Bayesian Methods & Association Rule Mining

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

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

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+2) instead of a/b, e.g. 1/7 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

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?

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

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

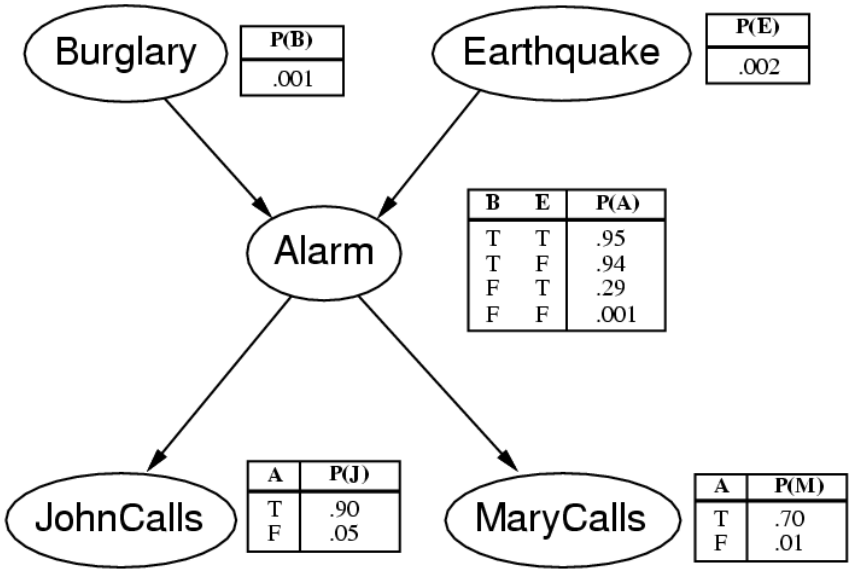
$$P(A \mid B) = 1(0.998)(0.94) + 1(0.002)(0.95)$$

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

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

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

$$= 0.85$$



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

alex@seewald.at / alex.seewald.at

Association Rule Mining

Different learning methodology (not prediction)

- Find rules of the form $A \Rightarrow B$ where A and B are itemsets, (i.e. sets of *items*) and $A \cap B = \emptyset$.

- An item is a specific object which may either be part of an itemset or not.
- Rules can related to any attribute; there is no target variable (*unsupervised* as in Reinforcement Learning)

- Usually applied to binary data (*market basket analysis*); Nominal variables can be transformed in the usual way into one binary attribute per value. Each binary attribute corresponds to an item with $attr_i = 1 \Leftrightarrow$ Item *i* is present.

alex@seewald.at / alex.seewald.at

Association Rule Mining: Defs.

- T is the set of all possible items. When $A,B \subseteq T$, $A \cap B = \emptyset$: $A \Rightarrow B$ is an association rule.

  E.g. supermarket: Each distinct product is an item $\in$ T. Each transaction Tr_i corresponds to the products bought by one shopper, and thus to a specific itemset. TD = $\{Tr_i\}$.

- Rules of the form $A \Rightarrow B$ tell us about correlations between itemsets, e.g. *(milk,beer)* $\Rightarrow$ *(diapers, babyfood)*
- Rules are characterized by *Support* (how common is the rule?), and *Confidence* (how well does rule $A \Rightarrow B$ hold?)
- Algorithms to find **all** rules with given minimum *Support* and *Confidence* exist, and are space- and time-efficient.

alex@seewald.at / alex.seewald.at

Support & Confidence

$$Support(A \Rightarrow B) = P(A \cup B) \quad [P(A) \text{ for Borgelt's Implementation}]$$

$$Confidence(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)}$$

$$P(I) = \frac{1}{|TD|} \sum_{Tr \in TD} \prod_{i \in I} (Tr_i = 1) \qquad Lift(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)P(B)}$$

Itemset I is frequent $\Leftrightarrow$ *Support(I)* $\geq$ *minimum support*

Adding items: Support can only monotonically decrease ($P(I \cup X) \leq P(I)$), since we add restrictions to the itemset.

$\Rightarrow$If an itemset *I* is frequent, all its subsets must also be frequent. If any subset has lower support, then *I* cannot be be frequent. This is an efficient pruning criterion

alex@seewald.at / alex.seewald.at

The Apriori Algorithm

Computes the set of all frequent itemsets.

L_1 = {set of frequent items (=frequent itemsets of size 1)}
for (k=1; L_k!=Ø; k++) {
  C_{k+1} = {A∪B | A,B ∈ L_k, |A∪B|=k+1,
    ∀ X⊂(A∪B) ⇒ X∈(∪L_k)}
  L_{k+1} = {C | C ∈ C_{k+1} && *Support(C) > MinSupp* }
}
return ∪ L_k;

Afterwards, compute all possible rules (partitions) of frequent itemsets and output those with min. confidence.

Open-source Apriori implementation by C. Borgelt
http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html
Also available in WEKA, Associate Tab.

alex@seewald.at / alex.seewald.at

Rules from Weather Dataset (nominal)

Minimum Support = 0.3, *Minimum Confidence* = 0.7

- **outlook=overcast ⇒ play=yes**
  supp: 0.28, conf: 1.0, lift: 1.56
- **temperature=cool ⇒ humidity=normal**
  supp: 0.28, conf: 1.0, lift: 2.0
- **humidity=normal, windy=false ⇒ play=yes**
  supp: 0.28, conf: 1.0, lift: 1.56
- **humidity=normal ⇒ play=yes**
  supp: 0.43, conf: 0.86, lift: 1.33
- **play=no ⇒ humidity=high**
  supp: 0.28, conf: 0.8, lift: 1.6

...

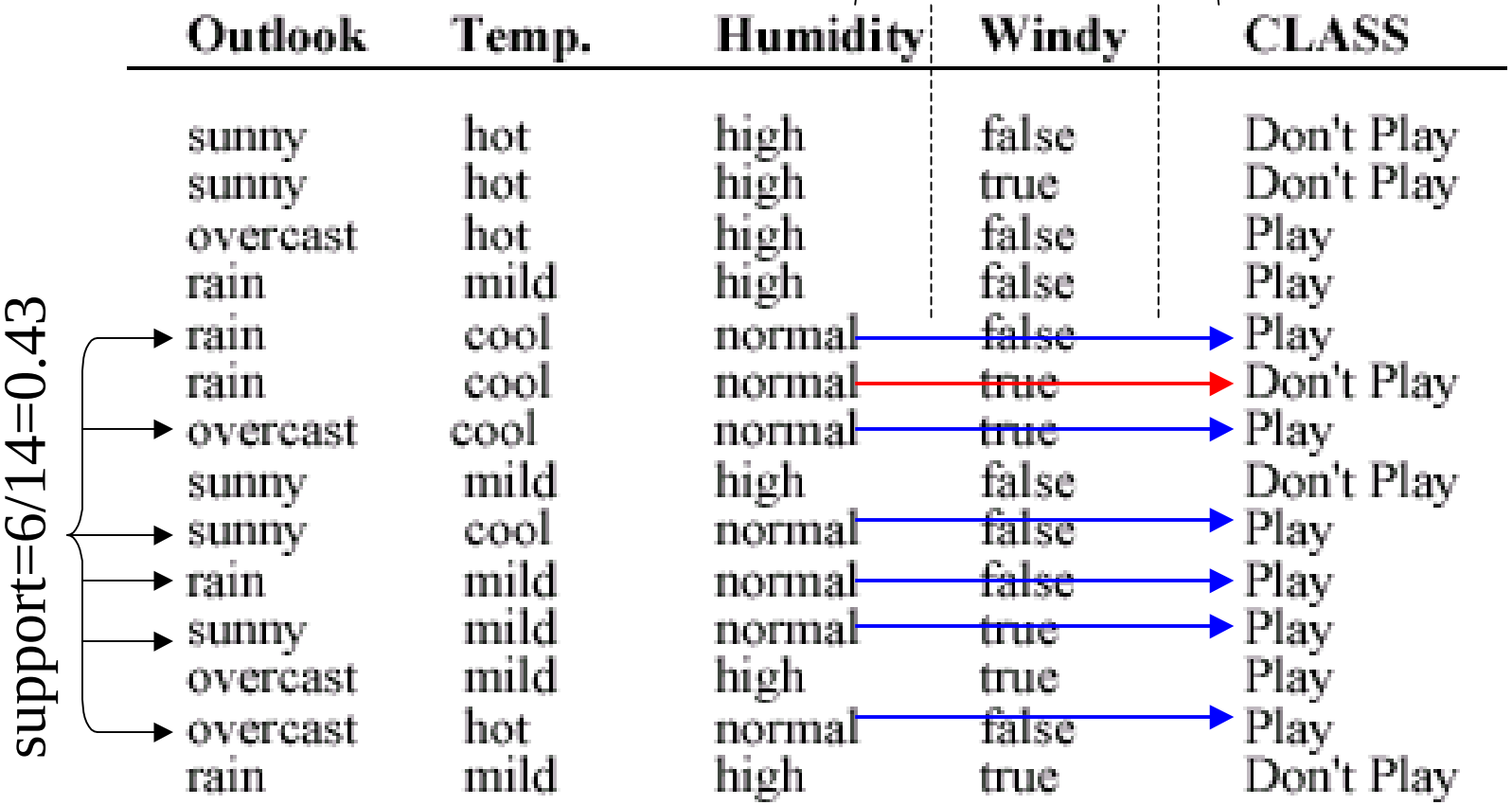
For large datasets, outputs a large set of rules (>1000), so understanding rules is more challenging than mining.

alex@seewald.at / alex.seewald.at

Ex.: Computing Support and Confidence

- **humidity=normal ⇒ play=yes**

confidence=6/7=0.86 (6x correct, 1x wrong)



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

alex@seewald.at / alex.seewald.at