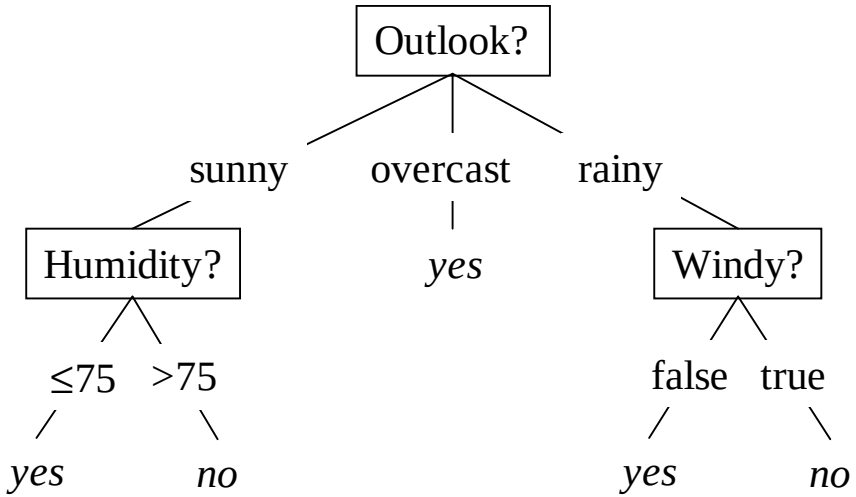
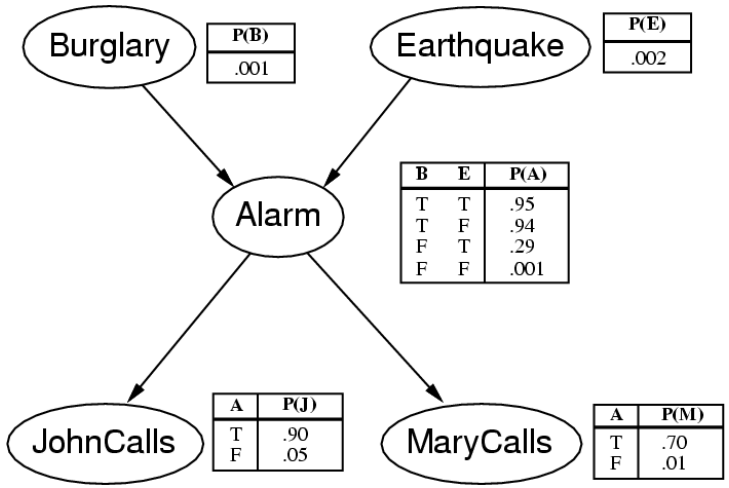
Decision Tree Learning (2) & Bayesian Methods





Univ.-Lektor Dr.techn. Alexander K. Seewald
Österreichisches Forschungsinstitut
für Artificial Intelligence

alex@seewald.at / alex.seewald.at

Common Problem with DTs: Overfitting

Outlook	Temp.	Humidity	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play
sunny	***cool***	***normal***	***true***	***Don't Play***

Additional training example with incorrect class

```
outlook = sunny
|   humidity = high: no (3)
|   humidity = normal: yes (2)
outlook = overcast: yes (4)
outlook = rainy
|   windy = TRUE: no (2)
|   windy = FALSE: yes (3)
```

```
outlook = sunny
|   humidity = high: no (3)
|   humidity = normal
|   |   temperature = hot: yes (0)
|   |   temperature = mild: yes (1)
|   |   temperature = cool
|   |   |   windy = TRUE: no (1)
|   |   |   windy = FALSE: yes (1)
outlook = overcast: yes (4)
outlook = rainy
|   windy = TRUE: no (2)
|   windy = FALSE: yes (3)
```


alex@seewald.at / alex.seewald.at

How to avoid Overfitting for DTs

Pre-Pruning

Stop splitting a node further (even if it still contains examples of different classes, and even if some attributes are still available) if there seems to be no statistically significant correlation between attributes and classes

Post-Pruning

First construct (possibly complex) tree that is maximally consistent with the training data (i.e., has minimum error on training data)

Then simplify the tree by cutting off branches and subtrees that seem harmful.

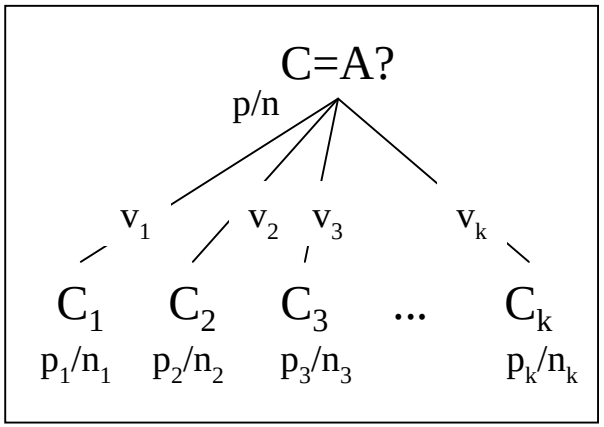
Effects of Pruning

- Simpler trees with lower accuracy on the training data but possibly higher accuracy on new, unseen data.
- Improves handling of attribute and class noise

alex@seewald.at / alex.seewald.at

Pre-Pruning

Pre-Pruning: X^2 Test (pronounced: Chi-Square)



If A is **completely irrelevant** to the class of an object in C, the expected value of p_i is $p_i' = p * |C_i|/|C|$ and the expected value of n_i is $n_i' = n * |C_i|/|C|$, where p and n are the number of positive resp. negative examples of the class.

⇒ The larger the differences $|p_i - p_i'|$ and $|n_i - n_i'|$, the smaller the likelihood that A is completely irrelevant.

⇒ Statistic $S = \sum_{i=1}^{k} \frac{(p_i - p_i')^2}{p_i'} + \frac{(n_i - n_i')^2}{n_i'}$ is approximately X^2 distributed with *k-1* d.o.f.

⇒ Perform X^2 test: S large enough? (Intuition: the smaller S, the higher the probability that A is irrelevant to the class (i.e., class is independent of A))

⇒ Prune (stop refining a node) if there is no relevant A at given confidence level

alex@seewald.at / alex.seewald.at

Post-Pruning

Reduced Error Pruning (Pseudocode)

1. Randomly split training examples TD into a training set TS (usually 70%) and a pruning (validation) set PS (usually 30%)
2. Learn a (possibly complex) tree from TS that is as consistent with the data as possible (i.e., that possibly overfits the data)
3. Perform tree simplification step:

   For each subtree T_i of T, tentatively replace T_i by a majority class leaf
4. Compare the accuracy on PS(!) for all modified subtrees with accuracy of original T on PS:

- If there is no T_i that improves accuracy on PS when removed: exit
- Otherwise: remove (and replace with leaf) T_i with maximum improvement.

5. Go to 3.

Question: Why additional pruning set PS? Why not use original training set TD for making pruning decisions?

alex@seewald.at / alex.seewald.at

Post-Pruning without Validation Set: PEP

Pessimistic Error Pruning (PEP)

(used in well-known C4.5 decision tree learner)

Replace subtree T_i by a majority class leaf, if and only if

$$E+0.5 < \sum J + L(T_i)/2 + SE$$

where...

- $\sum J$ number of training set errors for subtree T_i before replacing
- E number of training set errors when replacing T_i by a leaf (only for those examples which are within the subtree T_i)
- $L(T_i)$ total number of leaves in subtree T_i
- SE standard error: $\sqrt{\frac{(\sum J+L(T_i)/2)(\sum K-(\sum J+L(T_i)/2))}{\sum K}}$
- $\sum K$ number of examples in subtree T_i ($=p_i+n_i=|C_i|$)

\+ *no need for validation set - all training data can be used; very efficient*

– *heuristic is ad-hoc and not reasonably grounded in statistical theory*

alex@seewald.at / alex.seewald.at

Def.: Missing values

Missing values occur when values of some attributes are unknown for some examples. This can have a variety of reasons:

- *not applicable* - Attribute is not valid for current example; e.g. gender for legal persons (companies), results of medical examinations which were not executed
- *not valid* - Attribute value is invalid, e.g. due to data entry errors, errors in data conversion during preprocessing, or measurement errors. For example, the attribute value of -12 for humidity (in percent).
- *truly missing* - Attribute is valid but has not been measured; e.g. due to measurement device error, people choosing not to answer some questions in a questionnaire (income) etc..

...

- *unknown* - It is unknown why the value has been marked as missing, usually due to insufficient documentation of the data cleaning process.

*Missing values are usually encoded by **?***

alex@seewald.at / alex.seewald.at

How to treat MV?

Common solutions

- Ignore examples that are incompletely described (i.e. have at least one MV)
  $\Rightarrow$ waste of valuable data
- Introduce *missing* as a special value and treat it like the other attribute values
  $\Rightarrow$ useful for *not applicable* MVs, but can lead to paradoxical situations
- For Decision Trees, split incomplete example with MVs into *virtual* examples:
  - Compute probability (rel.frequency) pr_i for each value v_i of attribute A.
  - Assign fractional example (weight=pr_i) to each corresponding branch
  - Classification of new, incomplete examples works analogously: when encountering a split on an attribute that is *missing*, propagate example through all subtree and returned pr_i-weighted sum of classes.
- For Bayesian methods, MVs are modelled as an uniform probability distribution over all possible values of a missing attribute.

- Complete incomplete examples by replacing ? with a *default* value (e.g. the most common value of this attribute in the training data, or mean/mode of numerical attributes). **Simplest approach, which explains its widespread use**

alex@seewald.at / alex.seewald.at

Bayesian Methods & Bayes Theorem

Bayesian Methods provide the basis for probabilistic reasoning:

- Theoretical framework for machine learning, classification, knowledge representation and analysis.
- Allows to integrate uncertain and partial domain knowledge
- Direct modeling of uncertainty
- Easily handles noisy and incomplete data sets with MVs

One cornerstone of Bayesian Methods is Bayes' Rule:

$$P(f \mid TD) = \frac{P(TD \mid f)P(f)}{P(TD)}$$

The probability of function/model f given training data TD is equal to the probability of TD given f multiplied by the (prior) probability of h divided by the (prior) probability of TD. **All classification methods can be seen as estimating Bayes' Rule, with different techniques to estimate P(TD|f).**

alex@seewald.at / alex.seewald.at

Maximum Likelihood

Among all f from Concept Space CS, choose f which has highest probability given training data TD. This is the **maximum a posteriori (MAP)** model:

$$f_{BEST} = \arg\max_{f \in CS} P(f \mid TD) = \arg\max_{f \in CS} \frac{P(TD \mid f)P(f)}{P(TD)} = \arg\max_{f \in CS} P(TD \mid f)P(f)$$

In some cases, we can assume every f ∈ CS to be equally probable. In that case, the above simplifies to the **maximum likelihood (ML)** model:

$$f_{ML} = \arg\max_{f \in CS} P(TD \mid f)$$

Example: Cancer diagnosis

Two hypotheses: cancer, ¬cancer (|CS|=2)

Diagnostic test for cancer with two outcomes: ⊕ = + , ∅ = –

Known prob.:	P(cancer)	=	0.008	P(¬cancer)	=	0.992
(*sensitivity)*	P(⊕ | cancer)	=	0.98	P(∅ | cancer)	=	0.02
	P(⊕ | ¬cancer)	=	0.03	P(∅ | ¬cancer)	=	0.97 (*specificity)*

P(cancer | ⊕) = P(⊕ | cancer)P(cancer) = 0.98 * 0.008 = 0.0078 (21%)

P(¬cancer | ⊕) = P(⊕ | ¬cancer)P(¬cancer) = 0.03 * 0.992 = 0.0298 (**79%,MAP**)

alex@seewald.at / alex.seewald.at

Basic Probability Formulas

Product rule : probability of a conjunction of events A and B (= A AND B)

$$P(A \wedge B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

Sum rule : probability of a disjunction of two events A and B (= A OR B)

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Bayes Theorem : relating posterior and prior probabilities

$$P(f \mid TD) = \frac{P(TD \mid f)P(f)}{P(TD)}$$

Theorem of total probability : if events $A_1, \ldots, A_n$ are mutually exclusive

$$\text{with } \sum_{i=1}^{n} P(A_i) = 1 \Rightarrow P(B) = \sum_{i=1}^{n} P(B \mid A_i)P(A_i)$$

alex@seewald.at / alex.seewald.at

Bayes Optimal Classifier

So far, we have searched for the best f ∈ CS. However, it is possible to do better by combining all functions/models in CS, weighted by posterior probabilities:

$$Bayes_{optimal} = \arg\max_{Cl_j \in Class} \sum_{f_i \in CS} P(Cl_j \mid f_i) P(f_i \mid TD)$$

where *Class* is the set of possible classes, TD = training data, CS = concept space.

This is the **Bayes Optimal Classifier**.

Advantages

- Optimality: No other classification method with same CS and same prior knowledge can outperform this method (on average). This method maximizes the probability that the new instance is classified correctly, given the available data TD, concept space CS and posterior prob. over all the hypotheses f.
- Predictions correspond to a function/model not in CS – more general than CS!

Disadvantages

- Needs to sum over all possible functions f. Very costly and often intractable.
- Probabilities are usually unknown, and some of them are very hard to estimate.

alex@seewald.at / alex.seewald.at

Gibbs Algorithm & Naïve Bayes

Gibbs Algorithm is a more efficient but less optimal classifier. Under certain conditions it has at most twice the error rate of the optimal Bayes classifier. However, it is still very inefficient.

Gibbs Algorithm

1. Choose a function f from CS at random, according to posterior probability distribution over CS (i.e. P(f |TD))
2. Use f to predict the classification of next instance **x**.

A very efficient classifier is obtained by assuming the attributes to be conditionally independent ($P(A_i|A_j)=P(A_i)$ for $\forall$i,j). This is **Naïve Bayes**:

$$Bayes_{naïve} = \arg\max_{Cl_j \in Class} P(Cl_j) \prod_{\forall i} P(a_i \mid Cl_j)$$

$P(Cl_j)$ and $P(a_i|Cl_j)$ can be efficiently estimated from training data TD by counting. This is a commonly used classifier in machine learning, and works reasonably well even when the conditional independence assumption is violated.

alex@seewald.at / alex.seewald.at

Example: Weather dataset

Classify weather dataset with Naïve Bayes

- Estimate $P(Cl_j)$: P(*yes*)=9/14, P(*no*)=5/14
- Estimate $P(a_i|Cl_j)$, i.e. probability of attribute *i* having value a_i, given class of Cl_j:

	Play?	
Outlook	*yes*	*no*
overcast	4	0
rainy	3	2
sunny	2	3

	Play?	
Windy	*yes*	*no*
true	3	3
false	6	2

P(outlook=overcast | *yes*)=4/9 P(w.=t | *yes*)=3/9

P(outlook=rainy | *yes*)= 3/9 P(w.=f | *yes*)=6/9

P(outlook=sunny | *yes*)= 2/9

P(outlook=overcast| *no*) = 0/5 P(w.=t | *no*) =3/5

P(outlook=rainy | *no*) = 2/5 P(w.=f | *no*) =2/5

P(outlook=sunny | *no*) = 3/5

Outlook	T	H	Windy	Play?
overcast	64°F	65%	true	*yes*
overcast	72°F	90%	true	*yes*
overcast	81°F	75%	false	*yes*
overcast	83°F	86%	false	*yes*
rainy	68°F	80%	false	*yes*
rainy	70°F	96%	false	*yes*
rainy	75°F	80%	false	*yes*
rainy	65°F	70%	true	*no*
rainy	71°F	91%	true	*no*
sunny	69°F	70%	false	*yes*
sunny	75°F	70%	true	*yes*
sunny	72°F	95%	false	*no*
sunny	80°F	90%	true	*no*
sunny	85°F	85%	false	*no*

Problem: Estimates may be zero ⇒ P(*no* | outlook=overcast) would always be 0.

⇒ **Laplace correction:** Use (a+1)/(b+1) instead of a/b, e.g. 1/6 instead of 0/5.

alex@seewald.at / alex.seewald.at

Example: Weather dataset (2)

- For quantitative / numerical variables, probabilities cannot be determined by counting. Probability density functions must be defined. Common assumption: Values are normally distributed. Then, arithmetic mean and standard deviation define a normal probability density function as follows:

$$P(A_i = x \mid Cl_j) = \frac{1}{\sqrt{2\pi\,\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

where μ and σ^2 are chosen depending on Cl_j and A_i

E.g. for P(temp.=x | *no*) use μ=74.6 and σ^2=62.3;

for P(hum.=x | *yes*) use μ=79.1 and σ^2=104.4 etc..

Outlook	T	H	Windy	Play?
overcast	64°F	65%	true	*yes*
rainy	68°F	80%	false	*yes*
sunny	69°F	70%	false	*yes*
rainy	70°F	96%	false	*yes*
overcast	72°F	90%	true	*yes*
sunny	75°F	70%	true	*yes*
rainy	75°F	80%	false	*yes*
overcast	81°F	75%	false	*yes*
overcast	83°F	86%	false	*yes*
rainy	65°F	70%	true	*no*
rainy	71°F	91%	true	*no*
sunny	72°F	95%	false	*no*
sunny	80°F	90%	true	*no*
sunny	85°F	85%	false	*no*

	Play?	
Temp.	*yes*	*no*
μ	73.0	74.6
σ^2	38.0	62.3

	Play?	
Hum.	*yes*	*no*
μ	79.1	86.2
σ^2	104.4	94.7

alex@seewald.at / alex.seewald.at

Example: Weather dataset (3)

$$P(A_i = x \mid Cl_j) = \frac{1}{\sqrt{2\pi\,\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Classify example $\mathbf{x_n}$ = {outlook=overcast, temp.=18 °F, hum.=53%, windy=false}
(using simple Laplace correction for probabilities, e.g. 10/15 instead of 9/14 etc.)

$$P(yes \mid x_n) = P(yes)P(outlook = overcast \mid yes)P(temp. = 18°F \mid yes)P(hum. = 53\% \mid yes)$$

$$P(w. = f \mid yes) = \frac{10}{15}\frac{5}{10}\left(\frac{1}{\sqrt{2\pi 38.0}} e^{-\frac{1}{2*38.0}(18-73.0)^2}\right)\left(\frac{1}{\sqrt{2\pi 104.4}} e^{-\frac{1}{2*104.4}(53-79.1)^2}\right)\frac{7}{10} =$$

$$\approx 0.333(3.35*10^{-19})(1.495*10^{-3})0.7 = 1.167*10^{-22}$$

$$P(no \mid x_n) = P(no)P(outlook = overcast \mid no)P(temp. = 18°F \mid no)P(hum. = 53\% \mid no)$$

$$P(w. = f \mid no) = \frac{6}{15}\frac{1}{6}\left(\frac{1}{\sqrt{2\pi 62.3}} e^{-\frac{1}{2*62.3}(18-74.6)^2}\right)\left(\frac{1}{\sqrt{2\pi 94.7}} e^{-\frac{1}{2*94.7}(53-86.2)^2}\right)\frac{3}{6} =$$

$$\approx 0.067(3.45*10^{-13})(1.217*10^{-4})0.5 = 1.407*10^{-18}$$

$\Rightarrow$ predict Play=*no*.

Notice that since P(TD) is not known, these are not yet real probabilities (i.e. they do not sum to 1), but have to be normalized: P(no|x_n)=99.992%, P(yes|x_n)=0.008%

alex@seewald.at / alex.seewald.at

Bayesian Belief Networks

Naïve Bayes: Assumes conditional independence of all attribute. If this is true, then it outputs the optimal Bayes classification. However, in many cases this assumption is overly restrictive.

⇒ **Bayesian Networks:** Allow arbitrary conditional dependence. Dependency information can be learned from training data, or specified as background knowledge. Usually visualized as directed acyclic graph (DAG)

E.g. given a burglary, what is the prob. that John calls?

$$P(J \mid B) = ?$$

$$P(A \mid B) = P(B)P(\neg E)(0.94) + P(B)P(E)(0.95)$$

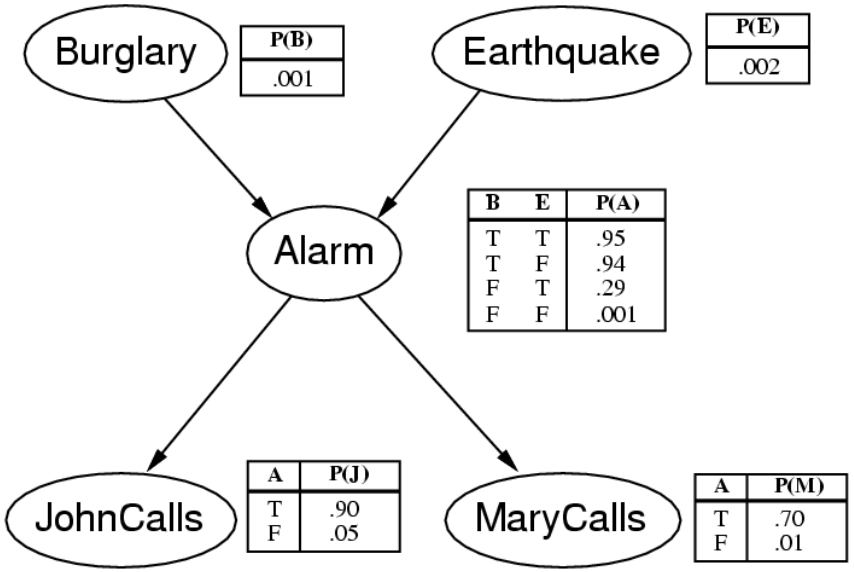
$$P(A \mid B) = 1(0.998)(0.94) + 1(0.002)(0.95)$$

$$P(A \mid B) = 0.94$$

$$P(J \mid B) = P(A \mid B)(0.9) + P(\neg A \mid B)(0.05)$$

$$P(J \mid B) = (0.94)(0.9) + (0.06)(0.05)$$

$$= 0.85$$



Exact computation of complex queries is NP-hard(!)

alex@seewald.at / alex.seewald.at

Common Preprocessing Steps

Some learning algorithms only accept quantitative X

- 2 values ⇒ map to one quantitative variable as 0/1 or -1/1
- *n* values ⇒ map to *n* quantitative binary variables, only one of which is on (=1). Also called *1-of-n coding*, *dummy variables*.

Some learning algorithms only accept quantitative Y

- 2 values ⇒ map as above, and map the prediction Y back via simple treshold (0.5 for 0/1 and 0 for -1/1)
- *n* values ⇒ multiple models have to be learned; e.g. map as above and use one binary variable for each of *n* models. More complex mappings are possible.

Some learning algorithms do not accept quantitative X or Y

- Discretize values to a qualitative variable with ordinal scale
- Some information is inevitably lost - performance does not necessarily suffer.

alex@seewald.at / alex.seewald.at