Sequential Data

CMPT 419/726
Mo Chen
SFU Computing Science
Mar. 2, 2020

Bishop PRML Ch. 13
Russell and Norvig, AIMA

Outline

Outline

Temporal Models

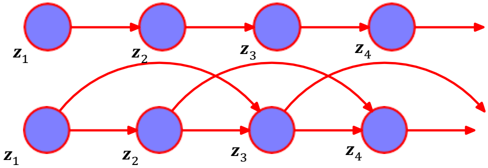
- The world changes over time
  - Explicitly model this change using Bayesian networks
  - Undirected models also exist (will not cover)
- Basic idea: copy state and evidence variables for each time step

e.g. Diabetes management

- $\boldsymbol{z}_t$ is set of unobservable state variables at time t
  - $bloodSugar_t$, $stomachContents_t$, ...
- $\boldsymbol{x}_t$ is set of observable evidence variables at time t
  - $measuredBloodSugar_t$, $foodEaten_t$, ...
- Assume discrete time step, fixed
- Notation: $\boldsymbol{x}_{a:b} = \boldsymbol{x}_a, \boldsymbol{x}_{a+1}, \ldots, \boldsymbol{x}_{b-1}, \boldsymbol{x}_b$

Markov Chain

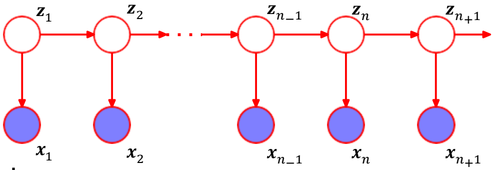
- Construct Bayesian network from these variables
  - parents? distributions? for state variables $\boldsymbol{z}_t$:
- Markov assumption: $\boldsymbol{z}_t$ depends on bounded subset of $\boldsymbol{z}_{\mathbf{1}:t}$
  - First-order Markov process: $p(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$
  - Second-order Markov process: $p(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{t-2}, \boldsymbol{z}_{t-1})$



- Stationary process: $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$ fixed for all t

Hidden Markov Model (HMM)

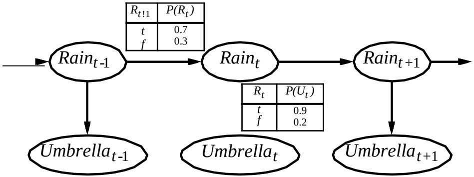
- Sensor Markov assumption: $p(x_t|z_{1:t}, x_{1:t-1}) = p(x_t|z_t)$
- Stationary process: transition model $p(z_t|z_{t-1})$ and sensor model $p(x_t|z_t)$ fixed for all t (separate $p(z_1)$)
- HMM special type of Bayesian network, z_t is a single discrete random variable:



- Joint distribution:

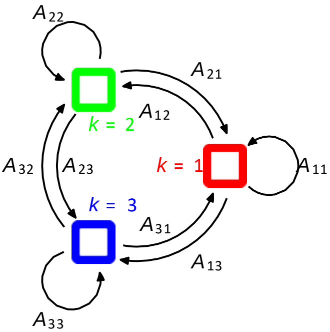
$$p(z_{1:t}, x_{1:t}) = p(z_1) \prod_{i=2:t} p(z_i|z_{i-1}) \prod_{i=1:t} p(x_i|z_i)$$

HMM Example



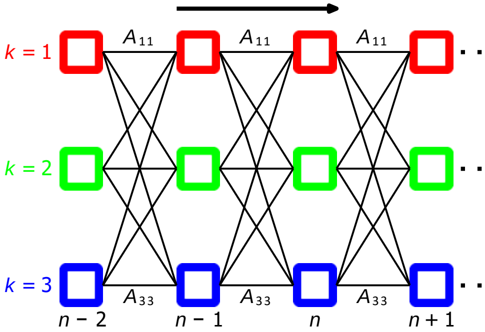
- First-order Markov assumption not true in real world
- Possible fixes:
  - Increase order of Markov process
  - Augment state, add $temp_t$, $pressure_t$

Transition Diagram



- z_n takes one of 3 values
- Using one-of-K coding scheme, $z_{nk} = 1$ if in state k at time n
- Transition matrix $\boldsymbol{A}$ where $p(z_{nk} = 1 | z_{n-1,j} = 1) = \boldsymbol{A}_{jk}$

Lattice / Trellis Representation



- The lattice or trellis representation shows possible paths through the latent state variables z_n

Outline

Hidden Markov Models

Inference for HMMs

Learning for HMMs

Inference Tasks

- Filtering: $p(z_t|x_{1:t})$
  - Estimate current unobservable state given all observations to date
- Prediction: $p(z_n|x_{1:t})$ for $n > t$
  - Similar to filtering, without evidence
- Smoothing: $p(z_n|x_{1:t})$ for $n < t$
  - Better estimate of past states
- Most likely explanation: $\arg\max_{z_{1:t}} p(z_{1:t}|x_{1:t})$
  - e.g. speech recognition, decoding noisy input sequence

Filtering

- Aim: devise a recursive state estimation algorithm:

$$p(z_{t+1}|x_{1:t+1}) = f\big(x_{t+1}, p(z_t|x_{1:t})\big)$$

$$\begin{aligned}
p(z_{t+1}|x_{1:t+1}) &= p(z_{t+1}|x_{1:t}, x_{t+1}) \\
&= \alpha p(x_{t+1}|x_{1:t}, z_{t+1}) p(z_{t+1}|x_{1:t}) && \text{(Bayes rule)} \\
&= \alpha p(x_{t+1}|z_{t+1}) p(z_{t+1}|x_{1:t}) && \text{(Markov assumption)}
\end{aligned}$$

- i.e. measurement + prediction. Prediction by summing out z_t:

$$\begin{aligned}
p(z_{t+1}|x_{1:t+1}) &= \alpha p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}, z_t|x_{1:t}) && \text{(Marginalize)} \\
&= \alpha p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t, x_{1:t}) p(z_t|x_{1:t}) && \text{(Product rule)} \\
&= \alpha p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t) p(z_t|x_{1:t}) && \text{(Markov assumption)}
\end{aligned}$$

Filtering Example

Prediction: $\sum_{R_1} p(R_2|R_1)p(R_1|U_1 = T)$

$0.7 \times 0.818 + 0.3 \times 0.182 \; (R_2 = T)$

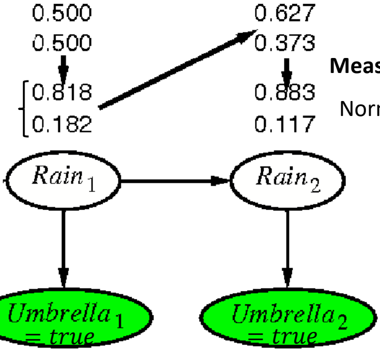
$0.3 \times 0.818 + 0.7 \times 0.182 \; (R_2 = F)$

Prior: $p(rain_1 = true) = 0.5$

Measurement: $p(R_1|U_1 = T)$

Normalize $0.5 \times 0.9 \; (R_1 = T)$
$0.5 \times 0.2 \; (R_1 = F)$

$$p(R_1|U_1) = \frac{p(U_1|R_1)p(R_1)}{P(U_1)}$$



Measurement: $p(R_2|U_2 = T)$

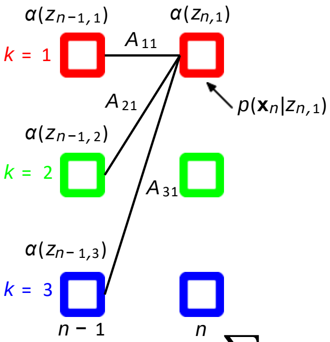
Normalize $0.627 \times 0.9 \; (R_2 = T)$
$0.117 \times 0.2 \; (R_2 = F)$

R_{t-1}	$P(R_t)$
t	0.7
f	0.3

R_t	$P(U_t)$
t	0.9
f	0.2

$$p(z_{t+1}|x_{1:t+1}) = \alpha p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)p(z_t|x_{1:t})$$

Filtering - Lattice



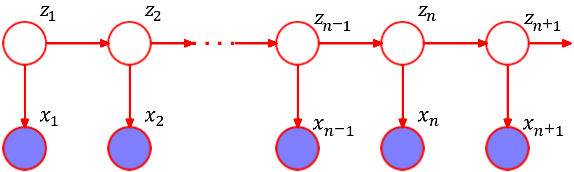
$$\underbrace{p(z_{t+1}|x_{1:t+1})}_{K\text{-dim vector}} = \alpha p(x_{t+1}|z_{t+1}) \sum_{z_t} \underbrace{p(z_{t+1}|z_t)}_{K\times K \text{ matrix}} \underbrace{p(z_t|x_{1:t})}_{K\text{-dim vector}}$$

$\Rightarrow O(K^2)$ for each time step, $O(NK^2)$ for N time steps

Forward message passing: $\alpha(z_{t+1}) = p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)\alpha(z_t)$

- $\alpha(z_t) = p(x_{1:t}, z_t)$; previous normalization constant can be dropped
- Initial condition: $\alpha(z_1) = p(x_1, z_1) = p(x_1|z_1)p(z_1)$

Smoothing



- Divide evidence $x_{1:t}$ into $x_{1:n}, x_{n+1:t}$

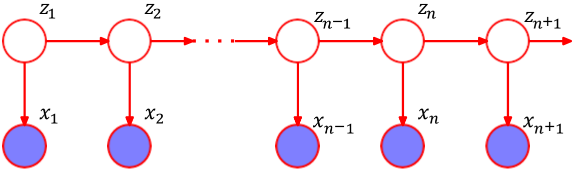
$$p(z_n|x_{1:t}) = \frac{p(x_{1:t}|z_n)p(z_n)}{p(x_{1:t})} \qquad \text{(Bayes rule)}$$

$$= \frac{p(x_{1:n}|z_n)p(x_{n+1:t}|z_n)p(z_n)}{p(x_{1:t})} \qquad \text{(Cond. indep.)}$$

$$= \frac{p(x_{1:n}, z_n)p(x_{n+1:t}|z_n)}{p(x_{1:t})} \qquad \text{(Product rule)}$$

$$= \frac{\alpha(z_n)\beta(z_n)}{p(x_{1:t})}$$

Smoothing



- Divide evidence $x_{1:t}$ into $x_{1:n}, x_{n+1:t}$, $p(z_n|x_{1:t}) = \eta\alpha(z_n)\beta(z_n)$
- Backwards message another recursion:

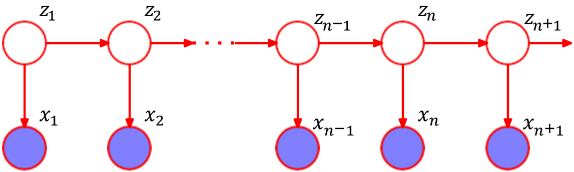
$$\underbrace{p(x_{n+1:t}|z_n)}_{\beta(z_n)} = \sum_{z_{n+1}} p(x_{n+1:t}, z_{n+1}|z_n) \qquad \text{(Marginalize)}$$

$$= \sum_{z_{n+1}} p(x_{n+1:t}|z_{n+1}, z_n)p(z_{n+1}|z_n) \qquad \text{(Product rule)}$$

$$= \sum_{z_{n+1}} p(x_{n+1:t}|z_{n+1})p(z_{n+1}|z_n) \qquad \text{(Markov assumption)}$$

$$= \sum_{z_{n+1}} p(x_{n+1}|z_{n+1})\underbrace{p(x_{n+2:t}|z_{n+1})}_{\beta(z_{n+1})}p(z_{n+1}|z_n) \qquad \text{(Cond. indep.)}$$

Smoothing



- Final condition: go back 2 slides and set $n = t$

$$p(z_t|x_{1:t}) = \frac{\alpha(z_t)\beta(z_t)}{p(x_{1:t})}$$

$$p(z_t|x_{1:t}) = \frac{p(x_{1:t}, z_t)\beta(z_t)}{p(x_{1:t})}$$

$$\Rightarrow \beta(z_t) = 1$$

Smoothing Example

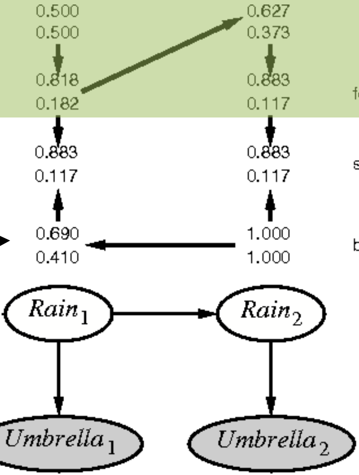
$$\alpha(z_1) = p(x_1|z_1)p(z_1)$$

From previous slide

$$p(z_1)$$

$$\frac{\alpha(z_1)}{p(x_{1:1})} = p(z_1|x_{1:1})$$

$$p(z_1|x_{1:2}) = \frac{\alpha(z_1)\beta(z_1)}{p(x_{1:2})}$$



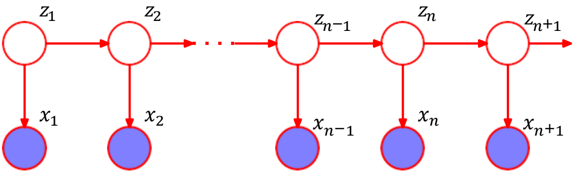
forward $\frac{\alpha(z_2)}{p(x_{1:2})} = p(z_2|x_{1:2})$

smoothed $p(z_2|x_{1:2}) = \frac{\alpha(z_2)\beta(z_2)}{p(x_{1:2})}$

backward $\beta(z_2)$

$$\beta(z_1) = \sum_{z_2} p(x_2|z_2)\beta(z_2)p(z_2|z_1)$$

Smoothing

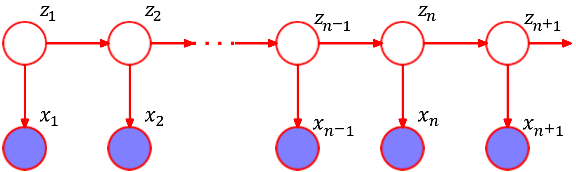


- Backwards message another recursion:

$$\underbrace{\beta(z_n)}_{K\text{-dim vector}} = \sum_{z_{n+1}} p(x_{n+1}|z_{n+1})\underbrace{\beta(z_{n+1})}_{K\text{-dim vector}}\underbrace{p(z_{n+1}|z_n)}_{K \times K \text{ matrix}}$$

$\Rightarrow O(K^2)$ for each time step, $O(NK^2)$ for N time steps

Forward-Backward Algorithm



- Filter from time 1 to N, and cache forward messages $\alpha(z_n)$
- Smooth from time N to 1, and cache backward messages $\beta(z_n)$
- Can now compute $p(z_n|x_1, x_2, \ldots, x_t)$ for all n
- Total complexity $O(NK^2)$
- a.k.a Baum-Welch algorithm

Outline

Hidden Markov Models

Inference for HMMs

Learning for HMMs

HMM Parameters

- The parameters of an HMM:
  - Transition matrix A where $p\left(z_{nk} = 1 \middle| z_{n-1,j} = 1\right) = \boldsymbol{A}_{jk}$
  - Sensor model ϕ_k parameters to each $p(x_n|z_{nk} = 1, \phi_k)$ (e.g. ϕ_k could be mean and variance of Gaussian)
  - Prior for initial state z_1, model as multinomial $p(z_{1k} = 1) = \pi_k$, parameters $\boldsymbol{\pi}$
- Call these parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{\pi}, \boldsymbol{\varphi})$
- Learning problem: given one sequence $\boldsymbol{x}$, find best $\boldsymbol{\theta}$
  - Extension to multiple sequences straight-forward (assume independent, log of product is sum)

Maximum Likelihood for HMMs

- We can use maximum likelihood to choose the best parameters:

$$\boldsymbol{\theta}_{ML} = \arg\max p(\boldsymbol{x}|\boldsymbol{\theta})$$

- Unfortunately this is hard to do: we can get $p(\boldsymbol{x}|\boldsymbol{\theta})$ by summing out from the joint distribution:

$$p(x|\theta) = \sum_{z_1}\sum_{z_2}\cdots\sum_{z_N} p(\boldsymbol{x}, z_1, z_2, \ldots, z_N|\boldsymbol{\theta})$$

$$\equiv \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$$

  - But this sum has K^N terms in it
  - No simple closed-form solution
- Instead, use expectation-maximization (EM)

EM for HMMs

- Start with initial guess for parameters $\boldsymbol{\theta}^{old} = (\boldsymbol{A}, \boldsymbol{\pi}, \boldsymbol{\phi})$
- **E-step**: Calculate posterior on latent variables $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{old})$

Forward-backward algorithm

$\mathbb{E}_{z \sim p(z|x, \theta^{old})}[\ln p(x, z|\theta)]$

- **M-step**: Maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{old}) \ln p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$ wrt $\boldsymbol{\theta}$
- Let's look at the M-step, and see how the HMM structure helps us

HMM M-step

- **M-step**: Maximize $Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}\right) = \sum_{z} p\left(\boldsymbol{z} \middle| \boldsymbol{x}, \boldsymbol{\theta}^{old}\right) \ln p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$ wrt $\boldsymbol{\theta}$:
- The complete data log-likelihood factors nicely:

$$\ln p(x, z|\theta) = \ln \left\{ p(z_1|\pi) \prod_{i=2}^{N} p(z_i|z_{i-1}, \boldsymbol{A}) \prod_{i=1}^{N} p(x_i|z_i, \boldsymbol{\phi}) \right\}$$

$$= \ln p(z_1|\boldsymbol{\pi}) + \sum_{i=2}^{N} \ln p(z_i|z_{i-1}, \boldsymbol{A}) + \sum_{i=1}^{N} \ln p(x_i|z_i, \phi)$$

- To maximize Q we now have 3 separate problems, one for each parameter
  - Let's consider each in turn

Prior $\boldsymbol{\pi}$

- Maximize Q wrt prior on initial state $\boldsymbol{\pi}$:

$$
\begin{aligned}
Q(\pi, \theta^{old}) &= \sum_{z} p(z|x, \theta^{old}) \ln p(z_1|\pi) \\
&= \sum_{z} p(z|x, \theta^{old}) \ln \prod_{k=1}^{K} \pi_k^{z_{1k}} \\
&= \sum_{z} p(z|x, \theta^{old}) \sum_{k=1}^{K} z_{1k} \ln \pi_k \\
&= \sum_{k=1}^{K} \ln \pi_k \sum_{z} p(z|x, \theta^{old}) z_{1k} \\
&= \sum_{k=1}^{K} p(z_{1k} = 1|\boldsymbol{x}, \boldsymbol{\theta}^{old}) \ln \pi_k
\end{aligned}
$$

- i.e. smoothed value for z_1 being in state k

$$Q(\pi, \theta^{old}) = \sum_{k=1}^{K} p(z_{1k} = 1 | \boldsymbol{x}, \boldsymbol{\theta}^{old}) \ln \pi_k$$

- Can solve for best $\boldsymbol{\pi}$
- Use Lagrange multiplier to enforce constraint $\sum_k \pi_k = 1$

$$\pi_k = \frac{p(z_{1k} = 1 | \boldsymbol{x}, \boldsymbol{\theta}^{old})}{\sum_{j=1}^{K} p(z_{1j} = 1 | \boldsymbol{x}, \boldsymbol{\theta}^{old})}$$

- Intuitively sensible result: new π_k is smoothed probability of being in state k at time 1 using old parameters
- E-step needs to calculate smoothed $p(z_{1k} = 1 | \boldsymbol{x}, \boldsymbol{\theta}^{old})$;this is fast $O(NK^2)$

Transition Matrix $\boldsymbol{A}$

- Maximize Q wrt transition matrix $\boldsymbol{A}$:

$$
\begin{aligned}
Q(\boldsymbol{A}, \boldsymbol{\theta}^{old}) &= \sum_{z} p(z|x, \theta^{old}) \sum_{i=2}^{N} \ln p(z_i|z_{i-1}, \boldsymbol{A}) \\
&= \sum_{z} p(z|x, \theta^{old}) \sum_{i=2}^{N} \ln \prod_{k=1}^{N} \prod_{j=1}^{K} \boldsymbol{A}_{jk}^{z_{i-1,j} z_{i,k}} \\
&= \sum_{z} p(z|x, \theta^{old}) \sum_{i=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} z_{i-1,j} z_{i,k} \ln \boldsymbol{A}_{jk} \\
&= \sum_{k=1}^{K} \sum_{j=1}^{K} \ln \boldsymbol{A}_{jk} \sum_{i=2}^{N} \sum_{z} p(z|x, \theta^{old}) z_{i-1,j} z_{i,k} \\
&= \sum_{k=1}^{K} \sum_{j=1}^{K} \ln \boldsymbol{A}_{jk} \sum_{i=2}^{N} p(z_{i-1} = j, z_i = k|x, \theta^{old})
\end{aligned}
$$

- E-step needs to calculate $p(z_{i-1} = j, z_i = k|\boldsymbol{x}, \boldsymbol{\theta}^{old})$; can be done quickly using forward and backward messages

$$Q(\boldsymbol{A}, \boldsymbol{\theta}^{old}) = \sum_{k=1}^{K} \sum_{j=1}^{K} \ln \boldsymbol{A}_{jk} \sum_{i=2}^{N} p(z_{i-1} = j, z_i = k | x, \theta^{old})$$

- Can solve for best $\boldsymbol{A}$
- Again use Lagrange multipliers to enforce constraint $\sum_k A_{jk} = 1$

$$\boldsymbol{A}_{jk} = \frac{\sum_{n=2}^{N} p(z_{n-1} = j, z_n = k | \boldsymbol{x}, \boldsymbol{\theta}^{old})}{\sum_{l=1}^{K} \sum_{n=2}^{N} p(z_{n-1} = j, z_n = k | \boldsymbol{x}, \boldsymbol{\theta}^{old})}$$

- Again sensible result: $\boldsymbol{A}_{jk}$ set to expected number of times we transition from state j to k using the smoothed results from old parameters

Sensor Model

- Similar derivation for sensor model parameters $\boldsymbol{\phi}$
- Again end up with weighted parameter estimated based on expected values of states given smoothed estimates

HMM EM Summary

- Start with initial guess for parameters $\boldsymbol{\theta}^{old} = (\boldsymbol{A}, \pi, \phi)$
- Run forward-backward algorithm to get all messages $\alpha(z_n), \beta(z_n)$ (E-step)
    - $O(NK^2)$ time complexity
    - Can use these to compute any smoothed posterior $p\left(z_{nk} = 1 \middle| \boldsymbol{x}, \boldsymbol{\theta}^{old}\right)$
    - Also can compute any $p\left(z_{nk} = 1, z_{n,k} = 1 \middle| \boldsymbol{x}, \boldsymbol{\theta}^{old}\right)$
    - Using these, update values for parameters (M-step)
    - $\boldsymbol{\pi}_k$ is smoothed probability of being in in state k at time 1
    - $\boldsymbol{A}_{jk}$ is smoothed probability of transitioning from state j to k averaged over all time steps
    - $\boldsymbol{\phi}$ is weighted sensor parameters using smoothed probabilities (e.g. similar to mixture of Gaussians)
- Repeat until convergence

Inference Tasks

- Filtering: $p(z_t|x_{1:t})$
  - Estimate current unobservable state given all observations to date
- Prediction: $p(z_n|x_{1:t})$ for $n > t$
  - Similar to filtering, without evidence
- Smoothing: $p(z_n|x_{1:t})$ for $n < t$
  - Better estimate of past states
- Most likely explanation: $\arg\max_{z_{1:t}} p(z_{1:t}|x_{1:t})$
  - e.g. speech recognition, decoding noisy input sequence

Sequence of Most Likely States

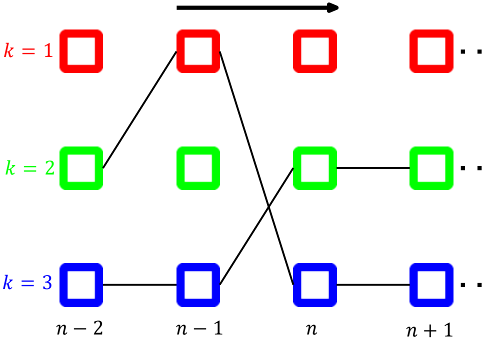
- Most likely sequence is not same as sequence of most likely states:

$$\arg\max_{z_{1:N}} p(z_{1:N}|x_{1:N})$$

versus

$$\left(\arg\max_{z_1} p(z_1|x_{1:N}), \ldots, \arg\max_{z_N} p(z_N|x_{1:N})\right)$$

Paths Through HMM

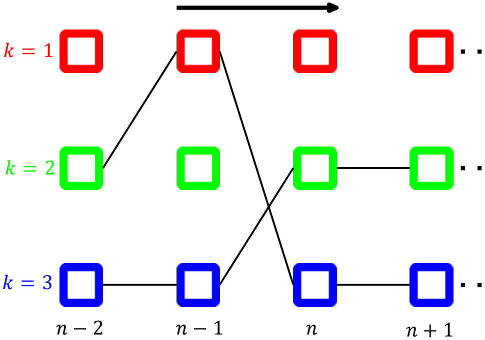


- There are K^N paths to consider through the HMM for computing

$$\arg\max_{z_{1:N}} p(z_{1:N}|x_{1:N})$$

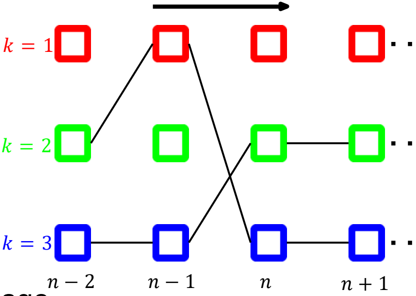
- Need a faster method

Viterbi Algorithm



- Insight: for any value k for z_n, the best path $(z_1, z_2, \ldots, z_n = k)$ ending in $z_n = k$ consists of the best path $(z_1, z_2, \ldots, z_{n-1} = j)$ for some j, plus one more step
  - Don't need to consider exponentially many paths, just K at each time step
  - Dynamic programming algorithm – Viterbi algorithm

Viterbi Algorithm - Math



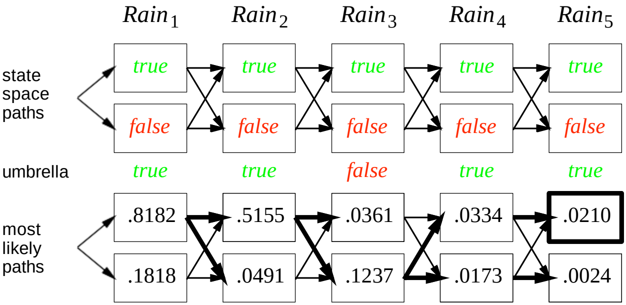
- Define message

$$w(n,k) = \max_{z_1,\dots,z_{n-1}} p(x_1,\dots,x_n,z_1,\dots,z_n = k)$$

- From factorization of joint distribution:

$$\begin{aligned} w(n,k) &= \max_{z_1,\dots,z_{n-1}} p(x_1,\dots,x_{n-1},z_1,\dots,z_{n-1})p(x_n|z_n = k)p(z_n = k|z_{n-1}) \\ &= \max_{z_{n-1}} \max_{z_1,\dots,z_{n-2}} p(x_{1:n-1},z_{1:n-1})p(x_n|z_n = k)p(z_n = k|z_{n-1}) \\ &= \max_j w(n-1,j)p(x_n|z_n = k)p(z_n = k|z_{n-1} = j) \end{aligned}$$

Viterbi Algorithm - Example



R_{t-1}	$P(R_t)$
t	0.7
f	0.3

R_t	$P(U_t)$
t	0.9
f	0.2

$$p(rain_1 = true) = 0.5$$

$$w(n,k) = \max_{z_1,\ldots,z_{n-1}} p(x_1,\ldots,x_n,z_1,\ldots,z_n = k)$$

$$= \max_j w(n-1,j)p(x_n|z_n = k)p(z_n = k|z_{n-1} = j)$$

Viterbi Algorithm - Complexity

- Each step of the algorithm takes $O(K^2)$ work
- With N time steps, $O(NK^2)$ complexity to find most likely sequence
- Much better than naive algorithm evaluating all K^N possible paths

Conclusion

- Readings: Ch. 13.2, 13.2.1, 13.2.2, 13.2.5
- HMM - Probabilistic model of temporal data
  - Discrete hidden (unobserved, latent) state variable at each time
  - Observation (can be discrete / continuous) at each time
  - Conditional independence assumptions (Markov)
  - Assumptions on distributions (stationary)
- Inference
  - Filtering
  - Smoothing
  - Most likely sequence (next)
- Maximum likelihood learning
  - EM – efficient computation $O(NK^2)$ time using forward-backward smoothing
- Most likely sequence in HMM
  - Viterbi algorithm – $O(NK^2)$ time, dynamic programming algorithm