Machine Learning 1.12: Belief Propagation

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

UNIVERSITY OF BATH

Markov Chain

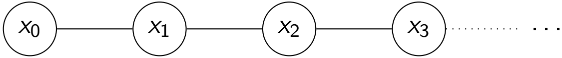
It's a chain!
Markov network (usual representation):

$$x_0 - x_1 - x_2 - x_3 \cdots \cdots$$

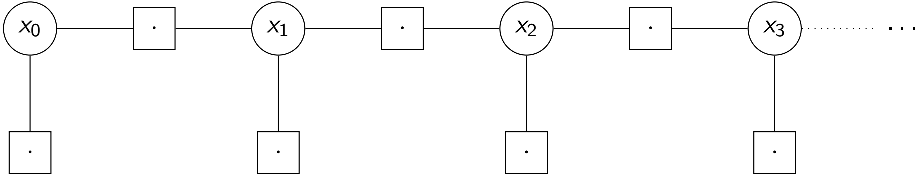
Markov Chain

It's a chain!

Markov network (usual representation):



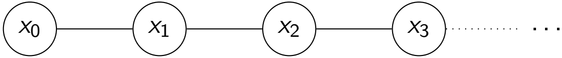
Factor graph:



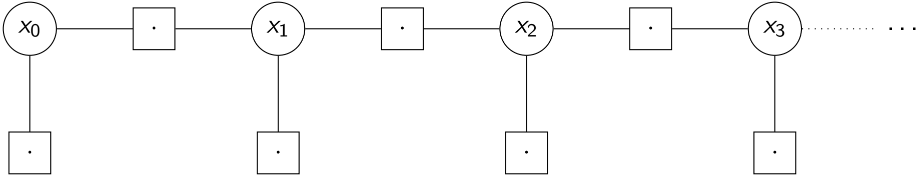
Markov Chain

It's a chain!

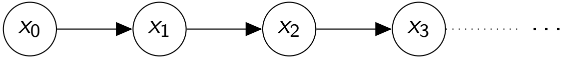
Markov network (usual representation):



Factor graph:



Bayesian network:



Markov?

- "*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.

Markov?

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

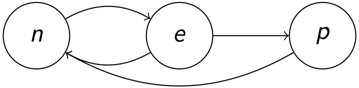
- Named after Andrey Markov (Russian mathematician) – invented them.

- Examples:
    - Much of physics, chemistry etc. Most simulations are Markov.
    - Stock market and other financial things.
    - Speech recognition.
    - Data transfer over noisy channels.

$$\vdots$$

Finite State Machines

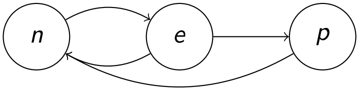
- Crossroad traffic lights as a finite state machine:



- 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

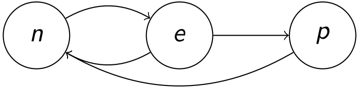
- Crossroad traffic lights unrolled as a Markov random chain:

$$n \rightarrow e \rightarrow n \rightarrow e \rightarrow p \rightarrow n \cdots\cdots\cdots \;\ldots$$

UNIVERSITY OF BATH

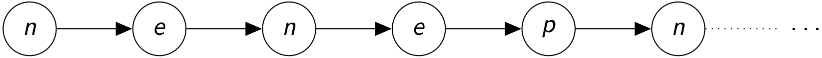
Finite State Machines

- Crossroad traffic lights as a finite state machine:



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

- Crossroad traffic lights unrolled as a Markov random chain:



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

Transition Matrix

A Markov random chain **can** be represented by:

- A set of n states, $\{s_0, s_1, \ldots, s_{n-1}\}$.
- A transition matrix $T \in \mathbb{R}^{n \times n}$, such that $P(s_a \rightarrow s_b) = T_{ab}$.
  ($P(s_a \rightarrow s_b)$ is the probability of going from s_a to s_b)

Transition Matrix

A Markov random chain **can** be represented by:

- A set of n states, $\{s_0, s_1, \ldots, s_{n-1}\}$.
- A transition matrix $T \in \mathbb{R}^{n \times n}$, such that $P(s_a \rightarrow s_b) = T_{ab}$.
  ($P(s_a \rightarrow s_b)$ is the probability of going from s_a to s_b)

Each row of T is a categorical distribution; to learn:

- Set a prior, e.g. a uniform Dirichlet distribution (conjugate).
- Update prior with collected data.
- Set T to the mean of the posterior (could draw).

UNIVERSITY OF BATH

Transition Matrix

A Markov random chain **can** be represented by:

- A set of n states, $\{s_0, s_1, \ldots, s_{n-1}\}$.
- A transition matrix $T \in \mathbb{R}^{n \times n}$, such that $P(s_a \rightarrow s_b) = T_{ab}$.
  ($P(s_a \rightarrow s_b)$ is the probability of going from s_a to s_b)

Each row of T is a categorical distribution; to learn:

- Set a prior, e.g. a uniform Dirichlet distribution (conjugate).
- Update prior with collected data.
- Set T to the mean of the posterior (could draw).

Explicitly:

- Fill T with 1.
- For each $s_a \rightarrow s_b$ in data increment T_{ab}.
- Normalise each row of T.

UNIVERSITY OF BATH

Simulation

To draw from the model:

- Not part of Markov chain:
  - 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)

- $x_{i+1} \sim P(x_i \rightarrow)$ for $i \in [1 \ldots \texttt{length} - 1]$.

Fake Place Names

Steps:

- Download UK place names from `https://www.paulstenning.com/uk-towns-and-counties-list/`
- Learn transition matrix.
- Learn categorical distribution over starting letter.
- Learn Poison distribution over length.

UNIVERSITY OF BATH

Fake Place Names

Steps:

- Download UK place names from `https://www.paulstenning.com/uk-towns-and-counties-list/`
- Learn transition matrix.
- Learn categorical distribution over starting letter.
- Learn Poison distribution over length.

- Draw “plausible” place names:

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

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

UNIVERSITY OF BATH

Observations

Real problems:

- Some states are fixed.
- Some states have distributions partially limiting their value.

Observations

Real problems:

- Some states are fixed.
- Some states have distributions partially limiting their value.

Possible model queries:

- Maximum Likelihood (ML)/Maximum a Posteriori (MAP) – the most probable state sequence.
- Marginals – per node distribution over state.

- Joint distribution – full distribution over unknown states (rarely practical).
- Conditional draw – draws from above.

(covering first two, ignoring second two)

Voice Recognition

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

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, effectively 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, effectively 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!

Marginals

- **Forward-backwards algorithm**.
- A kind of dynamic programming.

- marginal = "belief" = posterior distribution over RV given evidence.
- In layman's terms: Probabilities for what value it takes given what we know.

Marginals

- **Forward-backwards algorithm**.
- A kind of dynamic programming.

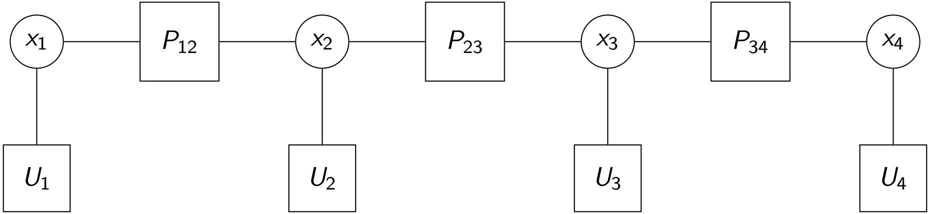
- marginal = "belief" = posterior distribution over RV given evidence.
- In layman's terms: Probabilities for what value it takes given what we know.

- Marginal is defined as:

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

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

Example



- x_i – Random variable.
- U_i – Unary, a factor on one RV.
- P_i – Pairwise, a factor on two RVs.

Message Passing

- Solve by sending messages along edges.
- Two messages per edge, one in each direction.
- Problem solved when all messages sent.

- Two kinds of message:
  - Random variable $\rightarrow$ Factor
  - Factor $\rightarrow$ Random variable

Stupid Message Passing

- Collect the joint distribution at every node.
- Can then marginalise to get the belief, b_i.

Stupid Message Passing

- Collect the joint distribution at every node.
- Can then marginalise to get the belief, b_i.

- $\therefore$ each factor must appear exactly once in the belief.
- $\therefore$ when a message is passed it must include all distributions "behind it".

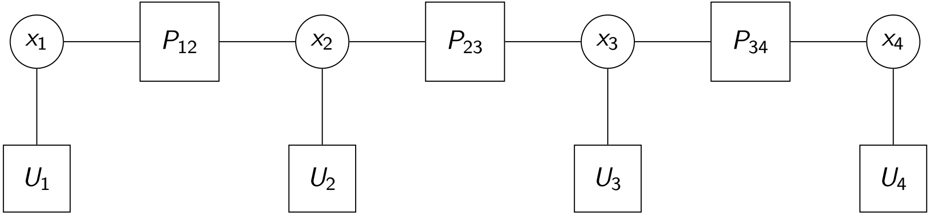
Stupid Message Passing

- Collect the joint distribution at every node.
- Can then marginalise to get the belief, b_i.

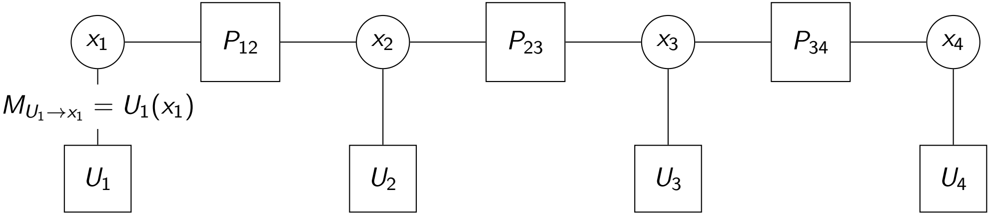
- $\therefore$ each factor must appear exactly once in the belief.
- $\therefore$ when a message is passed it must include all distributions "behind it".

- $\therefore$ Two passes: Forward (left to right) and backwards (right to left).

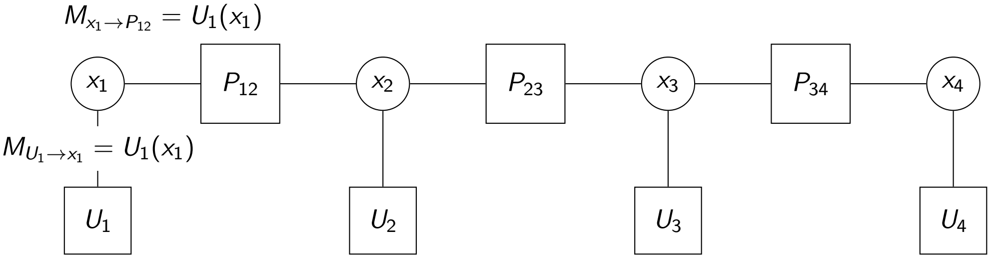
Example Forwards



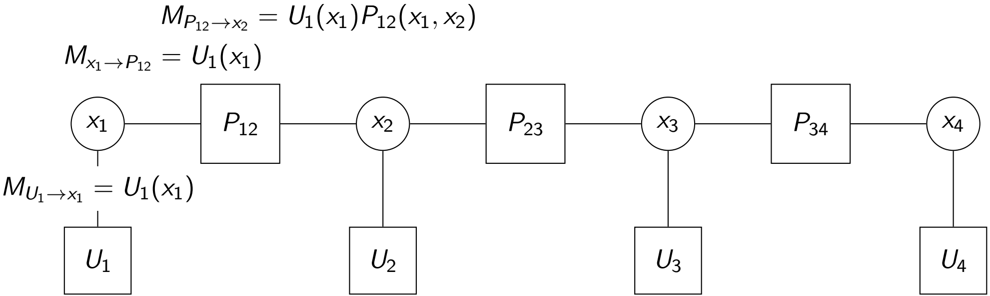
Example Forwards



Example Forwards



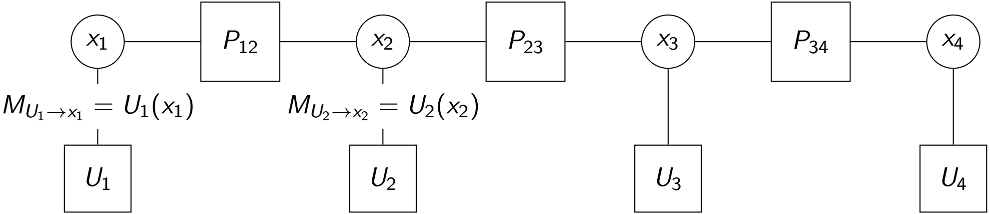
Example Forwards



Example Forwards

$$M_{P_{12}\to x_2} = U_1(x_1)P_{12}(x_1, x_2)$$

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

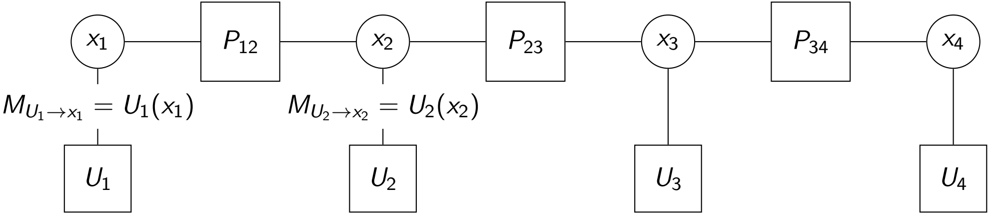


Example Forwards

$$M_{x_2 \to P_{23}} = U_1(x_1)P_{12}(x_1, x_2)U_2(x_2)$$

$$M_{P_{12} \to x_2} = U_1(x_1)P_{12}(x_1, x_2)$$

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



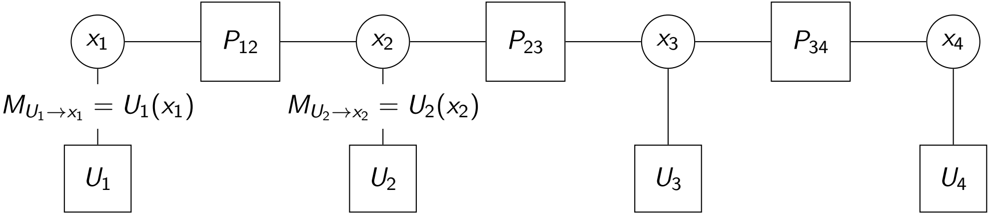
Example Forwards

$$M_{P_{23}\to x_3} = U_1(x_1)P_{12}(x_1,x_2)U_2(x_2)P_{23}(x_2,x_3)$$

$$M_{x_2\to P_{23}} = U_1(x_1)P_{12}(x_1,x_2)U_2(x_2)$$

$$M_{P_{12}\to x_2} = U_1(x_1)P_{12}(x_1,x_2)$$

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



Smart Message Passing

- Observation: Messages passed along edge include all factors behind direction of travel.
- Conclusion: Product of all messages entering a node includes all factors.
- $\therefore$ Have all information required to calculate beliefs.

Smart Message Passing

- Observation: Messages passed along edge include all factors behind direction of travel.
- Conclusion: Product of all messages entering a node includes all factors.
- $\therefore$ Have all information required to calculate beliefs.

- Problem: Messages end up as functions of all RVs – too big.
- Belief requires marginalising out – nodes don't need this extra state.
- Marginalise out all RVs behind a message.

Smart Message Passing

- Observation: Messages passed along edge include all factors behind direction of travel.
- Conclusion: Product of all messages entering a node includes all factors.
- $\therefore$ Have all information required to calculate beliefs.

- Problem: Messages end up as functions of all RVs – too big.
- Belief requires marginalising out – nodes don't need this extra state.
- Marginalise out all RVs behind a message.

- Messages only passed when all dependent messages received.
- Can be simplified: Messages passed between RVs directly.

Equation

- Messages sent by RVs:

$$M_{v\to F} = \sum_{x\notin F}\left[\prod_{G\in N_v/F} M_{G\to v}\right]$$

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

- Messages sent by factors:

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

Maximum a Posteriori

- What if we want the most probable solution?
- Viterbi algorithm.

Maximum a Posteriori

- What if we want the most probable solution?
- Viterbi algorithm.

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

Maximum a Posteriori

- What if we want the most probable solution?
- Viterbi algorithm.

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

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

Maximum a Posteriori

- What if we want the most probable solution?
- Viterbi algorithm.

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

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

- Best to record which RV won each max. Draws are common!
- $\therefore$ Only forward pass required.
- Can be implemented as a recursive function.

Dynamic Programming Notes

- Dynamic programming:
  "Any algorithm where the solution can be found recursively by solving slightly smaller problems first."
- Forwards-backwards and Viterbi are both instances of this; many others, e.g. Levenshtein distance.
- People often just say dynamic programming when they mean one of these.

Dynamic Programming Notes

- Dynamic programming:
  "*Any algorithm where the solution can be found recursively by solving slightly smaller problems first.*"
- Forwards-backwards and Viterbi are both instances of this; many others, e.g. Levenshtein distance.
- People often just say dynamic programming when they mean one of these.

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

Dynamic Programming Tricks

- Alignment: You know the data labels but not where they are.
- Solution: Labels represent where in the sequence, 0 probability of transitioning to any but the next.

Dynamic Programming Tricks

- Alignment: You know the data labels but not where they are.
- Solution: Labels represent where in the sequence, 0 probability of transitioning to any but the next.

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

Dynamic Programming Tricks

- Alignment: You know the data labels but not where they are.
- Solution: Labels represent where in the sequence, 0 probability of transitioning to any but the next.

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

- n-gram: States are dependent on not just neighbours but the neighbours of their neighbours.
- Explode states to encode history.

Belief Propagation

- Equations above will just work! Just have more messages to loop over.
- Fitting model to data is the same as well.

Belief Propagation

- Equations above will just work! Just have more messages to loop over.
- Fitting model to data is the same as well.

- sum-product = generalisation of forward-backwards.
- max-product = generalisation of Viterbi.

- min-sum = identical to max-product, but performed in negative log space (costs).

UNIVERSITY OF BATH

Belief Propagation Notes

- Major limitation: Have to pass distributions between nodes.

  (MCMC and variational will solve)

UNIVERSITY OF BATH

Belief Propagation Notes

- Major limitation: Have to pass distributions between nodes.
  (MCMC and variational will solve)

- Proper generative model: Markov Random Field (MRF).
  Distributions depend on data: Conditional Random Field (CRF)

Belief Propagation Notes

- Major limitation: Have to pass distributions between nodes.
  (MCMC and variational will solve)

- Proper generative model: Markov Random Field (MRF).
  Distributions depend on data: Conditional Random Field (CRF)

- Kalman smoothing generalised to an arbitrary graph is **Gaussian Belief Propagation**.
  (same marginal and MAP property)
- Other continuous distributions doable, but much harder.

Belief Propagation Notes

- Major limitation: Have to pass distributions between nodes.
  (MCMC and variational will solve)

- Proper generative model: Markov Random Field (MRF).
  Distributions depend on data: Conditional Random Field (CRF)

- Kalman smoothing generalised to an arbitrary graph is **Gaussian Belief Propagation**.
  (same marginal and MAP property)
- Other continuous distributions doable, but much harder.

- Loopy belief propagation approximately solves graphs with loops.
  Ignores dependencies and just keeps sending messages!
  (better choices, particularly graph cuts. Some situations where it can be solved precisely (Ising/Potts models and GBP))

Summary

- Markov chain.
- Dynamic programming.
- 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/`