Probabilistic Artificial Intelligence

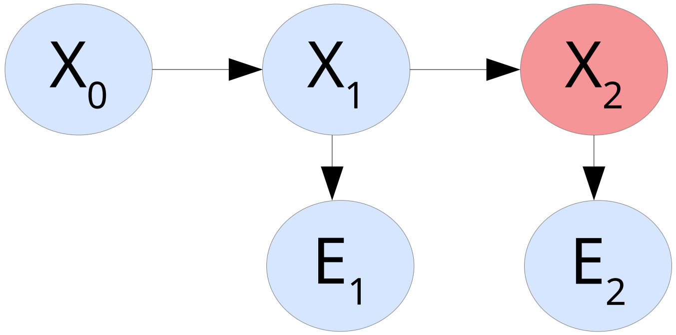
Lecture 4: Learning

handout:
“Maximum Likelihood Estimation
and the EM algorithm”

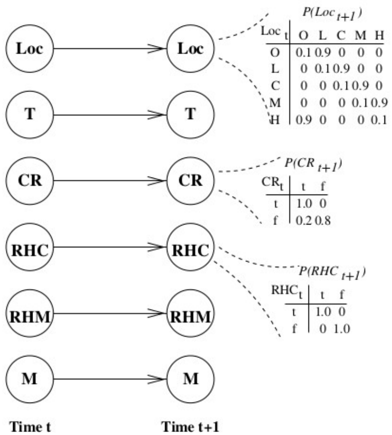
Interactive Intelligence group
Department of Electrical Engineering, Mathematics and Computer Science

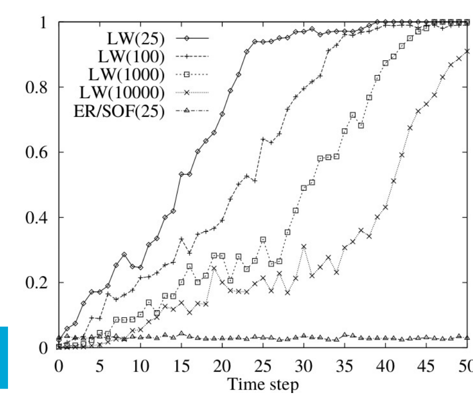
TU Delft
Delft University of Technology

Recap



- Probability for representing beliefs
  - Bayes' rule to update beliefs
- Time... need both:
  - estimation (sensor model)
  - prediction (transition model)
- HMMs
  - filtering, prediction, smoothing
  - Main tools: 1) Bayes' rule, 2) Markov assumption
- Compact representations: DBNs
  - Approximate inference: particle filter





Where do the numbers come from?

- Where do we get the parameters of the HMM or MDP?
- Today: learning techniques

- Field of Machine learning is huge... also see:
  - CS4070 – Multivariate Data Analysis
  - CS4180 – Deep Learning
  - IN4085 – Pattern Recognition → “machine learning”
  - IN4320 – Machine learning → “machine learning 2”
  - CS4400 – Deep RL
  - SC42050 – Knowledge-Based Control Systems (RL)

What is Learning?

Why learning?

- Three main motivations:
  1) No model available
     - Human designers can not provide models for all possible situations an intelligent agent may encounter
     - E.g., the numbers of a Bayesian network or HMM

  2) Adaptivity.
     - The way the environment works may change over time.
     - e.g., traffic patterns

  3) Humans don’t understand the task well enough; not possible to manually program.
     - E.g., vision tasks.

What is Learning?

- …?

What is Learning?

- We take the perspective that **learning = induction**.
  - going from observations to theories that explain these observations
  - (contrast with “deductive learning”:
    from a *known* general rule → more specialized rule that allows for more efficient processing.)
- But details may depend on settings/tasks…
  - receiving a bag of data
  - receiving a bag of data with labels
  - robot learning from its observations over time
  - robot taking actions

What is Learning?

- We take the perspective that **learning = induction**.
  - going from observations to theories that explain these observations
  - (contrast with "deductive learning":
    from a *known* general rule → more specialized rule that allows for more efficient process
- But details may c
  - receiving a bag (
  - receiving a bag (
  - robot learning fr
  - robot taking act

More generally, the choice of technique depends on:

- What the component is
- What prior knowledge the agent has
- How the data and component are represented
- What feedback is available to learn from

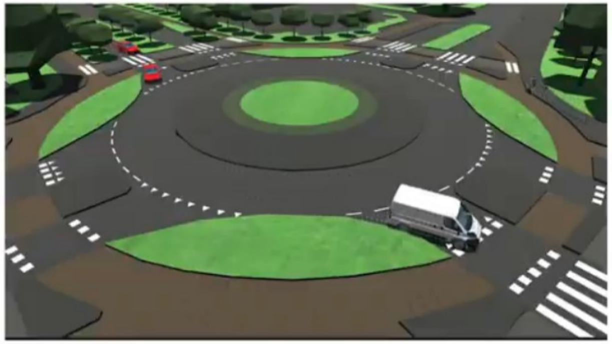
Example: Taxi Driver Agent

- Instructor shouts “Brake”…
  the agent may learn a
  condition-action rule for when to brake

- From images showing buses,
  the agent learns to **recognize buses**

- By trying actions and observing the results
  (e.g. braking hard on a wet road),
  agent can learn **effects of actions**

- No tip from passengers after driving wildly…
  learn a component of its
  (or really the passengers’!) **utility function**



Learned car behaviors
[Behbahani et al. 2019]

High Level Perspectives to Learning

- Idealistic
  - maintain a belief over true way the world works (possible hypotheses)
  - use observations in optimal fashion… → Bayes’ rule
  - **statistical learning**

- Pragmatic
  - anything that improves performance
  - Tom M. Mitchell:
    "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E*."
    - Machine Learning. 1997. McGraw Hill. p. 2. ISBN 978-0-07-042807-2.
  - **optimization**

- And everything in between… ML, statistics and optimization are tightly interwoven.

Idealistic (Bayesian) Perspective

- A robot that learns from its observation: Bayes rule!
  - use laws of probability to represent its beliefs
    (otherwise, its beliefs could led it to be ‘exploited’)
  - belief over how the world works: hypotheses *H*
  - using Bayes rule to update its beliefs → **that *is* learning!**
- learning = inference
  - $P(H|d) = P(d|H)\, P(H) / P(d)$
- And then ‘act’ in a Bayesian way:
  - $V(a) = \Sigma_h P(h|d)\, u(a,h)$

TU Delft

Idealistic Perspective: Hypothesis Spaces

- Nice, but… what is the class of hypotheses *H*…?
  - Idea 1: World is complex… need a huge class *H*
    - but leads to huge computational complexity…
  - Idea 2: Limit the class to allow for tractable inference
    - but what if the true model is not in *H*…?
    - all bets are off **

** Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. P Grünwald, T Van Ommen. Bayesian Analysis 12 (4), 1069-1103. 2017

Idealistic Perspective: Hypothesis Spaces

- Nice, but… what is the class of hypotheses *H*…?
  - Idea 1: World is complex… need a huge class *H*
    - but leads to huge computational complexity…
  - Idea 2: Limit the class to allow for tractable inference
    - but what if the true model is not in *H*…?
    - all bets are off **

- E.g., learning the parameters of an HMM from a long sequence…
  - what assumptions do we make?
  - how would this work?

** Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. P Grünwald, T Van Ommen. Bayesian Analysis 12 (4), 1069-1103. 2017

Idealistic Perspective: Hypoth

- Nice, but... v
  - Idea 1: Wo
    - but lea
  - Idea 2: Lim
    - but wh
    - all bet
- E.g., learnin
  - what assu
  - how would

- we assume...
  - some number of states *S*
  - Markov assumption
- a hypothesis *h* is a vector of all the initial state-, transition-, and observation parameters
- Now need to compute:

  $P(h|o_{1:T}) = P(o_{1:T}|h)\, P(h) \,/\, P(o_{1:T})$

  - *h* is high-dimensional... how to even represent $P(h|o_{1:T})$?
  - $P(h)$ what is our subjective belief...?
  - $P(o_{1:T}|h)$ itself is intractable: marginalize over sequence states

→ we may need some smarter tricks...

** Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. P Grünwald, T Van Ommen. Bayesian Analysis 12 (4), 1069-1103. 2017

Pragmatic Perspective

- "Any form of parameter updating that improves performance"
  - E.g., given a bag of data, make multiple passes trough to optimize some parameters using SGD.
- learning=optimization
- what are we optimizing...?
  - end-to-end learning:
    - parametrize 'actions' using some parameters θ
    - e.g., action probabilities, or a NN that generates those
    - directly optimize V(θ)
  - other typical approach: maximum likelihood

TUDelft

Maximum likelihood

- Somewhere between the full Bayesian perspective, and end-to-end optimization
- Still based on statistical models
  - instead of computing posterior $P(H|d)$
  - optimize:

    $$h_{ML} = max_h\ P(d|h)$$

- To do this optimization, optimize **log likelihood:** $L(h) = log\ P(d|h)$
  - $h_{ML} = max_h\ log\ P(d|h)$

    $$= max_h\ log \prod_i P(d_i|h) = max_h\ \sum_i log\ P(d_i|h)$$

  - usually much easier to optimize

Maximum likelihood

Example: a coin toss...

- We have a coin and toss it N times...
  - *k* heads, *l*=*N*-*k* tails

    → what is the prob. of heads?

ayesian perspective, and end-to-

;

$P(H|d)$

ze **log likelihood:** $L(h) = log\ P(d|h)$

- $h_{ML} = max_h\ log\ P(d|h)$

  $= max_h\ log \prod_i P(d_i|h) = max_h\ \sum_i log\ P(d_i|h)$
- usually much easier to optimize

Maximum likelihood

Example: a coin toss...

- We have a coin and toss it N times...
  - *k* heads, $l = N - k$ tails
    → what is the prob. of heads?
- Lets call P(head) = θ
- So.. likelihood:
  $P(d|\theta) = \theta^k (1-\theta)^l$

ayesian perspective, and end-to-

s

$P(H|d)$

ze **log likelihood:** $L(h) = \log P(d|h)$

- $h_{ML} = \max_h \log P(d|h)$
  $= \max_h \log \prod_i P(d_i|h) = \max_h \sum_i \log P(d_i|h)$
- usually much easier to optimize

Maximum likelihood

Example: a coin toss…

- We have a coin and toss it N times…
  - *k* heads, *l*=*N*-*k* tails
    → what is the prob. of heads?
- Lets call P(head) = θ
- So.. likelihood:
  $P(d|\theta) = \theta^k (1-\theta)^l$

ayesia

$P(H|d$

ze **log**

- ▹ $h_{ML} = \max_h \log P(d|h)$
  $= \max_h \log \prod_i P(d_i|h) = \max_h \sum_i \log$
- ▹ usually much easier to optimize

Maximum likelihood Bernoulli (20.2.1)

$$L(\theta) = \log \prod_{i=1}^{k} \theta \prod_{i=1}^{l} (1-\theta)$$
$$= \log \theta^k (1-\theta)^l$$
$$= k \log \theta + l \log(1-\theta)$$

Its derivative:

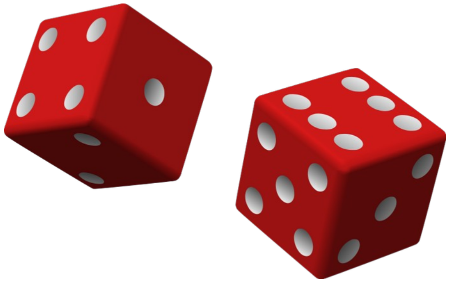
$$\frac{d}{d\theta} L(\theta) = \frac{k}{\theta} - \frac{l}{(1-\theta)}$$

equating with 0 and solving to find the maximum:

$$\frac{k}{\theta} - \frac{l}{(1-\theta)} = 0$$
$$\Leftrightarrow \frac{k}{\theta} = \frac{l}{(1-\theta)}$$
$$\Leftrightarrow k(1-\theta) = l\theta$$
$$\Leftrightarrow k = (l+k)\theta$$
$$\Leftrightarrow \frac{k}{l+k} = \theta = \frac{k}{N}$$

Maximum likelihood vs Bayesian

- **Bayesian learning**: computing posterior $P(H|d)$
  - uses prior information $P(h)$
  - use in weighted manner to select best action: $V(a) = \Sigma_h P(h|d)\, u(a,h)$
- **Maximum likelihood (ML)** $h_{ML} = max_h\, P(d|h)$
  - select action according $V(a) = u(a,h_{ML})$
  - prone to "overfitting"
- **Maximum a posteriori (MAP)** probability: ML + priors
  - $h_{MAP} = max_h\, P(d|h)\, P(h)$
  - can still overfit

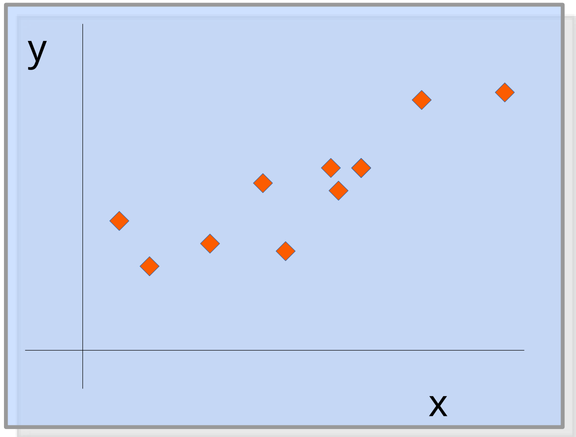


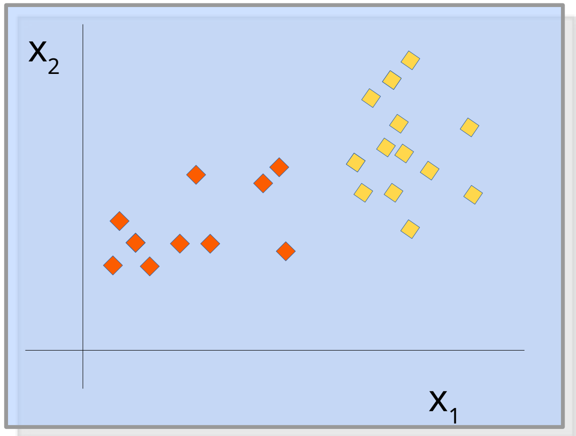
I threw a die a couple of times: (4,2,2,5,4,4)
→ does this die have 50% chance of landing 4?

Some Machine Learning Concepts in a Nutshell

Supervised Machine Learning

- Taxi agent told “that’s a bus”
- General set up:
  - bag of **training data** $d=\{<x_i,y_i>\}_{i=1\ldots N}$
  - assumption: labels generated by **‘true’ function** $y=f(x)$
  - goal: find **hypothesis** $h(x)\approx g(x)$
- Instantiations: regression, classification



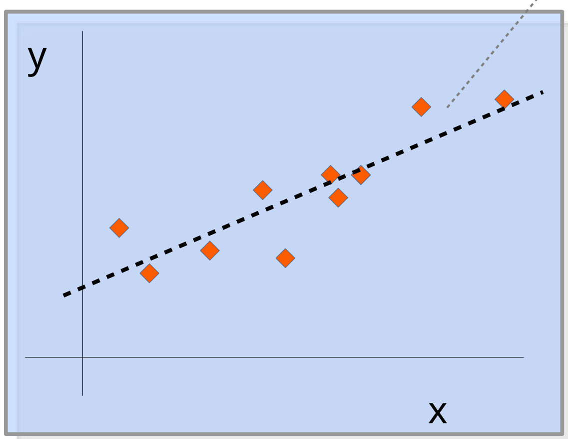


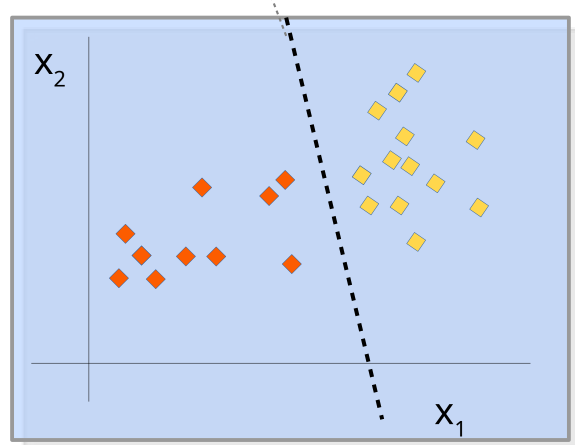
Supervised Machine Learning

- Taxi agent told "that's a bus"
- General set up:
  - bag of **training data** $d=\{<x_i,y_i>\}_{i=1...N}$
  - assumption: labels generated by **'true' function** $y=f(x)$
  - goal: find **hypothesis** $h(x)\approx g(x)$

Run your favorite ML algorithm
- define loss
- optimization

- Instantiations: regression, classification





Other Machine Learning Settings

- Unsupervised learning
  - Learn patterns in the input without explicit feedback – **no labels**
  - Most common task is clustering
  - e.g. taxi agent notices “bad traffic days”
- Semi-supervised learning
  - large bag of data, only a few are labeled…
  - can we use the unlabeled data to use labels more effectively?
- Active-learning
  - large bag of data, only a few are labeled…
  - what point should we ask an annotator to label?
- Reinforcement learning
  - Learns from a series of reinforcements: rewards or punishments
  - Cab driver gets paid and needs to pay for fuel. He/she also needs to pay for groceries to live..
- Etc., etc.,…

Generalization: the problem of induction

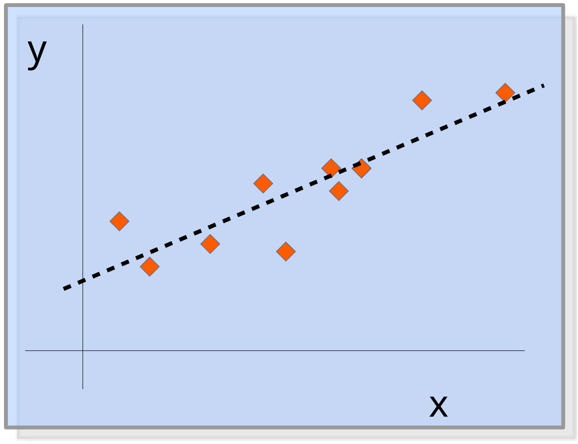
- Philosophical question of whether inductive reasoning leads to knowledge…
  - https://en.wikipedia.org/wiki/Problem_of_induction
  - E.g., the inference that "all swans we have seen are white, and, therefore, all swans are white", before the discovery of black swans…?
- In other words… **how do we know $h \approx f$?**

Generalization: the problem of induction

- Philosophical question of whether inductive reasoning leads to knowledge…
  - https://en.wikipedia.org/wiki/Problem_of_induction
  - E.g., the inference that "all swans we have seen are white, and, therefore, all swans are white", before the discovery of black swans…?

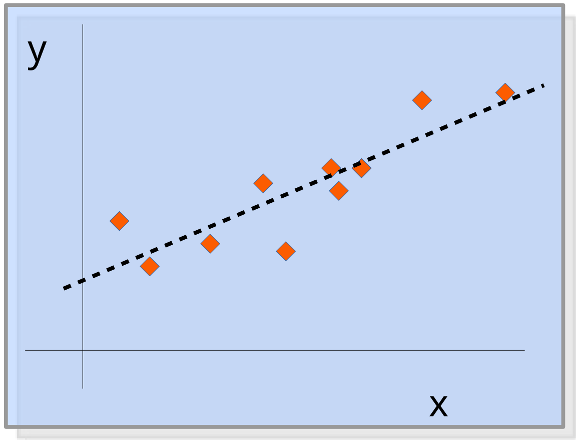
- In other words… **how do we know $h \approx f$?**
- Two approaches:
  - use theorems
    - computational/statistical learning theory
  - use experiments
    - test *h* on a new set of data

Generalization: train error / true error



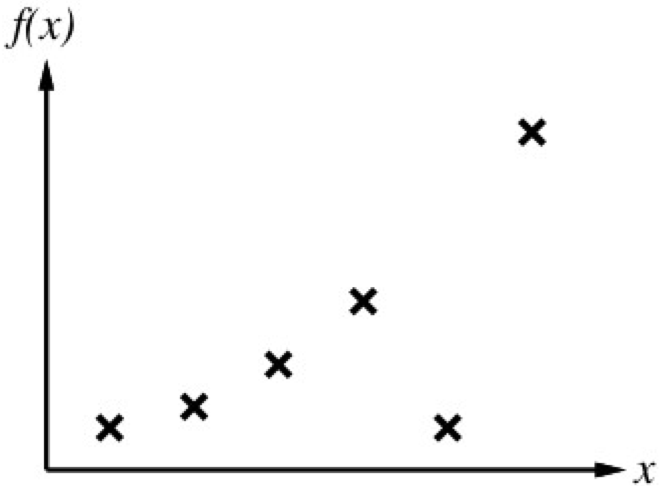
- ML algorithms optimize training loss
  - e.g., the mean squared error
- But we want to predict how well we do on unseen data
  - That performance can be quite different!

Generalization:
train error / true error



- ML algorithms optimize training loss

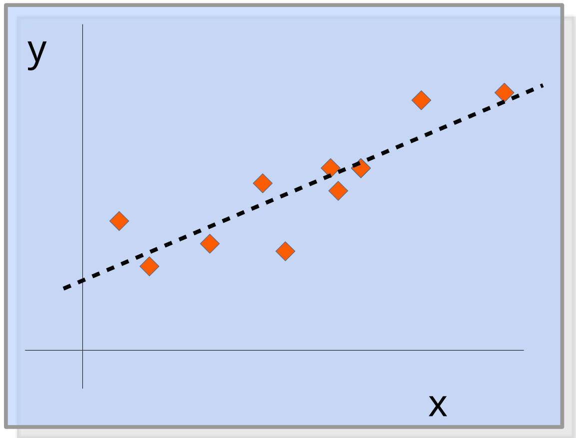
Example



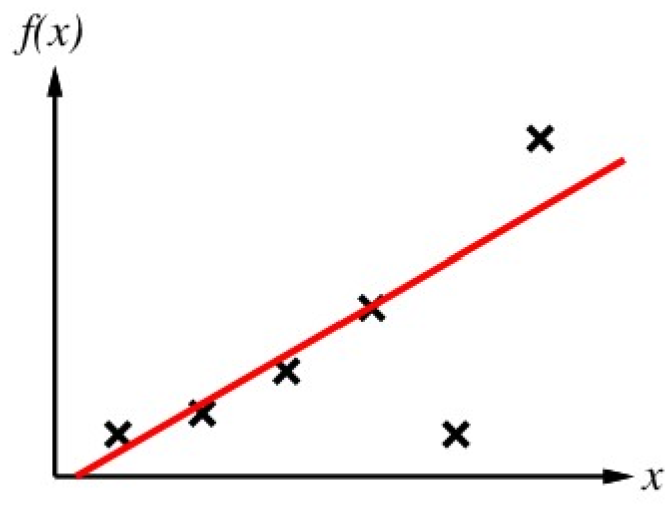
- E
  V
  ferent!

Generalization: train error / true error

- ML algorithms optimize training loss



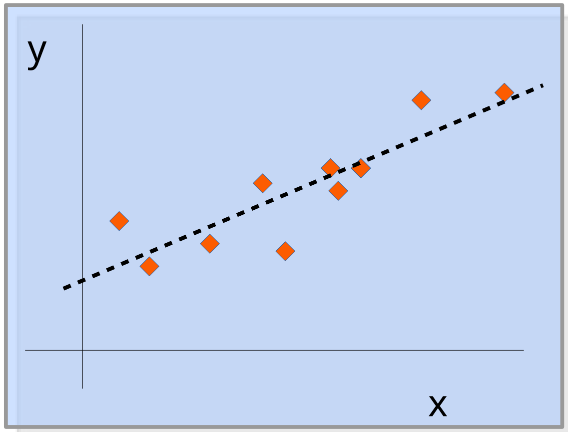
Example



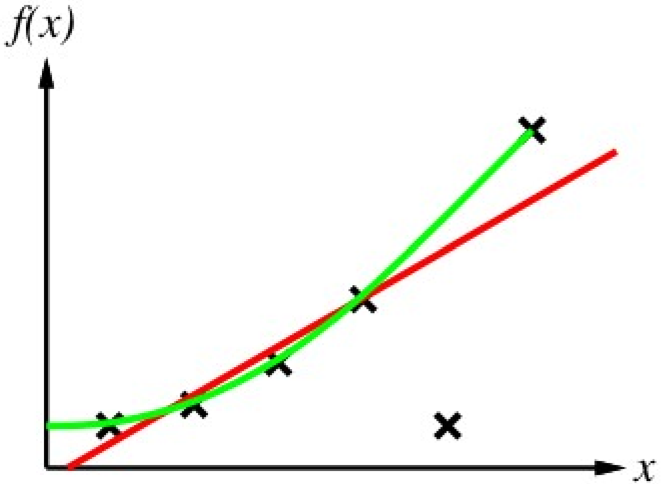
- E
  w
  ferent!

Generalization: train error / true error

- ML algorithms optimize training loss



Example

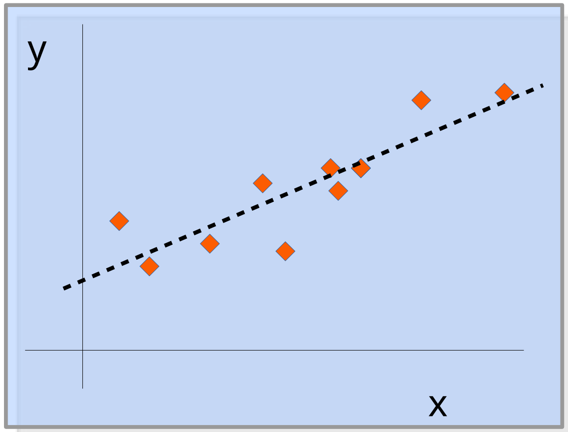


- E
  V

  ferent!

Generalization: train error / true error

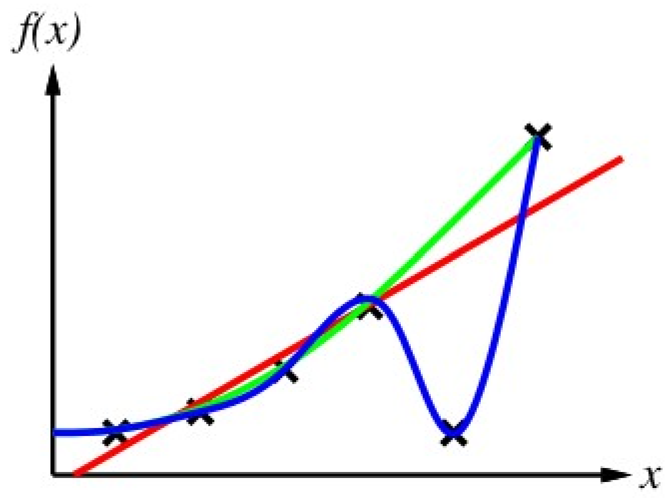
- ML algorithms optimize training loss



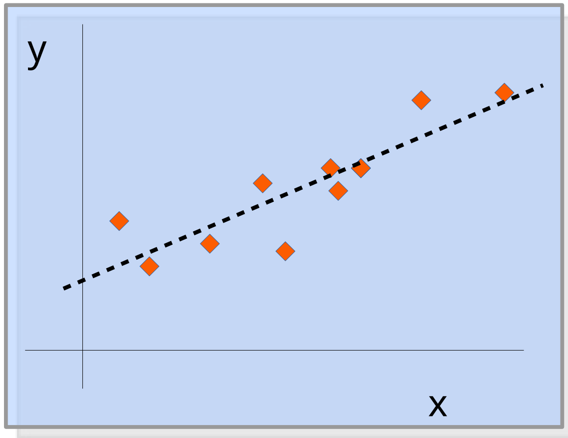
- E
  w

ferent!

Example

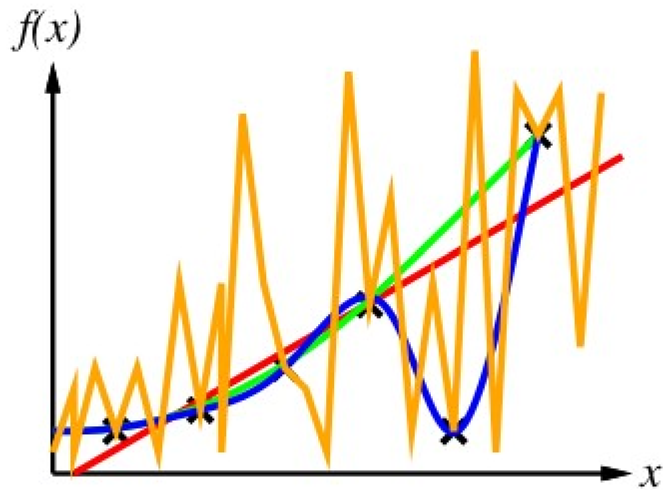


Generalization: train error / true error



- ML algorithms optimize training loss

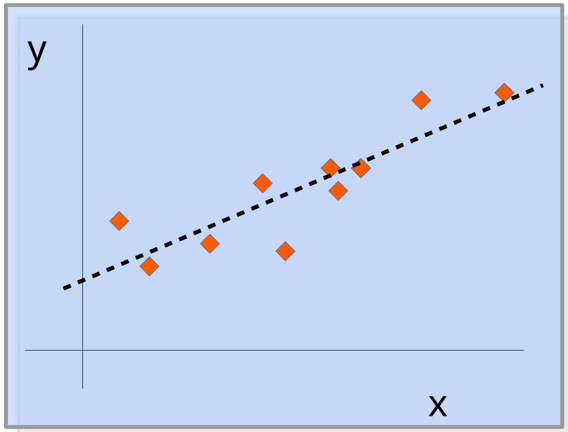
Example



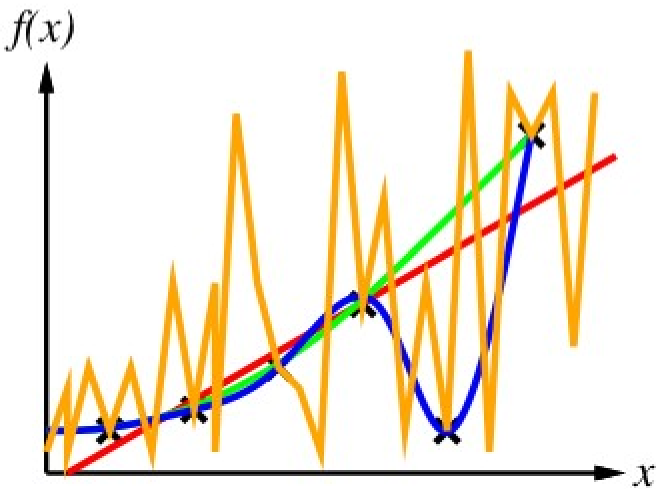
- E
  W
  ferent!

Generalization: train error / true error

- ML algorithms optimize training loss



Example

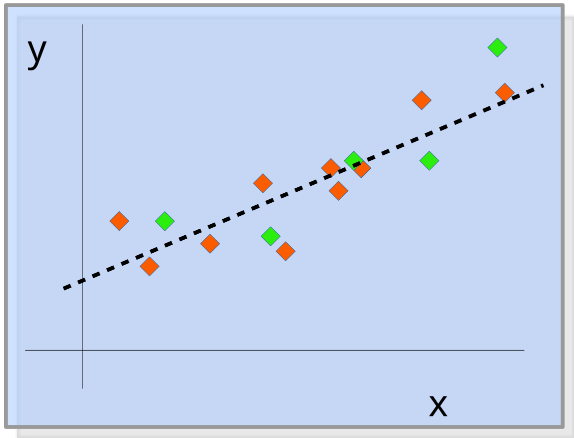


- E
  V

ferent!

by using a sufficiently complex model, is possible to get 0 training loss...
→ **“overfitting”**

Generalization: train error / true error



- ML algorithms optimize training loss
  - e.g., the mean squared error

- But we want to predict how well we do on unseen data
  - That performance can be quite different!

  - Solution: estimate on some held out **test data**
    - (test data is *never* used, also not for model selection!)

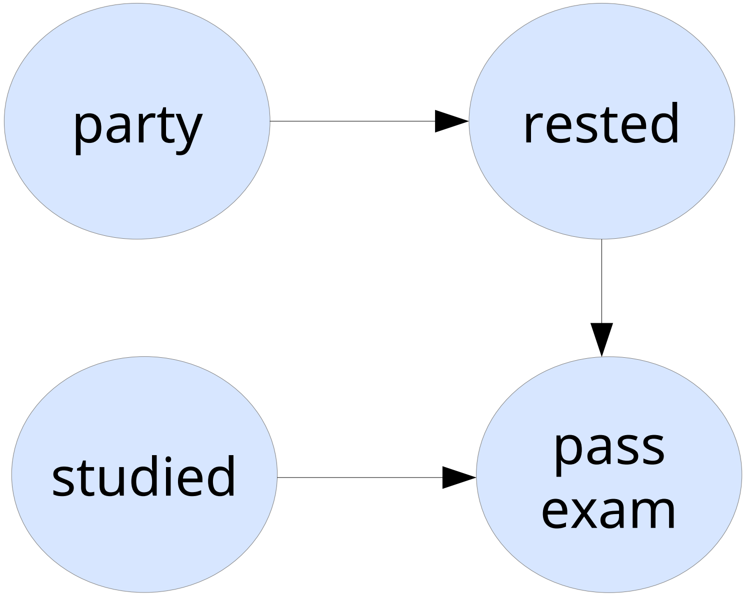
Theorems for Generalization

- We will not in detail cover, but see R&N 19.5
- Field: computational learning theory
- One of the main ideas: PAC-learning
  - assume that data is drawn from some true distribution
  - consider what would be an ‘unlucky draw’ (error >ε)
  - bound the probability of such ‘unlucky draws’ (<δ)

Learning parameters of a Bayesian Network

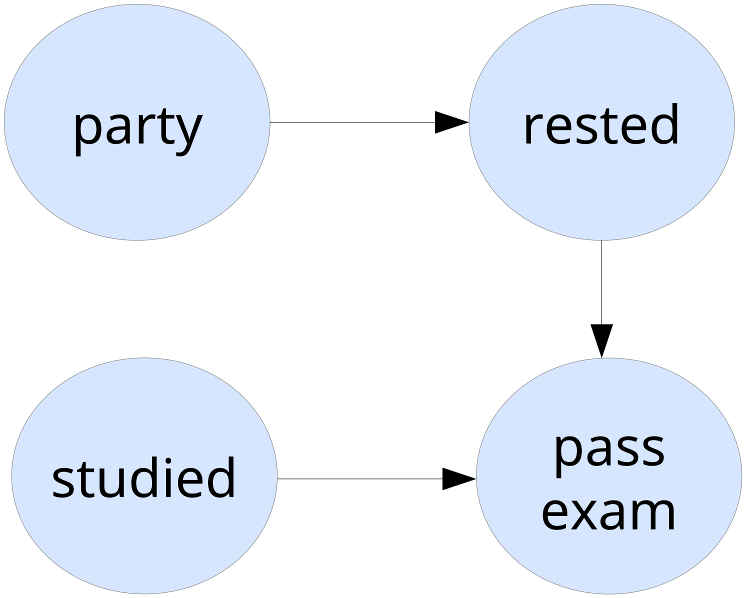
Learning a Bayesian Network

- How to Estimate the Parameters...?



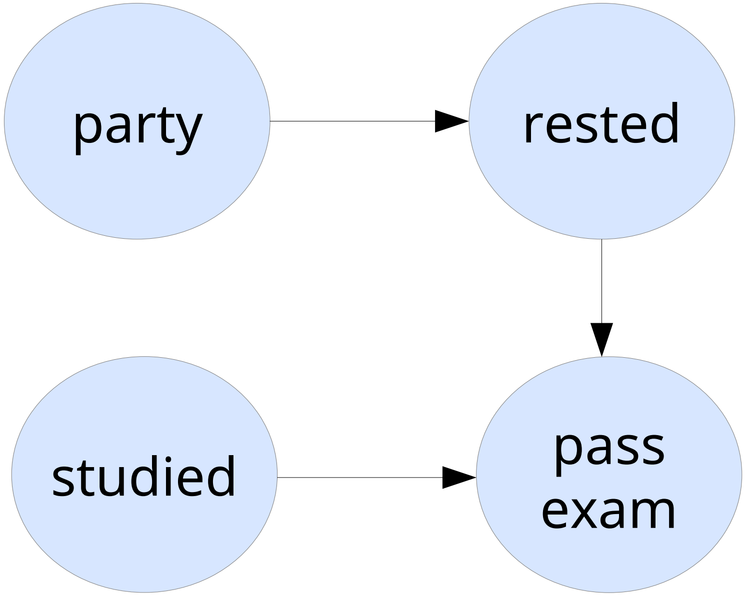
Learning a Bayesian Network

- How to Estimate the Parameters...?
- Well...
  - Bayesian learning
  - Maximum likelihood
  - MAP



Learning a Bayesian Network

- How to Estimate the Parameters...?
- Well...
  - Bayesian learning
  - Maximum likelihood
  - MAP

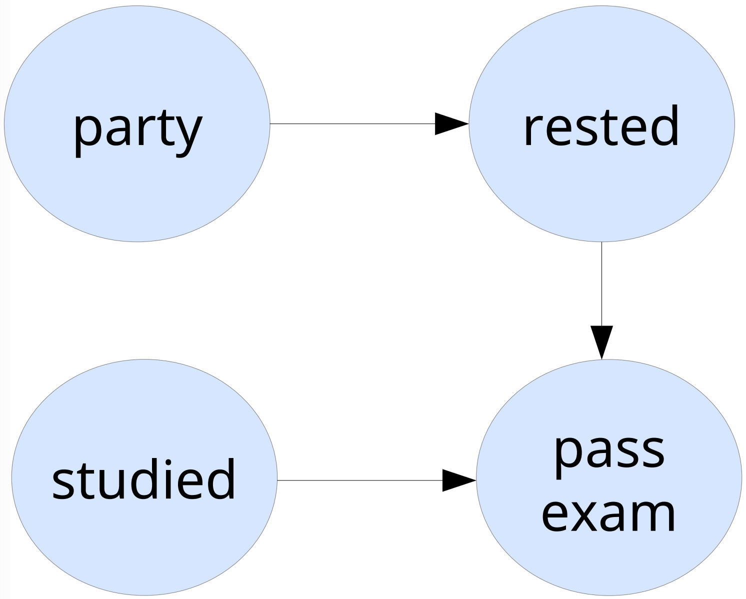


we'll give ML a shot....

Learning a Bayesian Network

L(θ) = log P(d;θ)

ters...?

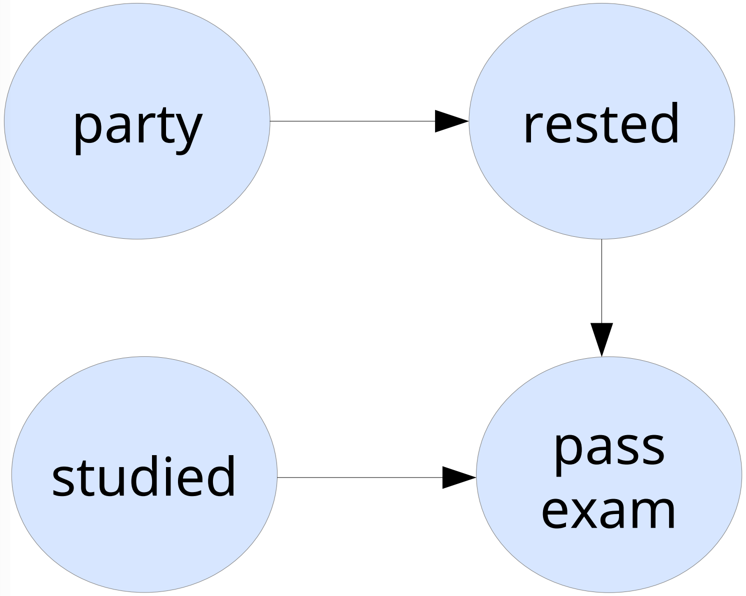


Learning a Bayesian Network

$L(\theta) = \log P(d;\theta)$

$= \log \Pi_i\, P(\langle party_i, rested_i, studied_i, pass_i \rangle ; \theta)$

ters...?



TU Delft

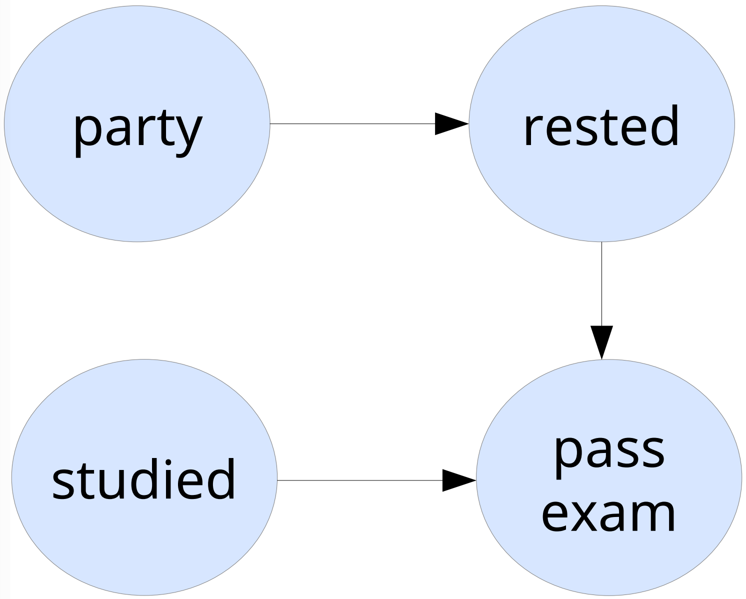
Learning a Bayesian Network

$L(\theta) = log\ P(d;\theta)$

$= log\ \Pi_i\ P(\ <party_i, rested_i, studied_i, pass_i>\ ;\ \theta\ )$

$= \Sigma_i\ log\ P(\ <party_i, rested_i, studied_i, pass_i>\ ;\ \theta\ )$

ters...?



Learning a Bayesian Network

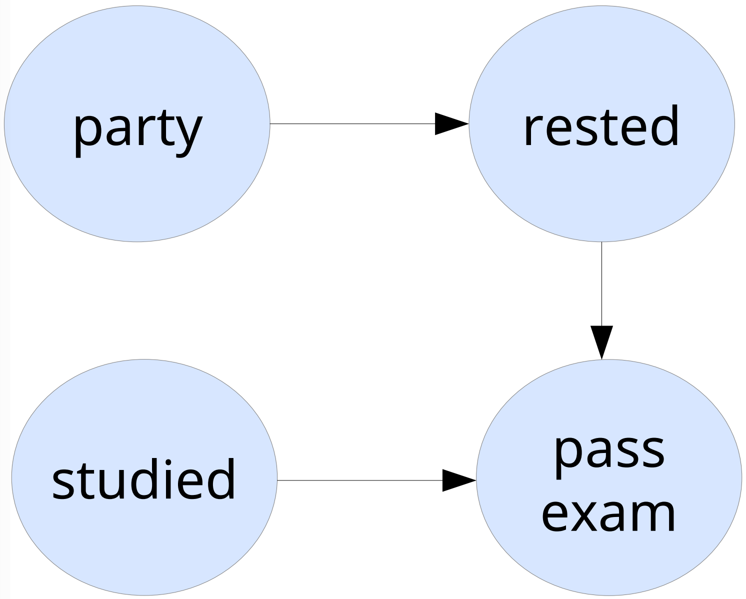
$$L(\theta) = \log P(d;\theta)$$

$$= \log \Pi_i P(<party_i, rested_i, studied_i, pass_i> ; \theta)$$

$$= \Sigma_i \log P(<party_i, rested_i, studied_i, pass_i> ; \theta)$$

$$= \Sigma_i \log P(party_i ; \theta) P(rested_i \mid party_i ; \theta) P(studied_i ; \theta) P(pass_i \mid studied_i , rested_i ; \theta)$$

ters...?



Learning a Bayesian Network

$L(\theta) = \log P(d;\theta)$

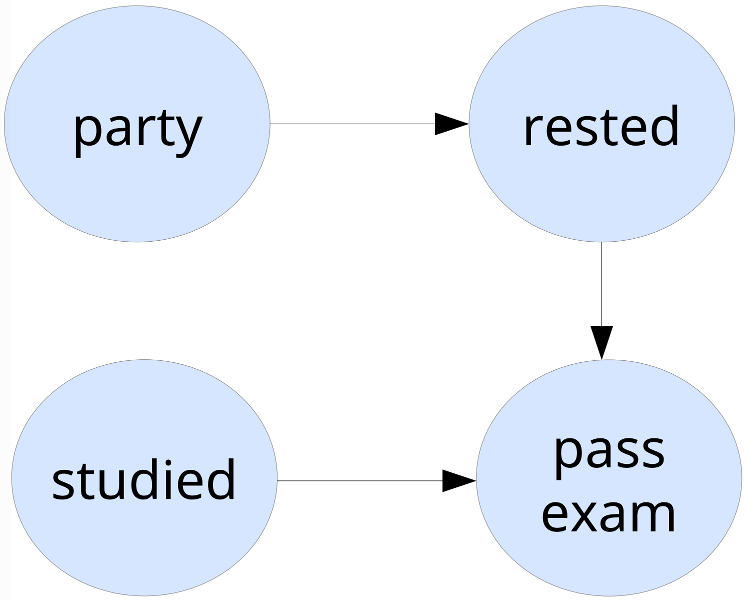
$= \log \Pi_i P(<party_i,rested_i,studied_i,pass_i> ; \theta)$

$= \Sigma_i \log P(<party_i,rested_i,studied_i,pass_i> ; \theta)$

$= \Sigma_i \log P(party_i; \theta)\, P(rested_i \,|\, party_i; \theta)$
$\quad P(studied_i; \theta) P(pass_i \,|\, studied_i, rested_i ; \theta)$

$= \Sigma_i \log P(party_i; \theta)$
$+ \Sigma_i \log P(rested_i \,|\, party_i; \theta)$
$+ \Sigma_i \log P(studied_i; \theta)$
$+ \Sigma_i \log P(pass_i \,|\, studied_i, rested_i ; \theta)$

ters...?



Learning a Bayesian Network

$L(\theta) = \log P(d;\theta)$

$= \log \Pi_i\, P(<party_i, rested_i, st$

$= \Sigma_i \log P(<party_i, rested_i, stu$

$= \Sigma_i \log P(party_i; \theta)\, P(rested$

$\quad P(studied_i; \theta) P(pass_i \,|\, stu$

$= \Sigma_i \log P(party_i; \theta)$

$+ \Sigma_i \log P(rested_i \,|\, party_i; \theta)$

$+ \Sigma_i \log P(studied_i; \theta)$

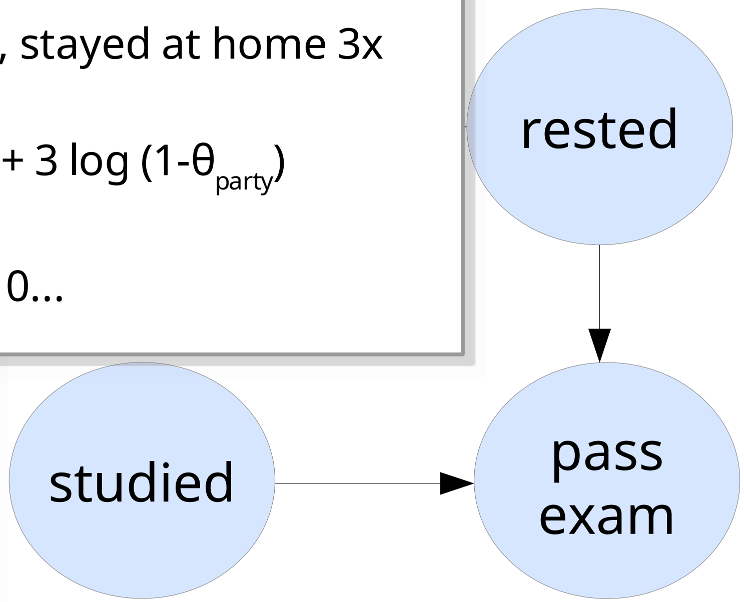
$+ \Sigma_i \log P(pass_i \,|\, studied_i, rested_i; \theta)$

Estimating a Bernoulli…!
(we saw this)

- $P(party_i; \theta) = \theta_{party}$
- e.g., went to party 2x, stayed at home 3x

  $\rightarrow L(\theta_{party}) = 2 \log \theta_{party} + 3 \log (1-\theta_{party})$

- diffentiate, equate to 0…



Learning a Bayesian Network

$L(\theta) = \log P(d;\theta)$

$= \log \Pi_i P(<party_i, rested_i, st$

$= \Sigma_i \log P(<party_i, rested_i, stu$

$= \Sigma_i \log P(party_i ; \theta\ ) P(rested$

$P(studied_i ; \theta) P(pass_i \,|\, stu$

$= \Sigma_i \log P(party_i ; \theta\ )$

$+ \Sigma_i \log P(rested_i \,|\, party_i ; \theta\ )$

$+ \Sigma_i \log P(studied_i ; \theta)$

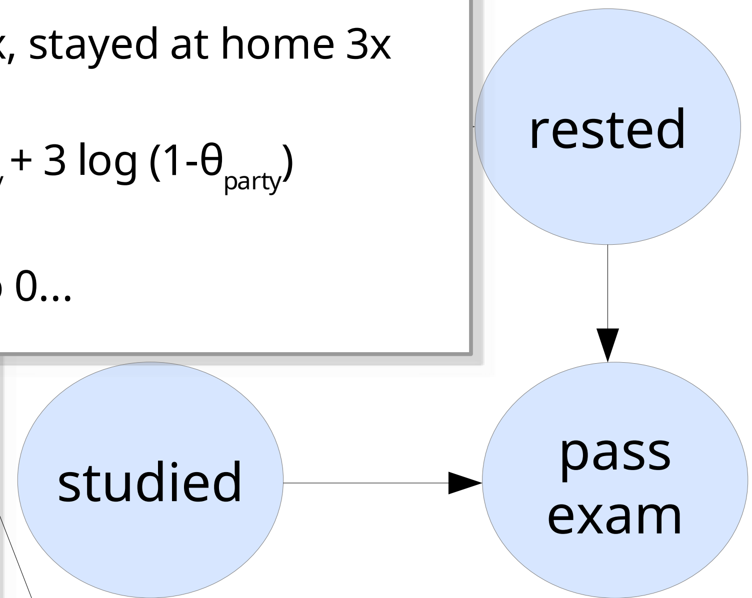
$+ \Sigma_i \log P(pass_i \,|\, studied_i, rested_i ; \theta\ )$

Estimating a Bernoulli...!
(we saw this)

- $P(party_i ; \theta) = \theta_{party}$
- e.g., went to party 2x, stayed at home 3x

$\rightarrow L(\theta_{party}) = 2 \log \theta_{party} + 3 \log (1-\theta_{party})$

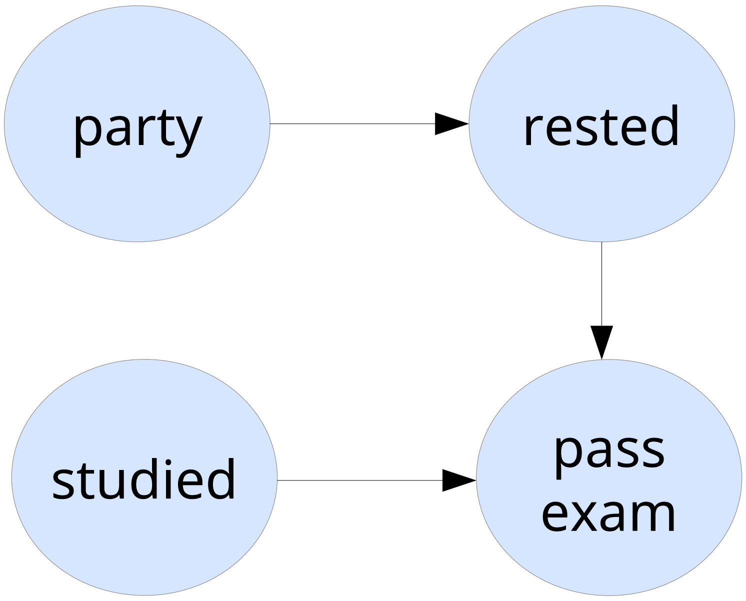
- diffentiate, equate to 0...



note: parameters are “localized”
→ estimate each CPT in isolation

Learning a Bayesian Network

- Great, you are now ready to plan for your exam!
  - risk of this approach...?



TUDelft

Learning with hidden variables

Learning with Hidden Variables (20.3)

- many ML methods: map x → y
  both are observable!
- What if don't observe all of the x? (or even y)...?
- E.g..
  - disease? (only observe diagnosis)
  - actual location of a robot?
  - or if it rained? (only observe umbrella)

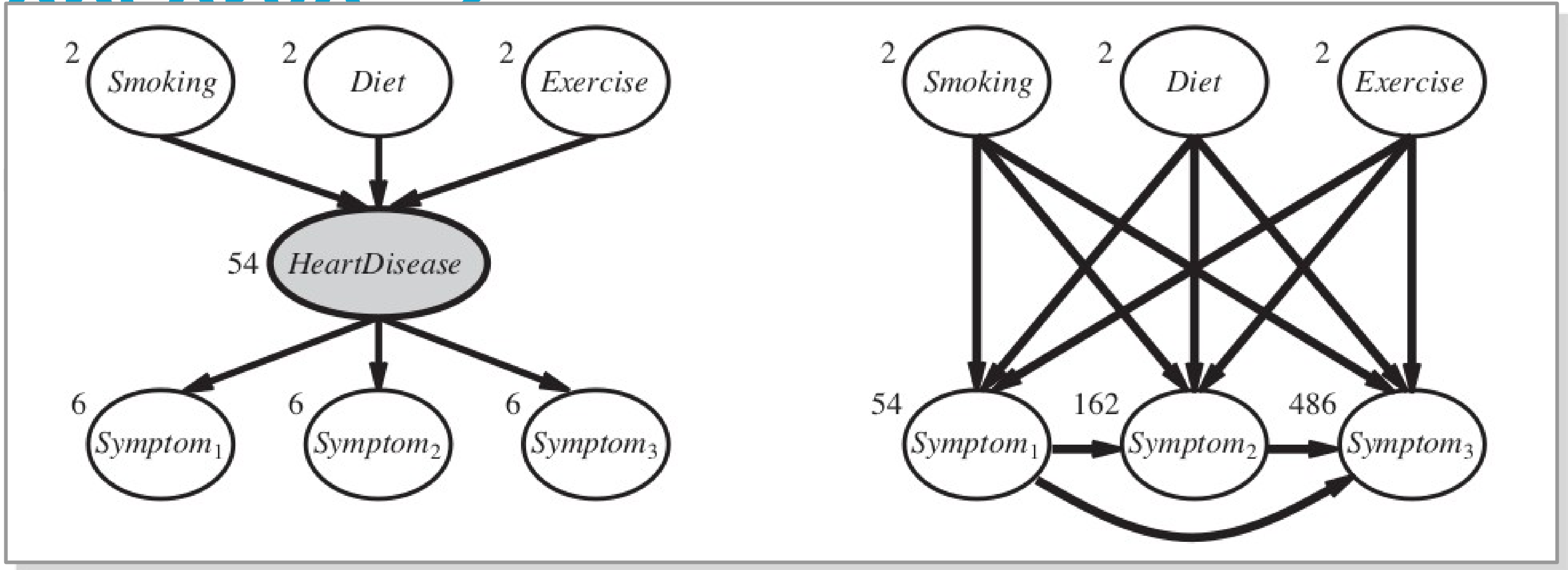
- Deal with hidden, or 'latent variables'!

Just work with what you can observe...?

- ...just use the variables x’ → y
  that you can observe?
  - typical approach taken in deep learning...
  - So can work well

- In favor of latent variable models:
  - can greatly reduce the number of parameters
  - may be critical to the further actions... (e.g. treatment depends on the disease!)

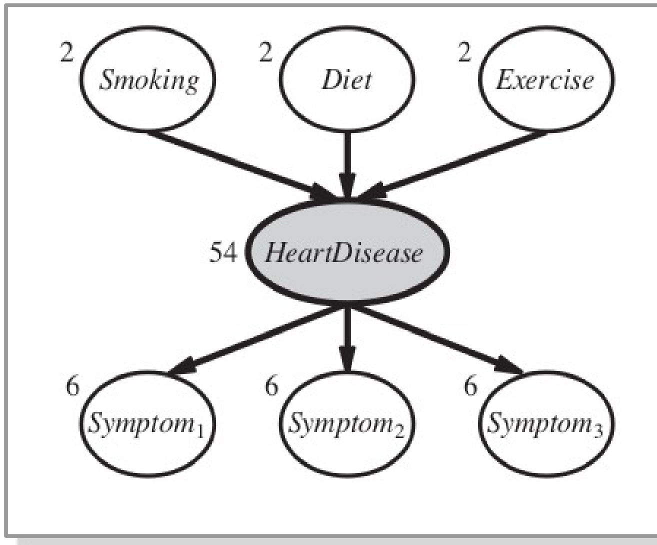
Just work with what you can observe…?

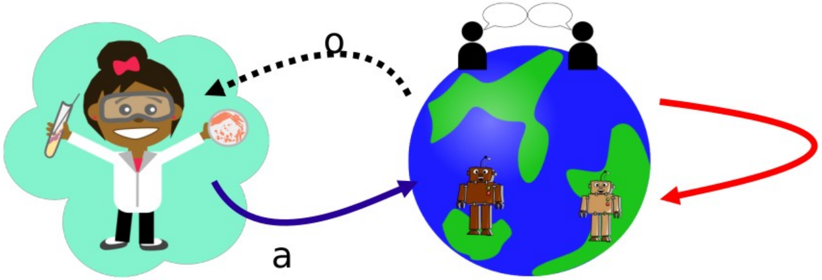


- In favor of latent variable models:
  - can greatly reduce the number of parameters
  - may be critical to the further actions… (e.g. treatment depends on the disease!)

Learning latent variable models

- The big challenge: how?
- We will assume we know the structure...
  - but even then...?
- Also learning the structure: very hard problem!
- But: at the core of AI!
  - agent that can hypothesize about the working of the world
  - even things that can not be seen directly (gravity, quarks...)





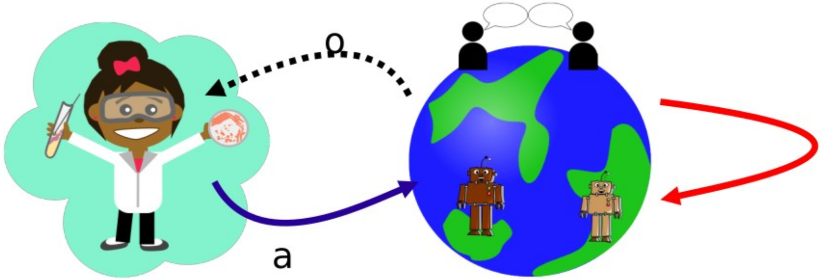
Learning with latent variables: the EM algorithm

EM – overall idea

- The **expectation-maximization** algorithm: one of the most frequently used methods to learn with hidden variables

- Overall intuition:
  - estimate hidden variables given current parameters
  - learn better parameters given estimated variables

- This will be technical...
  - but: agent that can do science!

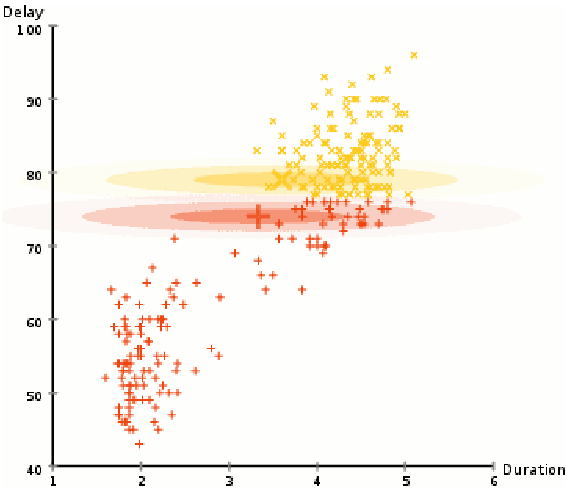


TUDelft

EM for clustering – intuition

- Intuition: EM for clustering using *k* Gaussians

- Algorithm does not get class labels
- randomly initialize:
  - cluster means, covariances
  - which point belongs to which cluster



https://en.wikipedia.org/wiki/Expectation%E2%80%93maxi

- Iterate:
  - estimate $p_{ij} = P(C=i|\mathbf{x}_j)$ the probability that data point j belongs to cluster i
    - using current cluster parameters
  - update cluster parameters using p_{ij}

EM General form

‘x’ – all observed data

- General form of EM:

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{Z} = z|\theta) \tag{2.1}$$

where:

- $\theta^{(k+1)}$ is the new parameter vector
- $\boldsymbol{z}$ is the vector of values for latent variables $\boldsymbol{Z}$
- $\boldsymbol{x}$ is the value of observed variables
- $P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$ the ‘estimation’ of the latent variables given $\boldsymbol{x},\theta^{(k)}$
- $L(\boldsymbol{x},\boldsymbol{Z} = \boldsymbol{z}|\theta)$ the log likelihood:

$$L(\boldsymbol{x},\boldsymbol{Z} = \boldsymbol{z}|\theta) = \log P(\boldsymbol{x},\boldsymbol{Z} = \boldsymbol{z}|\theta)$$

EM General form

1. **E-step**, where 'E' stands for *expectation*. Here the summation over $\boldsymbol{z}$ is performed to compute the expectation. Note that, in order to accomplish this, it needs to compute, or *estimate*, the posterior $P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x}, \theta^{(k)})$.
2. **M-step**. Which performs the maximization over parameters θ.

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x}, \theta^{(k)}) L(\boldsymbol{x}, \boldsymbol{Z} = z|\theta) \tag{2.1}$$

where:

- $\theta^{(k+1)}$ is the new parameter vector
- $\boldsymbol{z}$ is the vector of values for latent variables $\boldsymbol{Z}$
- $\boldsymbol{x}$ is the value of observed variables
- $P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x}, \theta^{(k)})$ the 'estimation' of the latent variables given $\boldsymbol{x}, \theta^{(k)}$
- $L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}|\theta)$ the log likelihood:

$$L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}|\theta) = \log P(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}|\theta)$$

EM General form

1. **E-step**, where 'E' stands for *expectation*. Here the summation over $\boldsymbol{z}$ is performed to compute the expectation. Note that, in order to accomplish this, it needs to compute, or *estimate*, the posterior $P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$.
2. **M-step**. Which performs the maximization over parameters θ.

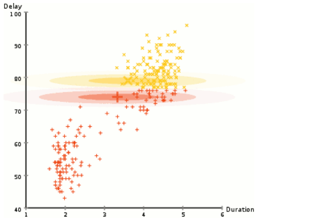
$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{Z} = z|\theta) \tag{2.1}$$

where:

- $\theta^{(k+1)}$ is the new parameter vector
- $\boldsymbol{z}$ is the vector of values for latent variables $\boldsymbol{Z}$
- $\boldsymbol{x}$ is the value of observed variables
- $P(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$ the 'estimation' of the latent variables given $\boldsymbol{x},\theta^{(k)}$
- $L(\boldsymbol{x},\boldsymbol{Z} = \boldsymbol{z}|\theta)$ the log likelihood:

In the Mixture of Gaussians example:

- x – the set of data points x_i z – the hidden "true cluster" z_i for each point
- θ - parameters: mean vectors, covariance matrices
- $L(x,z|\theta) = \log \Pi_i P(x_i, z_i \mid \theta)$
- E-step: estimate the probability of the assignment z
- M-step: update the means, covariances



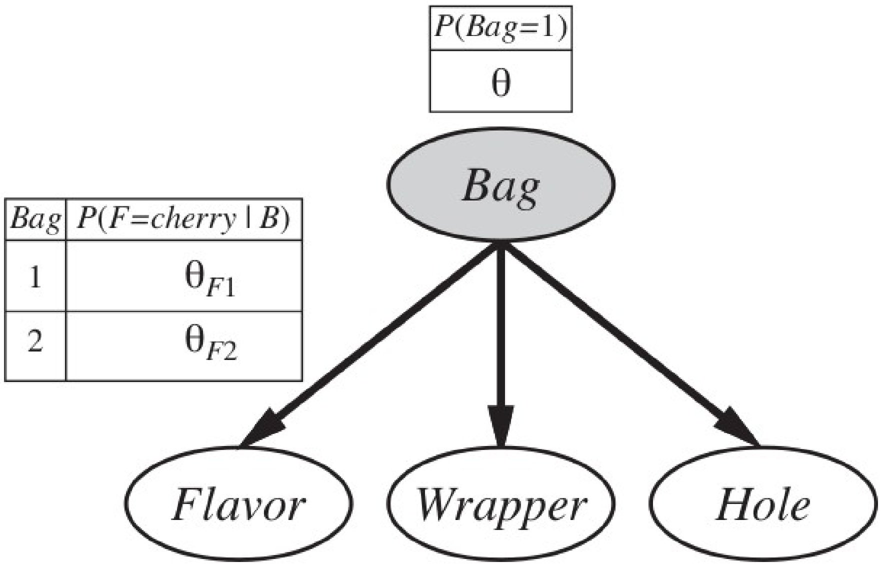
https://en.wikipedia.org/wiki/Expectation%E2%

EM for the “Flavor” Bayesian Network

(R&N: 20.3.2)

The Flavor BN

- 2 Bags of candy got mixed!



In the "flavor BN", we have

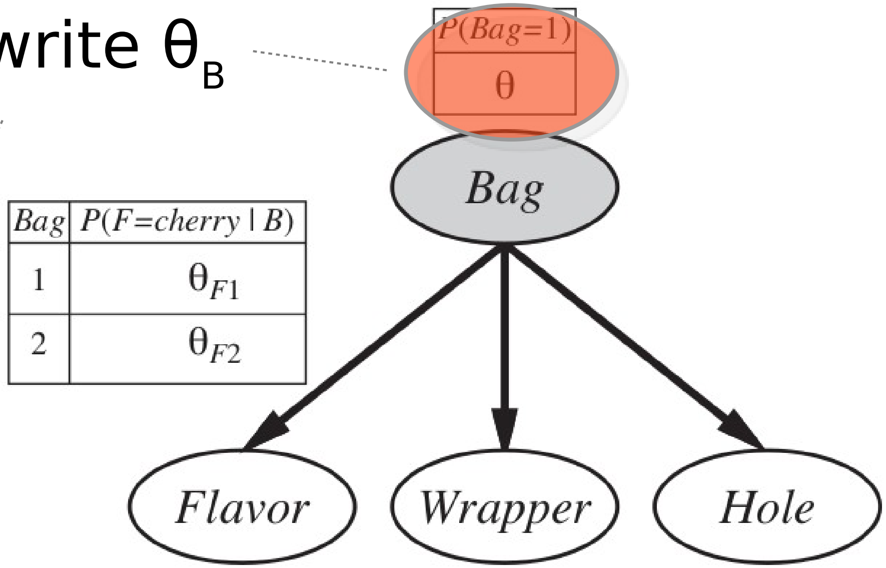
- the observed variables of a data point i are $X_i = \langle Flavor, Wrapper, Hole \rangle$
- the hidden variable $Z_i = Bag$, which takes values $z_i \in \{1,2\}$
- the parameters $\theta = \left\langle \theta_B, \{\theta_{Fx}, \theta_{Wx}, \theta_{Hx}\}_{x=1,2} \right\rangle$ encode the CTPs:

$$\theta_B = P(Bag = x), \quad \theta_{Fx} = P(Flavor = cherry | Bag = x), \text{ etc.}$$

- We write $\boldsymbol{x} = \langle x_i \rangle_{i=1}^N$, $\boldsymbol{z} = \langle z_i \rangle_{i=1}^N$ are the vectors of observed resp. hidden values for all data points,

The Flavor BN

I write θ_B



- 2 Bags of candy got mixed!

In the "flavor BN", we have

- the observed variables of a data point i are $X_i = \langle Flavor, Wrapper, Hole \rangle$
- the hidden variable $Z_i = Bag$, which takes values $z_i \in \{1,2\}$
- the parameters $\theta = \left\langle \theta_B, \{\theta_{Fx}, \theta_{Wx}, \theta_{Hx}\}_{x=1,2} \right\rangle$ encode the CTPs:

$$\theta_B = P(Bag = x), \quad \theta_{Fx} = P(Flavor = cherry | Bag = x), \text{ etc.}$$

- We write $\boldsymbol{x} = \langle x_i \rangle_{i=1}^N$, $\boldsymbol{z} = \langle z_i \rangle_{i=1}^N$ are the vectors of observed resp. hidden values for all data points,

Optimize Log Likelihood…

- Due to hidden variable, directly optimizing log likelihood is hard:

In this example the *data log-likelihood* is

$$
\begin{aligned}
L(\boldsymbol{x}|\theta) = \log \Pr(\boldsymbol{x}|\theta) &= \log \sum_{\boldsymbol{z}} \Pr(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}|\theta) \\
&= \log \sum_{\boldsymbol{z}} \prod_{i=1}^{N} \Pr(x_i, Z_i = z_i|\theta) \\
&= \log \sum_{\boldsymbol{z}} \prod_{i=1}^{N} \Pr(z_i|\theta_B) \Pr(flavor_i|z_i, \theta_{Fz_i}) \Pr(wrapper_i|z_i, \theta_{Wz_i}) \Pr(hole_i|z_i, \theta_{Hz_i})
\end{aligned}
$$

TU Delft

Optimize Log Likelihood…

- Due to hidden variable, directly optimizing log likelihood is hard:

In this example the *data log-likelihood* is

$$
\begin{aligned}
L(\boldsymbol{x}|\theta) = \log \Pr(\boldsymbol{x}|\theta) &= \log \sum_{\boldsymbol{z}} \Pr(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}|\theta) \\
&= \log \sum_{\boldsymbol{z}} \prod_{i=1}^{N} \Pr(x_i, Z_i = z_i|\theta) \\
&= \log \sum_{\boldsymbol{z}} \prod_{i=1}^{N} \Pr(z_i|\theta_B) \Pr(flavor_i|z_i, \theta_{Fz_i}) \Pr(wrapper_i|z_i, \theta_{Wz_i}) \Pr(hole_i|z_i, \theta_{Hz_i})
\end{aligned}
$$

log-of-sum does not decompose in smaller terms…
…everything is coupled in a big messy expression :(

If the problem was observable…

The EM algorithm iterates:

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

which uses the *full (or 'completed') log-likelihood:*

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{z}|\theta) &= \log \Pr(\boldsymbol{x},\boldsymbol{z}|\theta) = \log \prod_{i=1}^{N} \Pr(x_i,z_i|\theta) \\ &= \sum_{i=1}^{N} \log \Pr(x_i,z_i|\theta) \\ &= \sum_{i=1}^{N} \log \Pr(z_i|\theta_B) + \sum_{i=1}^{N} \log \Pr(flavor_i|z_i,\theta_{Fz_i}) \\ &\quad + \sum_{i=1}^{N} \log \Pr(wrapper_i|z_i,\theta_{Wz_i}) + \sum_{i=1}^{N} \log \Pr(hole_i|z_i,\theta_{Hz_i}) \end{aligned}$$

This term (indeed assuming we know $\boldsymbol{z}$) is easy to optimize: the parameters are *localized*: e.g., in order to optimize θ_B, we only need to consider the first term $\sum_{i=1}^{N} \log \Pr(z_i|\theta_B)$, the other terms are not affected by θ_B and there are no other parameters that affect it.

If the problem was observable…

The EM algorithm iterates:

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

which uses the *full (or 'completed') log-likelihood:*

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{z}|\theta) &= \log \Pr(\boldsymbol{x},\boldsymbol{z}|\theta) = \log \prod_{i=1}^{N} \Pr(x_i,z_i|\theta) \\ &= \sum_{i=1}^{N} \log \Pr(x_i,z_i|\theta) \\ &= \sum_{i=1}^{N} \log \Pr(z_i|\theta_B) + \sum_{i=1}^{N} \log \Pr(flavor_i|z_i,\theta_{Fz_i}) \\ &\quad + \sum_{i=1}^{N} \log \Pr(wrapper_i|z_i,\theta_{Wz_i}) + \sum_{i=1}^{N} \log \Pr(hole_i|z_i,\theta_{Hz_i}) \end{aligned}$$

let's zoom in on this term

This term (indeed assuming we know $\boldsymbol{z}$) is easy to optimize: the parameters are *localized*: e.g., in order to optimize θ_B, we only need to consider the first term $\sum_{i=1}^{N} \log \Pr(z_i|\theta_B)$, the other terms are not affected by θ_B and there are no other parameters that affect it.

TUDelft

… we could easily update θ_B

- Can rewrite as follows:

$$\begin{aligned}\sum_{i=1}^{N} \log \Pr(z_i|\theta_B) &= \sum_{i \text{ s.t. } z_i=1} \log \Pr(z_i|\theta_B) + \sum_{i \text{ s.t. } z_i=2} \log \Pr(z_i|\theta_B) \\ &= \sum_{i \text{ s.t. } z_i=1} \log \theta_B + \sum_{i \text{ s.t. } z_i=2} \log(1-\theta_B) \\ &= N_1 \log \theta_B + N_2 \log(1-\theta_B) \\ &= N_1 \log \theta_B + (N - N_1) \log(1-\theta_B)\end{aligned}$$

Where $N_1 = N(bag = 1|\boldsymbol{z})$.

If $\boldsymbol{z}$ was correct, this would then lead (by taking derivative and setting to 0) to

$$\theta_B \leftarrow \frac{N_1}{N}$$

TUDelft

... we could easily u

- Can rewrite as follows:

$$\sum_{i=1}^{N} \log \Pr(z_i|\theta_B) = \sum_{i \text{ s.t. } z_i=1} \log \Pr$$
$$= \sum_{i \text{ s.t. } z_i=1} \log \theta_B$$
$$= N_1 \log \theta_B + N_2$$
$$= N_1 \log \theta_B + (N$$

Where $N_1 = N(bag = 1|\boldsymbol{z})$.

If $\boldsymbol{z}$ was correct, this would then le

to 0) to

$$\theta_B \leftarrow \frac{N_1}{N}$$

Maximum likelihood Bernoulli (20.2.1)

$$\begin{aligned} L(\theta) &= \log \prod_{i=1}^{k} \theta \prod_{i=1}^{l} (1-\theta) \\ &= \log \theta^k (1-\theta)^l \\ &= k \log \theta + l \log(1-\theta) \end{aligned}$$

Its derivative:

$$\frac{d}{d\theta} L(\theta) = \frac{k}{\theta} - \frac{l}{(1-\theta)}$$

equating with 0 and solving to find the maximum:

$$\begin{aligned} &\frac{k}{\theta} - \frac{l}{(1-\theta)} = 0 \\ \Leftrightarrow &\frac{k}{\theta} = \frac{l}{(1-\theta)} \\ \Leftrightarrow &k(1-\theta) = l\theta \\ \Leftrightarrow &k = (l+k)\theta \\ \Leftrightarrow &\frac{k}{l+k} = \theta = \frac{k}{N} \end{aligned}$$

We don't know *z*, but...

- ...this is why EM takes the expectation w.r.t. $P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$.
- Again, let's continue to focus on updating θ_B.

We don't know *z*, but...

- ...this is why EM takes the expectation w.r.t. $P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$.
- Again, let's continue to focus on updating θ_B.
- The EM rule

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

translates to:

$$\theta_B^{(k+1)} = \arg\max_{\theta_B} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) \left[N_1 \log\theta_B + (N - N_1)\log(1-\theta_B)\right]$$

We don't know *z*, but…

- …this is why EM takes the expectation w.r.t. $P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$.
- Again, let's continue to focus on updating θ_B.
- The EM rule

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

translates to:

$$\theta_B^{(k+1)} = \arg\max_{\theta_B} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) \left[N_1 \log\theta_B + (N - N_1)\log(1-\theta_B)\right]$$

and we see that the dependence on $\boldsymbol{z}$ is *only* via the counts N_1…

We don't know *z*, but…

- .. so we can re-write:

$$\sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})\left[N_1 \log\theta_B + (N-N_1)\log(1-\theta_B)\right]$$

$$= \sum P(N_1 = n|\boldsymbol{x},\theta^{(k)})\left[n \log\theta_B + (N-n)\log(1-\theta_B)\right]$$

<etc…, check hand out.>

$$= \log\theta_B \cdot \hat{N}(Bag=1) + \log(1-\theta_B)\cdot(N - \hat{N}(Bag=1))$$

with $\hat{N}(Bag=1)$ is the *expected* counts for bag 1.

- Finally, this leads (by taking derivative and setting to 0) to

$$\theta_B^{(k+1)} \leftarrow \frac{\hat{N}(Bag=1)}{N}$$

We don't know *z*, but...

- .. so we can re-write:

$$\sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) \left[N_1 \log \theta_B + (N - N_1) \log(1 - \theta_B)\right]$$

$$= \sum P(N_1 = n|\boldsymbol{x},\theta^{(k)}) \left[n \log \theta_B + (N - n) \log(1 - \theta_B)\right]$$

<etc..., check hand out.>

$$= \log \theta_B \cdot \hat{N}(Bag = 1) + \log(1 - \theta_B) \cdot (N - \hat{N}(Bag = 1))$$

with $\hat{N}(Bag = 1)$ is the *expected* counts for bag 1.

using BN inference; see book !

- Finally, this leads (by taking derivative and setting to 0) to

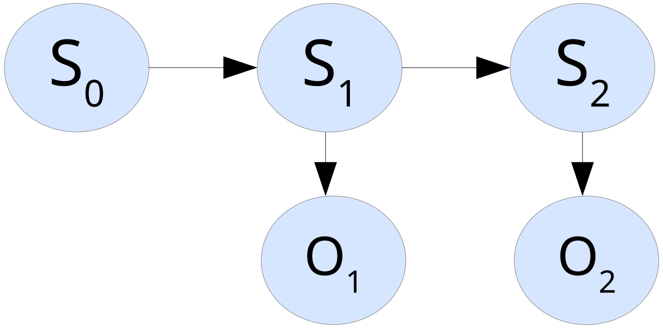
$$\theta_B^{(k+1)} \leftarrow \frac{\hat{N}(Bag = 1)}{N}$$

EM for HMMs

(R&N: 20.3.3)

So how about HMMs...?

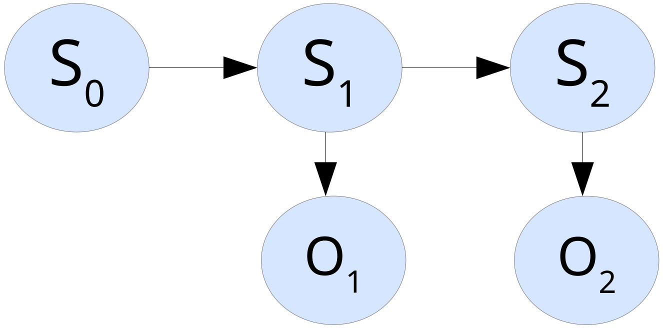
- Idea is going to be the same...
  - write down general EM rule
  - start filling out the “completed log-likelihood”
  - do puzzling: information we really need to take the expectation over **z**



TU Delft

HMM learning set up

- $\boldsymbol{x} = \{(o_{i1}, o_{i2}, \ldots, o_{iT})\}_{i=1}^{N}$ is the set of N trajectories of observations of the form $x_i = (o_{i1}, o_{i2}, \ldots, o_{iT})$.
- $\boldsymbol{z} = \{(s_{i0}, s_{i1}, s_{i2}, \ldots, s_{iT})\}_{i=1}^{N}$ is the set of N trajectories of hidden states of the form $z_i = (s_{i0}, s_{i1}, s_{i2}, \ldots, s_{iT})$.



HMM learning set up

- $\boldsymbol{x} = \{(o_{i1}, o_{i2}, \ldots, o_{iT})\}_{i=1}^{N}$ is the set of N trajectories of observations of the form $x_i = (o_{i1}, o_{i2}, \ldots, o_{iT})$.
- $\boldsymbol{z} = \{(s_{i0}, s_{i1}, s_{i2}, \ldots, s_{iT})\}_{i=1}^{N}$ is the set of N trajectories of hidden states of the form $z_i = (s_{i0}, s_{i1}, s_{i2}, \ldots, s_{iT})$.

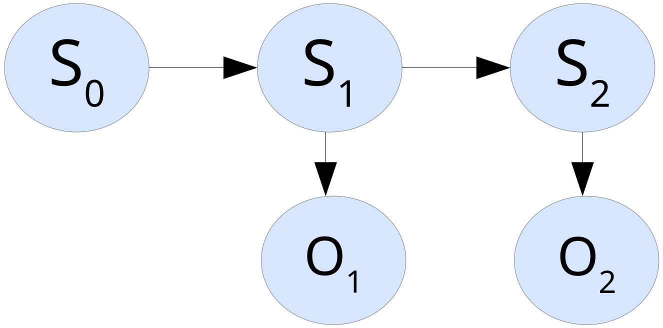
- The joint probability defined by an HMMs:

$$P(\boldsymbol{x},\boldsymbol{z}|\theta) = P(s_0)\prod_{t=1}^{T} P(s_t|s_{t-1},\theta)P(o_t|s_t,\theta)$$
$$= \theta_{s_0}^{init}\prod_{t=1}^{T} \theta_{s_{t-1}\to s_t}^{trans}\theta_{s_t\to o_t}^{obs}$$

where

$$\theta_{s_{t-1}\to s_t}^{trans} \triangleq P(s_t|s_{t-1})$$
$$\theta_{s_t\to o_t}^{obs} \triangleq P(o_t|s_t)$$

are the parameters for the transition and observation probabilities.



EM iterates...

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

- The *full (or 'completed') log-likelihood:*

$$L(\boldsymbol{x},\boldsymbol{z}|\theta) = \log\theta^{init}_{s_0} + \sum_{t=1}^{T} \log\theta^{trans}_{s_{t-1}\to s_t} + \sum_{t=1}^{T} \log\theta^{obs}_{s_t\to o_t}.$$

EM iterates...

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

- The *full (or 'completed') log-likelihood:*

$$L(\boldsymbol{x},\boldsymbol{z}|\theta) = \log \theta^{init}_{s_0} + \sum_{t=1}^{T} \log \theta^{trans}_{s_{t-1}\to s_t} + \sum_{t=1}^{T} \log \theta^{obs}_{s_t\to o_t}.$$

- Let's find the new transition probabilities of some state y

EM iterates...

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

- The *full (or 'completed') log-likelihood:*

$$L(\boldsymbol{x},\boldsymbol{z}|\theta) = \log \theta^{init}_{s_0} + \sum_{t=1}^{T} \log \theta^{trans}_{s_{t-1}\to s_t} + \sum_{t=1}^{T} \log \theta^{obs}_{s_t \to o_t}.$$

- Let's find the new transition probabilities of some state y
- Our goal is to maximize

$$\begin{aligned}\theta^{trans(k+1)}_{y\to\cdot} &= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) L(\boldsymbol{x},\boldsymbol{z}|\theta) \\ &= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) \left[\sum_{t=1}^{T} \log \theta^{trans}_{s_{t-1}\to s_t} \right]\end{aligned}$$

subject to $\sum_z \theta^{trans}_{y\to z} = 1$.

EM iterates...

$$\theta^{(k+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

- The *full (or 'completed') log-likelihood:*

$$L(\boldsymbol{x},\boldsymbol{z}|\theta) = \log\theta^{init}_{s_0} + \sum_{t=1}^{T} \log\theta^{trans}_{s_{t-1}\rightarrow s_t} + \sum_{t=1}^{T} \log\theta^{obs}_{s_t\rightarrow o_t}.$$

- Let's find the new transition probabilities of some state y
- Our goal is to maximize

$$\theta^{trans(k+1)}_{y\rightarrow\cdot} = \arg\max_{\theta^{trans}_{y\rightarrow\cdot}} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})L(\boldsymbol{x},\boldsymbol{z}|\theta)$$

$$= \arg\max_{\theta^{trans}_{y\rightarrow\cdot}} \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)}) \left[\sum_{t=1}^{T} \log\theta^{trans}_{s_{t-1}\rightarrow s_t}\right]$$

subject to $\sum_z \theta^{trans}_{y\rightarrow z} = 1.$

What part of z do we need?

TU Delft

What part of *z* do we need?

- Let us abbreviate: $Q^{(k+1)}(\boldsymbol{z}) = P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$
- Then:

$$
\begin{aligned}
&\arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} Q^{(k+1)}(\boldsymbol{z}) \left[\sum_{t=1}^{T} \log \theta^{trans}_{s_{t-1}\to s_t}\right] \\
&= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} Q^{(k+1)}(\boldsymbol{z}) \left[\sum_{t \text{ s.t. } \{S_{i,t-1}=y\}} \log \theta^{trans}_{y\to s_t}\right] \\
&= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{\substack{\boldsymbol{z} \text{ s.t.} \\ S_{t-1}=y}} Q^{(k+1)}(\boldsymbol{z}) \log \theta^{trans}_{y\to s_t} \\
&= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{(s_0\ldots s_{t-2},s_t\ldots s_T)} Q^{(k+1)}(s_0\ldots s_{t-1},S_{t-1}=y,s_t\ldots s_T) \log \theta^{trans}_{y\to s_t} \\
&= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{v} \sum_{(s_0\ldots s_{t-2},s_{t+1}\ldots s_T)} Q^{(k+1)}(s_0\ldots,S_{t-1}=y,S_t=v,\ldots s_T) \log \theta^{trans}_{y\to v} \\
&= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{v} Q^{(k+1)}(S_{t-1}=y,S_t=v) \log \theta^{trans}_{y\to v}
\end{aligned}
$$

- To update the parameters $\theta^{trans}_{y\to\cdot}$ for transitioning from y, we only need to estimate the marginal probabilities $Q^{(k+1)}(S_{t-1}=y,S_t=v)$

TU Delft

What part of *z* do we need?

- Let us abbreviate: $Q^{(k+1)}(\boldsymbol{z}) = P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$
- Then:

$$\arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} Q^{(k+1)}(\boldsymbol{z}) \left[\sum_{t=1}^{T} \log \theta^{trans}_{s_{t-1}\to s_t}\right]$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} Q^{(k+1)}(\boldsymbol{z}) \left[\sum_{t \text{ s.t. } \{S_{i,t-1}=y\}} \log \theta^{trans}_{y\to s_t}\right]$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{\substack{\boldsymbol{z} \text{ s.t.}\\ S_{t-1}=y}} Q^{(k+1)}(\boldsymbol{z}) \log \theta^{trans}_{y\to s_t}$$

check at home

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{(s_0\ldots s_{t-2},s_t\ldots s_T)} Q^{(k+1)}(s_0\ldots s_{t-1},S_{t-1}=y,s_t\ldots s_T) \log \theta^{trans}_{y\to s_t}$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{v} \sum_{(s_0\ldots s_{t-2},s_{t+1}\ldots s_T)} Q^{(k+1)}(s_0\ldots,S_{t-1}=y,S_t=v,\ldots s_T) \log \theta^{trans}_{y\to v}$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{v} Q^{(k+1)}(S_{t-1}=y,S_t=v) \log \theta^{trans}_{y\to v}$$

- To update the parameters $\theta^{trans}_{y\to\cdot}$ for transitioning from y, we only need to estimate the marginal probabilities $Q^{(k+1)}(S_{t-1}=y,S_t=v)$

What part of *z* do we need?

- Let us abbreviate: $Q^{(k+1)}(\boldsymbol{z}) = P(\boldsymbol{z}|\boldsymbol{x},\theta^{(k)})$
- Then:

$$\arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} Q^{(k+1)}(\boldsymbol{z}) \left[\sum_{t=1}^{T} \log \theta^{trans}_{s_{t-1}\to s_t}\right]$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{\boldsymbol{z}} Q^{(k+1)}(\boldsymbol{z}) \left[\sum_{t \text{ s.t. } \{S_{i,t-1}=y\}} \log \theta^{trans}_{y\to s_t}\right]$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{\substack{\boldsymbol{z} \text{ s.t.} \\ S_{t-1}=y}} Q^{(k+1)}(\boldsymbol{z}) \log \theta^{trans}_{y\to s_t}$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{(s_0\ldots s_{t-2},s_t\ldots s_T)} Q^{(k+1)}(s_0\ldots s_{t-1},S_{t-1}=y,s_t\ldots s_T) \log \theta^{trans}_{y\to s_t}$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{v} \sum_{(s_0\ldots s_{t-2},s_{t+1}\ldots s_T)} Q^{(k+1)}(s_0\ldots,S_{t-1}=y,S_t=v,\ldots s_T) \log \theta^{trans}_{y\to v}$$

$$= \arg\max_{\theta^{trans}_{y\to\cdot}} \sum_{t} \sum_{v} Q^{(k+1)}(S_{t-1}=y,S_t=v) \log \theta^{trans}_{y\to v}$$

M-Step:
similar to
max. likelihood
estimation before:

$\theta_{u\to v} = \Sigma_t Q(u,v) \,/\, \Sigma_t Q(u)$

- To update the parameters $\theta^{trans}_{y\to\cdot}$ for transitioning from y, we only need to estimate the marginal probabilities $Q^{(k+1)}(S_{t-1}=y,S_t=v)$

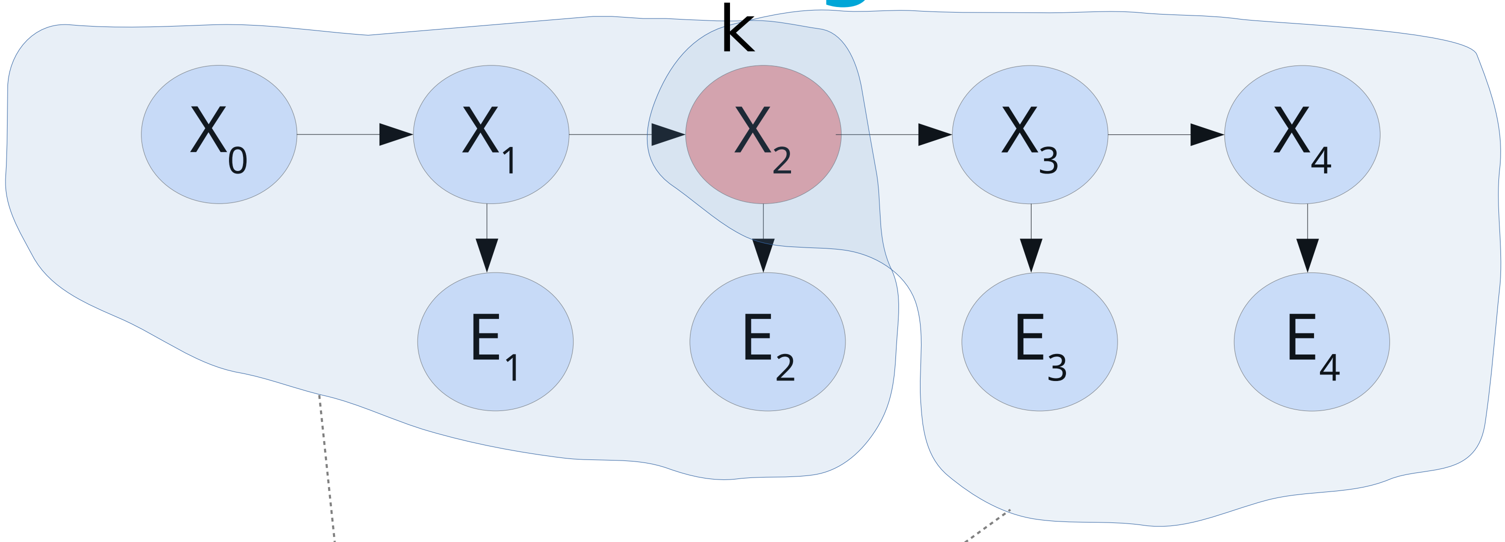
Computing the pairwise marginals...?

- Need to compute $Q(s_{t-1},s_t)=P(s_{t-1},s_t|o_{1:T},\theta^{(\kappa)})$...
- How?

- Modify the smoothing algorithm to compute these!
  - why smoothing?

TU Delft

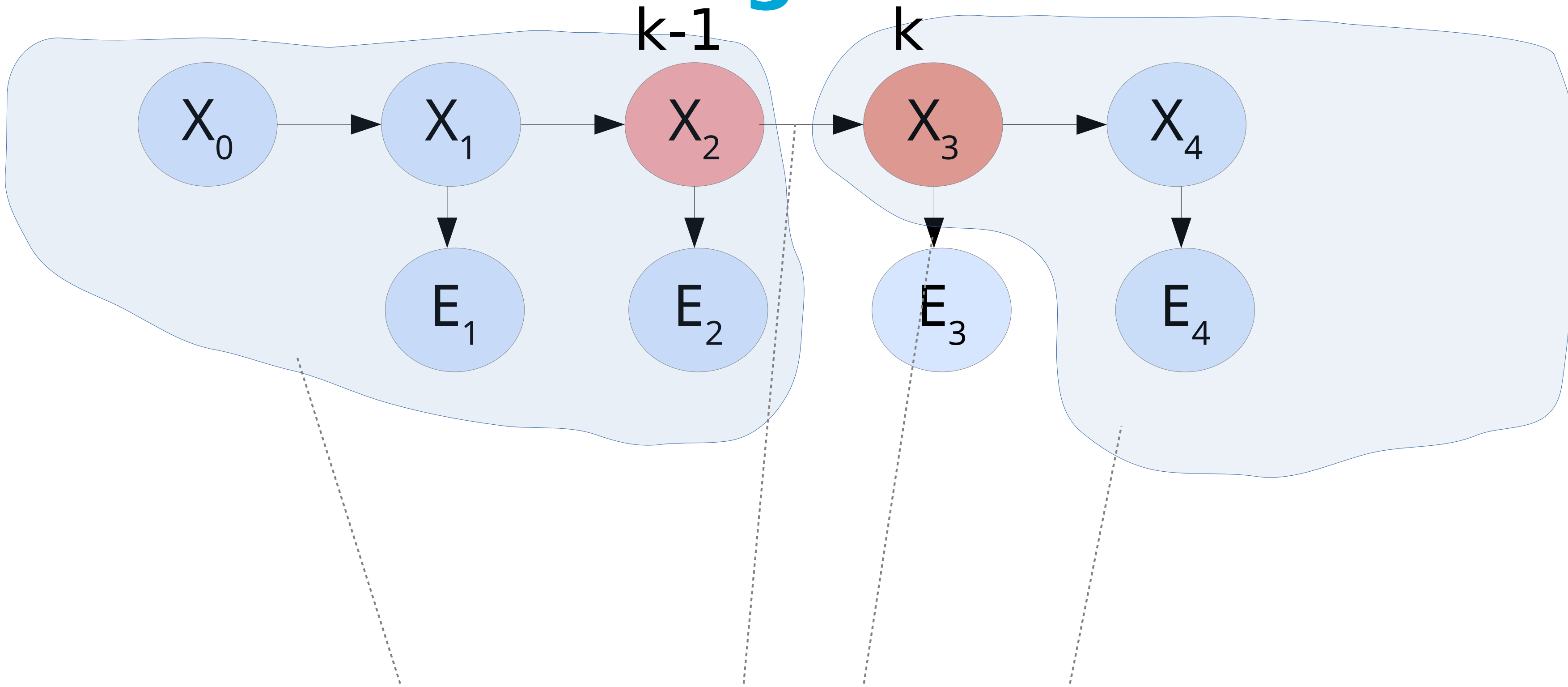
Normal Smoothing



$$
\begin{aligned}
\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k},\mathbf{e}_{k+1:t}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{e}_{1:k}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \\
&= \alpha\mathbf{f}_{1:k}\mathbf{b}_{k+1:t}
\end{aligned}
$$

TU Delft

Normal Smoothing



$$Q(x_{k-1},x_k)=\alpha\, \mathbf{f}_{1:k-1}(x_{k-1})\, P(x_k|x_{k-1})P(e_k|x_k)b_{k+1:t}(x_k)$$

Summary: EM for HMMs

- Iteratively apply EM rule as always…
- E-step: compute pairwise marginal probabilities
  - using modification of smoothing
- M-step: usual count ratios…
  - …but based on the marginal probabilities

- Limitations:
  - local optima
  - overfitting… (what can be done…?)

- But very **widely used** in ML and statistics!

Summary Learning

- Learning, ***induction***, is an important AI technique
  - no model available, adaptivity, too difficult to program
- Different learning…
  - …perspectives: idealistic vs. pragmatic
  - …task settings: supervised, unsupervised, etc.
- Problem of induction… $h \approx f$?
  - test error or theorems

- Learning with maximum likelihood
  - a binary variable (Bernoulli)
  - a (fully observed) Bayesian network
- Learning with hidden variables
  - Can lead to more compact models
  - EM algorithm
    - general form, Bayesian networks, HMMs

- Now, your robot can learn a model of change over time!