Machine Learning 1.12: Belief Propagation

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

UNIVERSITY OF BATH

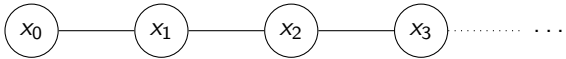
This lecture

- Three algorithms:
  - Dynamic programming
  - Belief propagation
  - Loopy belief propagation
- Really all the same!

UNIVERSITY OF BATH

Reminder

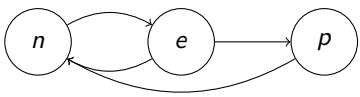
- Markov chain:



- "*Next state depends only on the previous state*"

Finite state machines

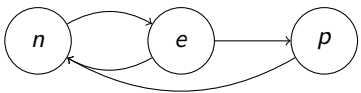
- Crossroad traffic lights as a **finite state machine**:



- n = North-south traffic flowing
- e = East-west traffic flowing
- p = Pedestrians crossing

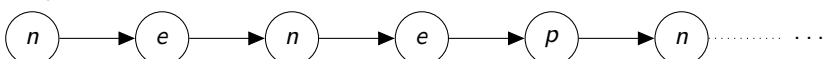
Finite state machines

- Crossroad traffic lights as a **finite state machine**:



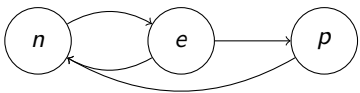
- n = North-south traffic flowing
- e = East-west traffic flowing
- p = Pedestrians crossing

- Possible sequence:



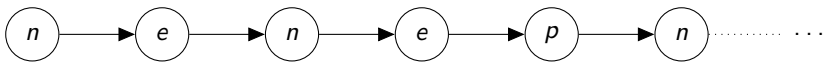
Finite state machines

- Crossroad traffic lights as a **finite state machine**:



  - n = North-south traffic flowing
  - e = East-west traffic flowing
  - p = Pedestrians crossing

- Possible sequence:



- Finite state machines (FSM) can run on Markov chains
- Markov chains are probabilistic; FSMs usually not
- Markov chains can have infinite states

Transition matrix

- Discrete Markov random chain:
  - Set of n states, $\{s_0, s_1, \ldots, s_{n-1}\}$.
  - Transition matrix $T \in \mathbb{R}^{n \times n}$, such that $P(s_a \to s_b) = T_{ab}$. ($P(s_a \to s_b)$ is the probability of going from s_a to s_b)

UNIVERSITY OF BATH

Transition matrix

- Discrete Markov random chain:
  - Set of n states, $\{s_0, s_1, \ldots, s_{n-1}\}$.
  - Transition matrix $T \in \mathbb{R}^{n \times n}$, such that $P(s_a \to s_b) = T_{ab}$. ($P(s_a \to s_b)$ is the probability of going from s_a to s_b)

- Each row of T is a categorical distribution; to learn:
  - Set prior, e.g. uniform Dirichlet distribution (conjugate)
  - Update prior with collected data
  - Set T to mean of the posterior (MAP solution)

UNIVERSITY OF BATH

Transition matrix

- Discrete Markov random chain:
  - Set of n states, $\{s_0, s_1, \ldots, s_{n-1}\}$.
  - Transition matrix $T \in \mathbb{R}^{n \times n}$, such that $P(s_a \to s_b) = T_{ab}$. ($P(s_a \to s_b)$ is the probability of going from s_a to s_b)
- Each row of T is a categorical distribution; to learn:
  - Set prior, e.g. uniform Dirichlet distribution (conjugate)
  - Update prior with collected data
  - Set T to mean of the posterior (MAP solution)
- Explicitly:
  - Fill T with 1.
  - For each $s_a \to s_b$ in training increment T_{ab}.
  - Normalise each row of T

UNIVERSITY OF BATH

Simulation

- To draw:
  - Select a start state (fit categorical to data)
  - Select a length (fit distribution to data length, e.g. Poisson)

  (can introduce start and stop states instead)

- $x_1 \sim P(x_1)$
- $x_i \sim P(x_{i-1} \to \cdot)$ for $i \in [2 \ldots \texttt{length}]$.

UNIVERSITY OF BATH

Fake place names

Steps:

- UK place names, `https://www.paulstenning.com/uk-towns-and-counties-list/`
- Learn transition matrix, start and length distributions

UNIVERSITY OF BATH

Fake place names

Steps:

- UK place names, `https://www.paulstenning.com/uk-towns-and-counties-list/`
- Learn transition matrix, start and length distributions
- Draw "plausible" place names:

ackilin	ckleryhtz	harkbrdy borthr	genongigropo
burie	ftonsest	ad onff	mveyldid
beston	kilelerg	crynhal	coeerigent
kqomironam	wey ok mbu	licalst	borayeamol

- Too simple really!
- Extension: n-gram model, e.g. $n = 2 \implies$ States are two previous letters

UNIVERSITY OF BATH

Dynamic programming

- What if we know something?
  - e.g. $x_7 =$ "a"?
  - e.g. $x_4 =$ "g" and $x_6 =$ "r"?
  - e.g. x_1 has a 20% chance of being "i", an 80% chance of being an "e"?

UNIVERSITY OF BATH

Dynamic programming

- What if we know something?
  - e.g. $x_7 =$ "a"?
  - e.g. $x_4 =$ "g" and $x_6 =$ "r"?
  - e.g. x_1 has a 20% chance of being "i", an 80% chance of being an "e"?
- Dynamic programming: Lets you reason given arbitrary information
  - Maximum Likelihood (ML) – most probable state sequence
  - Maximum a Posteriori (MAP) – most probable state sequence with prior
  - **Marginals** – distribution over state of each node
- Can't do:
  - Conditional draw – draw unspecified states
  - Joint distribution – full distribution over unknown states

Marginals

- marginal = "*belief*" = posterior distribution over RV given evidence

Marginals

- marginal = "*belief*" = posterior distribution over RV given evidence
- Definition:

$$b_i(x_i) = \sum_{\cdot/x_i} P(x_1, \ldots, x_N)$$

where:

- $x_i, i \in [1, \ldots, N]$ are the random variables
- $b_i(x_i)$ is the belief of RV i
- $\cdot/x_i$ means all variables except for x_i
- $P(x_1, \ldots, x_N)$ is the joint distribution over all variables

Marginals

- marginal = "*belief*" = posterior distribution over RV given evidence
- Definition:

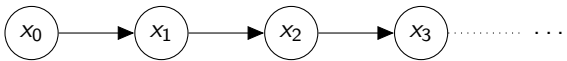
$$b_i(x_i) = \sum_{\cdot/x_i} P(x_1, \ldots, x_N)$$

  where:
  - $x_i, i \in [1, \ldots, N]$ are the random variables
  - $b_i(x_i)$ is the belief of RV i
  - $\cdot/x_i$ means all variables except for x_i
  - $P(x_1, \ldots, x_N)$ is the joint distribution over all variables

- $P(x_1, \ldots, x_N)$ is impractical, e.g.
  - RVs have 26 states
  - 10 RVs
  - $26^{10} = 141167095653376$ values!

UNIVERSITY OF BATH

Graphical model



- Factorise: $P(x_1, \ldots, x_N) = P(x_1)P(x_2|x_1)P(x_3|x_2)\ldots$
- Discrete distribution $\implies$ $P(x_i|x_{i-1}) =$ transition matrix

Factorisation

- Belief, for 5 variables:

$$b_3(x_3) = \sum_{x_1, x_2, x_4, x_5} P(x_1, x_2, x_3, x_4, x_5)$$

Factorisation

- Belief, for 5 variables:

$$b_3(x_3) = \sum_{x_1, x_2, x_4, x_5} P(x_1, x_2, x_3, x_4, x_5)$$

$$b_3(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} P(x_5|x_4)P(x_4|x_3)P(x_3|x_2)P(x_2|x_1)P(x_1)$$

(replace with structure implied by graphical model)

Factorisation

- Belief, for 5 variables:

$$b_3(x_3) = \sum_{x_1,x_2,x_4,x_5} P(x_1, x_2, x_3, x_4, x_5)$$

$$b_3(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} P(x_5|x_4)P(x_4|x_3)P(x_3|x_2)P(x_2|x_1)P(x_1)$$

(replace with structure implied by graphical model)

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} \sum_{x_2} P(x_3|x_2) \sum_{x_1} P(x_2|x_1)P(x_1)$$

(move sums to minimise computation/storage)

Factorisation

- Belief, for 5 variables:

$$b_3(x_3) = \sum_{x_1,x_2,x_4,x_5} P(x_1, x_2, x_3, x_4, x_5)$$

$$b_3(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} P(x_5|x_4)P(x_4|x_3)P(x_3|x_2)P(x_2|x_1)P(x_1)$$

(replace with structure implied by graphical model)

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} \sum_{x_2} P(x_3|x_2) \sum_{x_1} P(x_2|x_1)P(x_1)$$

(move sums to minimise computation/storage)

This is the **forward–backward** algorithm!

Forward–backward algorithm

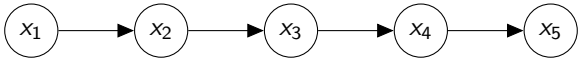
- Dynamic programming variant for calculating marginals (beliefs)
- Just rearranging the **sums** and **products** to minimise computation/storage!

UNIVERSITY OF BATH

Forward–backward algorithm

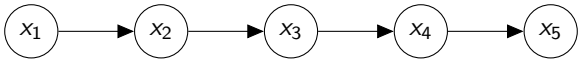
- Dynamic programming variant for calculating marginals (beliefs)
- Just rearranging the **sums** and **products** to minimise computation/storage!
- Better interpretation: Message passing

Message passing I



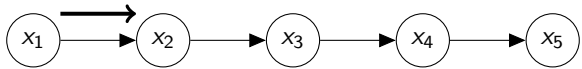
- Equivalent to sending messages between RVs
- Two messages per edge, one in each direction

Message passing II



$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} \sum_{x_2} P(x_3|x_2) \sum_{x_1} P(x_2|x_1) P(x_1)$$

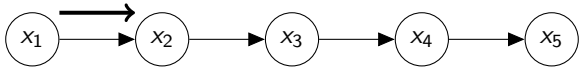
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} \sum_{x_2} P(x_3|x_2) \sum_{x_1} P(x_2|x_1)P(x_1)$$

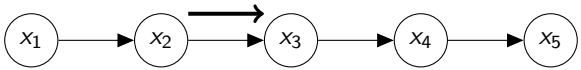
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} \sum_{x_2} P(x_3|x_2) m_{1\to 2}(x_2)$$

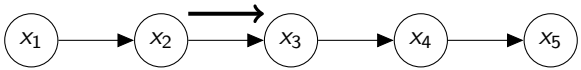
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)
- $m_{2\to 3}(x_3) = \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$$

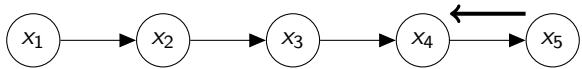
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)
- $m_{2\to 3}(x_3) = \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} m_{2\to 3}(x_3)$$

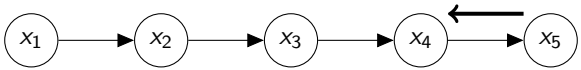
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)
- $m_{2\to 3}(x_3) = \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$
- $m_{5\to 4}(x_4) = \sum_{x_5} P(x_5|x_4)$

$$b_3(x_3) = \sum_{x_4} \left\{ \left[\sum_{x_5} P(x_5|x_4) \right] P(x_4|x_3) \right\} m_{2\to 3}(x_3)$$

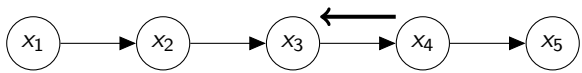
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)
- $m_{2\to 3}(x_3) = \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$
- $m_{5\to 4}(x_4) = \sum_{x_5} P(x_5|x_4)$

$$b_3(x_3) = \sum_{x_4} \{m_{5\to 4}(x_4)P(x_4|x_3)\}\, m_{2\to 3}(x_3)$$

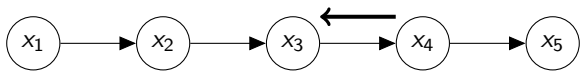
Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)
- $m_{2\to 3}(x_3) = \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$
- $m_{5\to 4}(x_4) = \sum_{x_5} P(x_5|x_4)$
- $m_{4\to 3}(x_3) = \sum_{x_4} P(x_4|x_3)m_{5\to 4}(x_4)$

$$b_3(x_3) = \sum_{x_4} \{m_{5\to 4}(x_4)P(x_4|x_3)\}\, m_{2\to 3}(x_3)$$

Message passing II



- $m_{1\to 2}(x_2) = \sum_{x_1} P(x_2|x_1)P(x_1)$
  (message from node 1 to 2)
- $m_{2\to 3}(x_3) = \sum_{x_2} P(x_3|x_2)m_{1\to 2}(x_2)$

- $m_{5\to 4}(x_4) = \sum_{x_5} P(x_5|x_4)$
- $m_{4\to 3}(x_3) = \sum_{x_4} P(x_4|x_3)m_{5\to 4}(x_4)$

$$b_3(x_3) = m_{4\to 3}(x_3)m_{2\to 3}(x_3)$$

Message passing III

- In general:

$$m_{i\to j}(x_j) = \sum_{\cdot/x_j} U(x_i)F(x_i, x_j)m_{k\to i}(x_i)$$

where $F(x_i, x_j)$ could be $P(x_i|x_j), P(x_j|x_i)$ or something non-probabilistic!
$U(x_i)$ is any unary term

Message passing III

- In general:

$$m_{i\to j}(x_j) = \sum_{\cdot/x_j} U(x_i)F(x_i, x_j)m_{k\to i}(x_i)$$

  where $F(x_i, x_j)$ could be $P(x_i|x_j), P(x_j|x_i)$ or something non-probabilistic! $U(x_i)$ is any unary term

- Belief:

$$b_j(x_j) = U(x_j)\prod_{i\in N_j} m_{i\to j}(x_j)$$

  - N_j = neighbours of j

Message passing III

- In general:

$$m_{i\to j}(x_j) = \sum_{\cdot/x_j} U(x_i)F(x_i, x_j)m_{k\to i}(x_i)$$

  where $F(x_i, x_j)$ could be $P(x_i|x_j), P(x_j|x_i)$ or something non-probabilistic!
  $U(x_i)$ is any unary term

- Belief:

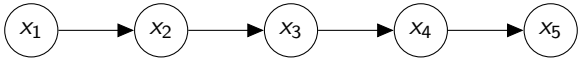
$$b_j(x_j) = U(x_j)\prod_{i\in N_j} m_{i\to j}(x_j)$$

  - N_j = neighbours of j

- Summary:
  - Message = partial joint distribution
  - Summary of terms "*behind*" message
  - RVs marginalised out irrelevant to destination
- Note: Generalises to continuous distributions (integrals instead of sums)

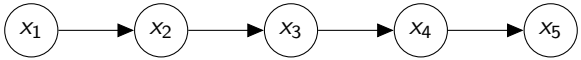
UNIVERSITY OF BATH

Forward–backward?



- Usually want marginal for **every RV**:

Forward–backward?



- Usually want marginal for **every RV**:

Forwards:

- $m_{1\to 2}(x_2)$
- $m_{2\to 3}(x_3)$
- $m_{3\to 4}(x_4)$
- $m_{4\to 5}(x_5)$

Backwards:

- $m_{5\to 4}(x_4)$
- $m_{4\to 3}(x_3)$
- $m_{3\to 2}(x_2)$
- $m_{2\to 1}(x_1)$

- Belief of every node = product incoming messages and unary terms
- Messages are **reused**!

- Ordered to satisfy dependencies (need $m_{1\to 2}(x_2)$ to send $m_{2\to 3}(x_3)$ etc.)
- Could do backward–forward!

UNIVERSITY OF BATH

Voice recognition

- **Input**: Waveform
- **Output**: Words

UNIVERSITY OF BATH

Voice recognition

- **Input**: Waveform
- **Output**: Words

- **States**: Phonemes, for each 10ms period
- **Observations**: Waveform, chopped into (overlapping) chunks $\rightarrow$
  Mel-Frequency Cepstral Coefficients $\rightarrow$
  Probability of each phoneme (Gaussian mixture model)
- **Transition matrix**: Language model – likely word sequences

Voice recognition

- **Input**: Waveform
- **Output**: Words

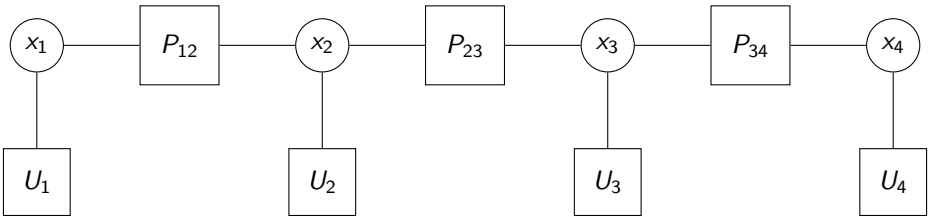
- **States**: Phonemes, for each 10ms period
- **Observations**: Waveform, chopped into (overlapping) chunks $\rightarrow$
  Mel-Frequency Cepstral Coefficients $\rightarrow$
  Probability of each phoneme (Gaussian mixture model)
- **Transition matrix**: Language model – likely word sequences

Notes:

- Usually given as a *hidden Markov model*
- Improvements: (Above was state of the art in $\sim$2012)
  - Other features, e.g. Perceptual Linear Prediction
  - Deep learning instead of a GMM for phoneme probability
  - Recurrent neural networks for language model, and for phoneme probability.

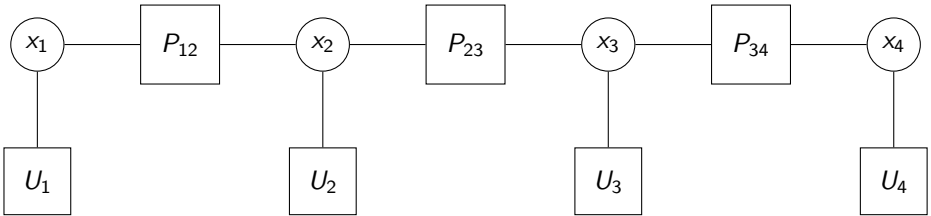
  Note: None of these replace the Markov chain!

Factor graph

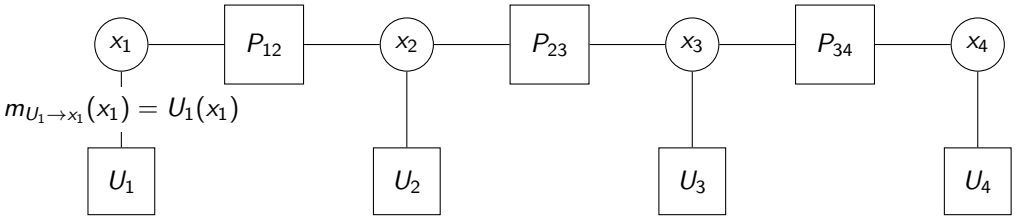


- x_i – Random variable: Which Phoneme (100 per second, ≈ 44 possible)
- U_i – Unary, a factor on one RV: **Gaussian mixture model over audio features**
- P_i – Pairwise, a factor on two RVs: **Language model**

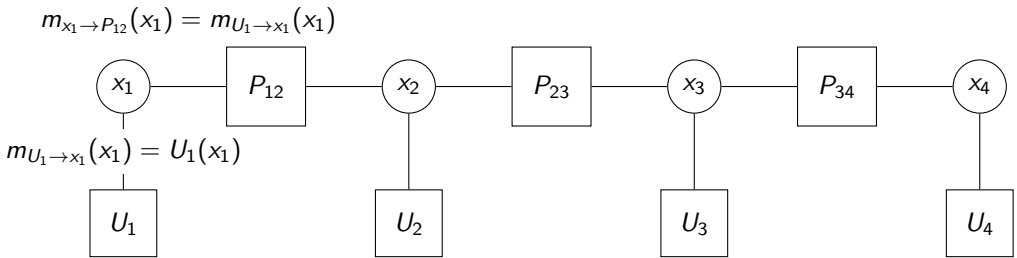
Factor graph



Factor graph



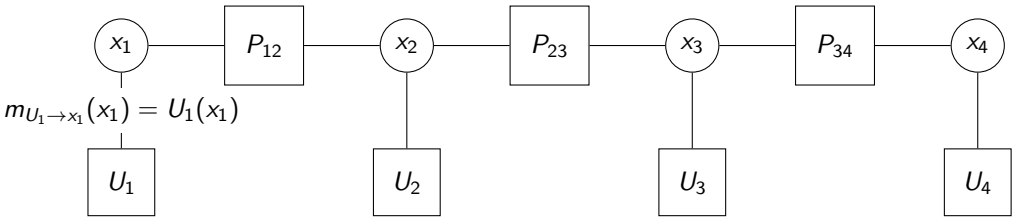
Factor graph



Factor graph

$$m_{P_{12}\to x_2} = \sum_{x_1} P_{12}(x_1, x_2) m_{x_1\to P_{12}}(x_1)$$

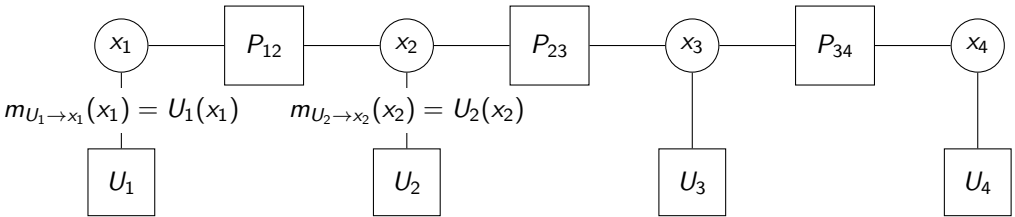
$$m_{x_1\to P_{12}}(x_1) = m_{U_1\to x_1}(x_1)$$



Factor graph

$$m_{P_{12}\to x_2} = \sum_{x_1} P_{12}(x_1, x_2) m_{x_1\to P_{12}}(x_1)$$

$$m_{x_1\to P_{12}}(x_1) = m_{U_1\to x_1}(x_1)$$

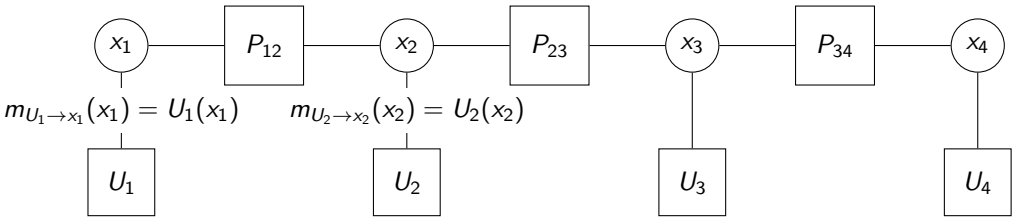


Factor graph

$$m_{x_2 \to P_{23}}(x_2) = m_{P_{12} \to x_2}(x_2) m_{U_2 \to x_2}(x_2)$$

$$m_{P_{12} \to x_2} = \sum_{x_1} P_{12}(x_1, x_2) m_{x_1 \to P_{12}}(x_1)$$

$$m_{x_1 \to P_{12}}(x_1) = m_{U_1 \to x_1}(x_1)$$



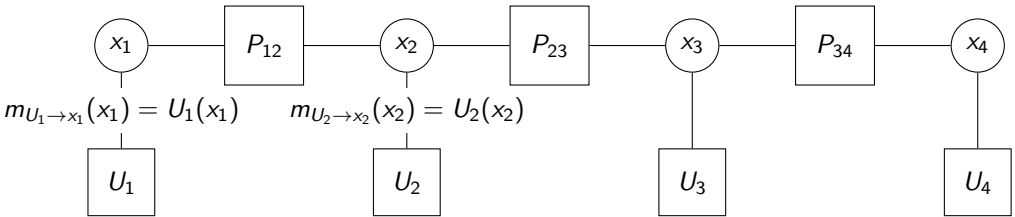
Factor graph

$$m_{P_{23}\to x_3}(x_3) = \sum_{x_2} P_{23}(x_2, x_3) m_{x_2\to P_{23}}(x_2)$$

$$m_{x_2\to P_{23}}(x_2) = m_{P_{12}\to x_2}(x_2) m_{U_2\to x_2}(x_2)$$

$$m_{P_{12}\to x_2} = \sum_{x_1} P_{12}(x_1, x_2) m_{x_1\to P_{12}}(x_1)$$

$$m_{x_1\to P_{12}}(x_1) = m_{U_1\to x_1}(x_1)$$



Factor graph equation

- Messages sent by RVs:

$$m_{v \to F}(x_v) = \prod_{G \in N_v / F} m_{G \to v}(x_v)$$

where $i \in N_v / F$ is all neighbours except the message destination.

- Messages sent by factors:

$$m_{F \to v}(x_v) = \sum_{x \notin v} \left[F(\cdot) \prod_{u \in N_F / v} m_{u \to F}(x_u) \right]$$

UNIVERSITY OF BATH

Viterbi

- What if we want
  - Maximum liklihood?
  - Maximum a Posteriori?
- Answer: Viterbi algorithm

Viterbi

- What if we want
  - Maximum liklihood?
  - Maximum a Posteriori?
- Answer: Viterbi algorithm

- Forward-backwards uses two operations: *sum* and *product*.
- For Viterbi: Change $sum \rightarrow max$

Viterbi

- What if we want
  - Maximum liklihood?
  - Maximum a Posteriori?
- Answer: Viterbi algorithm

- Forward-backwards uses two operations: *sum* and *product*.
- For Viterbi: Change *sum* $\rightarrow$ *max*

$$m_{v\to F}(x_v) = \prod_{G\in N_v/F} m_{G\to v}(x_v) \qquad m_{F\to v}(x_v) = \max_{x\notin v}\left[F(\cdot)\prod_{u\in N_F/v} m_{u\to F}(x_u)\right]$$

Viterbi

- What if we want
  - Maximum liklihood?
  - Maximum a Posteriori?
- Answer: Viterbi algorithm

- Forward-backwards uses two operations: *sum* and *product*.
- For Viterbi: Change *sum* $\rightarrow$ *max*

$$m_{v\to F}(x_v) = \prod_{G\in N_v/F} m_{G\to v}(x_v) \qquad m_{F\to v}(x_v) = \max_{x\notin v}\left[F(\cdot)\prod_{u\in N_F/v} m_{u\to F}(x_u)\right]$$

- Record which RV won each max – draws are common!
- $\therefore$ only forward pass required
- Can be implemented as a recursive function

UNIVERSITY OF BATH

Dynamic programming notes

- Dynamic programming:
  "*Any algorithm where the solution can be found recursively, by solving slightly smaller problems first*"
- Forwards-backwards and Viterbi both examples; many others

UNIVERSITY OF BATH

Dynamic programming notes

- Dynamic programming:
  "*Any algorithm where the solution can be found recursively, by solving slightly smaller problems first*"
- Forwards-backwards and Viterbi both examples; many others

- **Kalman smoothing** = Gaussian distributions
  (Forwards-backwards or Viterbi – identical answer)
- **Kalman filtering** = Online version where only the past is factored in

Dynamic programming tricks

- Alignment: Know label order, not position
- Solution: New labels represent where in sequence. Probability of transitioning to any but next $= 0$
- Used to label voice recognition training data

UNIVERSITY OF BATH

Dynamic programming tricks

- Alignment: Know label order, not position
- Solution: New labels represent where in sequence.
  Probability of transitioning to any but next $= 0$
- Used to label voice recognition training data

- Circle: Links form **one** loop
- Solution: Duplicate one node to start and end, solve for every state, select best

UNIVERSITY OF BATH

Dynamic programming tricks

- Alignment: Know label order, not position
- Solution: New labels represent where in sequence.
  Probability of transitioning to any but next $= 0$
- Used to label voice recognition training data

- Circle: Links form **one** loop
- Solution: Duplicate one node to start and end, solve for every state, select best

- n-gram: States are dependent on more than immediate neighbours
- Explode states to encode history

UNIVERSITY OF BATH

Belief propagation I

- Generalisation of forward–backward/Viterbi to a tree

Belief propagation I

- Generalisation of forward–backward/Viterbi to a tree
- sum–product: Generalisation of forward–backward
- max–product: Generalisation of Viterbi
- min–sum: max-product, but performed in negative log space (costs)

Belief propagation II

Forward–backward:

- RV to factor:

$$m_{v \to F}(x_v) = \prod_{G \in N_v / F} m_{G \to v}(x_v)$$

- Factor to RV:

$$m_{F \to v}(x_v) = \sum_{x \notin v} \left[F(\cdot) \prod_{u \in N_F / v} m_{u \to F}(x_u) \right]$$

- Belief:

$$b(v) = \prod_{G \in N_v} m_{G \to v}(x_v)$$

UNIVERSITY OF BATH

Belief propagation II

Forward–backward:

- RV to factor:

$$m_{v\to F}(x_v) = \prod_{G\in N_v/F} m_{G\to v}(x_v)$$

- Factor to RV:

$$m_{F\to v}(x_v) = \sum_{x\notin v}\left[F(\cdot)\prod_{u\in N_F/v} m_{u\to F}(x_u)\right]$$

- Belief:

$$b(v) = \prod_{G\in N_v} m_{G\to v}(x_v)$$

Sum–product:

- RV to factor:

$$m_{v\to F}(x_v) = \prod_{G\in N_v/F} m_{G\to v}(x_v)$$

- Factor to RV:

$$m_{F\to v}(x_v) = \sum_{x\notin v}\left[F(\cdot)\prod_{u\in N_F/v} m_{u\to F}(x_u)\right]$$

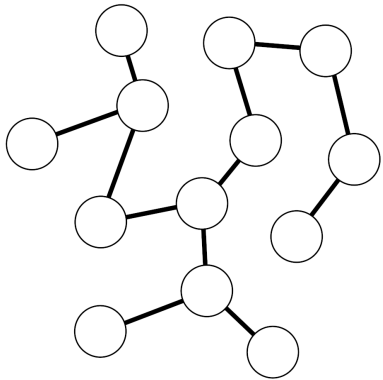
- Belief:

$$b(v) = \prod_{G\in N_v} m_{G\to v}(x_v)$$

Identical!

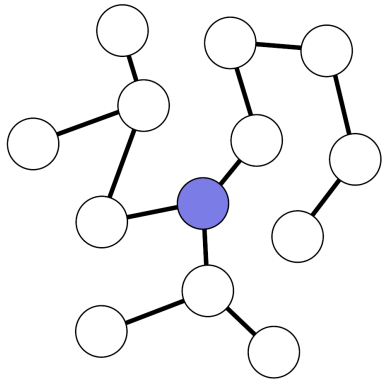
Belief propagation III

- Messages are identical...
- ...but forwards/backwards not defined



Belief propagation III

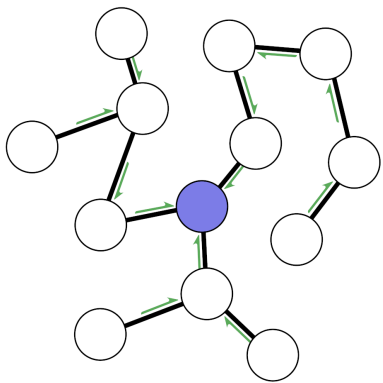
- Messages are identical...
- ...but forwards/backwards not defined
- Choose node to be "*root*" (any will do)



Belief propagation III

- Messages are identical...
- ...but forwards/backwards not defined

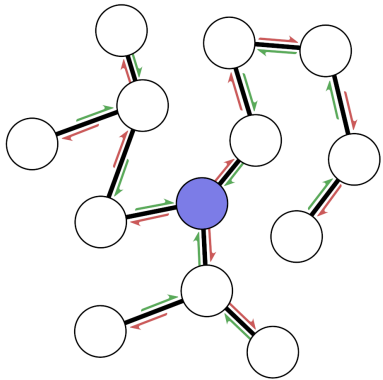
- Choose node to be "*root*" (any will do)
- Forward: Messages going towards it



Belief propagation III

- Messages are identical...
- ...but forwards/backwards not defined

- Choose node to be "*root*" (any will do)
- Forward: Messages going towards it
- Backwards: Messages going away from it



UNIVERSITY OF BATH

Loopy belief propagation

- BP does not work on a graph: Loops = circular dependencies
- Might be able to collapse

UNIVERSITY OF BATH

Loopy belief propagation

- BP does not work on a graph: Loops = circular dependencies
- Might be able to collapse

- Loopy BP: Ignore dependency and send messages anyway
  (initialise empty messages with uniform distribution)
- Keep sending messages until it converges...

UNIVERSITY OF BATH

Loopy belief propagation

- BP does not work on a graph: Loops = circular dependencies
- Might be able to collapse

- Loopy BP: Ignore dependency and send messages anyway
  (initialise empty messages with uniform distribution)
- Keep sending messages until it converges...

- **May** converge, may not
- "Double counts" 'evidence – answer **is wrong**
- Often good enough however

LBP advice

- Order messages to maximise information speed
- Double counting $\implies$ infinite energy $\rightarrow$ need to regularly renormalise distributions
- Add momentum term – improves convergence

LBP advice

- Order messages to maximise information speed
- Double counting $\implies$ infinite energy $\rightarrow$ need to regularly renormalise distributions
- Add momentum term – improves convergence

- Better algorithms:
  - Graph cuts (discrete only)
  - Tree reweighted message passing: Sequential (TRW-S)

UNIVERSITY OF BATH

Limitations & extensions

- Have to pass distributions between nodes
- Analytic solutions rare
- MCMC and variational more flexible

Limitations & extensions

- Have to pass distributions between nodes
- Analytic solutions rare
- MCMC and variational more flexible

- Clique size: Number of connections a factor has
- Efficiency premised on maximal clique size being small

Limitations & extensions

- Have to pass distributions between nodes
- Analytic solutions rare
- MCMC and variational more flexible

- Clique size: Number of connections a factor has
- Efficiency premised on maximal clique size being small

- Gaussian belief propagation: Kalman smoothing generalised to graph
  (Loopy Gaussian BP mostly safe – equivalent to inverting a matrix)
- Other continuous distributions doable, but much harder

Summary

- Dynamic programming
- Belief propagation
- Loop belief propagation

Further reading

- Very pragmatic tutorial on BP on grids of RVs:
  "Efficient Belief Propagation for Early Vision", by Felzenszwalb & Huttenlocher:
  `https://cs.brown.edu/~pff/papers/bp-cvpr.pdf`

- Voice recognition as described above:
  Hidden Markov Model Toolkit (HTK) `http://htk.eng.cam.ac.uk/`