*** Applied Machine Learning Fundamentals ***
Probabilistic Graphical Models

M. Sc. Daniel Wehner

SAP SE

Winter term 2019/2020

Find all slides on GitHub

Lecture Overview

Out of scope for this lecture!

Agenda for this Unit

Section:

Basic Statistics

DHBW
Duale Hochschule
Baden-Württemberg

Random Variables and Probability Distributions

- What is a random variable X?
  A random number whose value is subject to variations due to chance
- What is a distribution $p(X = x_i)$?
  Describes the probability (density) that the random variable X will be equal to a certain value x_i
- What is a joint, a conditional and a marginal distribution?
  $$\underbrace{p(X, Y)}_{\textbf{joint}} = \underbrace{p(Y|X)}_{\textbf{cond.}} \cdot \underbrace{p(X)}_{\textbf{marg.}}$$

Important Probability Rules

- Bayes' rule

$$p(X|Y) = \frac{p(Y|X) \cdot p(X)}{p(Y)} \tag{1}$$

- Chain rule of probabilities

$$p(W, X, Y, Z) = p(W|X, Y, Z) \cdot p(X|Y, Z) \cdot p(Y|Z) \cdot p(Z) \tag{2}$$

- Definition of conditional probability

$$p(X|Y) = \frac{p(X, Y)}{p(Y)} \tag{3}$$

Section:

Bayesian Networks (BNs)

DHBW
Duale Hochschule
Baden-Württemberg

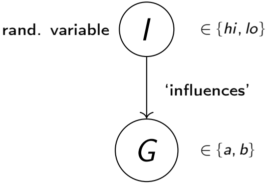
Representing Distributions by Enumeration

- Consider a probability distribution $p(X)$
  - Assign a probability to each $x_i \in Dom(X)$
  - Q: How many parameters do we have? (assuming $|Dom(X)| = k$)
  - A: $k - 1$ (Remember: $\sum_{x_i \in Dom(X)} p(x_i) = 1$)
- Now consider $p(X_1, X_2, \ldots, X_n)$
  - Q: How many parameters do we have now?
  - A: $k^n - 1$ **(Exponentially many!)**

Bayesian networks often need much fewer parameters. Why?

Simple Bayesian Network (2 Nodes)

- Let's first consider a simple BN
- Grade G is influenced by intelligence I



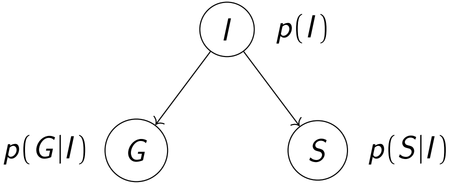
	$I = hi$	$I = lo$
$p(I)$	0.85	0.15

	G	$I = hi$	$I = lo$
$p(G\|I)$	a	0.90	0.50
	b	0.10	0.50

$$p(G = b, I = h) \stackrel{CR}{=} p(G = b|I = hi) \cdot p(I = hi)$$
$$= 0.85 \cdot 0.1 = \mathbf{0.085}$$

What if Variables are independent?

- Random variables: Intelligence I, Grade G, SAT score S



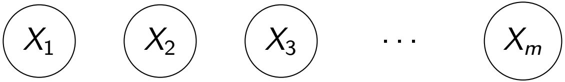
- G and S are influenced by I
- **But:** G is independent of S given I: $G \perp\!\!\!\perp S | I$
- **Independencies can lead to a smaller number of parameters**

Can we get linear Complexity?

- **Yes we can!**
- But we must assume $(\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}) \ \forall \ \boldsymbol{X}, \boldsymbol{Y}$ subsets of $\{X_1, X_2, \dots, X_m\}$
- The joint probability distribution can be written as:

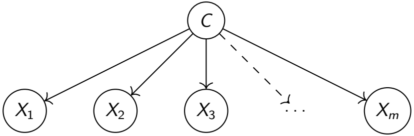
$$p(X_1, X_2, \dots, X_n) = \prod_{j=1}^{m} p(X_j) \tag{4}$$

- Q: How many parameters do we have? A: $m \cdot (k-1) = \mathcal{O}(m)$



Naïve Bayes

- This leads to the Naïve Bayes model
- Class variable C, evidence variables $\{X_1, X_2, \ldots, X_m\}$
- Assume: $(\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} | C)\ \forall\ \boldsymbol{X}, \boldsymbol{Y}$ subsets of $\{X_1, X_2, \ldots, X_m\}$



$$p(X_1, X_2, \ldots, X_m, C) = p(C) \cdot \prod_{j=1}^{m} p(X_j | C) \tag{5}$$

$\Rightarrow$ cf. slides 'Decision Theory'

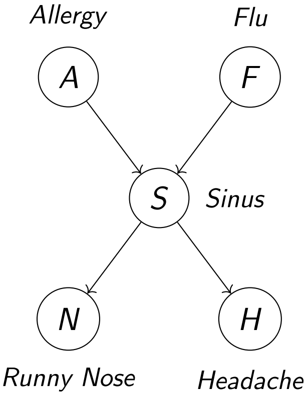
Local Markov Assumption

- How to read off the independencies from a BN?
- **Local Markov assumption**: A variable is independent of its non-descendants given its parents and only its parents:

$$(X_j \perp\!\!\!\perp \underbrace{NonDescendants(X_j)}_{ND(X_j)} \mid \underbrace{Parents(X_j)}_{Pa(X_j)}) \ \forall\, j = 1, 2, \ldots, m \tag{6}$$

$\Rightarrow$ cf. examples on the next slide

Example: Local Markov Assumption

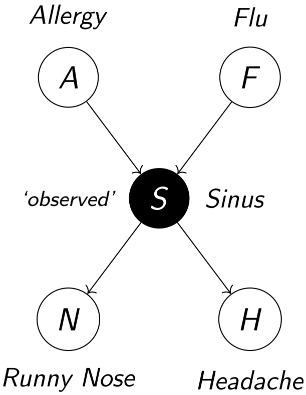


$$Pa(F) = \emptyset$$
$$ND(F) = \{A\}$$
$$Independencies \Rightarrow (F \perp\!\!\!\perp A)$$

$$Pa(N) = \{S\}$$
$$ND(N) = \{F, A, H\}$$
$$Independencies \Rightarrow (N \perp\!\!\!\perp \{F, A, H\} | S)$$

$$Pa(S) = \{F, A\}$$
$$ND(S) = \emptyset$$
$$Independencies \Rightarrow \text{ none}$$

Explaining away / Berkson's Paradox



- Two causes (A, F) 'compete' to explain the observed data (S)
- **Having a flu makes it less likely to have an allergy**
- It follows: $\neg(F \perp\!\!\!\perp A | S)$, although $F \perp\!\!\!\perp A$ (!!!)
- This is **not** implied by the local Markov assumption (S is descendant not parent!)

Joint Distribution

- According to the **chain rule** the joint probability distribution $P(A, F, S, H, N)$ is given by:

$$p(A, F, S, H, N) = p(F) \cdot p(A|F) \cdot p(S|F, A) \cdot p(H|S, F, A) \cdot p(N|S, F, A, H)$$

- Apply independency assumptions (**local Markov assumption**):

$$p(A, F, S, H, N) = p(F) \cdot p(A) \cdot p(S|F, A) \cdot p(H|S) \cdot p(N|S)$$

Much less parameters due to the local Markov assumption!

Definition of a Bayesian Network

- A BN is a **directed acyclic graph (DAG)**
  - Nodes represent random variables $\{X_1, X_2, \ldots, X_m\}$
  - Edges represent the dependencies between the random variables
- Due to the **local Markov assumption** the joint probability distribution factorizes according to:

$$p(X_1, X_2, \ldots, X_m) = \prod_{j=1}^{m} p(X_j | Pa(X_j)) \tag{7}$$

Independencies in real Problems

Real world

The true distribution P contains independency assertions $I(P)$

Model

The graph $\mathcal{G}$ encodes local independency assumptions $I_{LM}(\mathcal{G})$

Representation Theorem

- **Key representational assumption:** $I_{LM}(\mathcal{G}) \subseteq I(P)$
- We say: Graph $\mathcal{G}$ is an **I-Map (independency map)** for distribution P
- **Representation theorem**:

Conditional independencies encoded in BN are subset of conditional independencies in P

$$\Leftrightarrow$$

Joint probability distribution factorizes according to BN definition

$$I_{LM}(\mathcal{G}) \subseteq I(P) \Leftrightarrow P(X_1, X_2, \ldots, X_m) = \prod_{j=1}^{m} P(X_j | Pa(X_j)) \tag{8}$$

Independencies encoded in a BN

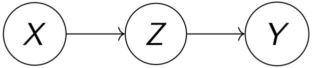
- To get the independencies, all you need is the **local Markov assumption**
- But there are more... Consider the following BN:

$$A \rightarrow B \rightarrow C \rightarrow D$$

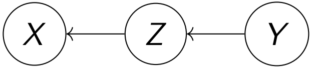
- By local Markov assumption: $D \perp\!\!\!\perp \{A, B\} | C$
- But we also have $D \perp\!\!\!\perp A | C$ and $D \perp\!\!\!\perp B | C$ (not covered by local Markov assumption)
- This leads us to the concept of d-separation (dependency separation)

d-Separation

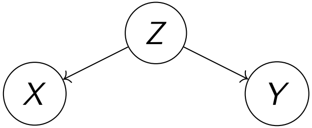
1 Indirect causal effect



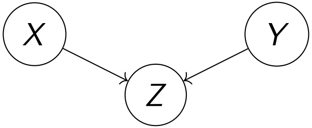
2 Indirect evidential effect



3 Common cause



4 Common effect (v-structure)



d-Separation (Ctd.)

- For patterns 1, 2 and 3 it holds:

$$X \perp\!\!\!\perp Y | Z$$

$$\neg(X \perp\!\!\!\perp Y)$$

1 indirect causal effect
2 indirect evidential effect
3 common cause
4 common effect

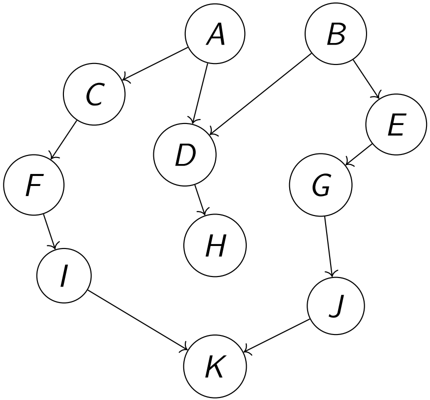
- Pattern 4 is different (inverted):

$$X \perp\!\!\!\perp Y$$

$$\neg(X \perp\!\!\!\perp Y | Z)$$

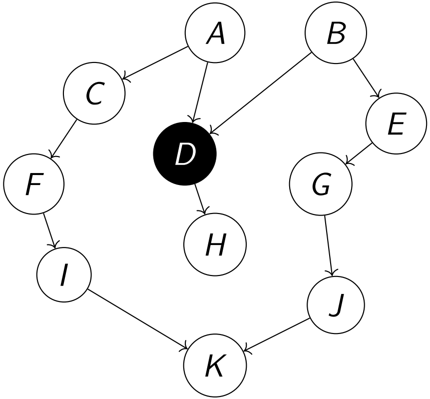
There is an **active trail** between X and Y, if X and Y are **dependent**.

d-Separation Example



- $F \perp\!\!\!\perp G$???
- Have a look at all consecutive triplets.
  - $F - I - K$: Active
  - $I - K - J$: Inactive (v-structure)
  - $K - J - G$: Active
  - $\Rightarrow$ This trail is not active
- Do the same with the other path (it's also inactive)
- We have $F \perp\!\!\!\perp G$

d-Separation Example II



- $F \perp\!\!\!\perp G | D$???
  - $F - C - A$: Active
  - $C - A - D$: Active
  - $A - D - B$: Active (v-structure, but D is observed)
  - $D - B - E$: Active
  - $B - E - G$: Active

  $\Rightarrow$ This trail is active! Information can flow!
- We have $\neg(F \perp\!\!\!\perp G | D)$

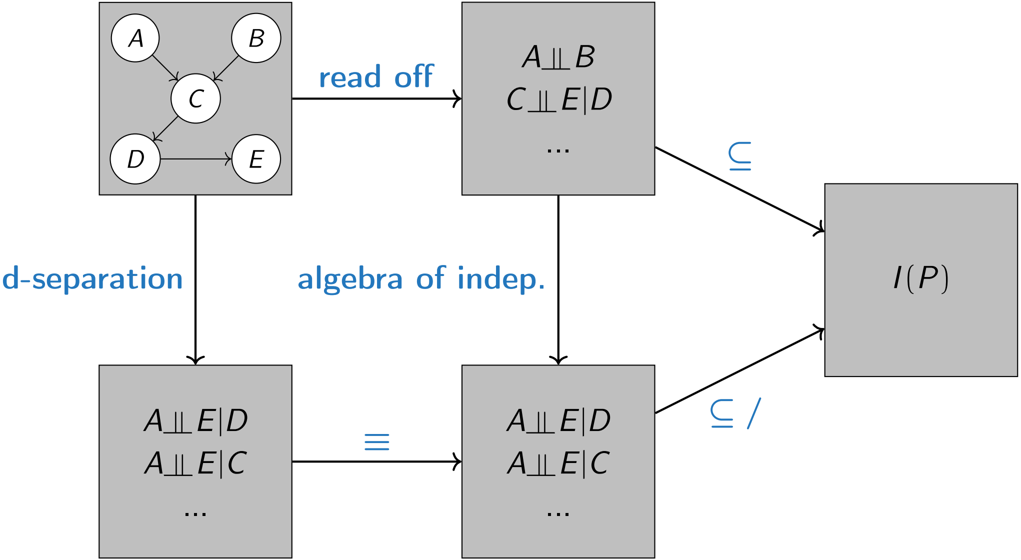
Soundness of d-Separation

Soundness

If P factorizes according to $\mathcal{G}$, then $I(\mathcal{G}) \subseteq I(P)$ and not only $I_{LM}(\mathcal{G}) \subseteq I(P)$

Completeness

- For 'almost all' distributions for which P factorizes according to $\mathcal{G}$, we have that $I(\mathcal{G}) = I(P)$
- This means P is **faithful**
- A faithful distribution does **not declare extra independence assumptions** that **cannot be read off** from $\mathcal{G}$


A
B
C
D
E
read off
$A \perp\!\!\!\perp B$
$C \perp\!\!\!\perp E \mid D$
…
$\subseteq$
$I(P)$
d-separation
algebra of indep.
$A \perp\!\!\!\perp E \mid D$
$A \perp\!\!\!\perp E \mid C$
…
$\equiv$
$A \perp\!\!\!\perp E \mid D$
$A \perp\!\!\!\perp E \mid C$
…
$\not\subseteq$

Inference in Bayesian Networks

- We want to use the Bayesian network to compute the probability of a query
- **Bad news:** In general, inference in Bayesian networks is hopeless

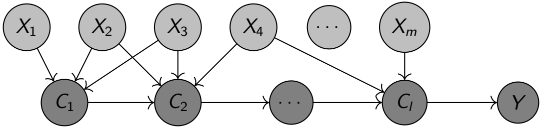
Theorem:
Inference in Bayesian networks (even approximate) is **NP-hard**

- However, in practice we can exploit the structure of the network
- There are some effective approximation algorithms
- Let us first talk about exact inference

Complexity of Inference

- Consider a reduction to 3-SAT (known to be NP-hard)
- We have m boolean variables. **Does a satisfying assignment exist?**

$$\underbrace{(\neg X_1 \vee X_2 \vee X_3)}_{C_1} \wedge \underbrace{(\neg X_2 \vee X_3 \vee \neg X_4)}_{C_2} \wedge \underbrace{(\dots)}_{C_l} \tag{9}$$



2^m possible assignments

$$p(Y=1) = \frac{\#\text{ sat assignments}}{2^m}$$

- **This problem is in #P (!!!)**

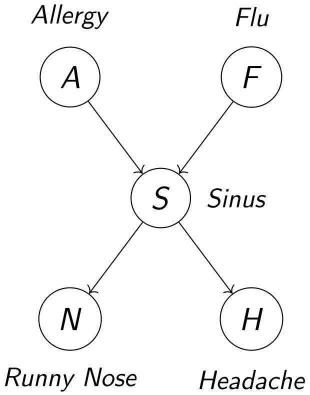
Exact Inference

- Back to our flu example
- Suppose we have a conditional probability query:

$$p(A = t | N = t)$$

- Rewrite using the definition of conditional probability:

$$p(A = t | N = t) = \frac{p(A = t, N = t)}{p(N = t)}$$



Exact Inference (Ctd.)

- We know what $p(A, F, S, N, H)$ is:

$$p(A, F, S, N, H) = p(A) \cdot p(F) \cdot p(S|A, F) \cdot p(N|S) \cdot p(H|S)$$

- In order to compute $p(A = t, N = t)$ we have to **marginalize (sum out)** all the other variables:

$$p(A = t, N = t) = \sum_{f \in F} \sum_{s \in S} \sum_{h \in H} p(A = t) \cdot p(F) \cdot p(S|A = t, F) \cdot p(N = t|S) \cdot p(H|S)$$

- Do the same for $p(N = t)$ and compute $p(A = t|N = t)$
- This algorithm is called **variable elimination**

Variable Elimination

Have: $p(A) \cdot p(F) \cdot p(S|A,F) \cdot p(N|S) \cdot p(H|S)$; **Want:** $p(H)$

Assume: Elimination order: A, F, N, S

Eliminate A:	$\varphi_A(F,S) = \sum_{a \in A} p(a) \cdot p(S|a,F)$	$\Rightarrow \varphi_A(F,S) \cdot p(F) \cdot p(N|S) \cdot p(H|S)$
Eliminate F:	$\varphi_F(S) = \sum_{f \in F} \varphi_A(f,S) \cdot p(f)$	$\Rightarrow \varphi_F(S) \cdot p(N|S) \cdot p(H|S)$
Eliminate N:	$\varphi_N(S) = \sum_{n \in N} p(n|S)$	$\Rightarrow \varphi_F(S) \cdot \varphi_N(S) \cdot p(H|S)$
Eliminate S:	$\varphi_S(H) = \sum_{s \in S} \varphi_F(s) \cdot \varphi_N(s) p(H|s)$	$\Rightarrow \boxed{\varphi_S(H)}$

Insight: Exact inference seems to be exponential in the number of variables!

Algorithm 1: Variable Elimination Algorithm

Input: Bayesian network BN, query $p(\boldsymbol{X}|\boldsymbol{O})$

instantiate evidence $\boldsymbol{O}$

prune non-active variables for $\{\boldsymbol{X}, \boldsymbol{O}\}$

choose an ordering on the variables $\{X_1, X_2, \ldots, X_m\}$

initialize factors $\{\varphi_1, \varphi_2, \ldots, \varphi_m\} : \varphi_j = p(X_j|Pa(X_j))$

foreach $j \in \{1, 2, \ldots, m\}$ **do**

  if $X_j \notin \{\boldsymbol{X}, \boldsymbol{E}\}$ **then**

    `// marginalize variable`

    remove factors $\varphi_1, \varphi_2, \ldots, \varphi_k$ that include X_j

    generate a new factor by eliminating X_j from these factors: $\psi = \sum_{X_j} \prod_{i=1}^{k} \varphi_i$

    add ψ to the set of factors

normalize probabilities

return *answer to query* $p(\boldsymbol{X}|\boldsymbol{O})$

Approximate Inference

- Since exact inference is NP-hard, let's try **approximate inference**
- Some common methods:
  - Forward sampling (without evidence)
  - Rejection sampling (with evidence)
  - Likelihood weighting
  - Gibbs sampling (MCMC – Markov Chain Monte Carlo)
- We are going to cover forward/rejection sampling and Gibbs sampling

Forward Sampling (without Evidence)

Algorithm 2: Forward Sampling without Evidence

Input: Bayesian network, # nodes m, # samples T

```
// generate number of samples specified
initialize set of samples: S ⟵ ∅
for t ∈ {1, 2, ..., T} do
    for j ∈ {1, 2, ..., m} do
        // sample value for random variable
        s_j^(t) ⟵ sampled from p(X_j | Pa(X_j))
    S ⟵ S ∪ s^(t)
return set of samples S = {s^(1), s^(2), ..., s^(T)}
```

Forward Sampling: Answering Queries

- Suppose we have collected several samples $\boldsymbol{S} = \{\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}, \dots, \boldsymbol{s}^{(T)}\}$
- **How can we do inference with them?**
- Very easy:

$$\widehat{p}(X_j = x_i) = \frac{\sum_{t=1}^{T} \mathbb{1}\{s_j^{(t)} = x_i\}}{T}$$

- $\mathbb{1}\{boolean\}$ is the **indicator function**. It returns 1 if the boolean expression is true, 0 otherwise. E. g. $\mathbb{1}\{1+1=2\} = 1$, $\mathbb{1}\{3=2\} = 0$
- **Basically, we count the number of samples for which $X_j = x_i$**
- What about evidence?

Rejection Sampling (Forward Sampling with Evidence)

- Major issue: The samples have to be consistent with the evidence
- If it is not consistent: **Reject the sample** (rejection sampling)
- **Problem:**
  - What if the evidence has low probability?
  - **Most samples will be rejected!**
  - This method is easy, but can be **very slow**

Gibbs Sampling

- So called Markov Chain Monte Carlo (MCMC) method
- Samples are **dependent** and form a **Markov chain**
- Probability estimates will **finally converge** to the true probabilities[1]
- Sampling process:
  - Fix values of evidence / observed variables $\boldsymbol{O}$
  - Initialize first sample $\boldsymbol{s}^{(0)}$ randomly
  - Generate next sample $\boldsymbol{s}^{(t+1)}$ based on the current one $\boldsymbol{s}^{(t)}$

[1] if all $p > 0$

Ordered Gibbs Sampler

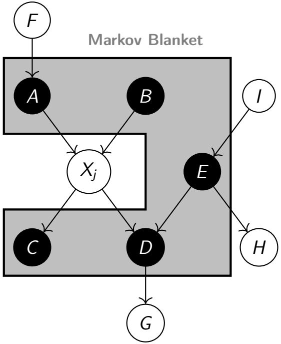
- **Main idea:** Generate next sample $\boldsymbol{s}^{(t+1)}$ based on the current one $\boldsymbol{s}^{(t)}$
- Sample variables **in order**:

$$\begin{aligned}
X_1: &\quad s_1^{(t+1)} \longleftarrow p(s_1 | s_2^{(t)}, s_3^{(t)}, \dots, s_m^{(t)}, \boldsymbol{O}) \\
X_2: &\quad s_2^{(t+1)} \longleftarrow p(s_2 | s_1^{(t+1)}, s_3^{(t)}, \dots, s_m^{(t)}, \boldsymbol{O}) \\
X_3: &\quad s_3^{(t+1)} \longleftarrow p(s_3 | s_1^{(t+1)}, s_2^{(t+1)}, \dots, s_m^{(t)}, \boldsymbol{O}) \\
\dots & \\
X_m: &\quad s_m^{(t+1)} \longleftarrow p(s_m | s_1^{(t+1)}, s_2^{(t+1)}, \dots, s_{m-1}^{(t+1)}, \boldsymbol{O})
\end{aligned}$$

- In short:

$$X_j: \quad s_j^{(t+1)} \longleftarrow p(s_j | \boldsymbol{s}^{(t)} \backslash s_j, \boldsymbol{O})$$

Markov Blanket



- We have to sample the value for X_j given all of the other variables in the network
- This is can be simplified using the **Markov blanket**

$$MB(X_j) = Pa(X_j) \cup Ch(X_j) \cup \left[\bigcup_{X_i \in Ch(X_j)} Pa(X_i) \right] \tag{10}$$

A node is independent of all other nodes in the network given its Markov blanket

Improvements of Gibbs Sampling

1. **Burn-In:** Discard first k samples, since starting point is random
2. **Reduction of dependence / auto-correlation:**
   - **Skip samples**
   - Randomize variable sampling order
3. **Reduction of variance:**
   - Sample several chains and average
   - **Blocking:** Sample variables block-wise
   - **Rao-Blackwellisation:** Only sample a subset of the variables

Learning in Bayesian Networks

- By now we know how to **represent** BNs and how to do **inference**
- **But: Where do the numbers come from?**
- Two kinds of learning:
  - **Parameter estimation:** obtain (conditional) probabilities
  - **Structure learning:** learn the structure of the network
- Why learning Bayesian networks?
  - Conditional independencies and graphical language **capture structure** of many real-world distributions
  - Graph structure provides much **insight**

Parameter Estimation

- Let's start with parameter estimation
- Let $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n)}\}$ be a set over m random variables
- We assume the data is I. I. D. (**i**ndependent and **i**dentically **d**istributed)
- Find parameters $\boldsymbol{\theta}$ of CPDs (conditional probability distributions) which match the data best

What does 'best matching' mean? Find parameters $\boldsymbol{\theta}$ which have most likely produced the data. $\Rightarrow$ **Maximum likelihood (ML)**

Maximum Likelihood Estimation

- **Recall:** In MLE we want to compute: (⇒ cf. slides 'Density estimation')

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathcal{D})$$

- By applying **Bayes' rule** we get:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P(\mathcal{D}|\boldsymbol{\theta}) \cdot \frac{\cancel{P(\boldsymbol{\theta})}}{\cancel{P(\mathcal{D})}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{D}|\boldsymbol{\theta})$$

- All parameters are apriori equally likely
- Data is equally likely for all parameters

Maximum Likelihood Estimation (Ctd.)

- This is the likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \arg\max_{\boldsymbol{\theta}} P(\mathcal{D}|\boldsymbol{\theta})$$

- Usually, the log-likelihood $\mathcal{LL}(\boldsymbol{\theta}|\mathcal{D})$ is used:

$$\mathcal{LL}(\boldsymbol{\theta}|\mathcal{D}) = \log P(\mathcal{D}|\boldsymbol{\theta})$$

- ML is one of the **most commonly used estimators** in statistics
- Its estimates converge to the best possible value as the number of examples grows

Decomposability of the Likelihood

$$
\begin{aligned}
\mathcal{LL}(\boldsymbol{\theta}|\mathcal{D}) &= \log p(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n)}|\boldsymbol{\theta}) \\
&\overset{(1)}{=} \log \prod_{i=1}^{n} p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta}) \\
&\overset{(2)}{=} \sum_{i=1}^{n} \log p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \dots, \boldsymbol{x}_i^{(m)}|\boldsymbol{\theta}) \\
&\overset{(3)}{=} \sum_{i=1}^{n} \log \left(\prod_{j=1}^{m} p(\boldsymbol{x}_i^{(j)}|Pa(\boldsymbol{x}_i^{(j)}), \boldsymbol{\theta}) \right) \overset{(2)}{=} \sum_{i=1}^{n} \sum_{j=1}^{m} \log p(\boldsymbol{x}_i^{(j)}|Pa(\boldsymbol{x}_i^{(j)}), \boldsymbol{\theta}) \\
&= \sum_{j=1}^{m} \sum_{i=1}^{n} \log p(\boldsymbol{x}_i^{(j)}|Pa(\boldsymbol{x}_i^{(j)}), \boldsymbol{\theta}_j) = \sum_{j=1}^{m} \mathcal{LL}(\boldsymbol{\theta}_j|\mathcal{D})
\end{aligned}
$$

(1) I. I. D.

(2) $\log \prod = \sum \log$

(3) Bayesian network semantics

Decomposability of the Likelihood (Ctd.)

- If the data set is **fully observed**[2]...
  - ...we can maximize each local likelihood function **independently**...
  - ...and then combine the solutions to get the global solution
- Decomposability allows us to come up with **efficient solutions** to the MLE problem

But: What does the likelihood function look like?

[2] missing data case: see later slides

Likelihood for Multinomials

- Assume a random variable V which can take $1, 2, \ldots, K$ values

$$p(V = k) = \theta_k \qquad \sum_{k=1}^{K} \theta_k = 1$$

- The **(log-)likelihood** is given by: (n_k is # of times event k occurs)

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \prod_{k=1}^{K} \theta_k^{n_k} \qquad \mathcal{LL}(\boldsymbol{\theta}_v|\mathcal{D}) = \sum_{k=1}^{K} \log \theta_k^{n_k} = \sum_{k=1}^{K} n_k \cdot \log \theta_k$$

- E. g. tossing an unfair coin: $\mathit{Events} = \{\mathtt{Head}, \mathtt{Tail}\}$; $P(\mathtt{H}) = 1/4$, $P(\mathtt{T}) = 3/4$
- $P(\mathtt{H}, \mathtt{T}, \mathtt{H}, \mathtt{H}, \mathtt{T}) = 1/4^3 \cdot 3/4^2$

Maximum Likelihood for Multinomials

- In order to get the maximum likelihood, we first have to compute the partial derivatives:[3]

$$\begin{aligned} \frac{\partial}{\partial \theta_i} \mathcal{LL}(\boldsymbol{\theta}_v | \mathcal{D}) &= \frac{\partial}{\partial \theta_i} (n_1 \log \theta_1 + n_2 \log(1 - \theta_1)) \\ &= \frac{n_1}{\theta_1} + \frac{n_2}{1 - \theta_1} \end{aligned}$$

- And set them to zero:

$$\frac{\partial}{\partial \theta_i} \mathcal{LL}(\boldsymbol{\theta}_v | \mathcal{D}) \overset{!}{=} 0 \Leftrightarrow \frac{n_1}{\theta_1} + \frac{n_2}{1 - \theta_1} \overset{!}{=} 0 \Rightarrow \boxed{\theta_1^* = \frac{n_1}{n_1 + n_2}}$$

[3] consider a binomial (special case with two events only)

Maximum Likelihood for Multinomials

- This easily generalizes to more than two events:

$$\theta_i^* = \frac{n_i}{\sum_j n_j}$$

- And to conditional multinomials as well:

$$\theta_{i|pa}^* = \frac{n_{i,pa}}{n_{pa}}$$

- **It's really simple. Let's make an example...**

Maximum Likelihood: Flu Example

A	F	S	N	H
0	1	0	1	1
1	0	0	0	0
1	0	1	0	1
1	1	1	1	0
0	0	1	1	0
0	0	0	1	1
1	0	0	0	0
0	1	0	1	1
1	1	0	0	0
1	0	1	0	1
1	1	1	1	1
1	1	0	1	0
1	0	1	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	0	0

Let's compute some (marginal | cond.) probabilities:

$$p(A=0) = \frac{5}{5+11} = {}^{5}\!/_{16}$$

$$p(A=1) = 1 - p(A=0) = {}^{11}\!/_{16}$$

$$p(F=0|A=1) = {}^{6}\!/_{11}$$

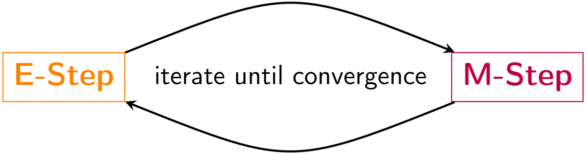
$$p(F=1|A=1) = 1 - p(F=0|A=1) = {}^{5}\!/_{11}$$

$$p(H=0|A=1, F=1) = {}^{4}\!/_{5}$$

$$\ldots$$

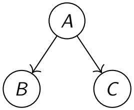
What about missing Values?

- But how can we handle **missing values**?
- In this case we can use the **Expectation-Maximization (EM) algorithm**



- The algorithm consists of two steps:
  - **Expectation:** Compute pseudo-counts
  - **Maximization:** Update parameters based on pseudo-counts

Expectation-Maximization Example



case	A	B	C
1	0	0	1
2	0	1	?
3	1	0	0
4	1	?	1
5	0	1	0
6	1	?	0
7	1	0	1
8	?	0	1
9	0	1	?
10	?	0	1

To do...

A	B	C	PC
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

EM-Algorithm for the incomplete Data Case

Algorithm 3: Expectation-Maximization Algorithm

```
initialize parameters θ
while not converged do
    compute pseudo counts
    set parameters to the maximum likelihood estimates
return final parameters θ
```

Caution: Depending on the initialization, the algorithm can **get stuck in local optima!** (Multiple runs?)

Structure Learning

- To do...

Section:

Hidden Markov Models (HMMs)

Duale Hochschule Baden-Württemberg

What is a hidden Markov Model?

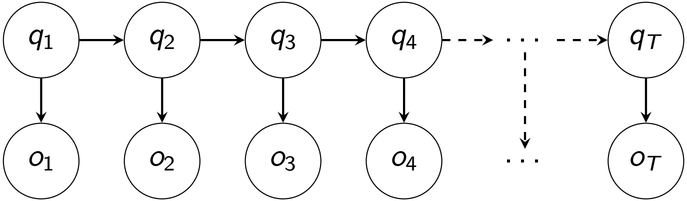
- **Motivation:** Consider e. g. the task of part-of-speech tagging
- **Problem:** Labels cannot be assigned by looking only at single words
- Polysemy: The same word can have different meanings, e. g. can, bank
- A hidden Markov model (HMM) is a **sequence classifier** and as such able to take the context of a word into account

Part-of-speech (POS) tagging is the task of assigning **part-of-speech tags** (`NN` – nouns, `VB` – verbs, etc.) to a **set of given words**.

What does an HMM look like?

(hidden) states

observations



Decoding in Hidden Markov Models

Decoding: Given as input an HMM with parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B})$ and a sequence of observations $\boldsymbol{o} = o_1, o_2, \dots, o_T$, find the most probable sequence of (hidden) states $\boldsymbol{q} = q_1, q_2, \dots, q_T$.

- Most probable state sequence: $\widehat{\boldsymbol{q}} = \arg\max_{\boldsymbol{q}} P(\boldsymbol{q}|\boldsymbol{o})$
- This equation is hard to compute. Let's apply **Bayes' rule**:

$$\widehat{\boldsymbol{q}} = \arg\max_{\boldsymbol{q}} \frac{P(\boldsymbol{o}|\boldsymbol{q}) \cdot P(\boldsymbol{q})}{P(\boldsymbol{o})} \propto \arg\max_{\boldsymbol{q}} \underbrace{P(\boldsymbol{o}|\boldsymbol{q})}_{\text{likelihood}} \cdot \underbrace{P(\boldsymbol{q})}_{\text{prior}} \tag{11}$$

Two important Assumptions

- It's still hard to compute :-(
- Hidden Markov models make **two simplifying assumptions**:

Assumption 1: The probability of an observation depends only on its own hidden state: $P(\boldsymbol{o}|\boldsymbol{q}) \approx \prod_{1=1}^{T} P(o_i|q_i)$

Assumption 2: The probability of a state appearing is dependent only on the previous state: $P(\boldsymbol{q}) \approx \prod_{1=1}^{T} P(q_i|q_{i-1})$
$\Rightarrow$ **Markov Assumption** ('the future is independent of the past given the present.')

The underlying Model

- Putting everything together, we get the hidden Markov model:

$$\widehat{\boldsymbol{q}} = \arg\max_{\boldsymbol{q}} P(\boldsymbol{q}|\boldsymbol{o}) \propto \arg\max_{\boldsymbol{q}} \prod_{i=1}^{T} P(o_i|q_i) \cdot P(q_i|q_{i-1}) \tag{12}$$

- This equation contains two types of probabilities:
  - **Transition probabilities:** $P(q_i|q_{i-1})$
  - **Emission probabilities:** $P(o_i|q_i)$

Example POS Tagging

$P(t_i \mid t_{i-1})$	VB	TO	NN	PPSS
<s>	0.01900	0.00430	0.04100	0.06700
VB	0.00038	0.03500	0.04700	0.00700
TO	0.83000	0.00000	0.00047	0.00000
NN	0.00400	0.01600	0.08700	0.00450
PPSS	0.23000	0.00079	0.00120	0.00014

$P(w_i \mid t_i)$	**I**	**want**	**to**	**race**
VB	0.00000	0.00930	0.00000	0.00012
TO	0.00000	0.00000	0.99000	0.00000
NN	0.00000	0.00005	0.00000	0.00057
PPSS	0.37000	0.00000	0.00000	0.00000

- Probabilities estimated from Brown corpus (million-word corpus of American English)
- **Transition probabilities** (first table)

$$P(q_i \mid q_{i-1}) = \frac{C(q_{i-1}, q_i)}{C(q_{i-1})} \tag{13}$$

- **Emission probabilities** (second table)

$$P(o_i \mid q_i) = \frac{C(q_i, o_i)}{C(q_i)} \tag{14}$$

Algorithm 4: Viterbi Algorithm (Dynamic Programming)

Input: $\boldsymbol{o} = o_1, o_2, \ldots, o_T$, state graph of length N

create a path probability matrix $\boldsymbol{V}[N+2, T]$

`// initialization step`

foreach *state* $q \in \{1, 2, \ldots, N\}$ **do**

  $\boldsymbol{V}[q, 1] \longleftarrow a_{0,q} \cdot b_q(o_1)$

  $trace[q, 1] \longleftarrow 0$

`// compute best path through trellis`

foreach *time step* $t \in \{2, 3, \ldots, T\}$ **do**

  foreach *state* $q \in \{1, 2, \ldots, N\}$ **do**

    $\boldsymbol{V}[q, t] \longleftarrow \max_{q'=1}^{N} \boldsymbol{V}[q', t-1] \cdot a_{q',q} \cdot b_q(o_t)$

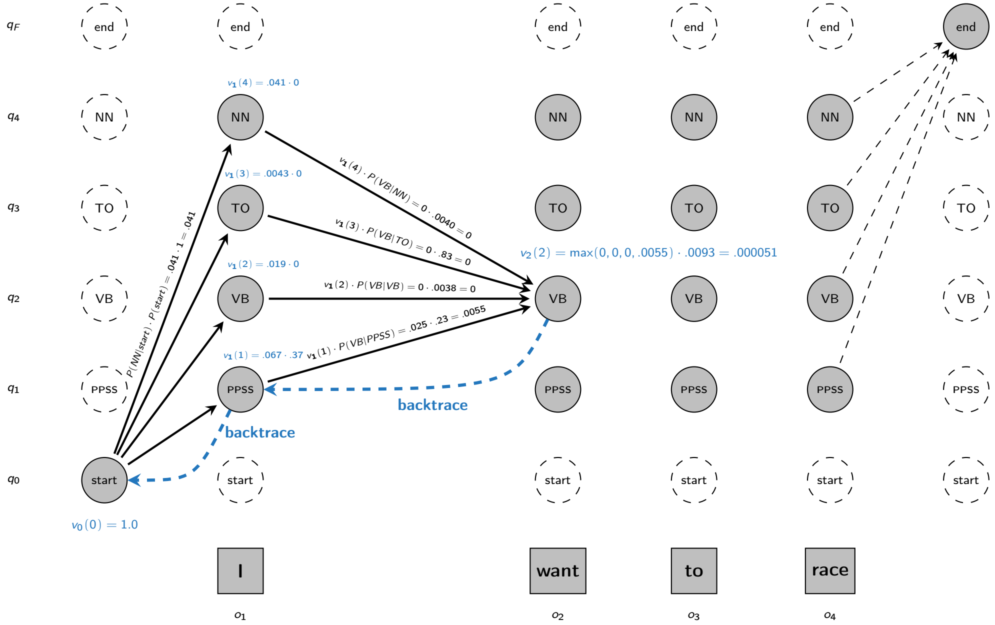
    $trace[q, t] \longleftarrow \arg\max_{q'=1}^{N} \boldsymbol{V}[q', t-1] \cdot a_{q',q}$

`// termination step`

$\boldsymbol{V}[q_F, T] \longleftarrow \max_{q=1}^{N} \boldsymbol{V}[q, T] \cdot a_{q,q_F}$

$trace[q_F, T] \longleftarrow \arg\max_{q=1}^{N} \boldsymbol{V}[q, T] \cdot a_{q,q_F}$

return *backtrace path by following the pointers back in time*



q_F
q_4
q_3
q_2
q_1
q_0
end
NN
TO
VB
PPSS
start
$v_1(4) = .041 \cdot 0$
$v_1(3) = .0043 \cdot 0$
$v_1(2) = .019 \cdot 0$
$v_1(1) = .067 \cdot .37$
$v_0(0) = 1.0$
$P(NN|start) \cdot P(start) = .041 \cdot 1 = .041$
$v_1(4) \cdot P(VB|NN) = 0 \cdot .0040 = 0$
$v_1(3) \cdot P(VB|TO) = 0 \cdot .83 = 0$
$v_1(2) \cdot P(VB|VB) = 0 \cdot .0038 = 0$
$v_1(1) \cdot P(VB|PPSS) = .025 \cdot .23 = .0055$
$v_2(2) = \max(0, 0, 0, .0055) \cdot .0093 = .000051$
backtrace
backtrace
I
want
to
race
o_1
o_2
o_3
o_4

Section:

Wrap-Up

DHBW
Duale Hochschule
Baden-Württemberg

Summary: Bayesian Networks (BNs)

1. **Representation:**
   - BNs represent exponentially large probability distributions
   - **Local Markov assumption:** Variable is independent of its non-descendants given its parents
   - **Representation theorem:**
     $I_{LM}(\mathcal{G}) \subseteq I(P) \Leftrightarrow P(X_1, X_2, \ldots, X_m) = \prod_{j=1}^{m} p(X_j | Pa(X_j))$
   - d-separation
2. **Inference:** Variable elimination algorithm, exact inference is **NP-hard**
3. **Learning:** Parameter estimation, structure learning

Summary: Hidden Markov Models (HMMs)

- An HMM is a **sequence classifier** (as such it takes the context into account)
- This is useful e. g. for part-of-speech (POS) tagging
- **Two assumptions:**
  1. Probability of an observation depends only on its own hidden state
  2. **Markov assumption:** Probability of a state appearing is dependent only on the previous state
- Find the **most probable hidden sequence** (*decoding*) by applying the **Viterbi** algorithm (dynamic programming)

Recommended Literature and further Reading I

[1] Probabilistic Graphical Models: Principles and Techniques
D. Koller, N. Friedman. The MIT Press, Cambridge, Massachusetts. 2009.
→ Click here, cf. chapters 3, 9, 16 and 17

[2] Pattern Recognition and Machine Learning
Christopher Bishop. Springer. 2006.
→ Link, cf. chapters 8 and 13

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Probabilistic Graphical Models
Term: Winter term 2019/2020

Contact:
M. Sc. Daniel Wehner
SAP SE
daniel.wehner@sap.com

Do you have any questions?