Exercise: 2

Meeting on January 12th / 14th

Problem 1: More on Naive Bayes

The following dataset is given:

Fall-Nr.	Clouds	Temperature	Humidity	Wind	Decision
1	sunny	high	high	no	+
2	sunny	high	high	yes	+
3	cloudy	medium	normal	no	+
4	rainy	high	high	no	-
5	sunny	low	normal	yes	-
6	rainy	low	high	yes	-
7	cloudy	high	high	yes	-
8	sunny	medium	normal	no	+
9	cloudy	medium	high	yes	-
Test case 1	rainy	medium	high	no	?
Test case 2	sunny	medium	high	yes	?
Test case 3	?	low	normal	yes	+
Test case 4	?	low	high	no	-

Table 1: Dataset and test cases for a Naive Bayes classifier

(a) Use Naive Bayes to predict the decisions for test cases 1 and 2 from the training data. What is the baseline for this problem?

(b) Use Naive Bayes to predict the decisions for test cases 3 and 4 from the training data. What is the baseline for this problem?

(c) Draw the corresponding Bayes network for one of the two previous tasks and explain the assumptions made by the Naive Bayes model.

Lösung:

(a) Baseline always predicts "-" since most training data (5/9) would be classified correctly.

We need P(Decision|rainy, medium, high, no) for test case 1.

$$\begin{aligned} & P(+|\text{rainy, medium, high, no}) \\ = \; & P(+) \cdot P(\text{rainy}|+) \cdot P(\text{medium}|+) \cdot P(\text{high}|+) \cdot P(\text{no}|+) \cdot \alpha \\ = \; & \frac{4}{9} \cdot \frac{0}{4} \cdot \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{3}{4} \cdot \alpha = 0 \cdot \alpha \end{aligned}$$

$$\begin{aligned} & P(-|\text{rainy, medium, high, no}) \\ = \; & P(-) \cdot P(\text{rainy}|-) \cdot P(\text{medium}|-) \cdot P(\text{high}|-) \cdot P(\text{no}|-) \cdot \alpha \\ = \; & \frac{5}{9} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{1}{5} \cdot \alpha \approx 0{,}007 \cdot \alpha \end{aligned}$$

$\alpha = 1/(0 + 0{,}007) = 1/0{,}007$

$P(+|\text{rainy, medium, high, no}) = 0/0{,}007 = 0\%$

$P(-|\text{rainy, medium, high, no}) = 0{,}007/0{,}007 = 100\%$

$\Rightarrow$ Naive Bayes predicts $-$ for test case 1.

We need P(Bewertung|sunny, medium, high, yes) for test case 2.

$$\begin{aligned} & P(+|\text{sunny, medium, high, yes}) \\ = \; & P(+) \cdot P(\text{sunny}|+) \cdot P(\text{medium}|+) \cdot P(\text{high}|+) \cdot P(\text{yes}|+) \cdot \alpha \\ = \; & \frac{4}{9} \cdot \frac{3}{4} \cdot \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{1}{4} \cdot \alpha \approx 0{,}021 \cdot \alpha \end{aligned}$$

$$\begin{aligned} & P(-|\text{sunny, medium, high, yes}) \\ = \; & P(-) \cdot P(\text{sunny}|-) \cdot P(\text{medium}|-) \cdot P(\text{high}|-) \cdot P(\text{yes}|-) \cdot \alpha \\ = \; & \frac{5}{9} \cdot \frac{1}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{4}{5} \cdot \alpha \approx 0{,}014 \cdot \alpha \end{aligned}$$

$\alpha = 1/(0{,}021 + 0{,}014) = 1/0{,}035$

$P(+|\text{sunny, medium, high, yes}) = 0{,}021/0{,}035 = 60\%$

$P(-|\text{sunny, medium, high, yes}) = 0{,}014/0{,}035 = 40\%$

$\Rightarrow$ Naive Bayes predicts $+$ for test case 2.

(b) Baseline always predicts sunny since most training data (4/9) would be classified correctly.

We need $P(\text{Clouds}|\text{low, normal, yes, +})$ for test case 3.

$$\begin{aligned} & P(\text{sunny}|\text{low, normal, yes, +}) \\ = \; & P(\text{sun.}) \cdot P(\text{low}|\text{sun.}) \cdot P(\text{normal}|\text{sun.}) \cdot P(\text{yes}|\text{sunny}) \cdot P(+|\text{sun.}) \cdot \alpha \\ = \; & \frac{4}{9} \cdot \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{3}{4} \cdot \alpha \approx 0{,}021 \cdot \alpha \end{aligned}$$

$$\begin{aligned} & P(\text{cloudy}|\text{low, normal, yes, +}) \\ = \; & P(\text{cl.}) \cdot P(\text{low}|\text{cl.}) \cdot P(\text{normal}|\text{cl.}) \cdot P(\text{yes}|\text{cl.}) \cdot P(+|\text{cl.}) \cdot \alpha \\ = \; & \frac{3}{9} \cdot \frac{0}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \alpha = 0 \cdot \alpha \end{aligned}$$

$$\begin{aligned} & P(\text{rainy}|\text{low, normal, yes, +}) \\ = \; & P(\text{rainy}) \cdot P(\text{low}|\text{rainy}) \cdot P(\text{normal}|\text{rainy}) \cdot P(\text{yes}|\text{rainy}) \cdot P(+|\text{rainy}) \cdot \alpha \\ = \; & \frac{2}{9} \cdot \frac{1}{2} \cdot \frac{0}{2} \cdot \frac{1}{2} \cdot \frac{0}{2} \cdot \alpha = 0 \cdot \alpha \end{aligned}$$

$\alpha = 1/(0{,}021 + 0 + 0) = 1/0{,}021$

$P(\text{sunny}|\text{low, normal, yes, +}) = 0{,}021/0{,}021 = 100\%$

$P(\text{cloudy}|\text{low, normal, yes, +}) = 0/0{,}021 = 0\%$

$P(\text{rainy}|\text{low, normal, yes, +}) = 0/0{,}021 = 0\%$

$\Rightarrow$ Naive Bayes predicts sunny for test case 3.

We need $P(\text{Clouds}|\text{low, high, no, -})$ for test case 4.

$$\begin{aligned} & P(\text{sunny}|\text{low, high, no, -}) \\ = \; & P(\text{sun}) \cdot P(\text{low}|\text{sun.}) \cdot P(\text{high}|\text{sun.}) \cdot P(\text{no}|\text{sun.}) \cdot P(\text{-}|\text{sun.}) \cdot \alpha \\ = \; & \frac{4}{9} \cdot \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{1}{4} \cdot \alpha \approx 0{,}007 \cdot \alpha \end{aligned}$$

$$\begin{aligned} & P(\text{cloudy}|\text{low, high, no, -}) \\ = \; & P(\text{cl.}) \cdot P(\text{low}|\text{cl.}) \cdot P(\text{high}|\text{cl.}) \cdot P(\text{no}|\text{cl.}) \cdot P(\text{-}|\text{cl.}) \cdot \alpha \\ = \; & \frac{3}{9} \cdot \frac{0}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \alpha = 0 \cdot \alpha \end{aligned}$$

$$\begin{aligned} & P(\text{rainy}|\text{low, high, no, -}) \\ = \; & P(\text{rainy}) \cdot P(\text{low}|\text{rainy}) \cdot P(\text{high}|\text{rainy}) \cdot P(\text{no}|\text{rainy}) \cdot P(\text{-}|\text{rainy}) \cdot \alpha \\ = \; & \frac{2}{9} \cdot \frac{1}{2} \cdot \frac{2}{2} \cdot \frac{1}{2} \cdot \frac{2}{2} \cdot \alpha \approx 0{,}056 \cdot \alpha \end{aligned}$$

$\alpha = 1/(0{,}007 + 0 + 0{,}056) = 1/0{,}063$

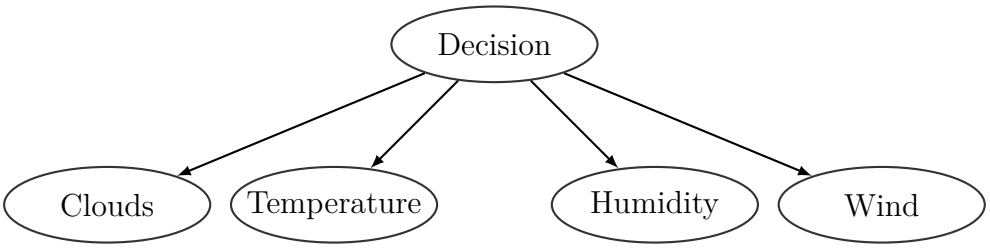
$P(\text{sunny}|\text{low, high, no, -}) = 0{,}007/0{,}063 \approx 11{,}1\%$

$P(\text{cloudy}|\text{low, high, no, -}) = 0/0{,}063 = 0\%$

$P(\text{rainy}|\text{low, high, no, -}) = 0{,}056/0{,}063 \approx 88{,}9\%$

$\Rightarrow$ Naive Bayes predicts rainy for test case 4.

(c) Bayes network for test cases 1 and 2:

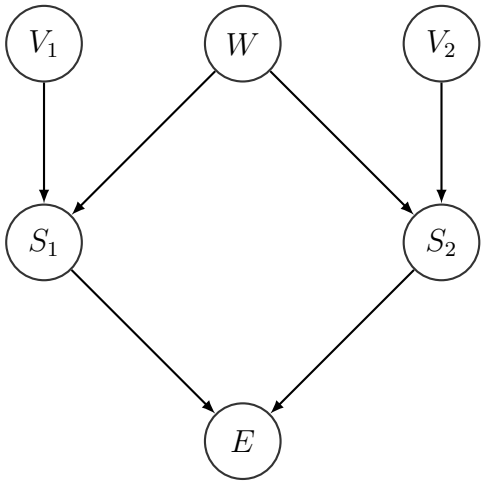


Naive Bayes assumes for these test cases that "Clouds", "Temperature", "Humidity" and "Wind" are independent of each other under the condition "Decision". Naive Bayes makes no assumptions on dependences between "Clouds", "Temperature", Humidity" and "Wind".

Problem 2: Tennis

Given is the following model for a game of tennis with two players (1 & 2):

Abbreviation	Description
V_1/V_2	Player 1/2 is hurt (v) or healthy (g)
W	The weather ist either sunny (s), cloudy (w) or rainy (r)
S_1/S_2	The player strength ot the two players
E	Outcome of the game: Either player 1 (e_1) or player 2 (e_2) wins.



a) What variable combinations need to be fulfilled so that S_1 and V_2 are d-separated? Make a table and write down for all possible combinations whether they fulfill the conditions.

b) Player strength is actually a continuous variable. Describe two approaches on how to model such a variable.

c) Assume that player strength has three possible values {strong, medium, weak}. How many entries does the conditional probablity table (CPT) for node S_1 contain? How many entries does the Bayes network contain in total? How many entries would the CPT need for variable E if all five remaining variables would have direct influence on the outcome E?

d) If S_1 contains the value "strong" and the weather is sunny, is V_1 dependent on the other variables? Explain your answer.

e) Now assume the following value ranges and CPTs. What is the prior probability for the weather being cloudy or rainy? What is the probability if you know that both players are "healthy", but have a "weak" player strength?

V_1	P
hurt v_1	0,3
healthy g_1	0,7

V_2	P
hurt v_2	0,2
healthy g_2	0,8

W	P
sunny s	0,6667
cloudy w	0,2222
rainy r	0,1111

Weather W is	sunny s		cloudy b		rainy r	
Health V_1 is	hurt	healthy	hurt	healthy	hurt	healthy
Player 1 is strong $\uparrow_1$	0,2	0,6	0,5	0,6	0,6	0,7
Player 1 ist weak $\downarrow_1$	0,8	0,4	0,5	0,4	0,4	0,3

Weather W is	sunny s		cloudy b		rainy r	
Health V_1 is	hurt	healthy	hurt	healthy	hurt	healthy
Player 2 is strong $\uparrow_2$	0,6667	0,6667	0,3333	0,3333	0,25	0,25
Player 2 is weak $\downarrow_2$	0,3333	0,3333	0,6667	0,6667	0,75	0,75

Player 1 is	strong		weak	
Player 2 is	strong	weak	strong	weak
Player 1 wins e_1	0,4	0,2	0,6	0,5
Player 2 wins e_2	0,6	0,8	0,4	0,5

Lösung:

a) Definition of d-separation from Wikipedia:

> Let P be a trail from node u to v. A trail is a loop-free, undirected (i.e. all edge directions are ignored) path between two nodes. Then P is said to be d-separated by a set of nodes Z if any of the following conditions holds:

(1) P contains (but does not need to be entirely) a directed chain, $u \cdots \leftarrow m \leftarrow \cdots v$ or $u \cdots \rightarrow m \rightarrow \cdots v$ such that the middle node m is in Z,

(2) P contains a fork, $u \cdots \leftarrow m \rightarrow \cdots v$, such that the middle node m is in Z, or

(3) P contains an inverted fork (or collider), $u \cdots \rightarrow m \leftarrow \cdots v$, such that the middle node m is not in Z and no descendant of m is in Z.

The nodes u and v are d-separated by Z if all trails between them are d-separated.

There are two undirected trails from S_1 to V_2:

- Trail 1: $S_1 \rightarrow W \rightarrow S_2 \rightarrow V_2$
- Trail 2: $S_1 \rightarrow E \rightarrow S_2 \rightarrow V_2$

Z				d-sep.	Trail 1			Trail 2		
V_1	W	E	S_2		C. (1)	C. (2)	C. (3)	C. (1)	C. (2)	C. (3)
				yes			+			+
			1					+		+
		1								
		1	1					+		
	1			yes		+	+			+
	1		1	yes		+		+		+
	1	1				+				
	1	1	1	yes		+		+		
1				yes			+			+
1			1					+		+
1		1								
1		1	1					+		
1	1			yes		+	+			+
1	1		1	yes		+		+		+
1	1	1				+				
1	1	1	1	yes		+		+		

Explanation for Trail 1:

- Condition (1) cannot be fulfilled since no candidate node exists on the trail.
- Condition (2) is fulfilled when W is given.
- Condition (3) is fulfilled when S_2 is *not* given and no descendant, here E, is given.

Erklärung für Weg 2:

- Condition (1) is fulfilled when S_2 is given.
- Condition (2) cannot be fulfilled since no candidate node exists on the trail.
- Condition (3) is fulfilled when E is *not* given. E has no descendants here.

S_1 and V_2 are d-separated when both trails fulfill at least one condition.

b) There are typically two approaches:

- *Discretization*: Define discrete intervals and group the values
- *Parametrization*: Using parametrized probability distributions

c)
- The CPT for node S_1 has $2 \cdot 3 \cdot 3 = 18$ entries since S_1 itself has 3 possible values and the probabilities need to be given dependent on the values of V_1 and W with 2 and 3 possible values respectively. However, these 18 probabilities are not independent of each other. Only 12 entries are necessary if computations are allowed. For exmaple, $P(S_1 = \text{strong}|X)$ can be described as

$$P(S_1 = \text{strong}|X) = 1 - P(S_1 = \text{weak}|X) - P(S_1 = \text{medium}|X)$$

- The given network has

$$\begin{aligned} & 2 \quad +3 \quad +2 \\ & +2 \cdot 3 \cdot 3 \quad +3 \cdot 2 \cdot 3 \\ & +3 \cdot 3 \cdot 2 \\ = 2 + 2 + 3 + 18 + 18 + 18 & \\ = 61 & \end{aligned}$$

  CPT entries.

- If all five variable had direct influence on the outcome, the CPT would have $2 \cdot 3 \cdot 2 \cdot 3 \cdot 3 \cdot 2 = 216$ for the resuts of $V_1 \cdot W \cdot V_2 \cdot S_1 \cdot S_2 \cdot E$.

d) No, the Markov-Blanket of V_1 is given completely. Hence, V_1 is independent of *all* other variables in the network.

e) The prior probability of the weather being cloudy *or* rainy is computed as:

$$P(W = \text{rainy} \cup W = \text{cloudy}) = 1 - P(W = \text{sunny}) = 1 - 0{,}6667 = 0{,}3333$$

We want to find $1 - P(W = \text{sunny} | V_1 = \text{healthy} \cap V_2 = \text{healthy} \cap S_1 = \downarrow_1 \cap S_2 = \downarrow_2)$. From here on we write $P(s|B)$ with condition $B = g_1 \cap g_2 \cap \downarrow_1 \cap \downarrow_2$.

$$P(s|B) = \frac{P(s \cap B)}{P(B)} = \frac{P(e_1 \cap s \cap B) + P(e_2 \cap s \cap B)}{\sum_{E=e_1,e_2} \sum_{W=s,w,r} P(E \cap W \cap B)}$$

Use the chain rule on the numerator:

$$\begin{aligned} P(e_{1/2} \cap s \cap B) &= P(e_{1/2} \cap s \cap g_1 \cap g_2 \cap \downarrow_1 \cap \downarrow_2) \\ &= P(e_{1/2} | \downarrow_1 \cap \downarrow_2) P(\downarrow_1 | g_1 \cap s) P(\downarrow_2 | g_2 \cap s) P(g_1) P(s) P(g_2), \end{aligned}$$

The numerator is then

$$\begin{aligned} & P(e_1 \cap s \cap B) + P(e_2 \cap s \cap B) \\ = \; & \left(P(e_1 | \downarrow_1 \cap \downarrow_2) + P(e_2 | \downarrow_1 \cap \downarrow_2)\right) P(\downarrow_1 | g_1 \cap s) P(\downarrow_2 | g_2 \cap s) P(g_1) P(s) P(g_2) \\ = \; & (0{,}5 + 0{,}5) \cdot 0{,}4 \cdot 0{,}3333 \cdot 0{,}7 \cdot 0.6667 \cdot 0{,}8 \\ = \; & 0{,}04978 \end{aligned}$$

Do the same for the denominator:

$$\begin{aligned} P(e_{1/2} \cap \{s,w,r\} \cap B) &= P(e_{1/2} \cap \{s,w,r\} \cap g_1 \cap g_2 \cap \downarrow_1 \cap \downarrow_2) \\ &= P(e_{1/2} | \downarrow_1 \cap \downarrow_2) P(\downarrow_1 | g_1 \cap \{s,w,r\}) \\ &\quad \cdot P(\downarrow_2 | g_2 \cap \{s,w,r\}) P(g_1) P(\{s,w,r\}) P(g_2), \end{aligned}$$

We get six summands:

- $P(e_1 \cap s \cap B) = 0{,}5 \cdot 0{,}4 \cdot 0{,}3333 \cdot 0{,}7 \cdot 0{,}6667 \cdot 0{,}8 = 0{,}02489$
- $P(e_1 \cap w \cap B) = 0{,}5 \cdot 0{,}4 \cdot 0{,}6667 \cdot 0{,}7 \cdot 0{,}2222 \cdot 0{,}8 = 0{,}01659$
- $P(e_1 \cap r \cap B) = 0{,}5 \cdot 0{,}3 \cdot 0{,}75 \cdot 0{,}7 \cdot 0{,}1111 \cdot 0{,}8 = 0{,}006999$
- $P(e_2 \cap s \cap B) = 0{,}5 \cdot 0{,}4 \cdot 0{,}3333 \cdot 0{,}7 \cdot 0{,}6667 \cdot 0{,}8 = 0{,}02489$
- $P(e_2 \cap w \cap B) = 0{,}5 \cdot 0{,}4 \cdot 0{,}6667 \cdot 0{,}7 \cdot 0{,}2222 \cdot 0{,}8 = 0{,}01659$
- $P(e_2 \cap r \cap B) = 0{,}5 \cdot 0{,}3 \cdot 0{,}75 \cdot 0{,}7 \cdot 0{,}1111 \cdot 0{,}8 = 0{,}006999$

Hence the denominator is $\sum_{E=e_1,e_2} \sum_{W=s,w,r} P(E \cap W \cap B) = 0.09696$.
Damit gilt:

$$P(r \cup b|B) = 1 - P(s|B) = 1 - \frac{0{,}04978}{0{,}09696} = 0{,}4866$$

Derivation of the chain rule:

- **The general case**:
  It applies that
  $$P(X \cap Y) = P(X|Y)P(Y)$$
  Now Write
  $$\begin{aligned} & P(A_1 \cap A_2 \cap \cdots \cap A_n) \\ = \; & \frac{P(A_1 \cap A_2 \cap \cdots \cap A_n)}{P(A_1 \cap \cdots \cap A_{n-1})} \cdots \frac{P(A_1 \cap A_2 \cap A_3)}{P(A_1 \cap A_2)} \cdot \frac{P(A_1 \cap A_2)}{P(A_1)} \cdot P(A_1) \\ = \; & P(A_n|A_1 \cap \cdots A_n) \cdots P(A_3|A_1 \cap A_2) \cdot P(A_2|A_1) \cdot P(A_1) \end{aligned}$$

- **Application to our case**:
  We use the upper formular after rewriting the equation in inverse order in the net. We are left with three independent variables V_1, V_2 and W. These do not need to be broken down further:
  $$\begin{aligned} & P(e_1 \cap s \cap g_1 \cap g_2 \cap \downarrow_1 \cap \downarrow_2) = P(e_1 \cap \downarrow_1 \cap \downarrow_2 \cap g_1 \cap g_2 \cap s) \\ = \; & \frac{P(e_1 \cap \downarrow_1 \cap \downarrow_2 \cap g_1 \cap g_2 \cap s)}{P(\downarrow_1 \cap \downarrow_2 \cap g_1 \cap g_2 \cap s)} \cdot \frac{P(\downarrow_1 \cap \downarrow_2 \cap g_1 \cap g_2 \cap s)}{P(\downarrow_2 \cap g_1 \cap g_2 \cap s)} \\ & \cdot \frac{P(\downarrow_2 \cap g_1 \cap g_2 \cap s)}{P(g_1 \cap g_2 \cap s)} \cdot P(g_1 \cap g_2 \cap s) \\ = \; & P(e_1 | \downarrow_1 \cap \downarrow_2 \cap g_1 \cap g_2 \cap s) \cdot P(\downarrow_1 | \downarrow_2 \cap g_1 \cap g_2 \cap s) \\ & \cdot P(\downarrow_2 | g_1 \cap g_2 \cap s) \cdot P(g_1 \cap g_2 \cap s) \end{aligned}$$
  Since e_1 is only directly dependent on $\downarrow_1$ and $\downarrow_2$ and $\downarrow_{1,2}$ is still only dependent on $g_{1,2}$ and s this can be further simplified to
  $$\begin{aligned} & P(e_1 \cap \downarrow_1 \cap \downarrow_2 \cap g_1 \cap g_2 \cap s) \\ = \; & P(e_1 | \downarrow_1 \cap \downarrow_2) \cdot P(\downarrow_1 | g_1 \cap s) \cdot P(\downarrow_2 | g_2 \cap s) \cdot P(g_1) \cdot P(g_2) \cdot P(s), \end{aligned}$$
  where we used the independence of g_1, g_2 and s in the last step.

Problem 3: Sampling-Methods

In the year 2031 you are leading a company making skirts for people with plus size worn by both men and women. You deliver to England (80%) and Scotland (20%). From your statistical analysis you know that both countries have less women f than men m with a ratio of 4/6. The probability of a person wearing skirts as well as the probability of a person needing plus size are given in 2.

Country	Probability
England	80%
Scotland	20%

Gender	Probability
Female	40%
Male	60%

	Scotland		England	
	male	female	male	female
Plus size	16%	3%	50%	6%
Skirt	5%	30%	0,5%	40%

Table 2: Statistics about skirt size and wearing skirts.

Now view the following Bayes network in Fig. 1. You know from your statistics that

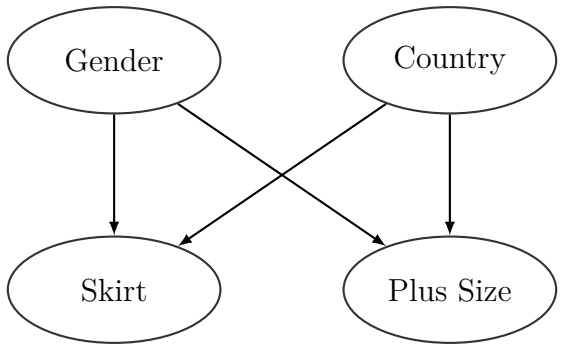


Figure 1: Bayes Network

60 million people are wearing plus size. Now you want to know the size of your target group.

(a) What are the two possibilites to compute $P(\text{Wearing Skirts}|\text{Plus Size})$?

> **Lösung:** You can compute the probability *analytically* from the given probabilites. However, the number of computations grows exponentially with network size. On the other hand, you can use sampling methods to *estimate* the probability, which we are going to do now.

(b) You are tired of analytical methods and instead remember a special Markov-Chain-Monte-Carlo (MCMC) sampling method called Gibbs sampling. Describe the method and its differences to Likelihood-Weighting and direct sampling.

Lösung: During the MCMC method, the configuration (state) of the Bayes network is changed iteratively, i.e. a new sample is created from the previous one. During Gibbs sampling only one specific variable is set in dependence of its markov blanket (mb).For a variable X_i it applies that:

$$P(X_i = x_i|\text{mb}(X_i)) = \alpha P(X_i = x_i|\text{parents}(X_i)) \cdot \prod_{Y_j \in \text{children}(X_i)} P(Y_j = y_j|\text{parents}(Y_j))$$

Direct sampling and likelihood weighting ditch the previous configuration during sampling. The random variables are newly assigend and weighted by their corresponding topological order.

(c) You know about tools like UnBBayes[1], which implement these algorithms but as a computer science graduate you would like to refresh your knowledge about how such an algorithm works. For precise inspection you choose the Likelihood-Weighting-Method. Complete 3 with the missing samples.

Node: Name	Gender	Country	Plus Size	Skirt	Weight
Next random number	0.60	0.91	-	0.08	
Node: Value	m	S	yes	no	0.16
Next random number	0.55	0.04	-	0.21	
Node: Value	m	E	yes	no	0.50
Next random number	0.80	0.79	-	0.58	
Node: Value	m	E	yes	no	0.50
Next random number	0.91	0.85	-	0.34	
Node: Value	m	S	yes	no	0.16
Next random number	0.86	0.97	-	0.02	
Node: Value	m	S	yes	yes	0.16
Next random number	0.21	0.26	-	0.06	
Node: Value	w	E	yes	yes	0.06
Next random number	0.17	0.60	-	0.88	
Node: Value	w	E	yes	no	0.06
Next random number	0.60	0.66	-	0.13	
Node: Value	m	E	yes	no	0.50
Next random number	0.70	0.67	-	0.68	
Node: Value	m	E	yes	no	0.50
Next random number	0.39	0.71	-	0.37	
Node: Value	w	E	yes	yes	0.06

Table 3: Samples for Liklihood-Weighting

Lösung: The missing values can be found in the table above.

[1] http://unbbayes.sourceforge.net/

(d) Now you want to estimate $P(\text{Wearing Skirts}|\text{Plus Size})$ from these samples. Use

$$P(A|B) = \frac{\sum \text{Weights of samples that fulfill condition(s) } A}{\sum \text{Weights of all samples}}.$$

What is the size of your target group?

Lösung: From the above formula we get

$$\text{numerator} \approx 0.16 + 0.06 + 0.06 = 0.28$$

and

$$\begin{aligned} \text{denominator} &\approx 0.16 + 0.50 + 0.50 + 0.16 + 0.16 + \\ &\quad 0.06 + 0.06 + 0.50 + 0.50 + 0.06 \\ &= 2.66 \end{aligned}$$

resulting in

$$P(\text{Wearing Skirt}|\text{Plus Size}) \approx \frac{0.28}{2.66} \approx 0.11.$$

Hence, the target group is at around 6.3 million people.

(e) Why is $P(\text{Wearing Skirts}|\text{Plus Size})$ not exact? How can you improve the accuracy?

Lösung: All sampling methods only give a statistic of the true probablities, where the accuracy is dependent on the number of samples. The "speed" of convergence depends on the exact sampling method.

(f) Implement the algorithm for the example above and compute a better value for $P(\text{Wearing Skirts}|\text{Plus Size})$. If necessary, plot multiple curves with different starting conditions against the number of samples.

Lösung: See python-Code.

```
import random
import matplotlib.pyplot as plt

class Net:
    def __init__(self):
        # storage for all nodes
        self.nodes = []
        # reset the sample history
        self.reset_samples()

    def add(self, node):
        self.nodes.append(node)
```

```
def reset_samples(self):
    self.history = []
    self.constrained = 0
    self.total = 0

def generate_lw_sample(self, A, B):
    # p defines the weight of the sample
    p = 1.0
    for node in self.nodes:
        # if the node is already set by B (condition)
        # we need to get the weight of the sample
        isInB = False
        for bl, _ in B:
            if bl == node.label:
                isInB = True
                break

        if isInB:
            # multiply weight
            p *= node.P(B)
        else:
            # generate sample
            ppair = node.generate_lw_sample(B)
            B.append((node.label, ppair.v))


    # optional: keep track of all samples
    self.history.append((B, p))

    # check if the sample fullfilles the prerequisite A
    fullfilles = True
    for al, av in A:
        subff = False
        for bl, bv in B:
            if bl == al and bv == av:
                subff = True
                break

        if not subff:
            fullfilles = False
            break

    # if so, add it to the numerator
    if fullfilles:
        self.constrained += p

    # always add to the denumerator
```

```
        self.total += p


        # return the current value of the probability
        return self.constrained / self.total


    def P(self, A):
        # compute the total probability of a condition
        p = 1.0
        for node in self.nodes:
            p *= node.P(A)

        return p


class PPair:
    def __init__(self, value, probability, parents = None):
        # store a CPT entry

        # value
        self.v = value

        # probability
        self.p = probability

        # contidions
        self.parents = parents


    def contains_parent(self, label, value):
        if self.parents is None:
            return True

        for plabel, pvalue in self.parents:
            if plabel == label and pvalue != value:
                return False

        return True

# a single node in a Bayes net
class Node:
    def __init__(self, label, values, parents):

        # the label of the node
        self.label = label
```

```
        # all entries of the CPT
        self.values = values

        # the parent nodes
        self.parents = parents

    # Compute the probability for a given condition
    # The condition may contain more constraints than required
    def P(self, A):
        # Find node value
        self_value = None
        for l, v in A:
            if l == self.label:
                self_value = v

        if self_value is None:
            raise Exception("Value for label %s not found in %s" % (
                ↪ self.label, A))

        # find the CPT entry that matches A
        a = self.findPPair(self_value, A)

        # there must only be one entry!
        if len(a) != 1:
            raise Exception("Multiple values found in CPT: %s" % a)

        return a[0].p


    # Find a CPT entry that matches P(self_value|labels)
    def findPPair(self, self_value, labels):
        if self_value is None:
            a = self.values[:]
        else:
            a = [v for v in self.values if v.v == self_value]

        for label, value in labels:
            a = [v for v in a if v.contains_parent(label, value)]

        return a

    # Generates a lw sample by chosing a random value
    # all parents must be chosen already in A
    def generate_lw_sample(self, A):
        # all parents nodes must be given!
        for p in self.parents:
            if len([True for l, v in A if l == p.label]) == 0:
```

```
                print("Parent node not set: %s" % A)

        a = self.findPPair(None, A)

        # this must be a set of choices, so the total probability
            ↪ must be one
        assert(sum([p.p for p in a]) == 1)

        # chose a random sample based on the probabilities
        rn = random.random()
        p = 0
        for sa in a:
            p += sa.p
            if p >= rn:
                return sa


        raise Exception("No valid sample created")



if __name__=="__main__":
    # create the Bayes net
    net = Net()
    gender = Node("gender", [PPair("male", 0.6), PPair("female", 0.4)
        ↪ ], [])
    country = Node("country", [PPair("england", 0.8), PPair("scottland
        ↪ ", 0.2)], [])
    outsize = Node("outsize", [PPair("yes", 0.16, [("gender", "male"),
        ↪  ("country", "scottland")]),
                           PPair("no", 0.84, [("gender", "male"), ("
                               ↪ country", "scottland")]),
                           PPair("yes", 0.03, [("gender", "female"), ("
                               ↪ country", "scottland")]),
                           PPair("no", 0.97, [("gender", "female"), ("
                               ↪ country", "scottland")]),
                           PPair("yes", 0.50, [("gender", "male"), ("
                               ↪ country", "england")]),
                           PPair("no", 0.50, [("gender", "male"), ("
                               ↪ country", "england")]),
                           PPair("yes", 0.06, [("gender", "female"), ("
                               ↪ country", "england")]),
                           PPair("no", 0.94, [("gender", "female"), ("
                               ↪ country", "england")])],
                           [gender, country])

    skirt = Node("skirt", [PPair("yes", 0.05, [("gender", "male"), ("
```

```
        ↪ country", "scottland")]),
                        PPair("no", 0.95, [("gender", "male"), ("
                            ↪ country", "scottland")]),
                        PPair("yes", 0.30, [("gender", "female"), ("
                            ↪ country", "scottland")]),
                        PPair("no", 0.70, [("gender", "female"), ("
                            ↪ country", "scottland")]),
                        PPair("yes", 0.005, [("gender", "male"), ("
                            ↪ country", "england")]),
                        PPair("no", 0.995, [("gender", "male"), ("
                            ↪ country", "england")]),
                        PPair("yes", 0.40, [("gender", "female"), ("
                            ↪ country", "england")]),
                        PPair("no", 0.60, [("gender", "female"), ("
                            ↪ country", "england")])],
                        [gender, country])


    net.add(gender)
    net.add(country)
    net.add(outsize)
    net.add(skirt)

    # Print a simple probability (all entries given!)
    print(net.P([("gender", "male"),("country","england"),("skirt","
        ↪ yes"),("outsize","yes")]))

    # do 10 runs of lw
    for r in range(10):
        net.reset_samples()
        # keep track of p to plot it!
        phistory = []

        # the number of samples per run
        for t in range(5000):
            # generate a single sample, returns the current p
            p = net.generate_lw_sample([("skirt", "yes")], [("outsize",
                ↪  "yes")])
            phistory.append(p)

        # print probability and plot
        print(net.constrained / net.total)
        plt.plot(phistory)

    # show all plots
    plt.show()
```

Problem 4: Part-of-speech

One of the many practical application of (dynamic) Bayes networks is Part-of-Speech tagging, i.e. assigning parts of speech to each word in a text. Similar to the umbrella example from the lecture, Part-of-Speech taggers model words as evidence variables and the corresponding part-of-speech tag as hidden states.

In the following example we assume a very simplified tag-set given in Tab. 4).

Abbreviation	Description	Examples
DET	determiner	the, a
N	noun	year, home, costs, time, education
PRO	pronoun	he, their, her, its, my, I, us
V	verb	said, took, told, made, asked

Table 4: Simplified tag-set.

For instance in the case of "I like you" a possible assignment of variables of the net in Fig. 2 would be given by $t_1 = \text{PRO}$, $w_1 = \text{I}$, $t_2 = \text{V}$, $w_2 = \text{like}$, $t_3 = \text{PRO}$ and $w_3 = \text{you}$. Every t_i is a part-of-speech tag and every w_i is a word.

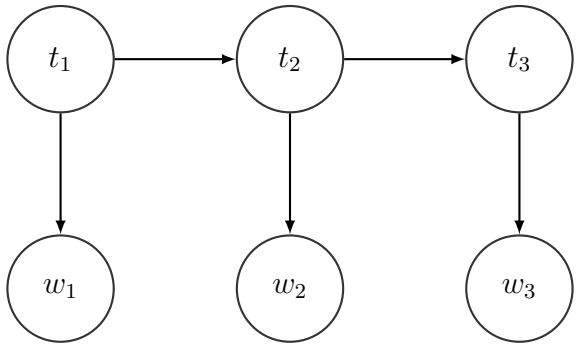


Figure 2: Beispiel zum Part-of-Speech Modell

The part-of-speech transition probabilities $P(t_i|t_{i-1})$ are given in Tab. 5 and an excerpt of word emission probabilities $P(w_i|t_i)$ is given in Tab. 6.

from/to	DET	N	PRO	V
DET	0,05	0,90	0,05	0
N	0,10	0,70	0,05	0,15
PRO	0,05	0,50	0,05	0,40
V	0,50	0,10	0,40	0

Table 5: Part-of-speech transition probabilities

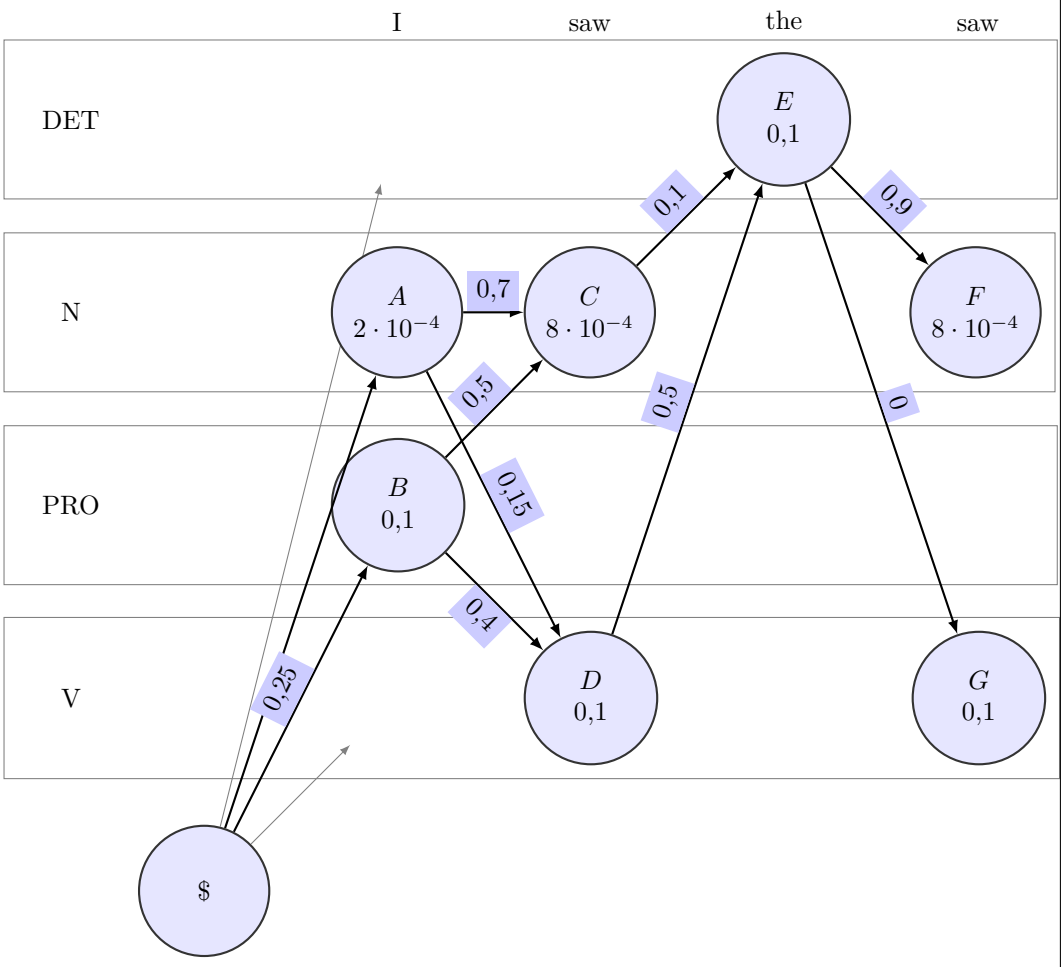
For simplicity, for the beginning of the sentence (first tag) you can assume a uniform distribution of tags.

(a) Compute the most likely sequence of tags of the sentence: "I saw the saw". Use the Viterbi algorithm.

w/t	DET	N	PRO	V
I	0	$2 \cdot 10^{-4}$	0,1	0
saw	0	$8 \cdot 10^{-4}$	0	0,1
the	0,1	0	0	0

Table 6: Excerpt of word emission probabilities

Lösung: The following figure shows an overview of word emission and word transition probabilities starting with the beginning of the sentence "$". Shown are only relevent paths, i.e. ones with probability greater than zero.



Let $\pi_{t_i}^{t_{i-1}}$ be the highest probability of a path to state t_i from its predecessor t_{i-1} under evidence $w_{1:i}$ at position i in the sequence. The best value across all possible predecessors shall be π_{t_i} with evidence until position i.

For the first positions A and B it is:

$$\pi_A = \pi_A^{\$} = 0{,}25 \cdot \left(2 \cdot 10^{-4}\right) = 5 \cdot 10^{-5} \tag{1}$$

$$\pi_B = \pi_B^{\$} = 0{,}25 \cdot 0{,}1 = 0{,}025 \tag{2}$$

Hence, we get for position C

$$\begin{aligned}\pi_C^A &= \pi_A \cdot 0{,}7 \cdot \left(8 \cdot 10^{-4}\right) = 2{,}8 \cdot 10^{-8} \\ \pi_C^B &= \pi_B \cdot 0{,}5 \cdot \left(8 \cdot 10^{-4}\right) = 1 \cdot 10^{-5} \\ \pi_C^B > \pi_C^A &\Rightarrow \pi_C \doteq \pi_C^B \end{aligned} \tag{3}$$

and respectively for position D

$$\begin{aligned}\pi_D^A &= \pi_A \cdot 0{,}15 \cdot 0{,}1 = 7{,}5 \cdot 10^{-7} \\ \pi_D^B &= \pi_B \cdot 0{,}4 \cdot 0{,}1 = 1 \cdot 10^{-3} \\ \pi_D^B > \pi_D^A &\Rightarrow \pi_D \doteq \pi_D^B \end{aligned} \tag{4}$$

From this we can cdetermine the third position E:

$$\begin{aligned}\pi_E^C &= \pi_C \cdot 0{,}1 \cdot 0{,}1 = 1 \cdot 10^{-7} \\ \pi_E^D &= \pi_D \cdot 0{,}5 \cdot 0{,}1 = 5 \cdot 10^{-5} \\ \pi_E^D > \pi_E^C &\Rightarrow \pi_E \doteq \pi_E^D \end{aligned} \tag{5}$$

The final positions F and G result in

$$\begin{aligned}\pi_F = \pi_F^E &= \pi_E \cdot 0{,}9 \cdot \left(8 \cdot 10^{-4}\right) = 3{,}6 \cdot 10^{-8} \\ \pi_G = \pi_G^E &= \pi_E \cdot 0 \cdot 0{,}1 = 0 \end{aligned}$$

From all computations after reading through the path we get the full sequence:

PRO	V	DET	N
I	saw	the	saw

In addition to all maximum probabilities for every word-part-of-speech-pair, the algorithm saves the corresponding predecessors.

(b) What factors influence the complexity of the computation and how are they connected?

Lösung: The complexity of computation is dependent on both the length of the sequence and the number of possible hidden states. The computation is *linearly* dependent on the *length* N since one run goes through all hidden states of a sequence. On the other hand, the computations for *transition* for all possible assignments from one latent state to its predecessors leads to *quadratic* cost concerning the *number of possible states of an unobserved variable* T:

$$\mathcal{O}(N \cdot T^2) \tag{6}$$