

*** Applied Machine Learning Fundamentals ***

Probabilistic Graphical Models

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2020/2021



Find all slides on [GitHub](#)

Agenda for this Unit

① Basic Statistics

Random Variables and Probability Distributions
Important Probability Rules

② Bayesian Networks (BNs)

Representation of large Probability Distributions
Answering Queries: Inference

Learning of Parameters and Structure

③ Hidden Markov Models (HMMs)

Introduction

④ Wrap-Up

Summary
Recommended Literature and further Reading

Section:
Basic Statistics



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?

$$\underbrace{p(X, Y)}_{\text{joint}} = \underbrace{p(Y|X)}_{\text{cond.}} \cdot \underbrace{p(X)}_{\text{marg.}}$$

Important Probability Rules

- Bayes' rule

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

- Chain rule of probabilities

$$p(W, X, Y, Z) = p(W|X, Y, Z) \cdot p(X|Y, Z) \cdot p(Y|Z) \cdot p(Z) \quad (2)$$

- Definition of conditional probability

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

Section:
Bayesian Networks (BNs)



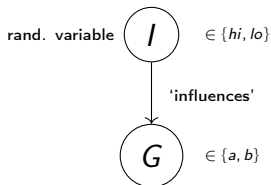
Representing Distributions by Enumeration

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



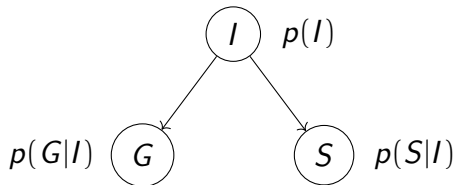
	$I = hi$	$I = lo$
$p(I)$	0.85	0.15

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

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

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 $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}) \forall \mathbf{X}, \mathbf{Y}$ subsets of $\{X_1, X_2, \dots, X_m\}$
- The joint probability distribution can be written as:

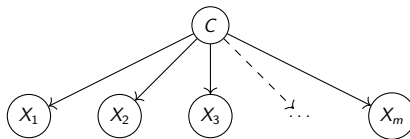
$$p(X_1, X_2, \dots, X_n) = \prod_{j=1}^m p(X_j) \quad (4)$$

- 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 $\{X_1, X_2, \dots, X_m\}$
- Assume: $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | C) \forall \mathbf{X}, \mathbf{Y}$ subsets of $\{X_1, X_2, \dots, X_m\}$



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

⇒ cf. slides 'Decision Theory'

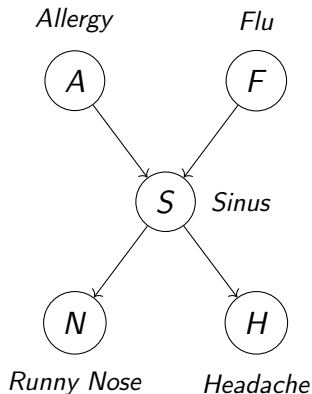
Local Markov Assumption

- 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:

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

⇒ cf. examples on the next slide

Example: Local Markov Assumption

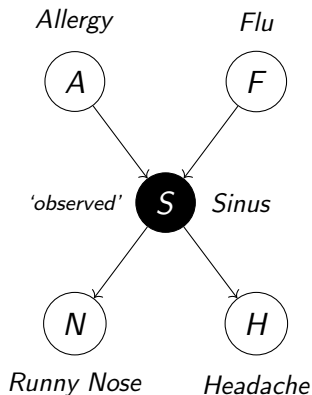


$$\begin{aligned} Pa(F) &= \emptyset \\ ND(F) &= \{A\} \\ \text{Independencies} &\Rightarrow (F \perp\!\!\!\perp A) \end{aligned}$$

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

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

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 $\{X_1, X_2, \dots, X_m\}$
 - Edges represent the dependencies between the random variables
- Due to the **local Markov assumption** the joint probability distribution factorizes according to:

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

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

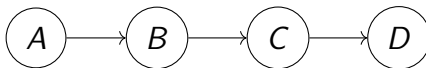
\Leftrightarrow

Joint probability distribution factorizes according to BN definition

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

Independencies encoded in a BN

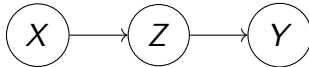
- To get the independencies, all you need is the **local Markov assumption**
- But there are more... Consider the following BN:



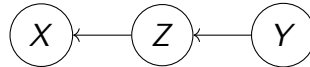
- By local Markov assumption: $D \perp\!\!\!\perp \{A, B\} | C$
- But we also have $D \perp\!\!\!\perp A | C$ and $D \perp\!\!\!\perp B | C$ (not covered by local Markov assumption)
- This leads us to the concept of **d-separation (dependency separation)**

d-Separation

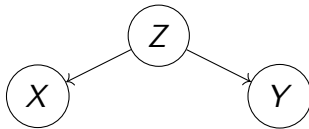
① Indirect causal effect



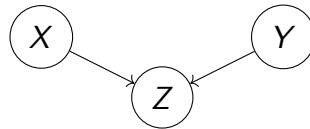
② Indirect evidential effect



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

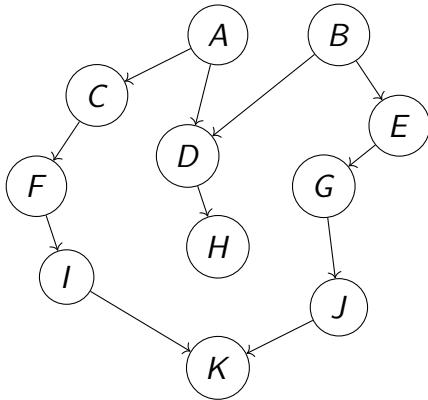
- Pattern ④ is different (inverted):

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

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

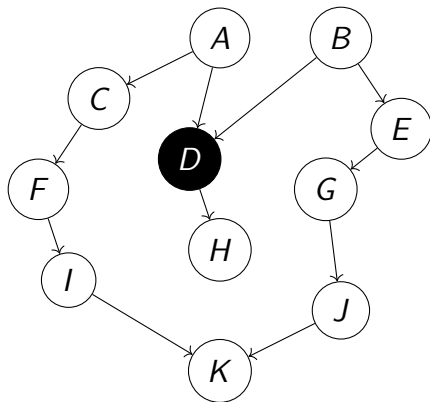
There is an **active trail** between X and Y , if X and Y are **dependent**.

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

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

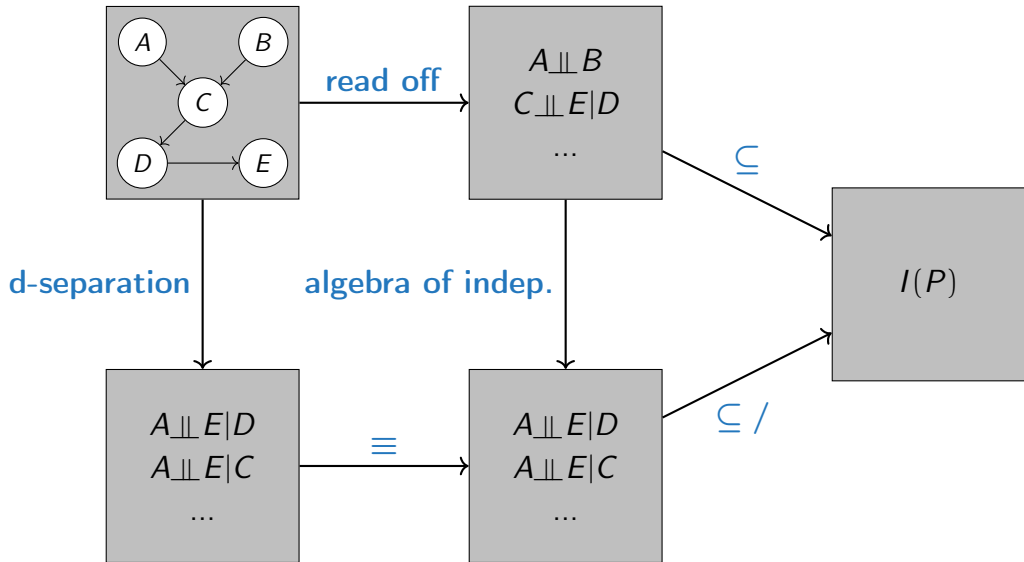
Soundness of d-Separation

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}



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:

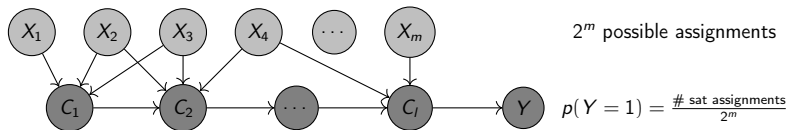
Inference in Bayesian networks (even approximate) is **NP-hard**

- 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?**

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



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

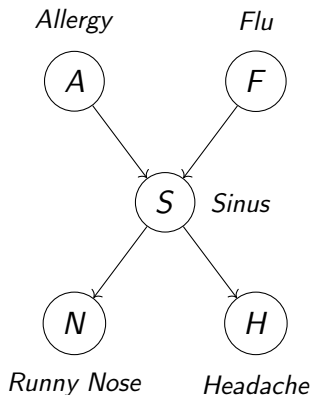
Exact Inference

- 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 $p(\mathbf{X}|\mathbf{O})$

- 1 instantiate evidence \mathbf{O}
 - 2 prune non-active variables for $\{\mathbf{X}, \mathbf{O}\}$
 - 3 choose an ordering on the variables $\{X_1, X_2, \dots, X_m\}$
 - 4 initialize factors $\{\varphi_1, \varphi_2, \dots, \varphi_m\} : \varphi_j = p(X_j | Pa(X_j))$
 - 5 **foreach** $j \in \{1, 2, \dots, m\}$ **do**
 - 6 **if** $X_j \notin \{\mathbf{X}, \mathbf{E}\}$ **then**
 - 7 // *marginalize variable*
 - 8 remove factors $\varphi_1, \varphi_2, \dots, \varphi_k$ that include X_j
 - 9 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
 - 10 normalize probabilities
 - 11 **return** *answer to query* $p(\mathbf{X}|\mathbf{O})$
-

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

```

1 initialize set of samples:  $\mathbf{S} \leftarrow \emptyset$ 
2 for  $t \in \{1, 2, \dots, T\}$  do
3     for  $j \in \{1, 2, \dots, m\}$  do
4         // sample value for random variable
5          $s_j^{(t)} \leftarrow \text{sampled from } p(X_j | Pa(X_j))$ 
6          $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{s}^{(t)}$ 
7 return set of samples  $\mathbf{S} = \{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \dots, \mathbf{s}^{(T)}\}$ 
    
```

Forward Sampling: Answering Queries

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

$$\hat{p}(X_j = x_i) = \frac{\sum_{t=1}^T \mathbb{1}\{\mathbf{s}_j^{(t)} = x_i\}}{T}$$

- $\mathbb{1}\{\text{boolean}\}$ 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 probabilities¹
- Sampling process:
 - Fix values of evidence / observed variables \mathbf{O}
 - Initialize first sample $\mathbf{s}^{(0)}$ randomly
 - Generate next sample $\mathbf{s}^{(t+1)}$ based on the current one $\mathbf{s}^{(t)}$

¹if all $p > 0$

Ordered Gibbs Sampler

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

$$X_1 : \quad s_1^{(t+1)} \leftarrow p(s_1 | s_2^{(t)}, s_3^{(t)}, \dots, s_m^{(t)}, \mathbf{O})$$

$$X_2 : \quad s_2^{(t+1)} \leftarrow p(s_2 | s_1^{(t+1)}, s_3^{(t)}, \dots, s_m^{(t)}, \mathbf{O})$$

$$X_3 : \quad s_3^{(t+1)} \leftarrow p(s_3 | s_1^{(t+1)}, s_2^{(t+1)}, \dots, s_m^{(t)}, \mathbf{O})$$

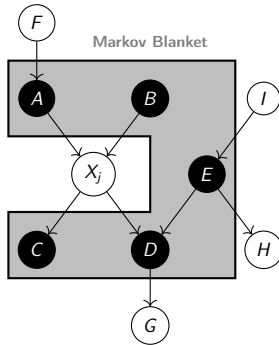
...

$$X_m : \quad s_m^{(t+1)} \leftarrow p(s_m | s_1^{(t+1)}, s_2^{(t+1)}, \dots, s_{m-1}^{(t+1)}, \mathbf{O})$$

- In short:

$$X_j : \quad s_j^{(t+1)} \leftarrow p(s_j | \mathbf{s}^{(t)} \setminus s_j, \mathbf{O})$$

Markov Blanket



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

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

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 $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$ be a set over m random variables
- We assume the data is I. I. D. (independent and identically distributed)
- Find parameters θ of CPDs (conditional probability distributions) which match the data best

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

Maximum Likelihood Estimation

- **Recall:** In MLE we want to compute: (\Rightarrow cf. slides 'density estimation')

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

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

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

- 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

$$\mathcal{LL}(\theta|\mathcal{D}) = \log p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}|\theta)$$

(1) I.I.D.

$$\stackrel{(1)}{=} \log \prod_{i=1}^n p(\mathbf{x}^{(i)}|\theta)$$

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

$$\stackrel{(2)}{=} \sum_{i=1}^n \log p(\mathbf{x}^{(i)}|\theta) = \sum_{i=1}^n \log p(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}, \dots, \mathbf{x}_i^{(m)}|\theta)$$

(3) Bayesian network semantics

$$\stackrel{(3)}{=} \sum_{i=1}^n \log \left(\prod_{j=1}^m p(\mathbf{x}_i^{(j)}|Pa(\mathbf{x}_i^{(j)}), \theta) \right) \stackrel{(2)}{=} \sum_{i=1}^n \sum_{j=1}^m \log p(\mathbf{x}_i^{(j)}|Pa(\mathbf{x}_i^{(j)}), \theta)$$

$$= \sum_{j=1}^m \sum_{i=1}^n \log p(\mathbf{x}_i^{(j)}|Pa(\mathbf{x}_i^{(j)}), \theta_j) = \sum_{j=1}^m \mathcal{LL}(\theta_j|\mathcal{D})$$

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?

²missing data case: see later slides

Likelihood for Multinomials

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

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

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

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

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

Maximum Likelihood for Multinomials

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

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

- And set them to zero:

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

³consider 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:

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

$$p(A = 1) = 1 - p(A = 0) = 11/16$$

$$p(F = 0|A = 1) = 6/11$$

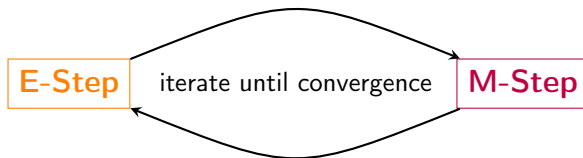
$$p(F = 1|A = 1) = 1 - p(F = 0|A = 1) = 5/11$$

$$p(H = 0|A = 1, F = 1) = 4/5$$

...

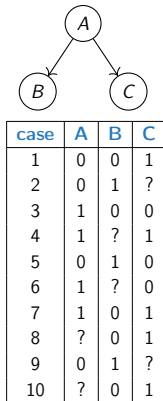
What about missing Values?

- 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



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

```
1 initialize parameters  $\theta$ 
2 while not converged do
3   | compute pseudo counts
4   | set parameters to the maximum likelihood estimates
5 return final parameters  $\theta$ 
```

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

Structure Learning

- To do...

Section:
Hidden Markov Models (HMMs)



What is a hidden Markov Model?

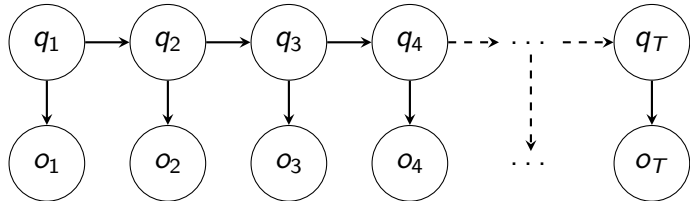
- **Motivation:** Consider e. g. the task of **part-of-speech tagging**
- **Problem:** Labels cannot be assigned by looking only at single words
- **Polysemy:** The same word can have different meanings, e. g. can, bank
- A **hidden Markov model (HMM)** is a **sequence classifier** and as such able to 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 $\theta = (\mathbf{A}, \mathbf{B})$ and a sequence of observations $\mathbf{o} = o_1, o_2, \dots, o_T$, find the most probable sequence of (hidden) states $\mathbf{q} = q_1, q_2, \dots, q_T$.

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

$$\hat{\mathbf{q}} = \arg \max_{\mathbf{q}} \frac{p(\mathbf{o}|\mathbf{q}) \cdot p(\mathbf{q})}{p(\mathbf{o})} \propto \arg \max_{\mathbf{q}} \underbrace{p(\mathbf{o}|\mathbf{q})}_{\text{likelihood}} \cdot \underbrace{p(\mathbf{q})}_{\text{prior}} \quad (11)$$

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:

$$p(\mathbf{o}|\mathbf{q}) \approx \prod_{i=1}^T p(o_i|q_i)$$

Assumption 2: The probability of a state appearing is dependent only on the previous state: $p(\mathbf{q}) \approx \prod_{i=1}^T p(q_i|q_{i-1})$

⇒ **Markov Assumption** ('the future is independent of the past given the present.')

The underlying Model

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

$$\hat{\mathbf{q}} = \arg \max_{\mathbf{q}} p(\mathbf{q}|\mathbf{o}) \propto \arg \max_{\mathbf{q}} \prod_{i=1}^T p(o_i|q_i) \cdot p(q_i|q_{i-1}) \quad (12)$$

- 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

$P(t_i t_{i-1})$	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

$P(w_i t_i)$	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)

$$p(q_i|q_{i-1}) = \frac{C(q_{i-1}, q_i)}{C(q_{i-1})} \quad (13)$$

- **Emission probabilities** (second table)

$$p(o_i|q_i) = \frac{C(q_i, o_i)}{C(q_i)} \quad (14)$$

Algorithm 4: Viterbi Algorithm (Dynamic Programming)

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

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

// initialization step

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

3 $\mathbf{V}[q, 1] \leftarrow a_{0,q} \cdot b_q(o_1)$

4 $trace[q, 1] \leftarrow 0$

// compute best path through trellis

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

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

7 $\mathbf{V}[q, t] \leftarrow \max_{q'=1}^N \mathbf{V}[q', t-1] \cdot a_{q',q} \cdot b_q(o_t)$

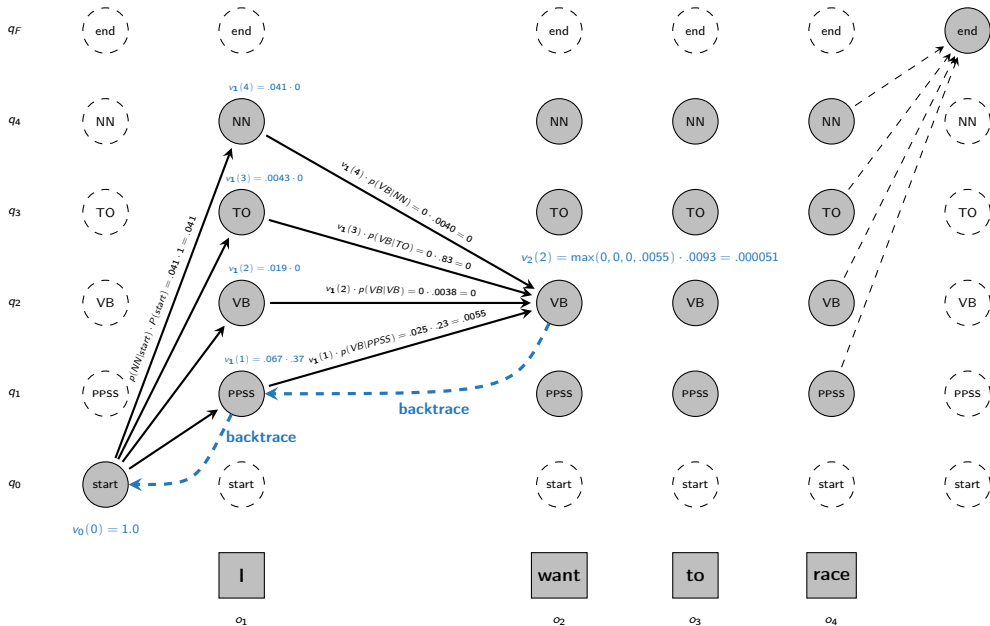
8 $trace[q, t] \leftarrow \arg \max_{q'=1}^N \mathbf{V}[q', t-1] \cdot a_{q',q}$

// termination step

9 $\mathbf{V}[q_F, T] \leftarrow \max_{q=1}^N \mathbf{V}[q, T] \cdot a_{q,q_F}$

10 $trace[q_F, T] \leftarrow \arg \max_{q=1}^N \mathbf{V}[q, T] \cdot a_{q,q_F}$

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



Section:
Wrap-Up



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:**
$$I_{LM}(\mathcal{G}) \subseteq I(p) \Leftrightarrow p(X_1, X_2, \dots, X_m) = \prod_{j=1}^m p(X_j | Pa(X_j))$$
- 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 2020/2021

Contact:

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

daniel.wehner@sap.com

Do you have any questions?