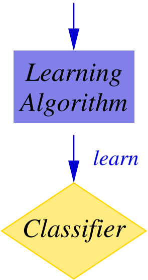
$\rightsquigarrow$ one data set can be the basis for many different learning tasks!

Classification, Regression

- Predicting a discrete target feature: **classification**
- Agent/Model/Program that classifies: **classifier**
- Predicting a continuous (numeric) target feature: **regression**
- Agent/Model/Program that performs regression: **regression model**

Learning and Using a Classifier

A_1	A_2	A_3	C
0	H	0.32	*good*
1	H	0.72	*good*
0	L	0.06	*bad*
1	M	1.21	*good*
...	...	...	...



A_1	A_2	A_3	C
0	H	0.32	*good*
1	H	0.72	*good*
0	L	0.06	*bad*
1	M	1.21	*good*
...	...	...	...

Learning Algorithm

learn

A_1	A_2	A_3	C
0	M	0.45	?
0	H	1.64	?
1	H	0.16	?
...	...	...	...

Classifier

predict

A_1	A_2	A_3	C
0	M	0.45	*bad*
0	H	1.64	*good*
1	H	0.16	*good*
...	...	...	...

Ingredients

Key ingredients of a learning method are:

- a **Hypothesis Space**: set of all possible classifiers that could be learned based on a given **Representation**
- an **Evaluation Measure** that is used to decide how good a candidate hypothesis is
- a **Search** or **Optimization** method used to find a hypothesis that scores high according to the evaluation measure.

Decision Trees

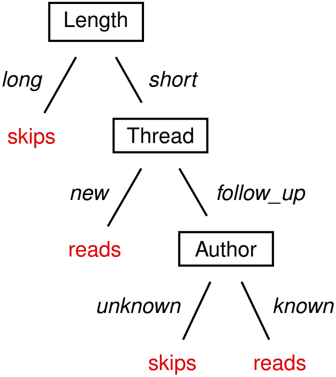
Example: Training Data

User preference data:

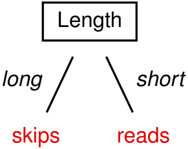
Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads
e_{19}	unknown	new	long	work	?
e_{20}	unknown	follow Up	long	home	?
e_{21}	unknown	follow Up	short	home	?

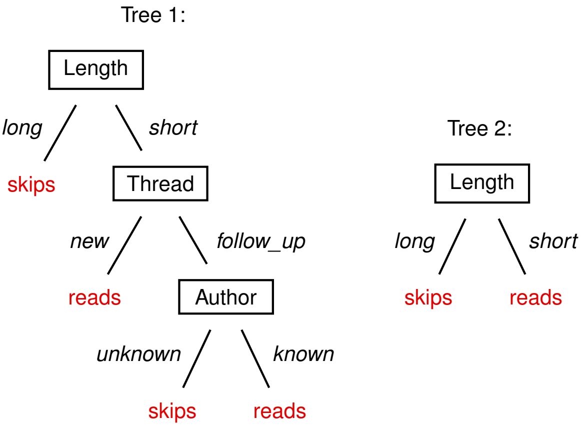
Example: Decision Trees

Tree 1:



Tree 2:





Tree 1 is equivalent to the following logic program:

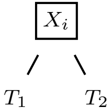
$$\begin{aligned} \mathit{skips} &\leftarrow \mathit{long} \\ \mathit{reads} &\leftarrow \mathit{short} \wedge \mathit{new} \\ \mathit{reads} &\leftarrow \mathit{short} \wedge \mathit{follow\_up} \wedge \mathit{known} \\ \mathit{skips} &\leftarrow \mathit{short} \wedge \mathit{follow\_up} \wedge \mathit{unknown} \end{aligned}$$

Example: Classifications

Example	Author	Thread	Length	WhereRead	UserAction	Tree 1	Tree 2
e_1	known	new	long	home	skips	skips	skips
e_2	unknown	new	short	work	reads	reads	reads
e_3	unknown	follow Up	long	work	skips	skips	skips
e_4	known	follow Up	long	home	skips	skips	skips
e_5	known	new	short	home	reads	reads	reads
e_6	known	follow Up	long	work	skips	skips	skips
e_7	unknown	follow Up	short	work	skips	skips	reads
e_8	unknown	new	short	work	reads	reads	reads
e_9	known	follow Up	long	home	skips	skips	skips
e_{10}	known	new	long	work	skips	skips	skips
e_{11}	unknown	follow Up	short	home	skips	skips	reads
e_{12}	known	new	long	work	skips	skips	skips
e_{13}	known	follow Up	short	home	reads	reads	reads
e_{14}	known	new	short	work	reads	reads	reads
e_{15}	known	new	short	home	reads	reads	reads
e_{16}	known	follow Up	short	work	reads	reads	reads
e_{17}	known	new	short	home	reads	reads	reads
e_{18}	unknown	new	short	work	reads	reads	reads
e_{19}	unknown	new	long	work	?	skips	skips
e_{20}	unknown	follow Up	long	home	?	skips	skips
e_{21}	unknown	follow Up	short	home	?	skips	reads

Top-down construction:

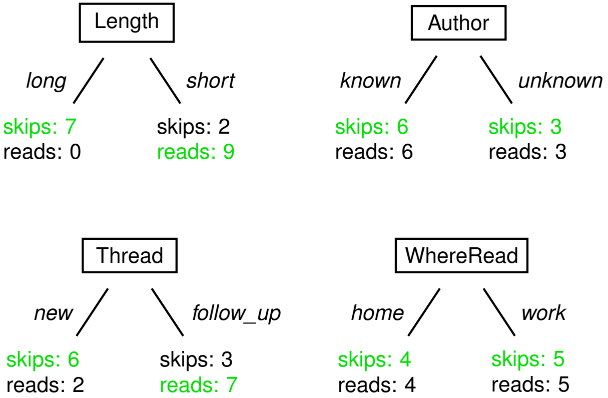
procedure *DecisionTreeLearner*($\mathbf{X}, Y, E$)
    // $\mathbf{X} = \{X_1, \ldots, X_n\}$: input features
    // Y: target feature
    // E: set of training examples
1. **if** stopping criterion is true
2.   **return** *leaf node* labeled with most frequent target feature value in E
3. **else**
4.   select feature $X_i \in \mathbf{X}$
    // let v_1, v_2 be the possible values of X_i
5. $E_1 = \{e \in E :$ *val*$(e, X_i) = v_1\}$
6. $T_1 =$ *DecisionTreeLearner*$(\mathbf{X}, Y, E_1)$
7. $E_2 = \{e \in E :$ *val*$(e, X_i) = v_2\}$
8. $T_2 =$ *DecisionTreeLearner*$(\mathbf{X}, Y, E_2)$
9.   **return**



Choosing X_i

Key question: which X_i to choose in line 4.?

Approach: choose the feature that would provide the best classifier if construction would terminate with that feature.



Showing

- Number of examples with class labels skips,reads belonging to different sub-trees
- Green: predicted class label (possibly a tie between two labels)

Class Purity

Principle: Prefer features that split the examples into *class pure* subsets:

pure:	*nearly pure*:	*impure*:
skips: 7, reads: 0	skips: 6, reads: 1	skips: 6, reads: 5
skips: 0, reads: 5	skips: 2, reads: 15	skips: 7, reads: 7
skips: 11, reads: 0	skips: 11, reads: 1	skips: 13, reads: 12

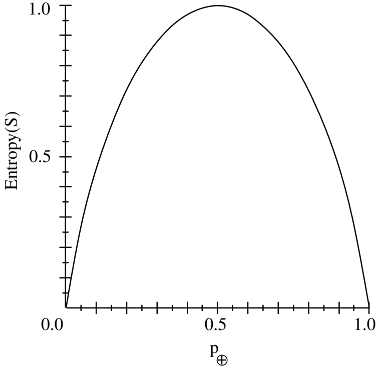
Normalized to probabilities:

pure:	*nearly pure*:	*impure*:
skips: 1, reads: 0	skips: 0.85, reads: 0.15	skips: 0.54, reads: 0.46
skips: 0, reads: 1	skips: 0.12, reads: 0.88	skips: 0.5, reads: 0.5
skips: 1, reads: 0	skips: 0.91, reads: 0.09	skips: 0.52, reads: 0.48

Purity Measure

For a probability distribution $(p, 1-p)$ of a two-valued class label, define

$$h(p, 1-p) = -p \cdot \mathit{log}(p) - (1-p) \cdot \mathit{log}(1-p)$$



- High values for impure distributions
- Maximal for $(0.5, 0.5)$
- Zero for $(1, 0)$ and $(0, 1)$

Examples:

- Entropy$(0.5, 0.5) = -0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5) = 0.5 \cdot 1 + 0.5 \cdot 1 = 1$
- Entropy$(0.35, 0.65) = -0.35 \cdot \log_2(0.35) - 0.65 \cdot \log_2(0.65) = 0.93$
- Entropy$(0, 1) = -0 \cdot \log_2(0) - 1 \cdot \log_2(1) = -0 - 0 = 0$
- Entropy$(1, 0) = -1 \cdot \log_2(1) - 0 \cdot \log_2(0) = -0 - 0 = 0$

Generalization to larger domains

For a probability distribution on domain with n elements:

$$\mathbf{p} = (p_1, p_2, \ldots, p_n) \qquad (p_n = 1 - \sum_{i=1}^{n-1} p_i)$$

define entropy:

$$h(\mathbf{p}) = -\sum_{i=1}^{n} p_i \cdot \mathit{log}(p_i)$$

Again:

- Maximal for $\mathbf{p} = (1/n, \ldots, 1/n)$
- Zero for $\mathbf{p} = (1, 0, \ldots, 0)$, $\mathbf{p} = (0, 1, 0, \ldots, 0)$,..., $\mathbf{p} = (0, \ldots, 0, 1)$

Expected Entropy Example

We prefer features that split into subsets with low entropy, but consider example for binary class variable (values c_1, c_2 with initial counts $c_1 : 20, c_2 : 20$), and two 3-valued features X_1, X_2:

$\boxed{X_1}$

$c_1 : 0, c_2 : 1$	$c_1 : 1, c_2 : 0$	$c_1 : 19, c_2 : 19$	counts
$(0, 1)$	$(1, 0)$	$(0.5, 0.5)$	normalized
0	0	1	entropy

$\boxed{X_2}$

$c_1 : 0, c_2 : 15$	$c_1 : 15, c_2 : 0$	$c_1 : 5, c_2 : 5$	counts
$(0, 1)$	$(1, 0)$	$(0.5, 0.5)$	normalized
0	0	1	entropy

Expected Entropy Example

We prefer features that split into subsets with low entropy, but consider example for binary class variable (values c_1, c_2 with initial counts $c_1 : 20, c_2 : 20$), and two 3-valued features X_1, X_2:

X_1

$c_1 : 0, c_2 : 1$	$c_1 : 1, c_2 : 0$	$c_1 : 19, c_2 : 19$	counts
$(0, 1)$	$(1, 0)$	$(0.5, 0.5)$	normalized
0	0	1	entropy

X_2

$c_1 : 0, c_2 : 15$	$c_1 : 15, c_2 : 0$	$c_1 : 5, c_2 : 5$	counts
$(0, 1)$	$(1, 0)$	$(0.5, 0.5)$	normalized
0	0	1	entropy

X_2 provides a better division of examples than X_1. It gives a lower *expected entropy*:

$$(1/40) \cdot 0 + (1/40) \cdot 0 + (38/40) \cdot 1 > (15/40) \cdot 0 + (15/40) \cdot 0 + (10/40) \cdot 1$$

Expected Entropy and Information Gain

For feature X with domain $v_1, \ldots, v_n$, let:

- E_i be the set of examples with $X = v_i$
- $q_i = |E_i|/|E|$
- h_i the entropy of the class label distribution in E_i

The **expected entropy** from splitting on X then is:

$$h(\textit{Class} \mid X) = \sum_{i=1}^{n} q_i \cdot h_i$$

Let

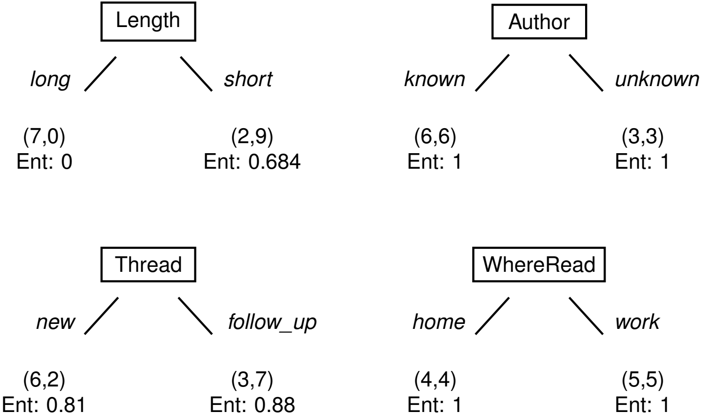
- $h(\textit{Class})$: entropy of class label distribution before splitting

The **Information Gain** from splitting on X then is:

$$h(\textit{Class}) - h(\textit{Class} \mid X)$$

Information Gain in Decision Tree Learning

- In line 4. of algorithm choose feature X_i that gives highest information gain.


Length
long
short
(7,0)
Ent: 0
(2,9)
Ent: 0.684
Author
known
unknown
(6,6)
Ent: 1
(3,3)
Ent: 1
Thread
new
follow_up
(6,2)
Ent: 0.81
(3,7)
Ent: 0.88
WhereRead
home
work
(4,4)
Ent: 1
(5,5)
Ent: 1

Information gain

Length

long / \ *short*

(7,0)
Ent: 0

(2,9)
Ent: 0.684

Author

known / \ *unknown*

(6,6)
Ent: 1

(3,3)
Ent: 1

E-E: $\frac{7}{18} \cdot 0 + \frac{11}{18} \cdot 0.684 = 0.582$

Thread

new / \ *follow_up*

(6,2)
Ent: 0.81

(3,7)
Ent: 0.88

WhereRead

home / \ *work*

(4,4)
Ent: 1

(5,5)
Ent: 1

Information gain

Length

long / \ *short*

(7,0)
Ent: 0

(2,9)
Ent: 0.684

E-E: $\frac{7}{18} \cdot 0 + \frac{11}{18} \cdot 0.684 = 0.582$

Author

known / \ *unknown*

(6,6)
Ent: 1

(3,3)
Ent: 1

E-E: $\frac{12}{18} \cdot 1 + \frac{6}{18} \cdot 1 = 1$

Thread

new / \ *follow_up*

(6,2)
Ent: 0.81

(3,7)
Ent: 0.88

WhereRead

home / \ *work*

(4,4)
Ent: 1

(5,5)
Ent: 1

Length

long — (7,0) Ent: 0

short — (2,9) Ent: 0.684

E-E: $\frac{7}{18} \cdot 0 + \frac{11}{18} \cdot 0.684 = 0.582$

Author

known — (6,6) Ent: 1

unknown — (3,3) Ent: 1

E-E: $\frac{12}{18} \cdot 1 + \frac{6}{18} \cdot 1 = 1$

Thread

new — (6,2) Ent: 0.81

follow_up — (3,7) Ent: 0.88

E-E: $\frac{8}{18} \cdot 0.81 + \frac{10}{18} \cdot 0.88 = 0.85$

WhereRead

home — (4,4) Ent: 1

work — (5,5) Ent: 1

Information gain

Length

long | *short*

(7,0) Ent: 0 | (2,9) Ent: 0.684

E-E: $\frac{7}{18} \cdot 0 + \frac{11}{18} \cdot 0.684 = 0.582$

Author

known | *unknown*

(6,6) Ent: 1 | (3,3) Ent: 1

E-E: $\frac{12}{18} \cdot 1 + \frac{6}{18} \cdot 1 = 1$

Thread

new | *follow_up*

(6,2) Ent: 0.81 | (3,7) Ent: 0.88

E-E: $\frac{8}{18} \cdot 0.81 + \frac{10}{18} \cdot 0.88 = 0.85$

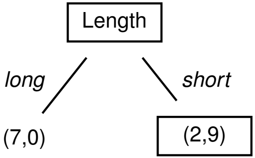
WhereRead

home | *work*

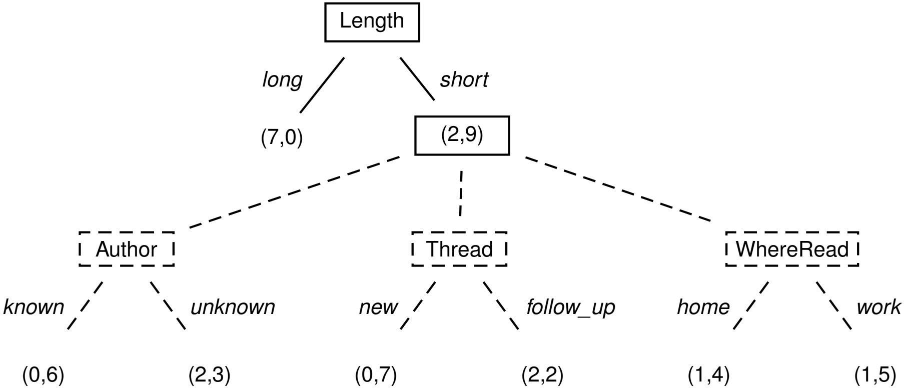
(4,4) Ent: 1 | (5,5) Ent: 1

E-E: $\frac{8}{18} \cdot 1 + \frac{10}{18} \cdot 1 = 1$

Constructing the decision tree



Constructing the decision tree



Continuous/many-valued attributes I

The information gain measure favors attributes with many values:

- For example, the attribute Date (with the possible dates as states) will have a very high information gain but is unable to generalize!

One approach for avoiding this problem is to select attributes based on GainRation:

$$\text{GainRation}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

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

where S_i is the subset of examples produced by splitting on the i'th value of A.

Note that SplitInformation is the entropy of S w.r.t. the values of A.

Continuous/many-valued attributes I

We require that the attributes being tested are discrete valued. So in order to test a continuous valued attribute we need to “discretize” it.

Suppose that the training examples are associated with the attribute Temperature:

Temperature:	40	48	60	72	80	90
Rain tomorrow:	*yes*	*yes*	*no*	*no*	*no*	*yes*

Continuous/many-valued attributes I

We require that the attributes being tested are discrete valued. So in order to test a continuous valued attribute we need to “discretize” it.

Suppose that the training examples are associated with the attribute Temperature:

Temperature:	40	48	60	72	80	90
Rain tomorrow:	*yes*	*yes*	*no*	*no*	*no*	*yes*

Create a new boolean valued attribute by first testing the two candidate thresholds:

- (48+60)/2
- (80+90)/2

Next, pick the one with highest information gain (i.e., Temperature$_{>54}$)

Overfitting

Noise in data may lead to a bad classifier. In particularly, if the decision tree fits the data perfectly. This is called overfitting.

Definition

A hypothesis h is said to overfit the training data if there exists some alternative hypotheses h', such that:

- h has smaller error than h' over the training data, but
- h' has a smaller error than h over the entire distribution of instances.