Graphical Models & HMMs

Henrik I. Christensen

Robotics & Intelligent Machines @ GT
Georgia Institute of Technology,
Atlanta, GA 30332-0280
hic@cc.gatech.edu

Outline

Introduction

- Basically we can describe Bayesian inference through repeated use of the sum and product rules
- Using a graphical / diagrammatical representation is often useful
  1. A way to visualize structure and consider relations
  2. Provides insights into a model and possible independence
  3. Allow us to leverage of the many graphical algorithms available
- Will consider both directed and undirected models
- This is a very rich areas with numerous good references

Outline

Bayesian Networks

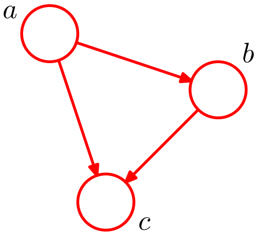
- Consider the joint probability $p(a, b, c)$
- Using product rule we can rewrite it as

$$p(a, b, c) = p(c|a, b)p(a, b)$$

- Which again can be changed to

$$p(a, b, c) = p(c|a, b)p(b|a)p(a)$$

- We can illustrate this as



Bayesian Networks

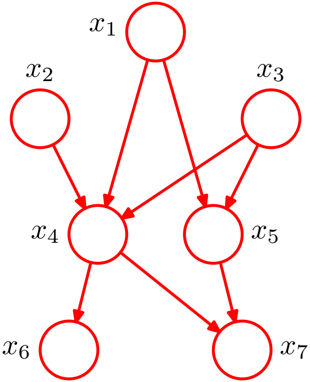
- Nodes represent variables
- Arcs/links represent conditional dependence
- We can use the decomposition for any joint distribution.
- The direct / brute-force application generates fully connected graphs
- We can represent much more general relations

Example with "sparse" connections

- We can represent relations such as

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$

- Which is shown below



The general case

- We can think of this as coding the *factors*

$$p(x_k|pa_k)$$

where pa_k is the set of parents to a variable x_k

- The inference is then

$$p(\mathbf{x}) = \prod_k p(x_k|pa_k)$$

we will refer to this as factorization

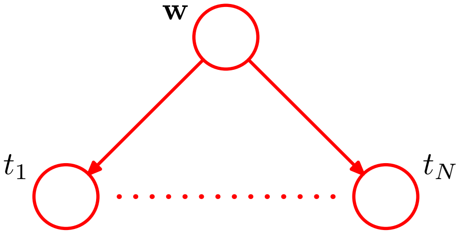
- There can be no directed cycles in the graph
- The general form is termed a *directed acyclic graph* - DAG

Basic example

- We have seen the polynomial regression before

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_n p(t_n|\mathbf{w})$$

- Which can be visualized as

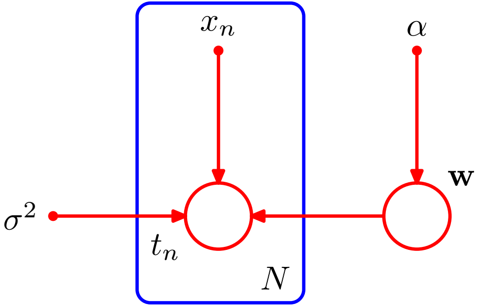


Bayesian Regression

- We can make the parameters and variables explicit

$$p(\mathbf{t}, \mathbf{w}|\mathbf{x}, \alpha, \sigma^2) = p(\mathbf{w}|\alpha) \prod_n p(t_n|\mathbf{w}, x_n, \sigma^2)$$

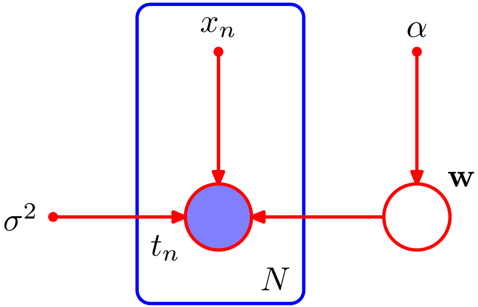
as shown here



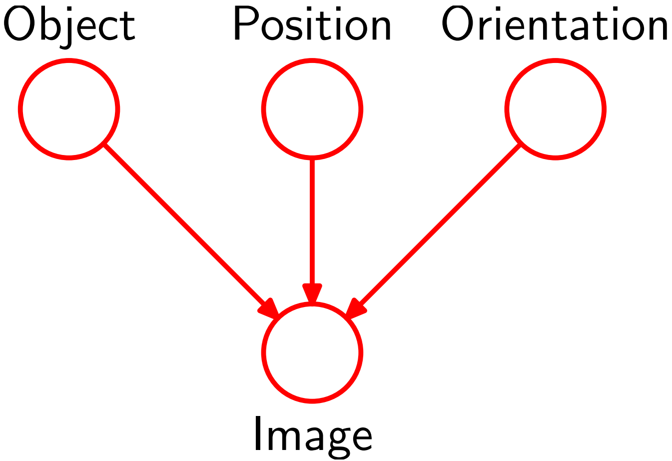
Bayesian Regression - Learning

- When entering data we can condition inference on it

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w}) \prod_n p(t_n|\mathbf{w})$$



Generative Models - Example Image Synthesis



Discrete Variables - 1

- General joint distribution has $K^2 - 1$ parameters (for K possible outcomes)

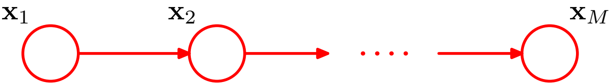
$$p(\mathbf{x}_1, \mathbf{x}_2 | \mu) = \prod_{i=1}^{K} \prod_{j=1}^{K} \mu_{ij}^{x_{1i} x_{2j}}$$

- Independent joint distributions have $2(K-1)$ parameters

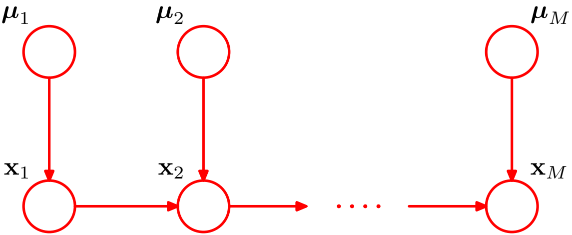
$$p(\mathbf{x}_1, \mathbf{x}_2 | \mu) = \prod_{i=1}^{K} \mu_{1i}^{x_{1i}} \prod_{j=1}^{K} \mu_{2j}^{x_{2j}}$$

Discrete Variables - 2

- General joint distribution over M variables will have $K^M - 1$ parameters
- A Markov chain with M nodes will have $K - 1 + (M-1)K(K-1)$ parameters



Discrete Variables - Bayesian Parms

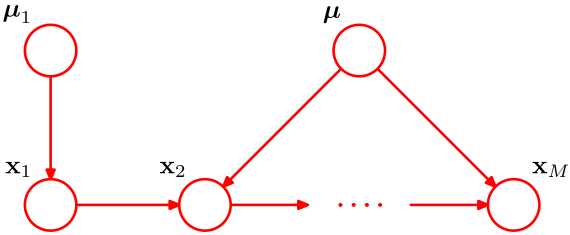


- The parameters can be modelled explicitly

$$p(\{\mathbf{x}_m, \mu_m\}) = p(\mathbf{x}_1|\mu_1)p(\mu_1)\prod_{m=2}^{M} p(x_m|x_{m-1}, \mu_m)p(\mu_m)$$

- It is assumed that $p(\mu_m)$ is a Dirachlet

Discrete Variables - Bayesian Parms (2)



- For shared paraemeters the situation is simpler

$$p(\{\mathbf{x}_m\}, \mu_1, \mu) = p(\mathbf{x}_1|\mu_1)p(\mu_1) \prod_{m=2}^{M} p(x_m|x_{m-1}, \mu)p(\mu)$$

Extension to Linear Gaussian Models

- The model can be extended to have each node as a Gaussian process/variable that is a linear function of its parents

$$p(x_i|pa_i) = N\left(x_i \middle| \sum_{j \in pa_i} w_{ij}x_j + b_i, v_i\right)$$

Outline

Conditional Independence

- Considerations of independence is important as part of the analysis and setup of a system
- As an example a is independent of b given c

$$p(a|b,c) = p(a|c)$$

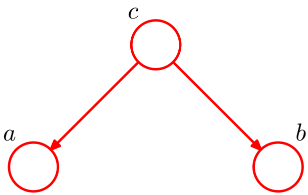
- Or equivalently

$$\begin{aligned} p(a,b|c) &= p(a|b,c)p(b|c) \\ &= p(a|c)p(b|c) \end{aligned}$$

- Frequent notation in statistics

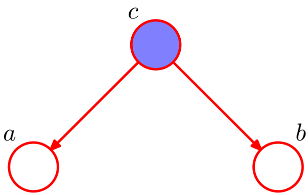
$$a \perp\!\!\!\perp b | c$$

Conditional Independence - Case 1



$$
\begin{aligned}
p(a,b,c) &= p(a|c)p(b|c)p(c) \\
p(a,b) &= \sum_c p(a|c)p(b|c)p(c) \\
a \not\perp\!\!\!\perp b &\mid \emptyset
\end{aligned}
$$

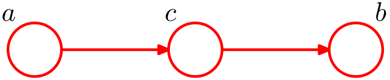
Conditional Independence - Case 1



$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = p(a|c)p(b|c)$$

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence - Case 2

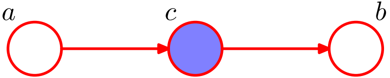


$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

$$p(a, b) = p(a)\sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp b | \emptyset$$

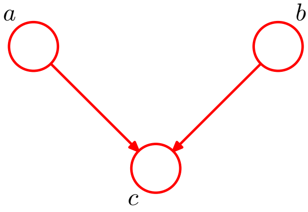
Conditional Independence - Case 2



$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

$$a \perp\!\!\!\perp b | c$$

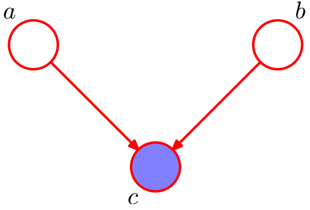
Conditional Independence - Case 3



$$
\begin{aligned}
p(a, b, c) &= p(a)p(b)p(c|a, b) \\
p(a, b) &= p(a)p(b) \\
a \perp\!\!\!\perp b &\mid \emptyset
\end{aligned}
$$

This is the opposite of Case 1 - when c unobserved

Conditional Independence - Case 3



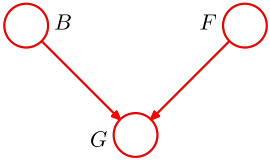
$$p(a, b|c) = \frac{p(a, b, c)}{p(c)}$$

$$= \frac{p(a)p(b)p(c|a, b)}{p(c)}$$

$$a \not\perp\!\!\!\perp b \quad | \quad c$$

This is the opposite of Case 1 - when c observed

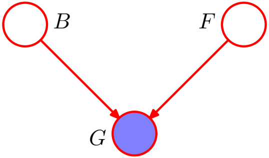
Diagnostics - Out of fuel?



$$
\begin{aligned}
p(G=1|B=1,F=1) &= 0.8\\
p(G=1|B=1,F=0) &= 0.2\\
p(G=1|B=0,F=1) &= 0.2\\
p(G=1|B=0,F=0) &= 0.1\\
p(B=1) &= 0.9\\
p(F=1) &= 0.9\\
\Rightarrow p(F=0) &= 0.1
\end{aligned}
$$

B = Battery
F = Fuel Tank
G = Fuel Gauge

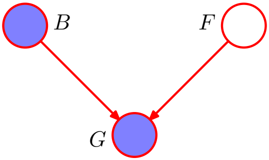
Diagnostics - Out of fuel?



$$p(F=0|G=0) = \frac{p(G=0|F=0)p(F=0)}{p(G=0)} \approx 0.257$$

Observing G=0 increased the probability of an empty tank

Diagnostics - Out of fuel?



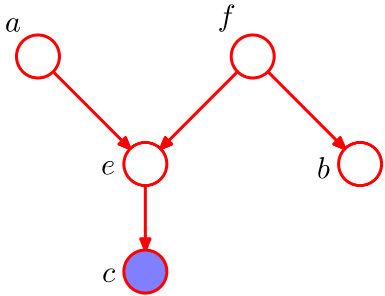
$$p(F=0|G=0,B=0) = \frac{p(G=0|B=0,F=0)p(F=0)}{\sum p(G=0|B=0,F)p(F)} \approx 0.111$$

Observing B=0 implies less likely empty tank

D-separation

- Consider non-intersecting subsets A, B, and C in a directed graph
- A path between subsets A and B is considered *blocked* if it contains a node such that:
  1. the arcs are head-to-tail or tail-to-tail and in the set C
  2. the arcs meet head-to-head and neither the node or its descendents are in the set C.
- If all paths between A and B are blocked then A is d-separated from B by C.
- If there is d-separation then all the variables in the graph satisfies $A \perp\!\!\!\perp B | C$

D-Separation - Discussion



Outline

Inference in graphical models

- Consider inference of $p(x, y)$ we can formulate this as

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x)$$

- We can further marginalize

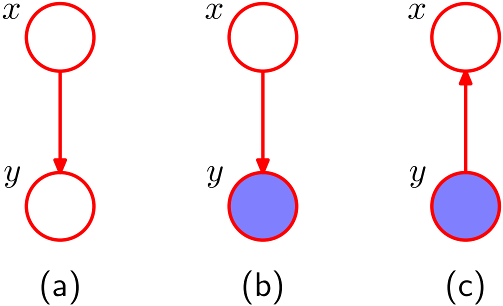
$$p(y) = \sum_{x'} p(y|x')p(x')$$

- Using Bayes Rule we can reverse the inference

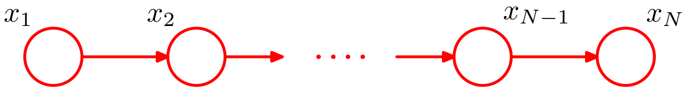
$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

- Helpful as mechanisms for inference

Inference in graphical models



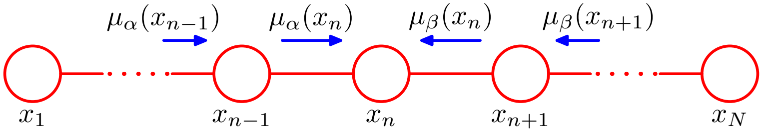
Inference on a chain



$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

$$p(x_n) = \sum_{x_1}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\sum_{x_N} p(\mathbf{x})$$

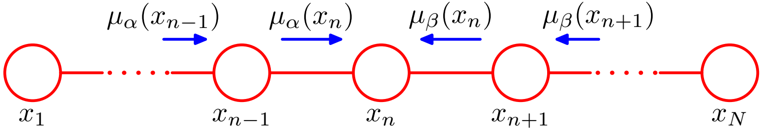
Inference on a chain



$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right]}_{\mu_a(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right]\right]}_{\mu_b(x_n)}$$

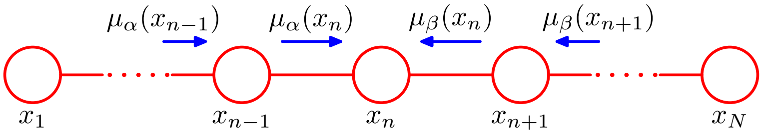
Inference on a chain



$$\mu_a(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right]$$
$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_a(x_{n-1})$$

$$\mu_b(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right]$$
$$= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \mu_b(x_{n+1}$$

Inference on a chain



$\mu_a(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \quad \mu_b(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_n)$

$$Z = \sum_{x_n} \mu_a(x_n)\mu_b(x_n)$$

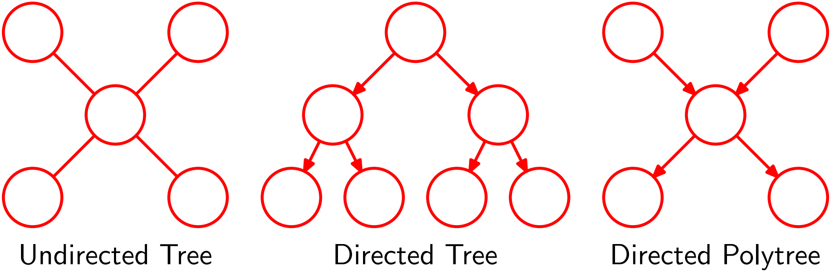
Inference on a chain

- To compute local marginals
  - Compute and store forward messages $\mu_a(x_n)$
  - Compute and store backward messages $\mu_b(x_n)$
  - Compute Z at all nodes
  - Compute

$$p(x_n) = \frac{1}{Z}\mu_a(x_n)\mu_b(x_n)$$

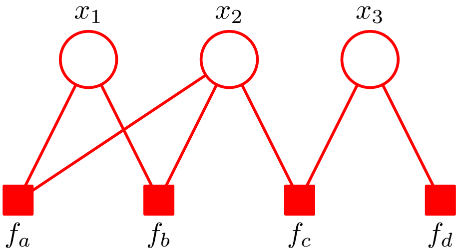
  for all variables

Inference in trees



Outline

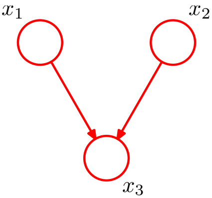
Factor Graphs



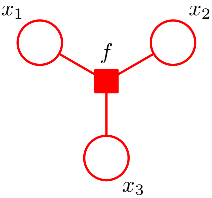
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

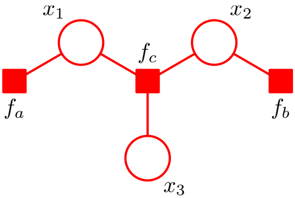
Factor Graphs from Directed Graphs



$p(x) = p(x_1)p(x_2)$
$p(x_3|x_1, x_2)$

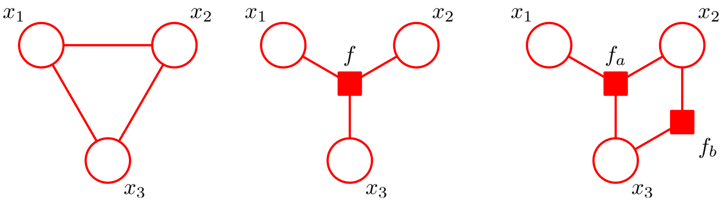


$f(x_1, x_2, x_3) =$
$p(x_1)p(x_2)p(x_3|x_1, x_2)$



$f_a(x_1) = p(x_1)$
$f_b(x_2) = p(x_2)$
$f_c(x_1, x_2, x_3) = p(x_3|x_2, x_1)$

Factor Graphs from Undirected Graphs



$\psi(x_1, x_2, x_3)$

$f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$

$f_a(x_1, x_2, x_3) f_b(x_2, x_3) = \psi(x_1, x_2, x_3)$

Outline

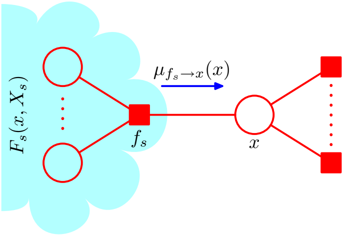
The Sum-Product Algorithm

Objective
- exact, efficient algorithm for computing marginals
- make allow multiple marginals to be computed efficiently

Key Idea
- The distributive Law

$$ab + ac = a(b + c)$$

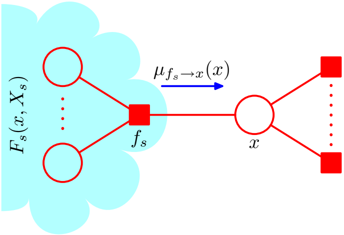
The Sum-Product Algorithm



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

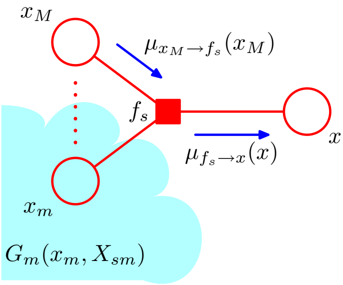
The Sum-Product Algorithm



$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

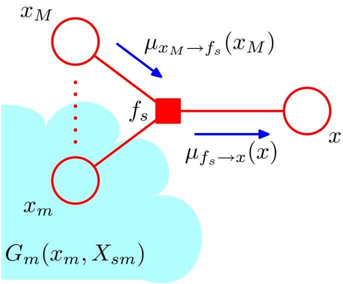
$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

The Sum-Product Algorithm



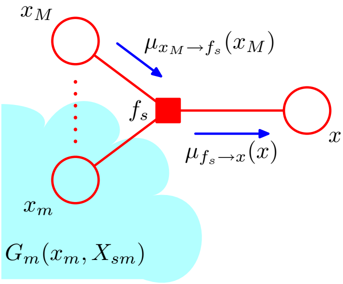
$$F_s(x, X_s) = f_s(x, x_1, ..., x_M) G_1(x_1, X_{s1}) \ldots G_M(x_M, X_{sM})$$

The Sum-Product Algorithm



$$\begin{aligned}\mu_{f_s \to x}(x) &= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in Ne(f_s) \setminus x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right] \\ &= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in Ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)\end{aligned}$$

The Sum-Product Algorithm

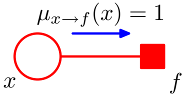


$$\mu_{x_m \to f_s}(x_m) \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in Ne(x_m) \setminus f_s} F_l(x_m, X_{lm})$$
$$= \prod_{l \in Ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

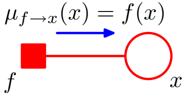
The Sum-Product Algorithm

Initialization

- For variable nodes



- For factor nodes



The Sum-Product Algorithm

- To compute local marginals
  - Pick an arbitrary node as root
  - Compute and propagate msgs from leafs to root (store msgs)
  - Compute and propagate msgs from root to leaf nodes (store msgs)
  - Compute products of received msgs and normalize as required
- Propagate up the tree and down again to compute all marginals (as needed)

Outline

Introduction

- There are many examples where we are processing sequential data
- Everyday examples include
  - Processing of stock data
  - Speech analysis
  - Traffic patterns
  - Gesture analysis
  - Manufacturing flow
- We cannot assume IID as a basis for analysis
- The Hidden Markov Model is frequently used

Apple Stock Quote Example

Outline

Markov model

- We have already talked about the Markov model.
- Estimation of joint probability as a sequential challenge

$$p(x_1, \ldots, x_N) = \prod_{n=1}^{N} p(x_n | x_{n-1}, \ldots, x_1)$$

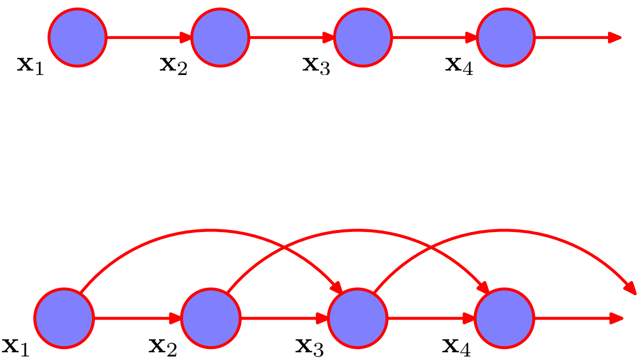
- An Nth order model is dependent on the N last terms
- A 1st order model is then

$$P(x_1, \ldots, x_N) = \prod_{n=1}^{N} p(x_n | x_{n-1})$$

- I.e.:

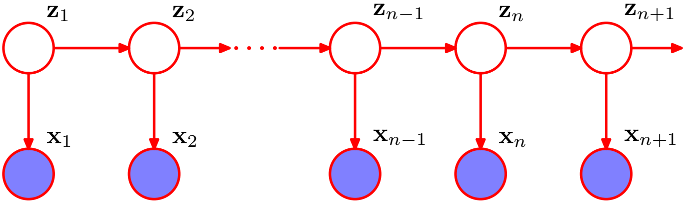
$$p(x_n | x_{n-1}, \ldots, x_1) = p(x_n | x_{n-1})$$

Markov chains



Outline

Hidden Markov Model



Modelling of HMM's

- We can model the transition probabilities as a table

$$A_{jk} = p(z_{nk} = 1 | z_{n-1,j} = 1)$$

- The conditionals are then (with a 1-out-of-K coding)

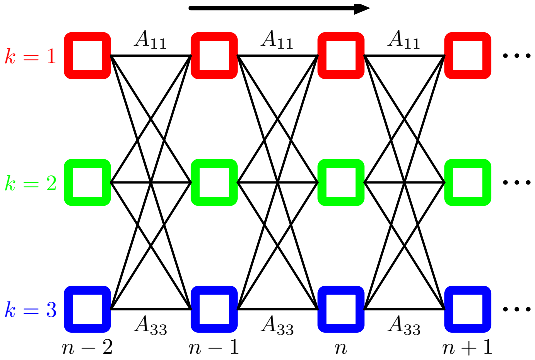
$$p(z_n | z_{n-1}, A) = \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j} z_{nk}}$$

- The per element probability is expressed by $\pi_k = p(z_{1k} = 1)$

$$p(z_1 | \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$$

with $\sum_k \pi_k = 1$

Illustration of HMM



Outline

Maximum likelihood for the HMM

- If we observe a set of data $X = \{x_1, \ldots, x_N\}$ we can estimate the parameters using ML

$$p(X|\theta) = \prod_{Z} p(X, Z|\theta)$$

- I.e. summation of paths through lattice
- We can use EM as a strategy to find a solution
- E-Step: Estimation of $p(Z|X, \theta^{old})$
- M-Step: Maximize over θ to optimize

ML solution to HMM

- Define

$$
\begin{aligned}
Q(\theta, \theta^{old}) &= \sum_Z p(Z|X, \theta^{old}) \ln p(X, Z|\theta) \\
\gamma(z_n) &= p(z_n|X, \theta^{old}) \\
\gamma(z_{nk}) &= E[z_{nk}] = \sum_z \gamma(z) z_{nk} \\
\xi(z_{n-1}, z_n) &= p(z_{n-1}, z_n|X, \theta^{old}) \\
\xi(z_{n-1,j}, z_{nk}) &= \sum_z \gamma(z) z_{n-1,j} z_{nk}
\end{aligned}
$$

ML solution to HMM

- The quantities can be computed

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}$$

- Assume $p(x|\phi_k) = N(x|\mu_k, \Sigma_k)$ so that

$$\mu_k = \frac{\sum_n \gamma(z_{nk}) x_n}{\sum_n \gamma(z_{nk})}$$

$$\Sigma_k = \frac{\sum_n \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_n \gamma(z_{nk})}$$

- How do we efficiently compute $\gamma(z_{nk})$?

Outline

Forward-Backward / Baum-Welch

- How can we efficiently compute $\gamma()$ and $\xi(.,.)$?
- Remember the HMM is a tree model
- Using message passing we can compute the model efficiently (remember earlier discussion?)
- We have two parts to the message passing forward and backward for any component
- We have

$$\gamma(z_n) = p(z_n|X) = \frac{P(X|z_n)p(z_n)}{p(X)}$$

- From earlier we have

$$\gamma(z_n) = \frac{\alpha(z_n)\beta(z_n)}{p(X)}$$

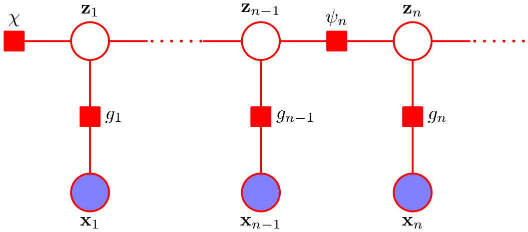
Forward-Backward

- We can then compute

$$
\begin{aligned}
\alpha(z_n) &= p(x_n|z_n) \sum_{z_{n-1}} \alpha(z_{n-1}) p(z_n|z_{n-1}) \\
\beta(z_n) &= \sum_{z_{n+1}} \beta(z_{n+1}) p(x_{n+1}|z_{n+1}) p(z_{n+1}|z_n) \\
p(X) &= \sum_{z_n} \alpha(z_n)\beta(z_n) \\
\xi(z_{n-1}, z_n) &= \frac{\alpha(z_{n-1}) p(x_n|z_n) p(z_n|z_{n-1}) \beta(z_n)}{p(X)}
\end{aligned}
$$

Sum-product algorithms for the HMM

- Given the HMM is a tree structure
- Use of sum-product rule to compute marginals
- We can derive a simple factor graph for the tree



Sum-product algorithms for the HMM

- We can then compute the factors

$$\begin{aligned} h(z_1) &= p(z_1)p(x_1|z_1) \\ f_n(z_{n-1}, z_n) &= p(z_n|z_{n-1})p(x_n|z_n) \end{aligned}$$

- The update factors $\mu_{f_n \rightarrow z_n}(z_n)$ can be used to derive message passing with $\alpha(.)$ and $\beta(.)$

Outline

Viterbi Algorithm

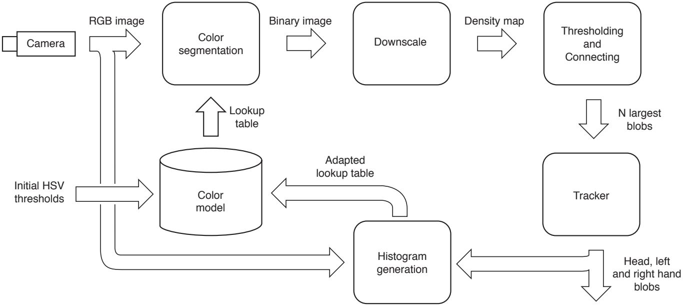
- Using the message passing framework it is possible to determine the most likely solution (ie best recognition)
- Intuitively
- Keep only track of the most likely / probably path through the graph
- At any time there are only K possible paths to maintain
- Basically a greedy evaluation of the best solution

Outline

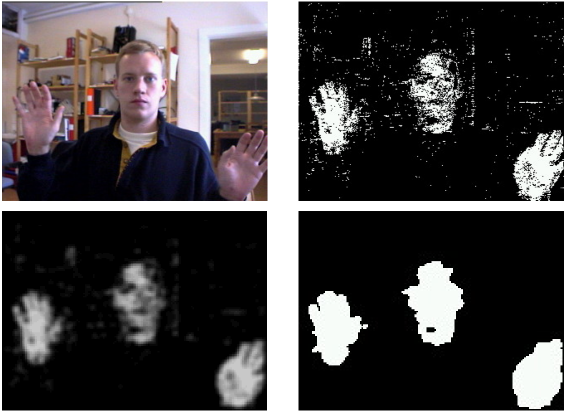
Small example of gesture tracking

- Tracking of hands using an HMM to interpret track
- Pre-process images to generate tracks
  - Color segmentation
  - Track regions using Kalman Filter
  - Interpret tracks using HMM

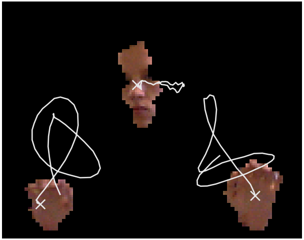
Pre-process architecture

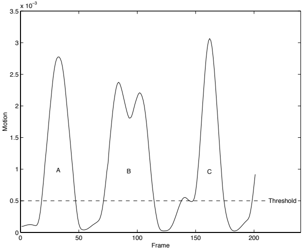


Basic idea

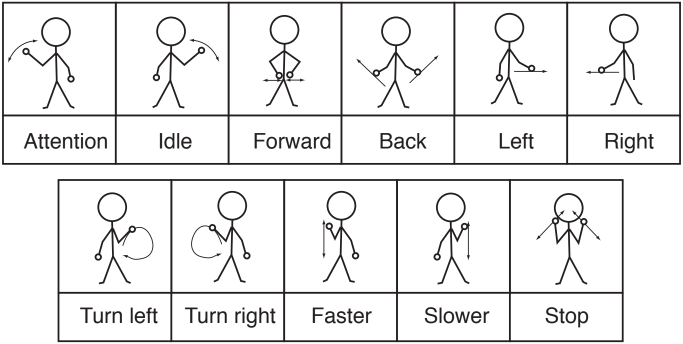


Tracking




x 10^-3
3.5
3
2.5
2
1.5
1
0.5
0
Motion
A
B
C
Threshold
0
50
100
150
200
Frame

Motion Patterns



Evaluation

- Acquired 2230 image sequences
- Covering 5 people in a normal living room
- 1115 used for training
- 1115 sequences were used for evaluation
- Capture of position and velocity data

Rec Rates	Position	Velocity	Combined
Result [%]	96.6	88.7	99.5

Example Timing

Phase	Time/frame[ms]
Image Transfer	4.3
Segmentation	0.6
Density Est	2.1
Connect Comp	2.1
Kalman Filter	0.3
HMM	21.0
Total	30.4

Outline

Summary

- The Hidden Markov Model (HMM)
- Many uses for sequential data models
- HMM is one possible formulation
- Autoregressive Models are common in data processing