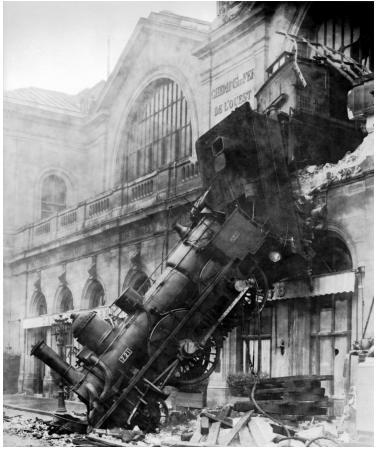
Machine Learning 1.11: Graphical Models

Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

Derailment

- Predict derailment due to "*overspeed on sharp curves*"
  (technical term for rolling a train)



Derailment

- Predict derailment due to "*overspeed on sharp curves*"
  (technical term for rolling a train)

- Collect data:

Driver on duty (hours)	6	9	0	8	1	10
Lateness (%)	13	0	1	0	0	0
Speed Limit (km/h)	30	30	30	10	20	10
Radius (meters)	150	180	180	40	100	40
Train Age (years)	8	31	12	33	0	30
Track Age (years)	2	9	3	0	6	0
Crashed	0	0	0 (simulated)	1	1	1

Derailment

- Predict derailment due to "*overspeed on sharp curves*"
  (technical term for rolling a train)

- Collect data:

Driver on duty (hours)	6	9	0	8	1	10
Lateness (%)	13	0	1	0	0	0
Speed Limit (km/h)	30	30	30	10	20	10
Radius (meters)	150	180	180	40	100	40
Train Age (years)	8	31	12	33	0	30
Track Age (years)	2	9	3	0	6	0
Crashed	0	0	0 (simulated)	1	1	1

- Could just use classification, but...
- We know **structure**:
  - First 3 parameters predict **speed**
  - Speed plus last 3 predict **crash**

Structure

- Instead of learning

$$P(\text{crashed}|6 \text{ features})$$

learn

$$P(\text{crashed}|3 \text{ features}, \text{speed})P(\text{speed}|3 \text{ features})$$

- Note: Traditionally probabilistic, but doesn't have to be, e.g. neural networks

UNIVERSITY OF BATH

Why?

- Why is this useful?
  - Less dimensions $\implies$ reduces curse of dimensionality
    An explicit *manifold*

UNIVERSITY OF BATH

Why?

- Why is this useful?
  - Less dimensions $\implies$ reduces curse of dimensionality
    An explicit *manifold*
  - More data, e.g.
    - $P(\text{speed}|3 \text{ features})$ – large data set
    - $P(\text{crashed}|3 \text{ features}, \text{speed})$ – small data set

    (not a problem if imbalanced)

Why?

- Why is this useful?
  - Less dimensions $\implies$ reduces curse of dimensionality
    An explicit *manifold*
  - More data, e.g.
    - $P(\text{speed}|\text{3 features})$ – large data set
    - $P(\text{crashed}|\text{3 features}, \text{speed})$ – small data set

    (not a problem if imbalanced)
  - Supports missing data
    - Runtime, e.g. $P(\text{crashed})$ when *driver on duty* unknown
    - Training, e.g. Exemplars don't always include *train age*
    - Latent, e.g. Speed is never known

    (going to ignore last case for now)

UNIVERSITY OF BATH

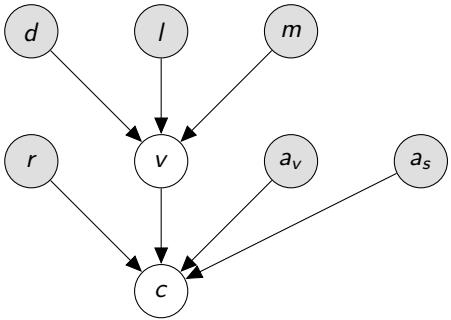
Why?

- Why is this useful?
  - Less dimensions $\implies$ reduces curse of dimensionality
    An explicit *manifold*
  - More data, e.g.
    - $P(\text{speed}|\text{3 features})$ – large data set
    - $P(\text{crashed}|\text{3 features}, \text{speed})$ – small data set

    (not a problem if imbalanced)
  - Supports missing data
    - Runtime, e.g. $P(\text{crashed})$ when *driver on duty* unknown
    - Training, e.g. Exemplars don't always include *train age*
    - Latent, e.g. Speed is never known

    (going to ignore last case for now)
  - Improved accuracy (hard removal of possibilities)
  - Faster runtime (maybe)

Visualising dependency



Arrows = Dependency

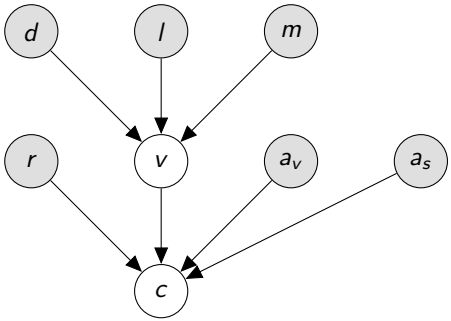
d = Driver on duty (hours)
l = Lateness (%)
m = Speed Limit (km/h)

v = Speed (km/h)

r = Radius (meters)
a_v = Train Age (years)
a_s = Track Age (years)

c = Crashed

Visualising dependency



Arrows = Dependency

d = Driver on duty (hours)
l = Lateness (%)
m = Speed Limit (km/h)

v = Speed (km/h)

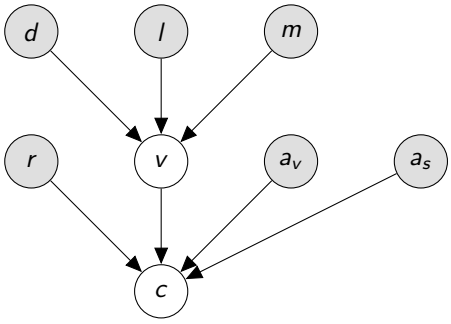
r = Radius (meters)
a_v = Train Age (years)
a_s = Track Age (years)

c = Crashed

This is a **graphical model**

UNIVERSITY OF BATH

Graphical models



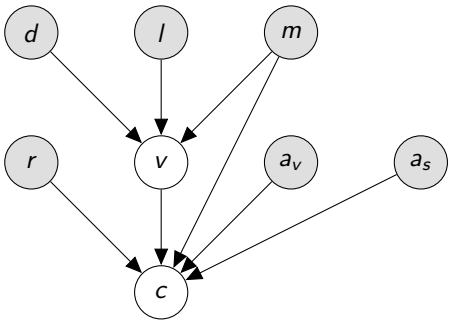
- **Structure = conditional independence**

$$P(\text{crashed} | \text{3 features}, \text{speed})$$

$$P(\text{speed} | \text{3 features})$$

- If we know v (speed) then m (speed limit) gives us **no extra information** about c (crash)

Graphical models



- **Structure = conditional independence**

$$P(\text{crashed}|\text{3 features}, \text{speed})$$

$$P(\text{speed}|\text{3 features})$$

- If we know v (speed) then m (speed limit) gives us **no extra information** about c (crash)

- Structure is in the **omitted edges**
- Connect m to c and above no longer true

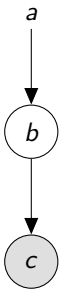
This lecture

- Thinking about graphical models
- Graphical models can be problem specific

- High level – algorithms in next two lectures

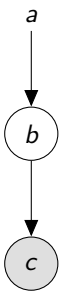
UNIVERSITY OF BATH

Conventions



- Circle means **random variable** (b, c) – variable
- No circle means (hyper-)**parameter** (a) – fixed

Conventions



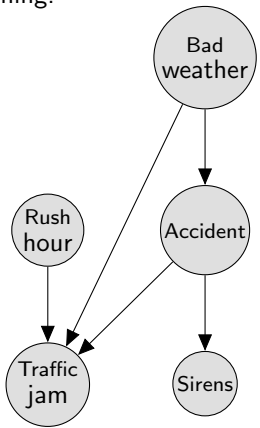
- Circle means **random variable** (b, c) – variable
- No circle means (hyper-)**parameter** (a) – fixed

- Shaded means **observed** (c)
- Unshaded means **unobserved** (b)

Modes

- Observed/unobserved changes with each stage:

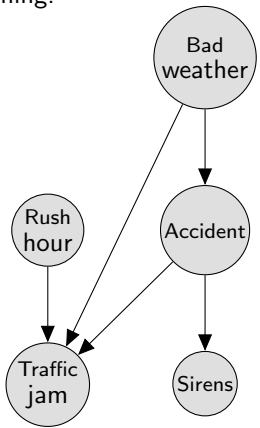
Training:



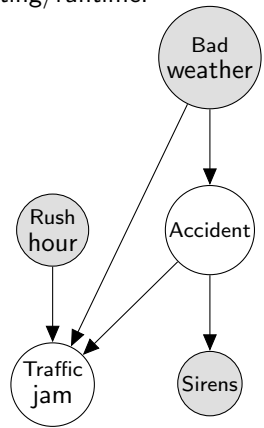
Modes

- Observed/unobserved changes with each stage:

Training:



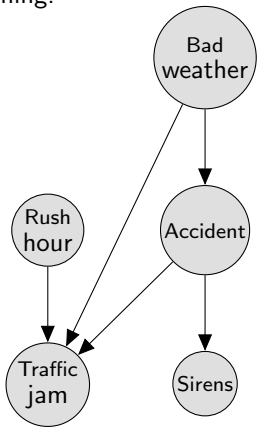
Testing/runtime:



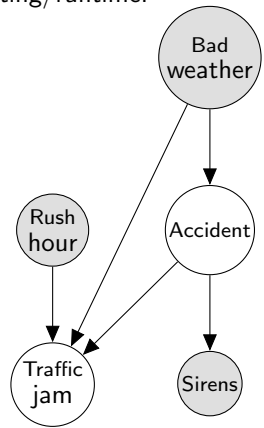
Modes

- Observed/unobserved changes with each stage:

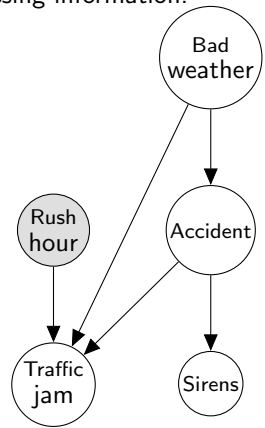
Training:



Testing/runtime:



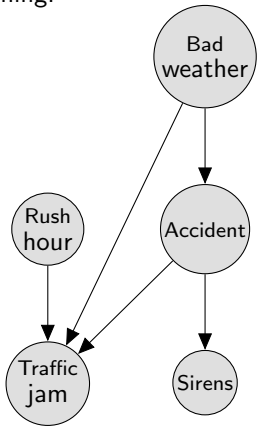
Missing information:



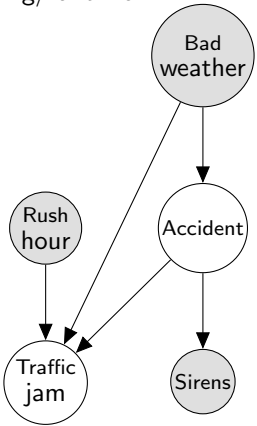
Modes

- Observed/unobserved changes with each stage:

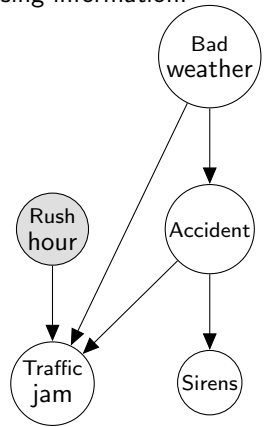
Training:



Testing/runtime:



Missing information:



- Usually only present training model

UNIVERSITY OF BATH

Latent variables

- Latent variable = always unobserved RV
  (focus of later lecture)
- Two kinds:
  - **Hidden** – Something we know exists, but can't measure
  - **Hypothetical** – Might not actually exist

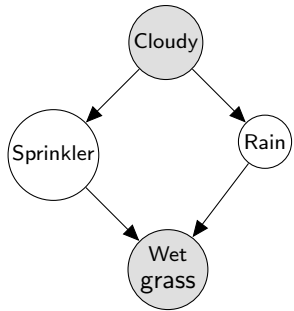
UNIVERSITY OF BATH

Three representations

- Bayesian networks
- Factor graphs
- Markov random fields

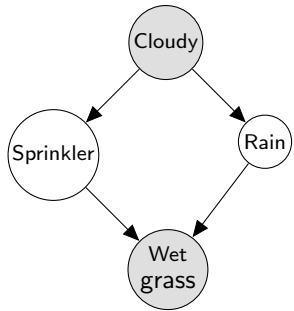
Bayesian networks

- All earlier examples
- Intuitive: $x \rightarrow y$ means x *causes* y (typically)
- Also called **Bayes network** or **Belief network**
- Not always Bayesian! (despite name)



Bayesian networks

- All earlier examples
- Intuitive: $x \rightarrow y$ means x *causes* y (typically)
- Also called **Bayes network** or **Belief network**
- Not always Bayesian! (despite name)
- Equation:
  - Product of terms
  - One per RV: $P(x|\text{arrows pointing at } x)$
  - e.g. $P(w|r,s)P(s|c)P(r|c)P(c)$
    $c = \text{Cloudy}$
    $s = \text{Sprinkler}$
    $r = \text{Rain}$
    $w = \text{Wet grass}$



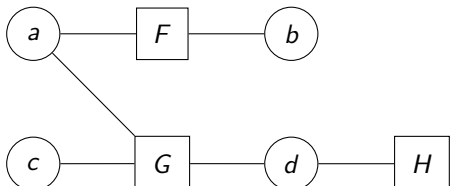
UNIVERSITY OF BATH

Factor graphs

- Product of arbitrary functions
- First used for "*Boolean satisfiability problem*"
  (SAT, NP-complete, important to P vs NP)

Factor graphs

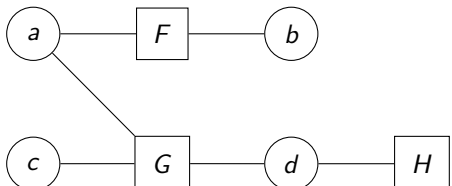
- Product of arbitrary functions
- First used for "*Boolean satisfiability problem*"
  (SAT, NP-complete, important to P vs NP)



$$= F(a, b)G(a, c, d)H(d)$$

Factor graphs

- Product of arbitrary functions
- First used for "*Boolean satisfiability problem*"
  (SAT, NP-complete, important to P vs NP)

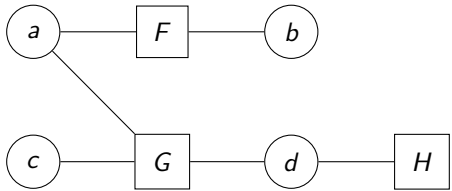


$$= F(a, b)G(a, c, d)H(d)$$

- Circles = RVs (as before)
- Squares = functions
- Edges link functions to dependent variables

Factor graphs

- Product of arbitrary functions
- First used for "*Boolean satisfiability problem*"
  (SAT, NP-complete, important to P vs NP)



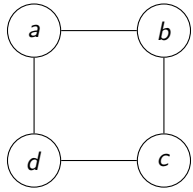
$$= F(a, b)G(a, c, d)H(d)$$

- Circles = RVs (as before)
- Squares = functions
- Edges link functions to dependent variables

- Completely general: No requirement to be probabilistic
- But commonly represents unnormalised probability

Markov random fields

- Also called **Markov networks**
- Factor graph without boxes!
- Only functions of two variables
- Functions of one variable implicit



$$= F(a,b)G(b,c)H(c,d)I(d,a)\ A(a)B(b)C(c)D(d)$$

- Function names:
  - F, G, H, I: Pairwise terms (function of two variables)
  - A, B, C, D: Unary terms (function of one variable)

Which representation?

- Whichever is convenient
- If it makes sense: Multiple representations at same time!
- Code uses factor graphs...

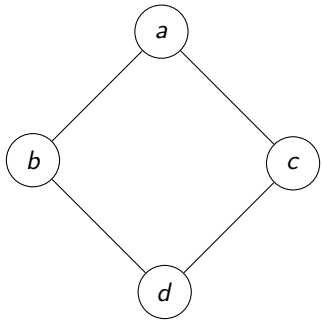
Conversion

- To factor graph always possible
- To others is not

Conversion

- To factor graph always possible
- To others is not

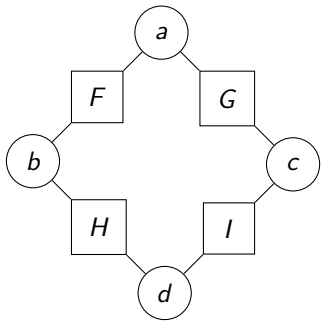
- Markov network to factor graph:



Conversion

- To factor graph always possible
- To others is not

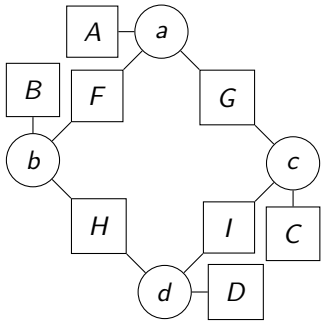
- Markov network to factor graph:
  1. Add factor to each edge



Conversion

- To factor graph always possible
- To others is not

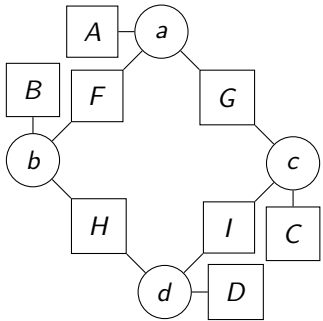
- Markov network to factor graph:
  1. Add factor to each edge
  2. Add factors for implicit unary terms



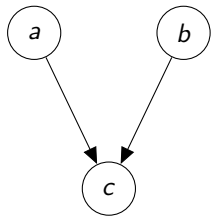
Conversion

- To factor graph always possible
- To others is not

- Markov network to factor graph:
  1. Add factor to each edge
  2. Add factors for implicit unary terms



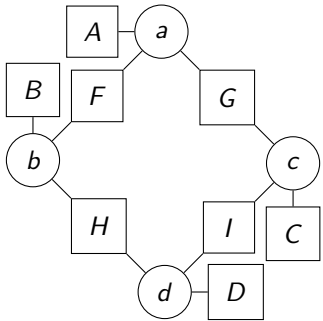
- Bayesian network to factor graph:



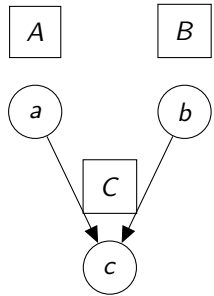
Conversion

- To factor graph always possible
- To others is not

- Markov network to factor graph:
  1. Add factor to each edge
  2. Add factors for implicit unary terms



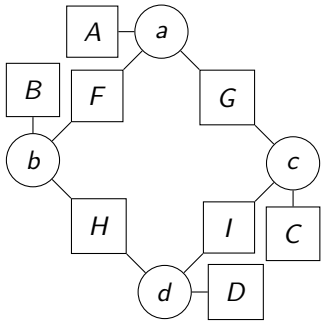
- Bayesian network to factor graph:
  - Add factor for each RV



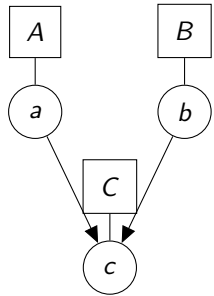
Conversion

- To factor graph always possible
- To others is not

- Markov network to factor graph:
  1. Add factor to each edge
  2. Add factors for implicit unary terms



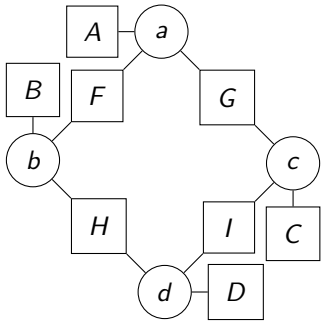
- Bayesian network to factor graph:
  - Add factor for each RV
  - Add edges to RVs



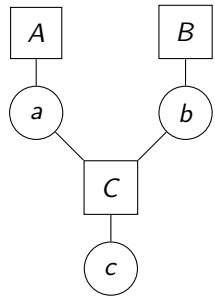
Conversion

- To factor graph always possible
- To others is not

- Markov network to factor graph:
  1. Add factor to each edge
  2. Add factors for implicit unary terms



- Bayesian network to factor graph:
  - Add factor for each RV
  - Add edges to RVs
  - Convert RV arrows to factor edges



UNIVERSITY OF BATH

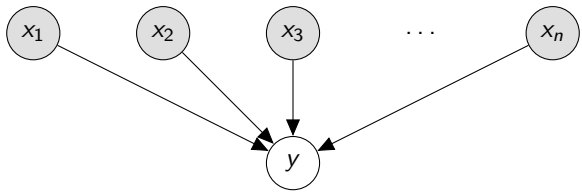
Examples

I Classification & regression
II Naive Bayes
III Expert systems
IV Markov chain
V Markov grid

(ignoring entire graphical model categories... for now)

Example I: Classification & regression

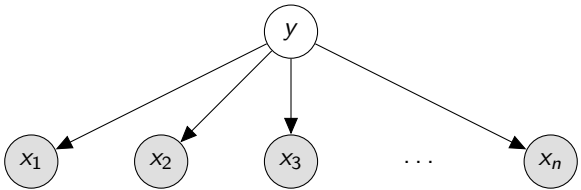
- Graphical model for normal classification/regression:



- Everything connected $\therefore$ no conditional independence $\therefore$ no structure
- Pointless!

Example II: Naive Bayes

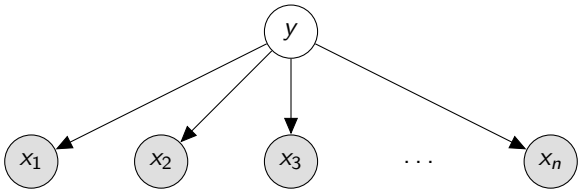
- Reverse graphical model for classification/regression:



- Opposite: Everything independent!
- May make more sense: $y =$ kitten *causes* pixels $x_1 \ldots x_n$

Example II: Naive Bayes

- Reverse graphical model for classification/regression:



- Opposite: Everything independent!
- May make more sense: y = kitten *causes* pixels $x_1 \dots x_n$

- Strong assumption – rarely true
- Best to avoid

UNIVERSITY OF BATH

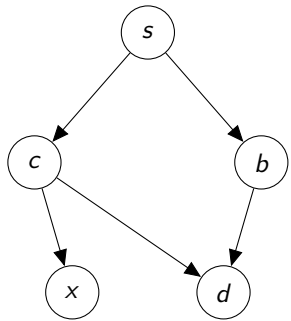
Example III: Expert systems I

- Experts encode their knowledge as a Bayesian network
- Nice idea, but...
  - Can't assign probabilities
  - Cause/effect not always clear
  - Complexity
- Never caught on

Example III: Expert systems I

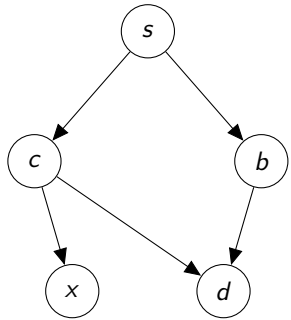
- Experts encode their knowledge as a Bayesian network
- Nice idea, but...
  - Can't assign probabilities
  - Cause/effect not always clear
  - Complexity
- Never caught on

- e.g. medical diagnosis
- All RVs binary:
  - s = Smokes?
  - c = Has lung cancer?
  - b = Has bronchitis?
  - x = Shadow on x-ray?
  - d = Has dyspnoea? (difficulty breathing)



Example III: Expert systems II

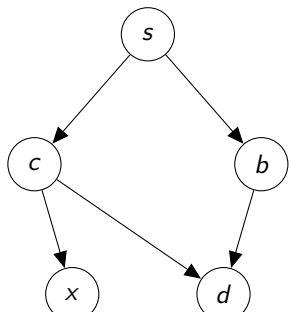
Equation steps:



Example III: Expert systems II

Equation steps:

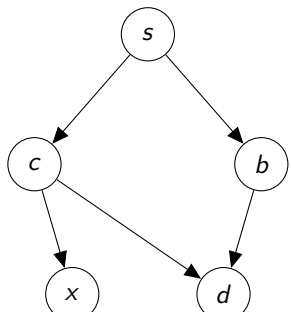
- Conditional probability for every RV:
  $= P(s|.)P(c|.)P(b|.)P(x|.)P(d|.)$



Example III: Expert systems II

Equation steps:

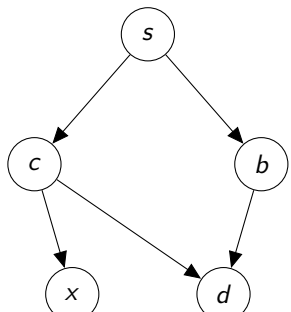
- Conditional probability for every RV:
  $= P(s|.)P(c|.)P(b|.)P(x|.)P(d|.)$
- Conditional on parent RVs only:
  $= P(s)P(c|s)P(b|s)P(x|c)P(d|c,b)$



Example III: Expert systems II

Equation steps:

- Conditional probability for every RV:
  $= P(s|.)P(c|.)P(b|.)P(x|.)P(d|.)$
- Conditional on parent RVs only:
  $= P(s)P(c|s)P(b|s)P(x|c)P(d|c,b)$
- Equal to joint distribution over all RVs
  $= P(s,c,b,x,d)$



Example III: Expert systems III

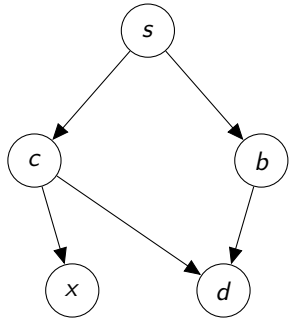
Parameters – binary RVs ∴:

- $P(s = \texttt{false}) = 0.5$
- $P(s = \texttt{true}) = 0.5$
  (example from 1990s, hence 50 : 50)

- $P(c = \texttt{false} | s = \texttt{false}) = 0.99$
- $P(c = \texttt{false} | s = \texttt{true}) = 0.9$
- $P(c = \texttt{true} | s = \texttt{false}) = 0.01$
- $P(c = \texttt{true} | s = \texttt{true}) = 0.1$

$\vdots$

(22 all in, Bernoulli distributions)

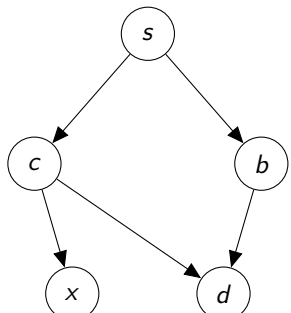


Example III: Expert systems IV

Update probabilities as evidence arrives

1. Patient walks in, everything unknown

$P(c) = 0.055, \quad P(b) = 0.45$



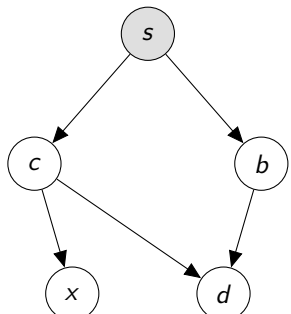
Example III: Expert systems IV

Update probabilities as evidence arrives

1. Patient walks in, everything unknown
2. Doctor asks if they smoke (true)

$P(c) = 0.1, \quad P(b) = 0.6$

$P(s = \texttt{true}) = 1$



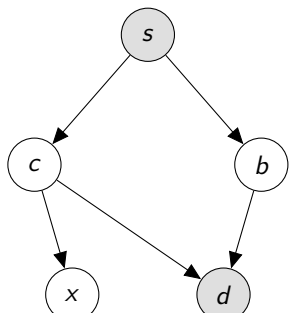
Example III: Expert systems IV

Update probabilities as evidence arrives

1. Patient walks in, everything unknown
2. Doctor asks if they smoke (true)
3. Doctor asks if they have difficulty breathing (false)

$P(c) = 0.037, \quad P(b) = 0.242$

$P(s = \texttt{true}) = 1$
$P(d = \texttt{false}) = 1$



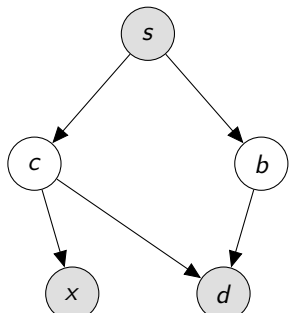
Example III: Expert systems IV

Update probabilities as evidence arrives

1. Patient walks in, everything unknown
2. Doctor asks if they smoke (true)
3. Doctor asks if they have difficulty breathing (false)
4. Doctor sends them for an x-ray (true)

$P(c) = 0.430, \quad P(b) = 0.158$

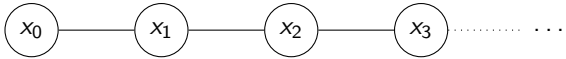
$P(s = \texttt{true}) = 1$
$P(d = \texttt{false}) = 1$
$P(x = \texttt{true}) = 1$



UNIVERSITY OF BATH

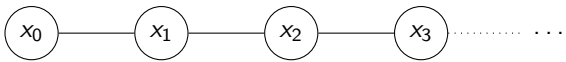
Example IV: Markov chain

Chain of RVs – usually represented as Markov network:

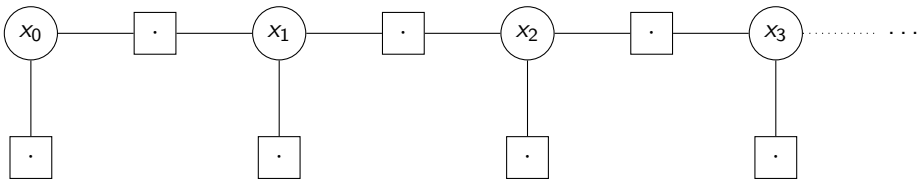


Example IV: Markov chain

Chain of RVs – usually represented as Markov network:

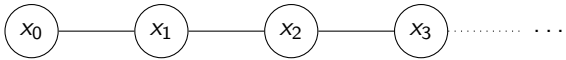


Factor graph:

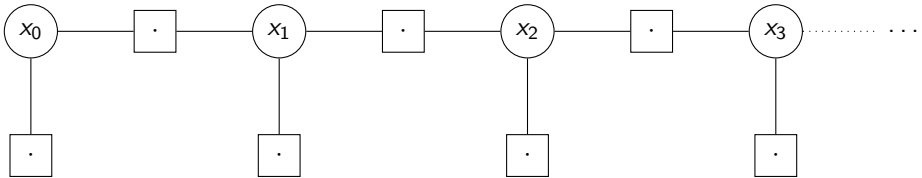


Example IV: Markov chain

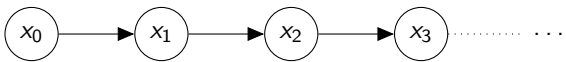
Chain of RVs – usually represented as Markov network:



Factor graph:



Bayesian network:



Markov property?

- "*Next state depends* **only** *on the previous state*"
- Alternatively, "*state has no memory of the past*"
- Most commonly applied to time, but also space

- Named after Andrey Markov (Russian mathematician) – invented them

UNIVERSITY OF BATH

Markov property?

- "*Next state depends* **only** *on the previous state*"
- Alternatively, "*state has no memory of the past*"
- Most commonly applied to time, but also space

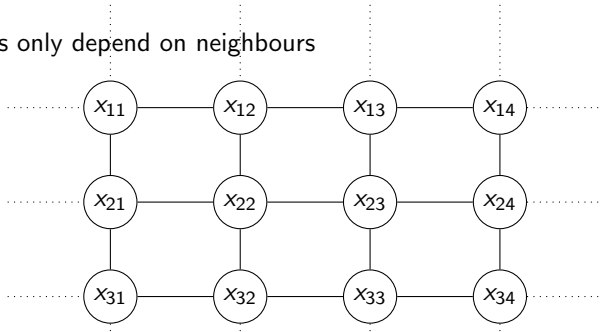
- Named after Andrey Markov (Russian mathematician) – invented them

- Examples:
  - Most simulations, e.g. in physics, chemistry etc. . .
  - Stock market and other financial things
  - Speech recognition
  - Data transfer over noisy channels (internet)

  ⋮

Example V: Markov grid

- 2D or 3D grid
- Markov: Elements only depend on neighbours



- Usually spacial, popular in computer vision:
  - Ising model & Potts model – state change for magnets (1920 – earliest use)
  - Image labelling, e.g. which pixels are cat
  - Image segmentation, e.g. objects vs background pixels in CCTV

UNIVERSITY OF BATH

Further concepts

- Conditional random field
- Explaining away
- d-separation (generalisation of explaining away)

- Choices
- Merging RVs

- Learning factors
- Assignment probability

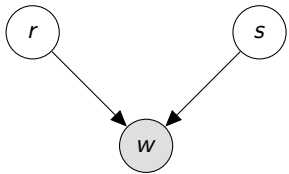
Conditional random fields

- Markov random field $\Longrightarrow$
  - Distributions learned from data
  - Distributions always the same

Conditional random fields

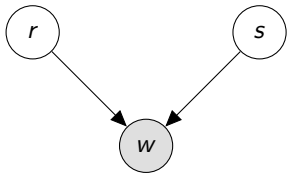
- Markov random field $\Longrightarrow$
  - Distributions learned from data
  - Distributions always the same
- Conditional random field $\Longrightarrow$
  - Distributions learned from data
  - Distributions **depend** on the data
- MRF is generative – can draw from it
- CRF is not – defined in terms of data

Explaining away



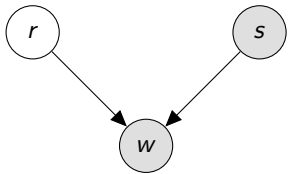
- Binary RVs:
  - r = It's been raining
  - s = Sprinklers have been on
  - w = Grass is wet

Explaining away



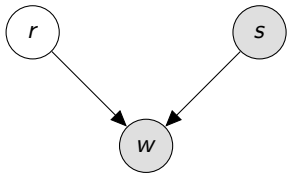
- Binary RVs:
  - r = It's been raining
  - s = Sprinklers have been on
  - w = Grass is wet
- Observe grass is wet:
  $P(r) = 0.38, \quad P(s) = 0.76$

Explaining away



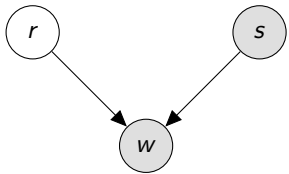
- Binary RVs:
  - r = It's been raining
  - s = Sprinklers have been on
  - w = Grass is wet
- Observe grass is wet:
  $P(r) = 0.38, \quad P(s) = 0.76$
- Update with sprinkler on:
  $P(r) = 0.2, \quad P(s) = 1.0$

Explaining away



- Binary RVs:
  - r = It's been raining
  - s = Sprinklers have been on
  - w = Grass is wet
- Observe grass is wet:
  $P(r) = 0.38, \quad P(s) = 0.76$
- Update with sprinkler on:
  $P(r) = 0.2, \quad P(s) = 1.0$
- Note how $P(r)$ drops?
  Explained wet grass $\therefore$ rain less likely
  Called **explaining away**

Explaining away



Observing a parent does not make the children conditionally independent!

- Binary RVs:
  - r = It's been raining
  - s = Sprinklers have been on
  - w = Grass is wet
- Observe grass is wet:
  $P(r) = 0.38, \quad P(s) = 0.76$
- Update with sprinkler on:
  $P(r) = 0.2, \quad P(s) = 1.0$
- Note how $P(r)$ drops?
  Explained wet grass $\therefore$ rain less likely
  Called **explaining away**

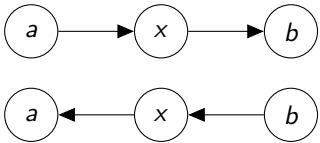
d-separation I

- d = directional; opposite is d-connected
- **RVs a and b are d-separated if independent given observed RVs, z**
- Must consider all paths between a and b, ignoring directions

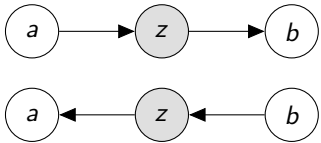
d-separation I

- d = directional; opposite is d-connected
- **RVs a and b are d-separated if independent given observed RVs, z**
- Must consider all paths between a and b, ignoring directions

d-connected (dependent):



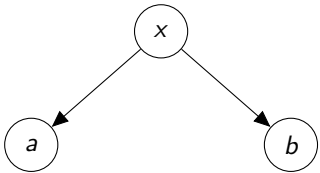
d-seperated (conditionally independent):



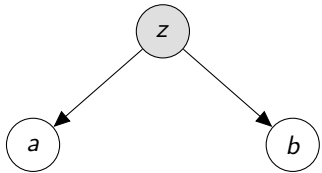
d-separation II

- d = directional; opposite is d-connected
- **RVs a and b are d-separated if independent given observed RVs, z**
- Must consider all paths between a and b, ignoring directions

d-connected (dependent):



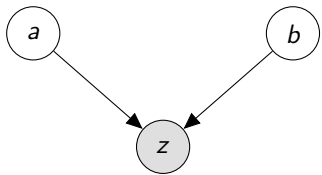
d-seperated (conditionally independent):



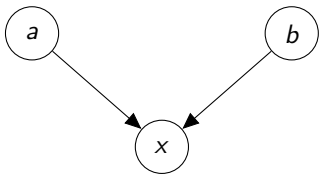
d-separation III

- d = directional; opposite is d-connected
- **RVs a and b are d-separated if independent given observed RVs, z**
- Must consider all paths between a and b, ignoring directions

d-connected (dependent):



d-seperated (conditionally independent):

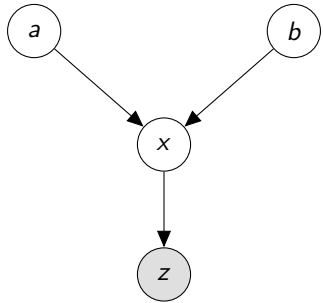


This is backwards!

d-separation IV

- d = directional; opposite is d-connected
- **RVs a and b are d-separated if independent given observed RVs, z**
- Must consider all paths between a and b, ignoring directions

d-connected (dependent):



Can think in terms of information "bouncing off" of observed RVs (arrow heads only)

UNIVERSITY OF BATH

Choices

- RVs can represent anything:
  - Single value
  - Vector
  - Matrix
  - Text
  - 3D model
  - Graphical model!

$$\vdots$$

UNIVERSITY OF BATH

Choices

- RVs can represent anything:
  - Single value
  - Vector
  - Matrix
  - Text
  - 3D model
  - Graphical model!
  - ⋮
- Functions can be anything:
  - Standard distributions
  - Logistic regression
  - Support vector machine
  - Random forest
  - Neural network
  - Graphical model!
  - ⋮

UNIVERSITY OF BATH

Choices

- RVs can represent anything:
  - Single value
  - Vector
  - Matrix
  - Text
  - 3D model
  - Graphical model!
  - ⋮
- Functions can be anything:
  - Standard distributions
  - Logistic regression
  - Support vector machine
  - Random forest
  - Neural network
  - Graphical model!
  - ⋮

- Graphical models look all powerful...
- ...but writing down a model does not mean it can be solved:-(

UNIVERSITY OF BATH

Choices

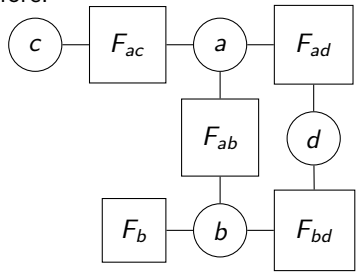
- RVs can represent anything:
  - Single value
  - Vector
  - Matrix
  - Text
  - 3D model
  - Graphical model!
  - ⋮
- Functions can be anything:
  - Standard distributions
  - Logistic regression
  - Support vector machine
  - Random forest
  - Neural network
  - Graphical model!
  - ⋮

- Graphical models look all powerful...
- ...but writing down a model does not mean it can be solved:-(
- Graphical model doesn't specify above – really a graphical *family* of models

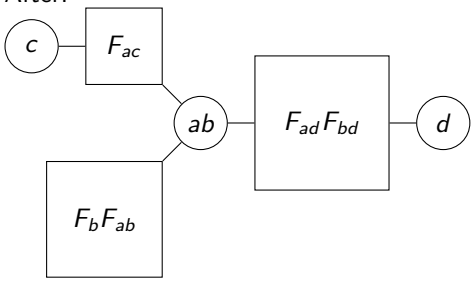
Merging RVs

- Can always merge two RVs, e.g a and b:

Before:



After:

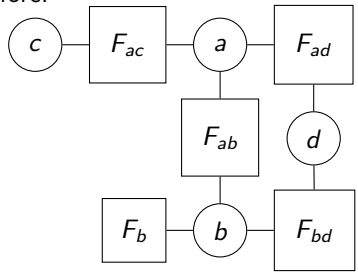


- Don't need to merge factors, but convenient/faster

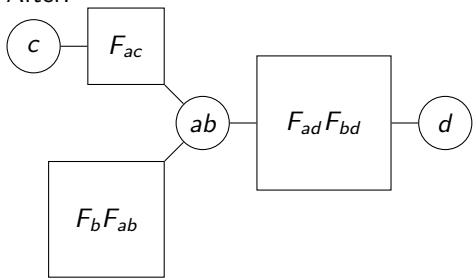
Merging RVs

- Can always merge two RVs, e.g a and b:

Before:



After:



- Don't need to merge factors, but convenient/faster

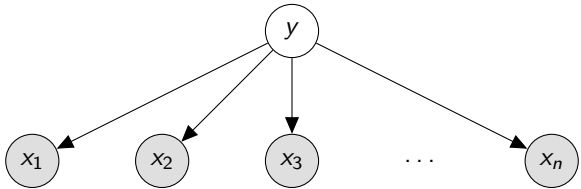
- Some algorithms require **trees**, i.e. **no loops**
- Can merge RVs to collapse loops (as above)
- Not always practical, e.g. Markov grid

Learning factors

- Just learn each $P(\cdot)$ independently!

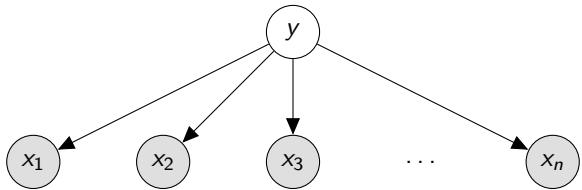
Learning factors

- Just learn each $P(\cdot)$ independently!
- e.g. Naive Bayes



Learning factors

- Just learn each $P(\cdot)$ independently!
- e.g. Naive Bayes



- Finish model – use binary RVs:

$$P(x_i|y) = Bernoulli(p_{iy}), \qquad x_i \in \{0,1\}, \qquad y \in \{0,1\}$$

Binary naive Bayes

$$P(x_i|y) = \mathit{Bernoulli}(p_{iy}) \quad [P(x_i{=}1|y){=}p_{iy},\ P(x_i{=}0|y){=}1{-}p_{iy}]$$

- $C_i(x', y') =$ number of training exemplars for which $x_i = x'$ and $y = y'$
- Maximum likelihood:

$$p_{iy} = \frac{C_i(1, y)}{C_i(0, y) + C_i(1, y)}$$

Binary naive Bayes

$$P(x_i|y) = Bernoulli(p_{iy}) \quad [P(x_i{=}1|y){=}p_{iy},\ P(x_i{=}0|y){=}1{-}p_{iy}]$$

- $C_i(x', y') =$ number of training exemplars for which $x_i = x'$ and $y = y'$
- Maximum likelihood:

$$p_{iy} = \frac{C_i(1, y)}{C_i(0, y) + C_i(1, y)}$$

- Maximum a posteriori (MAP) with prior $p_{iy} \sim \mathrm{Beta}(1, 1)$: (conjugate, uniform)

$$p_{iy} = \frac{1 + C_i(1, y)}{2 + C_i(0, y) + C_i(1, y)}$$

Binary naive Bayes

$$P(x_i|y) = Bernoulli(p_{iy}) \quad [P(x_i{=}1|y){=}p_{iy},\ P(x_i{=}0|y){=}1{-}p_{iy}]$$

- $C_i(x', y') =$ number of training exemplars for which $x_i = x'$ and $y = y'$
- Maximum likelihood:

$$p_{iy} = \frac{C_i(1, y)}{C_i(0, y) + C_i(1, y)}$$

- Maximum a posteriori (MAP) with prior $p_{iy} \sim \texttt{Beta}(1, 1)$: (conjugate, uniform)

$$p_{iy} = \frac{1 + C_i(1, y)}{2 + C_i(0, y) + C_i(1, y)}$$

- Bayesian, same prior:

$$p_{iy} \sim \texttt{Beta}(1 + C_i(1, y), 1 + C_i(0, y))$$

Assignment probability

- Probability of RV assignment: Just evaluate all factors and multiply together

Assignment probability

- Probability of RV assignment: Just evaluate all factors and multiply together
- e.g. Naive Bayes with binary RVs given x known:

$$P(y|x) \propto \prod_{i=1}^{n} P(x_i|y)P(y)$$

$$P(y|x) \propto \prod_{i=1}^{n} (p_{iy})^{x_i}(1-p_{iy})^{1-x_i}P(y)$$

Assignment probability

- Probability of RV assignment: Just evaluate all factors and multiply together
- e.g. Naive Bayes with binary RVs given x known:

$$P(y|x) \propto \prod_{i=1}^{n} P(x_i|y)P(y)$$

$$P(y|x) \propto \prod_{i=1}^{n} (p_{iy})^{x_i}(1-p_{iy})^{1-x_i}P(y)$$

- Above numerically unstable – calculate in log space instead:

$$\log P(y|x) = \sum_{i=1}^{n} x_i \log(p_{iy}) + (1-x_i)\log(1-p_{iy}) + \log P(y) + C$$

Assignment probability

- Probability of RV assignment: Just evaluate all factors and multiply together
- e.g. Naive Bayes with binary RVs given x known:

$$P(y|x) \propto \prod_{i=1}^{n} P(x_i|y)P(y)$$

$$P(y|x) \propto \prod_{i=1}^{n} (p_{iy})^{x_i}(1-p_{iy})^{1-x_i}P(y)$$

- Above numerically unstable – calculate in log space instead:

$$\log P(y|x) = \sum_{i=1}^{n} x_i \log(p_{iy}) + (1-x_i)\log(1-p_{iy}) + \log P(y) + C$$

- Need to define $P(y)$, e.g. $y \sim \texttt{Beta}(1,1)$
- Proportionality/$+C$ from applying Bayes without knowing $P(x)$

Causality

- Graphical models are usually **causal**
- Not necessary (or always possible), but easier to think about

Causality

- Graphical models are usually **causal**
- Not necessary (or always possible), but easier to think about

- Learning causality is a research problem (a really hard one)
- A formal treatment of scientific method

- Often ambiguous (as in no amount of data can resolve – need experiments)
- Unobserved RVs can break conclusions

UNIVERSITY OF BATH

Algorithms

- Covering (next two lectures):
  - Analytic
  - Dynamic programming (DP)
  - Belief propagation (BP, generalisation of dynamic programming)
  - Loopy belief propagation (LBP, briefly)
  - Expectation-maximisation (EM)

- Ignoring (in ML1):
  - Tree reweighted message passing – sequential
  - Graph cuts

  - *Markov Chain Monte Carlo* (MCMC) methods, e.g. Gibbs sampling
  - Variational methods

(key techniques; there are others!)

Summary

- Graphical models represent problem structure
- Powerful
- Lots of details!
- Naive Bayes (but don't use – educational example only)

UNIVERSITY OF BATH

Further reading

- "Information Theory, Inference, and Learning Algorithms" by MacKay. Chapter 16 – Viterbi algorithm
- "Causality" by Judea Pearl. The only book you will ever want on causality; but only if you really, really want to know more (extremely hard)
- "The Book of Why: The New Science of Cause and Effect " by Judea Pearl and Dana Mackenzie. Pop-sci version of above – easier, relatively speaking