# Sequential Data - Part 2

Greg Mori - CMPT 419/726

Bishop PRML Ch. 13
Russell and Norvig, AIMA



## Outline





## Inference Tasks

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



## Sequence of Most Likely States

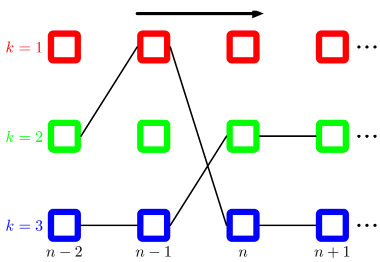
- Most likely sequence is not same as sequence of most likely states:

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

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



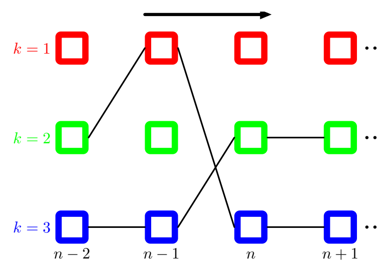

- There are $K^N$ paths to consider through the HMM for computing

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

- Need a faster method



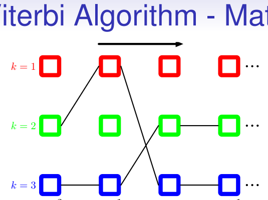
## Viterbi Algorithm




- 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

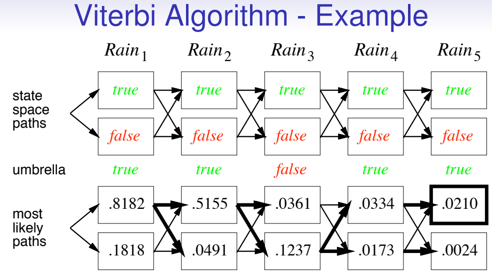



- Define message $w(n,k) = \max_{z_1,\ldots,z_{n-1}} p(x_1, \ldots, x_n, z_1, \ldots, z_n = k)$
- From factorization of joint distribution:

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



## 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 |

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

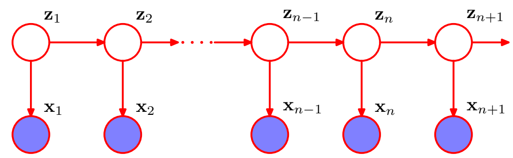
$$\begin{aligned} w(n,k) &= \max_{z_1,\ldots,z_{n-1}} p(x_1,\ldots,x_n,z_1,\ldots,z_n=k) \\ &= \max_j w(n-1,j)p(x_n|z_n=k)p(z_n=k|z_{n-1}=j) \end{aligned}$$

## 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



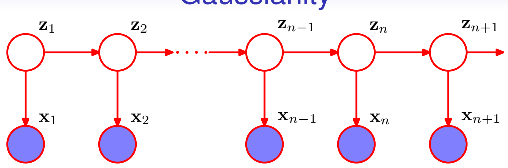
## Continuous State Variables




- In HMMs, the state variable $z_t$ is assumed discrete
- In many applications, $z_t$ is continuous
  - Object tracking
  - Stock price, gross domestic product (GDP)
  - Amount of rain
- Can either discretize
  - Large state space
  - Discretization errors
- Or use method that directly handles continuous variables



## Gaussianity




- As in the HMM, we require model parameters – transition model and sensor model
- Unlike HMM, each of these is a conditional probability density given a continuous-valued $z_t$
- One common assumption is to let both be linear Gaussians:

$$
\begin{aligned}
p(z_t|z_{t-1}) &= \mathcal{N}(z_t; Az_{t-1}, \Sigma_z) \\
p(x_t|z_t) &= \mathcal{N}(x_t; Cz_t, \Sigma_x)
\end{aligned}
$$



## Continuous State Variables - Filtering

- Recall the filtering problem $p(z_t|x_{1:t})$ distribution on current state given all observations to date
- As in discrete case, can formulate a recursive computation:

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

- Now we have an integral instead of a sum
- Can we do this integral exactly?
  - If we use linear Gaussians, yes: Kalman filter
  - In general, no: can use particle filter

# Particle Filter

- The particle filter is a particular sampling-importance-resampling algorithm for approximating $p(z_t|x_{1:t})$



# Recall: SIR - Algorithm

- Sampling-importance-resampling algorithm has two stages
- Sampling:
  - Draw samples $z^{(1)}, \ldots, z^{(L)}$ from proposal distribution $q(z)$
- Importance resampling:
  - Put weights on samples

$$w_l = \frac{\tilde{p}(z^{(l)})/q(z^{(l)})}{\sum_m \tilde{p}(z^{(m)})/q(z^{(m)})}$$

  - Draw samples $\hat{z}^{(\ell)}$ from the discrete set $z^{(1)}, \ldots, z^{(L)}$ according to weights $w_l$
- Approximate $p(\cdot)$ by:

$$p(z) \approx \frac{1}{L}\sum_{\ell=1}^{L} \delta(z - \hat{z}^{(\ell)})$$

$$p(z) \approx \sum_{\ell=1}^{L} w_\ell \delta(z - z^{(\ell)})$$



# Particle Filter

- The particle filter is a particular sampling-importance-resampling algorithm for approximating $p(z_t|x_{1:t})$
- What should be the proposal distribution $q(z_t)$?
  - Trick: use prediction given previous observations

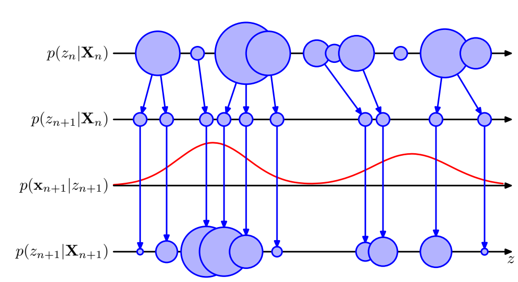
$$p(z_t|x_{1:t-1}) \approx \sum_{\ell=1}^{L} w_{t-1}^{\ell} p(z_t|z_{t-1}^{(\ell)})$$

  - With this proposal distribution, the weights for importance resampling are:

$$w_t^{\ell} = \frac{\tilde{p}(z^{(\ell)})}{q(z^{(\ell)})} = \frac{p(z_t^{(\ell)}|x_{1:t})}{p(z_t^{(\ell)}|x_{1:t-1})} = p(x_t|z_t^{(\ell)})$$



# Particle Filter Illustration



## Particle Filter Example

Okuma et al. ECCV 2004



## Conclusion

- Readings: Ch. 13.2.5, 13.3
- Most likely sequence in HMM
  - Viterbi algorithm – $O(NK^2)$ time, dynamic programming algorithm
- Continuous state spaces
  - Linear Gaussians – closed-form filtering (and smoothing) using Kalman filter
  - General case – no closed-form solution, can use particle filter, a sampling method