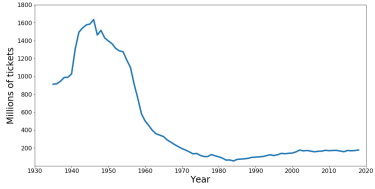
Machine Learning 2.14: Time

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

Time

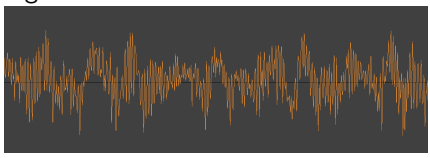
- Time = sequences

UK cinema ticket sales:



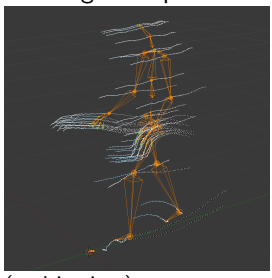
(discrete)

Light saber audio:



(continuous but sampled)

Walking mocap:



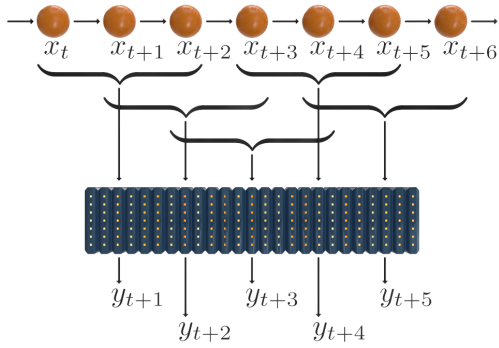
(multivariate)

- Equal sampling most common
- Doesn't have to be time – anything linear

Windows

- Easy approach: Short temporal windows
  (much like convolutional neural networks)
- Can overlap, have stride, arbitrary width etc.

- Lack of temporal coherence
  (noisy)
- Stacking: Second layer to clean up first
  (Mean, voting, ML, Markov random chain. . .)



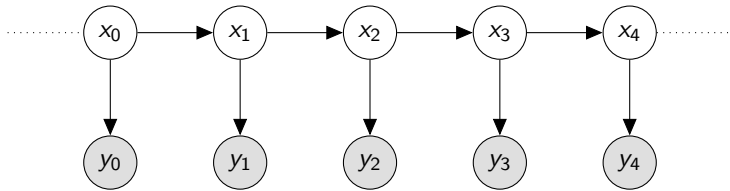
Markov property

- Last discussed: Lecture 12 of Machine Learning 1

- "*Each moment in time depends only on the last*"
- A statement about conditional independence:

$$P(x_0, x_1, x_2, x_3, x_4) = P(x_4|x_3)P(x_3|x_2)P(x_2|x_1)P(x_1|x_0)P(x_0)$$

Hidden Markov model



- x: Unobserved (hidden) Markov chain
- y: Observations, dependent only on simultaneous hidden state

Maximum a posteriori

- Find $\underset{x}{\text{argmax}}\, P(x|y)$
- Belief propagation, again (yay!)

UNIVERSITY OF BATH

Maximum a posteriori

- Find $\underset{x}{\text{argmax}}\, P(x|y)$
- Belief propagation, again (yay!)

- Going to walk through HMM MAP specialisation
- Names:
  - Viterbi
  - Dynamic programming

Smoothing

- Many uses of HMM, e.g.
  - Voice recognition
  - Part-of-speech tagging
  - Tracking (video)
  - Genetic information

UNIVERSITY OF BATH

Smoothing

- Many uses of HMM, e.g.
  - Voice recognition
  - Part-of-speech tagging
  - Tracking (video)
  - Genetic information

    ⋮
- Common use: Smoothing / regularisation
  - Observed: Noisy state
  - Unobserved: True state

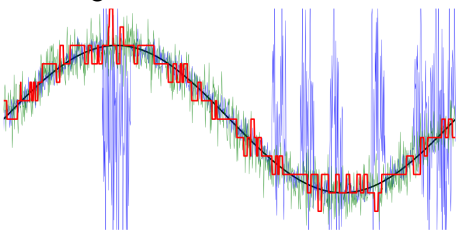
Smoothing

- Many uses of HMM, e.g.
  - Voice recognition
  - Part-of-speech tagging
  - Tracking (video)
  - Genetic information

  ⋮
- Common use: Smoothing / regularisation
  - Observed: Noisy state
  - Unobserved: True state

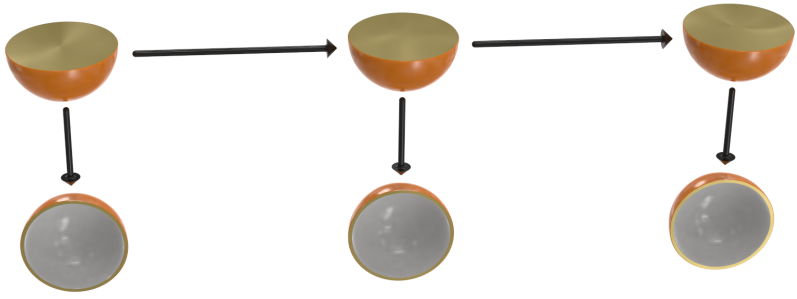
- Noise:
  - Measurement error
  - ML ($\text{state} \approx f(\text{real observation})$)
  - Windows are incorrect...

    ...but work really well!
- Smoothing?

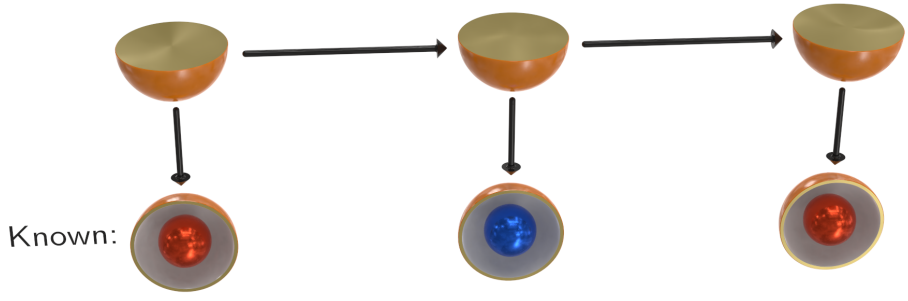


UNIVERSITY OF BATH

Message passing

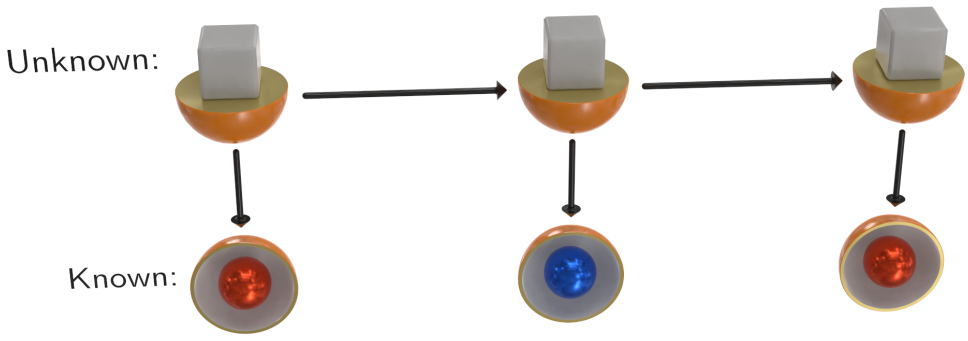


Message passing

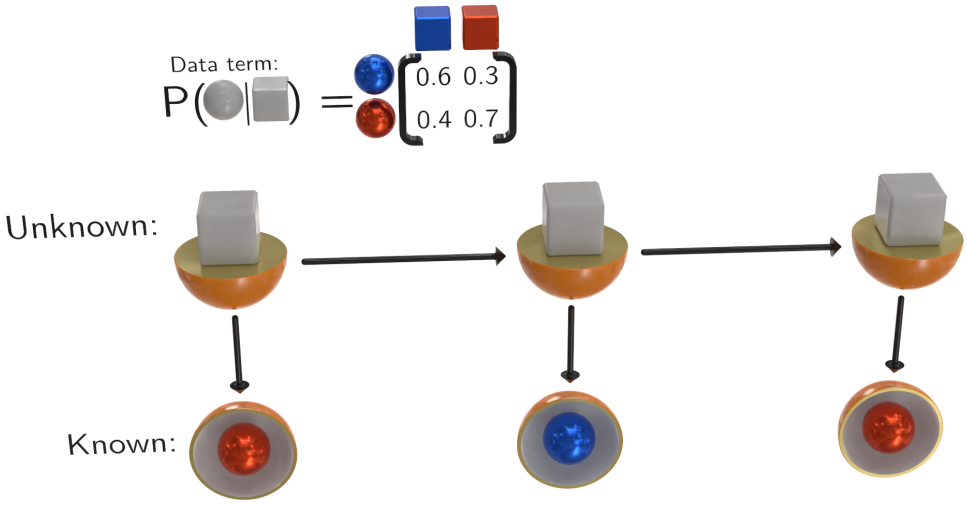


UNIVERSITY OF BATH

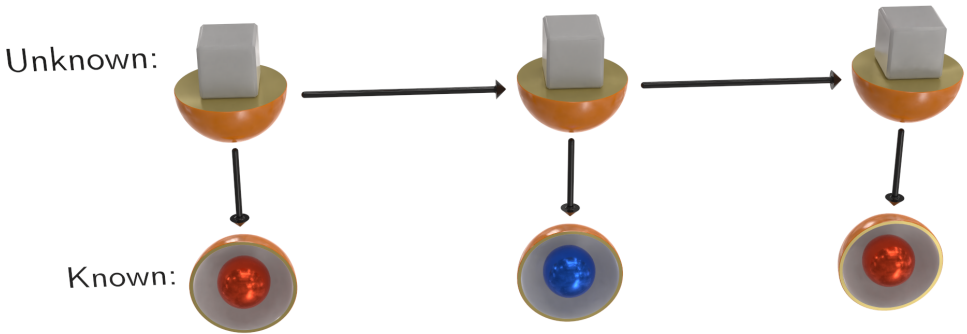
Message passing



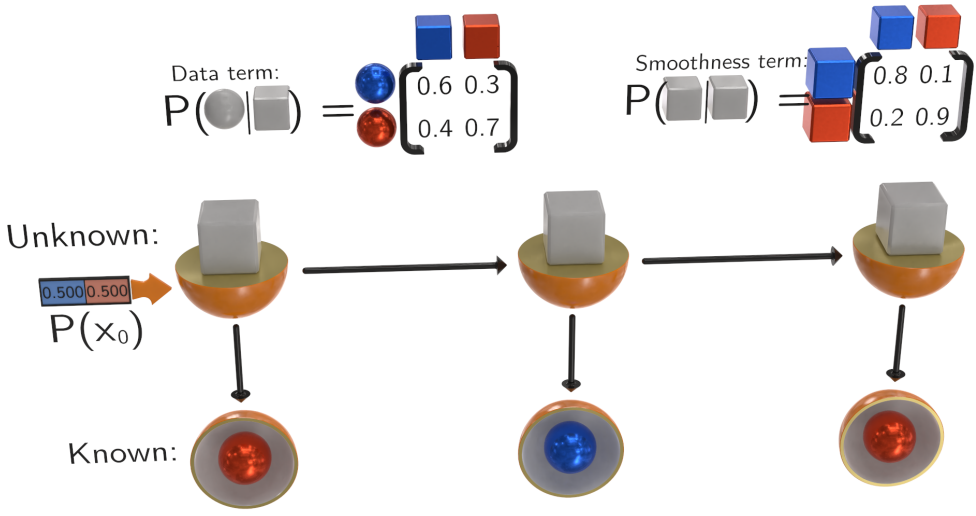
Message passing



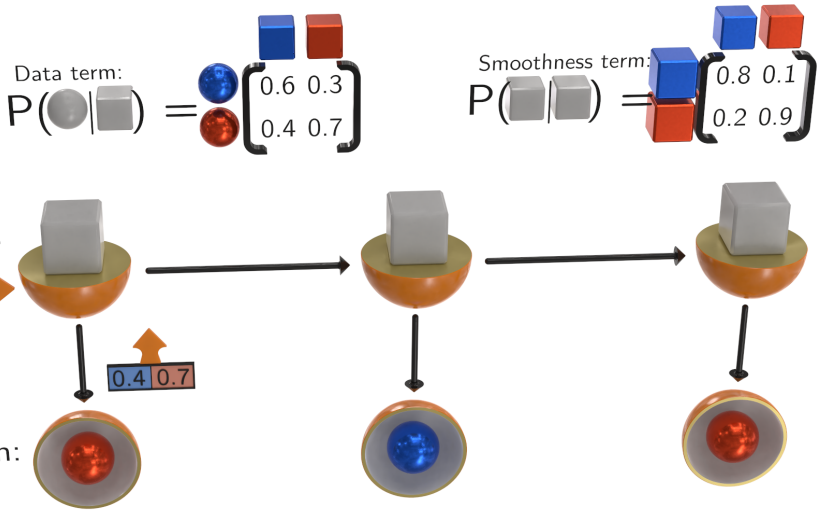
Message passing



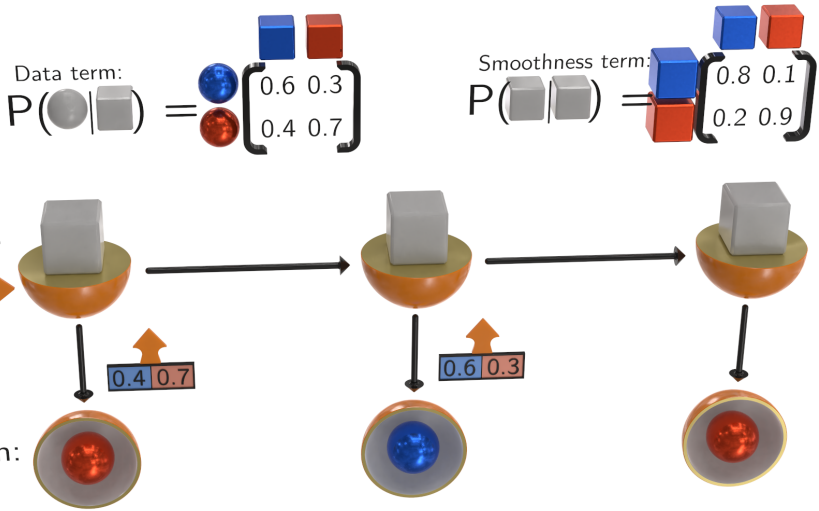
Message passing



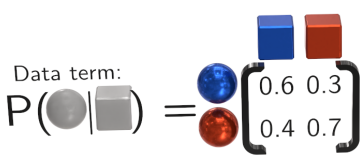
Message passing

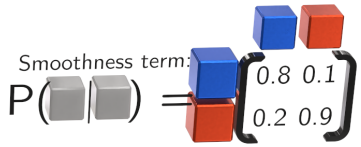


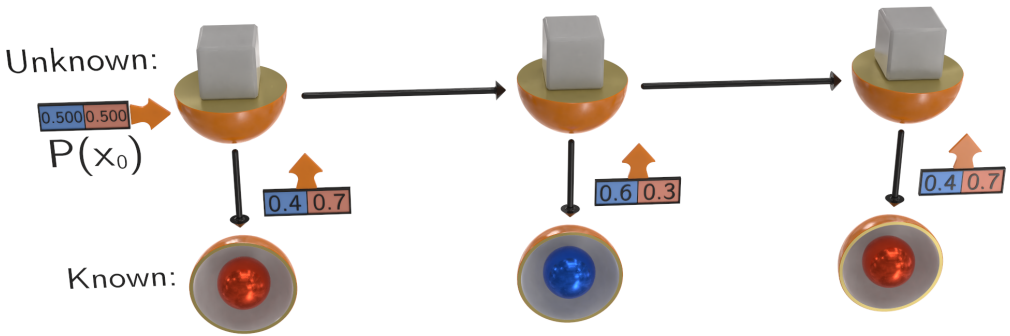
Message passing



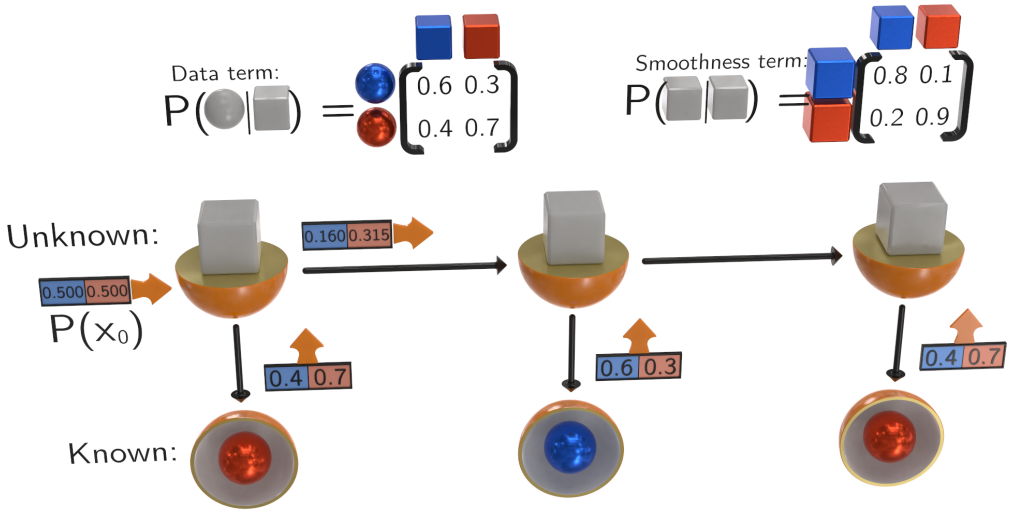
Message passing



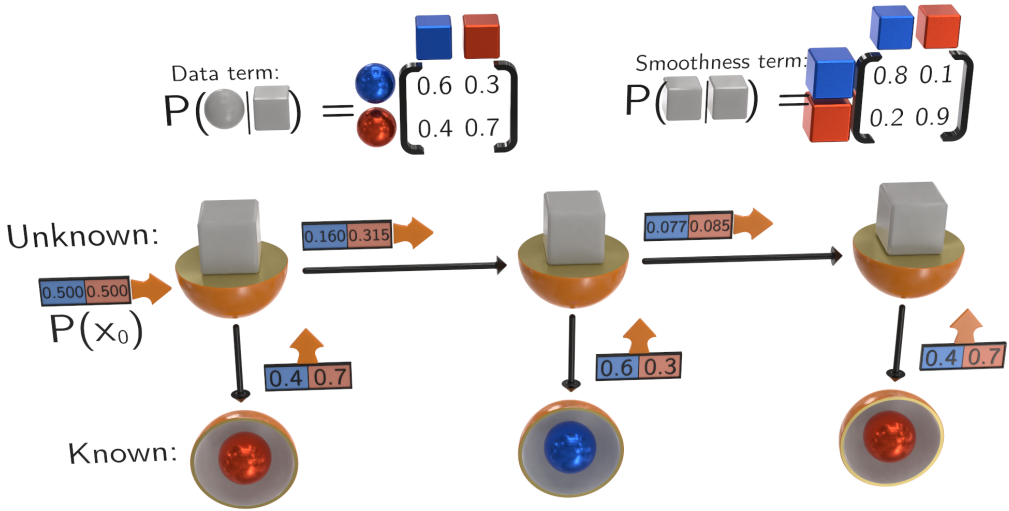




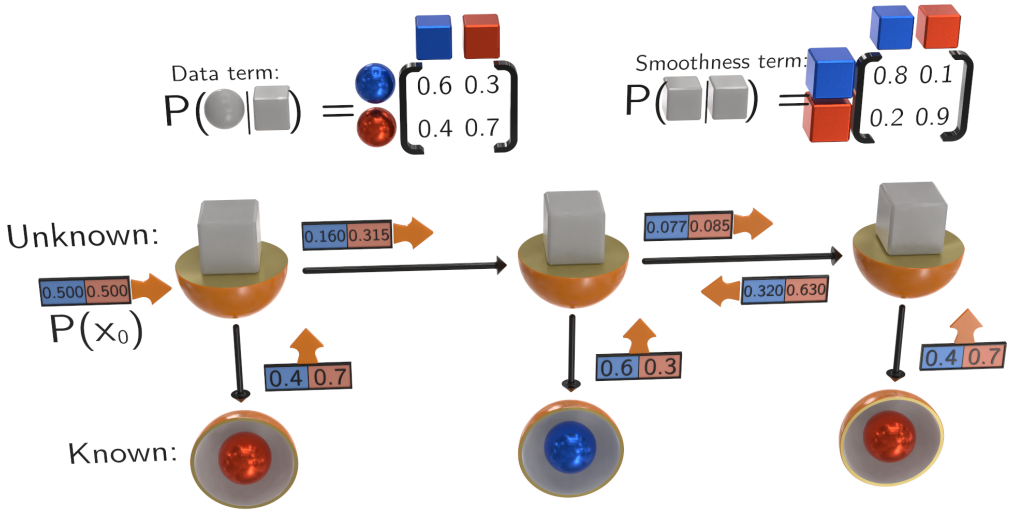
Message passing



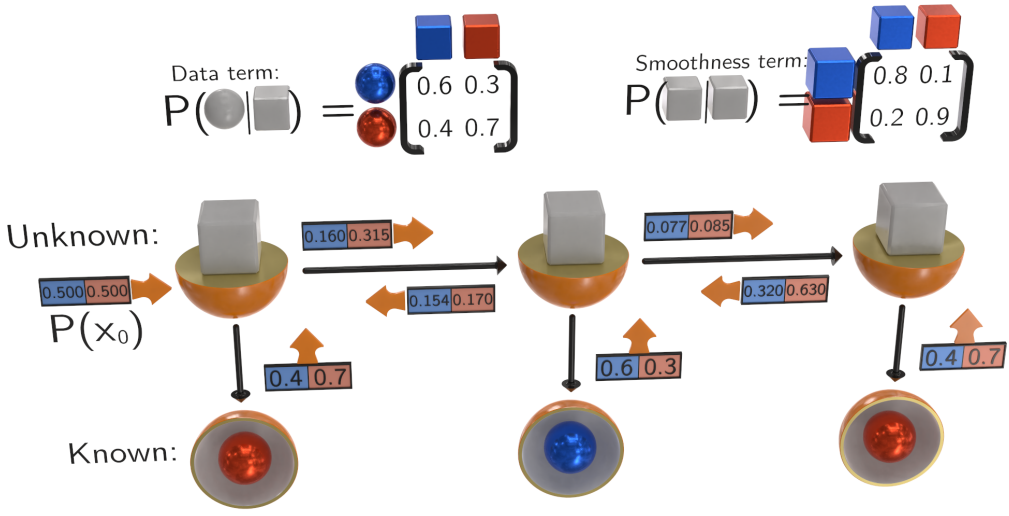
Message passing



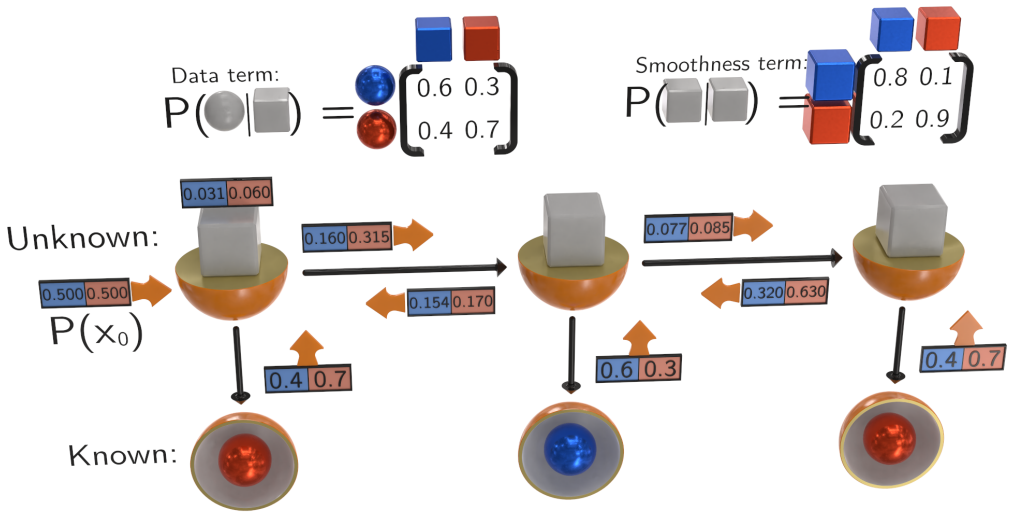
Message passing



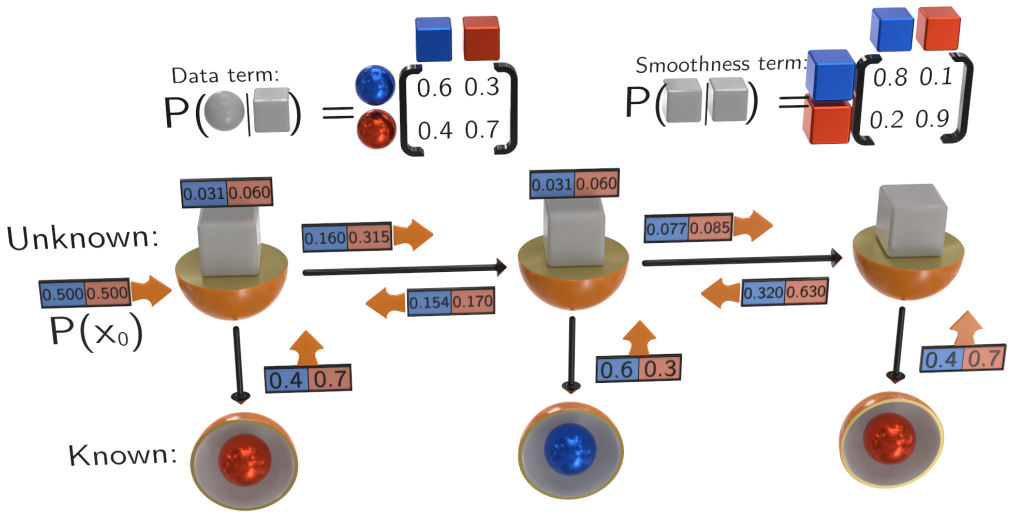
Message passing



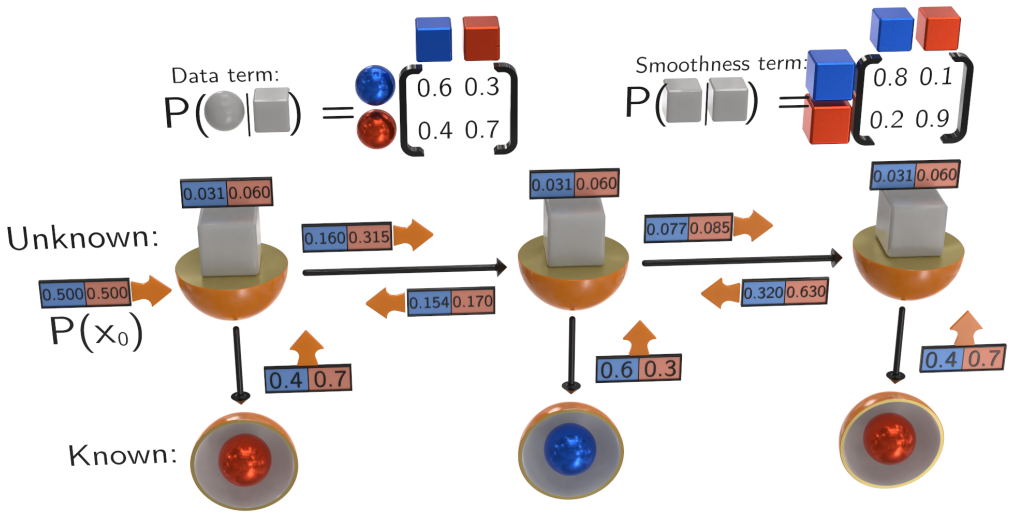
Message passing



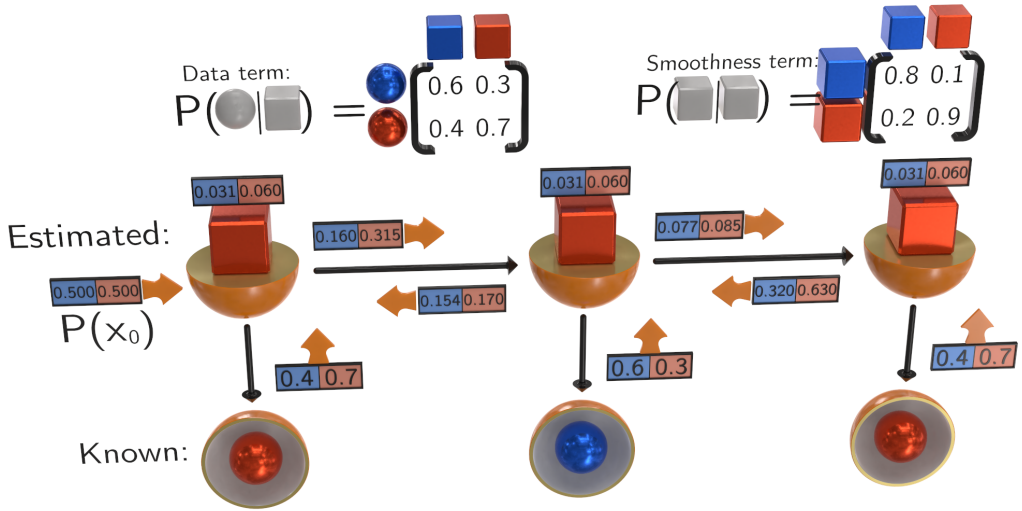
Message passing



Message passing



Message passing



Maths

- Forwards pass:

$$M(x_{t+1})_{t\to t+1} = \max_{x_t} [P(x_{t+1}|x_t)P(y_t|x_t)M_{t-1\to t}(x_t)]$$

- Backwards pass:

$$M(x_{t-1})_{t\to t-1} = \max_{x_t} [P(x_t|x_{t-1})P(y_t|x_t)M_{t+1\to t}(x_t)]$$

- Belief: (output)

$$P(x_t) = M(x_t)_{t-1\to t}M(x_t)_{t+1\to t}P(y_t|x_t)$$

(Messages at chain ends: Replace with $P(x_0)$ or 1)

UNIVERSITY OF BATH

Normalisation

- Numbers will underflow!
- One fix: Normalise messages

$$M(x_t)' = \frac{M(x_t)}{\sum_{x_t} M(x_t)}$$

Normalisation

- Numbers will underflow!
- One fix: Normalise messages

$$M(x_t)' = \frac{M(x_t)}{\sum_{x_t} M(x_t)}$$

- However, without normalisation:

$$\text{belief}(\text{state}) = P(\text{best solution}|\text{RV} = \text{state})$$

UNIVERSITY OF BATH

Normalisation

- Numbers will underflow!
- One fix: Normalise messages

$$M(x_t)' = \frac{M(x_t)}{\sum_{x_t} M(x_t)}$$

- However, without normalisation:

$$\text{belief}(\text{state}) = P(\text{best solution}|\text{RV} = \text{state})$$

- Log space!
  (multiplication $\rightarrow$ addition $\therefore$ faster!)
- Preserves numerical stability... for a while (still breaks when chain gets too long)

Normalisation

- Numbers will underflow!
- One fix: Normalise messages

$$M(x_t)' = \frac{M(x_t)}{\sum_{x_t} M(x_t)}$$

- However, without normalisation:

$$\text{belief}(\text{state}) = P(\text{best solution}|\text{RV} = \text{state})$$

- Log space!
  (multiplication $\rightarrow$ addition $\therefore$ faster!)
- Preserves numerical stability... for a while (still breaks when chain gets too long)
- Normalise $\rightarrow$ subtract maximum from messages (= scale so max is 1)

$$\log M(x_t)' = \log M(x_t) - \max_{x_t} (\log M(x_t))$$

- Keep total $\log P(x)$ removed – can add back to get (log) probability of solution

Don't go backwards

- Last RV (x_{T-1}) correct after forward pass

UNIVERSITY OF BATH

Don't go backwards

- Last RV (x_{T-1}) correct after forward pass
- Filtering = only need state of last node
  (as opposed to smoothing)
- Usually incremental
  – data/decisions in real time
- Need $P(\text{best})$? (probability without state)

$$\max_{x_{T-1}} P(x_{T-1})$$

Don't go backwards

- Last RV (x_{T-1}) correct after forward pass
- Filtering = only need state of last node
  (as opposed to smoothing)
- Usually incremental
  – data/decisions in real time
- Need $P(\text{best})$? (probability without state)

$$\max_{x_{T-1}} P(x_{T-1})$$

- Need maximum a posteriori? (MAP)
- Keep *pointers*:
  - Forward pass: Track winning x_{t-1} of each max
  - Backwards pass: Follow chain of pointers!
  - Faster (no computation)
- Handles ties
  (much better results when it matters)

n^{th} order

- Thus far: 1^{st} order
- 2^{nd} order: Depends on last two time steps

$$P(x_0, x_1, x_2, x_3, x_4) = P(x_4|x_3, x_2)P(x_3|x_2, x_1)P(x_2|x_1, x_0)P(x_1|x_0)P(x_0)$$

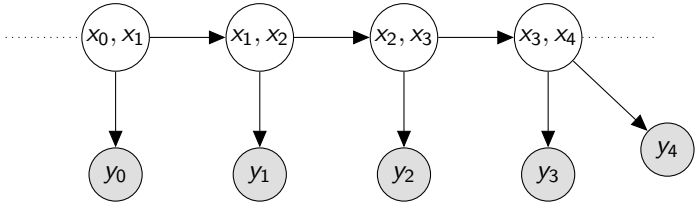
- And so on, 3^{rd} etc. (gets slow – rare)

n^{th} order

- Thus far: 1^{st} order
- 2^{nd} order: Depends on last two time steps

$$P(x_0, x_1, x_2, x_3, x_4) = P(x_4|x_3, x_2)P(x_3|x_2, x_1)P(x_2|x_1, x_0)P(x_1|x_0)P(x_0)$$

- And so on, 3^{rd} etc. (gets slow – rare)
- Convert to 1^{st} order:



$$P(x_{t+2}, x_{t+1}|x'_{t+1}, x_t) = P(x_{t+2}|x_{t+1}, x_t)\delta(x_{t+1} - x'_{t+1})$$

($\delta(.)$ = Kronecker delta function; can hang each y_t off either possible parent)

Notes

- Terminology:

$$-\log P(x) = \text{Energy} = \text{Cost}$$

- Note negative: max $\rightarrow$ min

- Wide enough window + ML can beat HMM
  - Has to learn Markov property
  - Needs lots of data

UNIVERSITY OF BATH

Now

- Hidden Markov model
- Inferring HMM parameters
  (Baum–Welch)
- Kalman filter

Learning

- Need to learn:
  1. $P(x_0)$
  2. $P(y_t|x_t)$ (data term)
  3. $P(x_{t+1}|x_t)$ (smoothness term)

UNIVERSITY OF BATH

Learning

- Need to learn:
  1. $P(x_0)$
  2. $P(y_t|x_t)$ (data term)
  3. $P(x_{t+1}|x_t)$ (smoothness term)

- Terms can vary with t
- Smoothness term is function of data $\implies$ *Conditional hidden Markov model*

UNIVERSITY OF BATH

Learning

- Need to learn:
  1. $P(x_0)$
  2. $P(y_t|x_t)$ (data term)
  3. $P(x_{t+1}|x_t)$ (smoothness term)

- Terms can vary with t
- Smoothness term is function of data $\implies$ *Conditional hidden Markov model*

- Three scenarios:
  1. Hidden state available to train: Trivial
  2. Never see hidden state: Baum–Welch
  3. Unaligned labels: Baum–Welch variant

Baum–Welch

- An Expectation-maximisation (EM) algorithm
  (see lecture 13, machine learning 1)

Baum–Welch

- An Expectation-maximisation (EM) algorithm
  (see lecture 13, machine learning 1)
- E-step: Calculate **marginals** using *forward–backwards*
  - Not Viterbi!
  - Another belief propagation specialisation
    (see lecture 12, machine learning 1)
  - MAP with max replaced by $\sum$
- M-step: Optimise *data* and *smoothness* terms to maximise probability
- Initialise; iterate until convergence

E-step I

- Just belief propagation/forward–backwards/dynamic programming:
  - Forwards pass:
$$M(x_{t+1})_{t\to t+1} = \sum_{x_t} [P(x_{t+1}|x_t)P(y_t|x_t)M_{t-1\to t}(x_t)]$$
  - Backwards pass:
$$M(x_{t-1})_{t\to t-1} = \sum_{x_t} [P(x_t|x_{t-1})P(y_t|x_t)M_{t+1\to t}(x_t)]$$

E-step II

- Assume $x_t \in \mathcal{S}$, $|\mathcal{S}| = N$, $\mathcal{T} = \{0, \ldots, T-1\}$
- Calculate:

$$\beta_t(s) = P(x_t = s) \propto M(x_t)_{t-1 \to t}[s] M(x_t)_{t+1 \to t}[s] P(y_t | x_t = s)$$

$$\rho_t(s, l) = P(x_t = s, x_{t+1} = l) \propto$$
$$M(x_t)_{t-1 \to t}[s] P(x_{t+1} = l | x_t = s) M(x_{t+1})_{t+2 \to t+1}[l] P(y_t | x_t = s) P(y_{t+1} | x_{t+1} = l)$$

  - $\beta_t(s) =$ belief
  - $\rho_t(s, l) =$ pairwise belief

(resolve $\propto$ by dividing through with $\sum$)

M-step

- x_t is discrete $\therefore$

$$A_{rc} = P(x_{t+1} = r | x_t = c)$$

- Assume y_t discrete $\therefore$

$$B_{rc} = P(y_t = r | x_t = c)$$

M-step

- x_t is discrete $\therefore$

$$A_{rc} = P(x_{t+1} = r | x_t = c)$$

- Assume y_t discrete $\therefore$

$$B_{rc} = P(y_t = r | x_t = c)$$

- M-step:

$$A_{rc} = \frac{\sum_{t \in \{0,\ldots,T-2\}} \rho_t(r,c)}{\sum_{t \in \mathcal{T}} \beta_t(c)}, \qquad B_{rc} = \frac{\sum_{\{t \in \mathcal{T}; y_t = r\}} \beta_t(c)}{\sum_{t \in \mathcal{T}} \beta_t(c)}$$

- Note: y_t can be continuous: Weighted density estimate

Initialisation

- Initialisation: Random PDFs
  (uniform will not work)
- Can get stuck in local minima
- Include prior

Initialisation

- Initialisation: Random PDFs
  (uniform will not work)
- Can get stuck in local minima
- Include prior

- Use log space
- Initial state:
  - One chain: Impossible
  - Many chains: Trivial

UNIVERSITY OF BATH

Unaligned labels

- Know order of labels but not time
- Faster for human labeller

UNIVERSITY OF BATH

Unaligned labels

- Know order of labels but not time
- Faster for human labeller

- Transform state space:
  - States = which training label
  - $P(x_{t+1}|x_t)$, smoothing term:
    Only allows you to stay on current label/move to next
- Fix first RV to be first label
- Fix last RV to be last label

- Baum–Welch as before
  (transform state space back to calculate pairwise beliefs)

UNIVERSITY OF BATH

Kalman filter

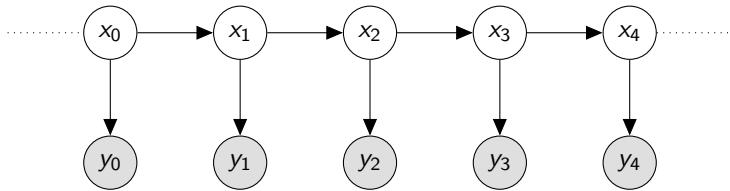
- Invented in 1960 by Kalman!
- Motivated by missiles (cold war)

- Used by Apollo
- Apollo development lead to *extended Kalman filter*

- These days:
  - Tracking with GPS + accelerometer...
    Cars, ships, planes, spacecraft, robots, *your phone*

Familiar?

- **Continuous** hidden Markov model
- **Gaussian** distributions
- Primarily for smoothing: Estimating true value from noisy measurements



- x: Unobserved (hidden) Markov chain
- y: Observations, dependent only on simultaneous hidden state

UNIVERSITY OF BATH

Gaussian algebra

- Represent Gaussian as

$$\sigma[\mathbf{P}\mu, \mathbf{P}] \propto \exp\left[-\frac{1}{2}(x-\mu)^T\mathbf{P}(x-\mu)\right]$$

($\mathbf{P}\mu$ = p-mean, $\mathbf{P}$ = precision)

Gaussian algebra

- Represent Gaussian as

$$\sigma[\mathbf{P}\mu, \mathbf{P}] \propto \exp\left[-\frac{1}{2}(x-\mu)^T\mathbf{P}(x-\mu)\right]$$

($\mathbf{P}\mu$ = p-mean, $\mathbf{P}$ = precision)

- Offset:

$$\sigma[\mathbf{P}\mu, \mathbf{P}] + v = \sigma[\mathbf{P}\mu + \mathbf{P}v, \mathbf{P}]$$

- Scale:

$$\mathbf{S}\sigma[\mathbf{S}^{-\mathbf{T}}\mathbf{P}\mu, \mathbf{S}^{-\mathbf{T}}\mathbf{P}\mathbf{S}^{-1}]$$

UNIVERSITY OF BATH

Gaussian algebra

- Represent Gaussian as

$$\sigma[\mathbf{P}\mu, \mathbf{P}] \propto \exp\left[-\frac{1}{2}(x-\mu)^T\mathbf{P}(x-\mu)\right]$$

($\mathbf{P}\mu$ = p-mean, $\mathbf{P}$ = precision)

- Offset:

$$\sigma[\mathbf{P}\mu, \mathbf{P}] + v = \sigma[\mathbf{P}\mu + \mathbf{P}v, \mathbf{P}]$$

- Scale:

$$\mathbf{S}\sigma[\mathbf{S}^{-\mathbf{T}}\mathbf{P}\mu, \mathbf{S}^{-\mathbf{T}}\mathbf{P}\mathbf{S}^{-1}]$$

- Multiplication:

$$\sigma[\mathbf{P}_1\mu_1, \mathbf{P}_1]\sigma[\mathbf{P}_2\mu_2, \mathbf{P}_2] = \sigma[\mathbf{P}_1\mu_1+\mathbf{P}_2\mu_2, \mathbf{P}_1+\mathbf{P}_2]$$

- Addition:

$$\sigma[\mathbf{P}_1\mu_1, \mathbf{P}_1]+\sigma[\mathbf{P}_2\mu_2, \mathbf{P}_2] = \sigma[\mathbf{P}'(\mu_1+\mu_2), \mathbf{P}']$$

$$\mathbf{P}' = (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1}$$

UNIVERSITY OF BATH

Gaussian algebra

- Represent Gaussian as

$$\sigma[\mathbf{P}\mu, \mathbf{P}] \propto \exp\left[-\frac{1}{2}(x-\mu)^T\mathbf{P}(x-\mu)\right]$$

($\mathbf{P}\mu$ = p-mean, $\mathbf{P}$ = precision)

- Offset:

$$\sigma[\mathbf{P}\mu, \mathbf{P}] + v = \sigma[\mathbf{P}\mu + \mathbf{P}v, \mathbf{P}]$$

- Scale:

$$\mathbf{S}\sigma[\mathbf{S}^{-\mathbf{T}}\mathbf{P}\mu, \mathbf{S}^{-\mathbf{T}}\mathbf{P}\mathbf{S}^{-1}]$$

- Multiplication:

$$\sigma[\mathbf{P}_1\mu_1, \mathbf{P}_1]\sigma[\mathbf{P}_2\mu_2, \mathbf{P}_2] = \sigma[\mathbf{P}_1\mu_1+\mathbf{P}_2\mu_2, \mathbf{P}_1+\mathbf{P}_2]$$

- Addition:

$$\sigma[\mathbf{P}_1\mu_1, \mathbf{P}_1]+\sigma[\mathbf{P}_2\mu_2, \mathbf{P}_2] = \sigma[\mathbf{P}'(\mu_1+\mu_2), \mathbf{P}']$$

$$\mathbf{P}' = (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1}$$

- Extend with degenerate RV:

$$\text{Extend}_{1*}\left(\sigma[\mathbf{P}\mu, \mathbf{P}]\right) = \sigma\left[\begin{pmatrix}\mathbf{P}\mu \\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{pmatrix}\right]$$

- Marginalise: (sum over)

$$\text{Marg}_1\left(\sigma\left[\begin{pmatrix}\mathbf{h}_1 \\ \mathbf{h}_2\end{pmatrix}, \begin{pmatrix}\mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22}\end{pmatrix}\right]\right)$$

$$= \sigma[\mathbf{h}_2 - \mathbf{P}_{12}\mathbf{P}_{11}^{-1}\mathbf{h}_1, \mathbf{P}_{22} - \mathbf{P}_{12}\mathbf{P}_{11}^{-1}\mathbf{P}_{12}^T]$$

Transforms

- Data term:

$$y_t = H_t x_t + \sigma[\mathbf{0}, \mathbf{R}_t^{-1}]$$

  - H_t = mapping from true state to observed state
  - $\sigma[\mathbf{0}, \mathbf{R}_t^{-1}]$ = measurement noise

- "Smoothness" term:

$$x_{t+1} = F_t x_t + u_t + \sigma[\mathbf{0}, \mathbf{Q}_t^{-1}]$$

  - F_t = mapping from old state to new state
  - u_t = control signal (e.g. thrusters)
  - $\sigma[\mathbf{0}, \mathbf{Q}_t^{-1}]$ = noise

- Note: All linear

Message passing

- Gaussian algebra $\implies$ can just rearrange!
- All messages Gaussian

- Forwards:

$$M(x_{t+1})_{t\to t+1} = F_t\left(M(x_t)_{t-1\to t}H_t^{-1}\sigma[\mathbf{R}^{-1}y_t, \mathbf{R}^{-1}]\right) + u_t + \sigma[\mathbf{0}, \mathbf{Q}_t^{-1}]$$

- Backwards:

$$M(x_{t-1})_{t\to t-1} = F_t^{-1}\left[\left(M(x_t)_{t+1\to t}H_t^{-1}\sigma[\mathbf{R}^{-1}y_t, \mathbf{R}^{-1}]\right) - u_t + \sigma[\mathbf{0}, \mathbf{Q}_t^{-1}]\right]$$

- Belief:

$$P(x_t) = M(x_t)_{t-1\to t}M(x_t)_{t+1\to t}H_t^{-1}\sigma[\mathbf{R}^{-1}y_t, \mathbf{R}^{-1}]$$

Learning

- Problem specific

- Optimisation with data set (gradient descent)
- May involve calibration
  (experiments with real hardware)

SLAM

- "*Simultaneous localisation and mapping*"
  - Robots
  - Build/refines map of world
  - Keeps track of own location

SLAM

- "*Simultaneous localisation and mapping*"
  - Robots
  - Build/refines map of world
  - Keeps track of own location
- First that only used a camera:
  `https://www.youtube.com/watch?v=mimAWVm-0qA`
  (Kalman filter for camera location/rotation and feature locations)
- "*MonoSLAM: Real-Time Single Camera SLAM*",
  by Davison, Reid, Molton & Stasse (2007)

UNIVERSITY OF BATH

From video

Notes

- Another demo: Load up Google maps and go for a walk!
- Not usually presented with Gaussian algebra

Notes

- Another demo: Load up Google maps and go for a walk!
- Not usually presented with Gaussian algebra

- Extended Kalman filter:
  - F and H no longer linear – arbitrary functions
  - Updating mean as expected
  - Precision approximated: Updated using linear approximation around current point (First term of Taylor expansion, need Jacobian matrices)

Notes

- Another demo: Load up Google maps and go for a walk!
- Not usually presented with Gaussian algebra

- Extended Kalman filter:
  - F and H no longer linear – arbitrary functions
  - Updating mean as expected
  - Precision approximated: Updated using linear approximation around current point (First term of Taylor expansion, need Jacobian matrices)

- Unscented Kalman filter:
  Better linear approximation using sampling

UNIVERSITY OF BATH

Summary

- Hidden Markov models
- Baum–Welch to infer parameters
- Kalman filter/smoothing

Further reading

- Original Kalman filter paper:
  "*A New Approach to Linear Filtering and Prediction Problems*",
  by Kalman (1960) (worth reading, but find transcribed version!)

- If Gaussian is a bad assumption you need a particle filter:
  "*A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*",
  by Arulampalam, Maskell, Gordon & Clapp (2002)