Computer vision: models, learning and inference

Chapter 11

Models for Chains and Trees

Structure

- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

Chain and tree models

- Given a set of measurements $\{\mathbf{x}_n\}_{n=1}^N$ and world states $\{w_n\}_{n=1}^N$, infer the world states from the measurements.

- Problem: if N is large, then the model relating the two will have a very large number of parameters.

- Solution: build sparse models where we only describe subsets of the relations between variables.

Chain and tree models

Chain model: only model connections between a world variable and its 1 predeeding and 1 subsequent variables

Tree model: connections between world variables are organized as a tree (no loops). Disregard directionality of connections for directed model

Assumptions

We'll assume that

- World states w_n are discrete

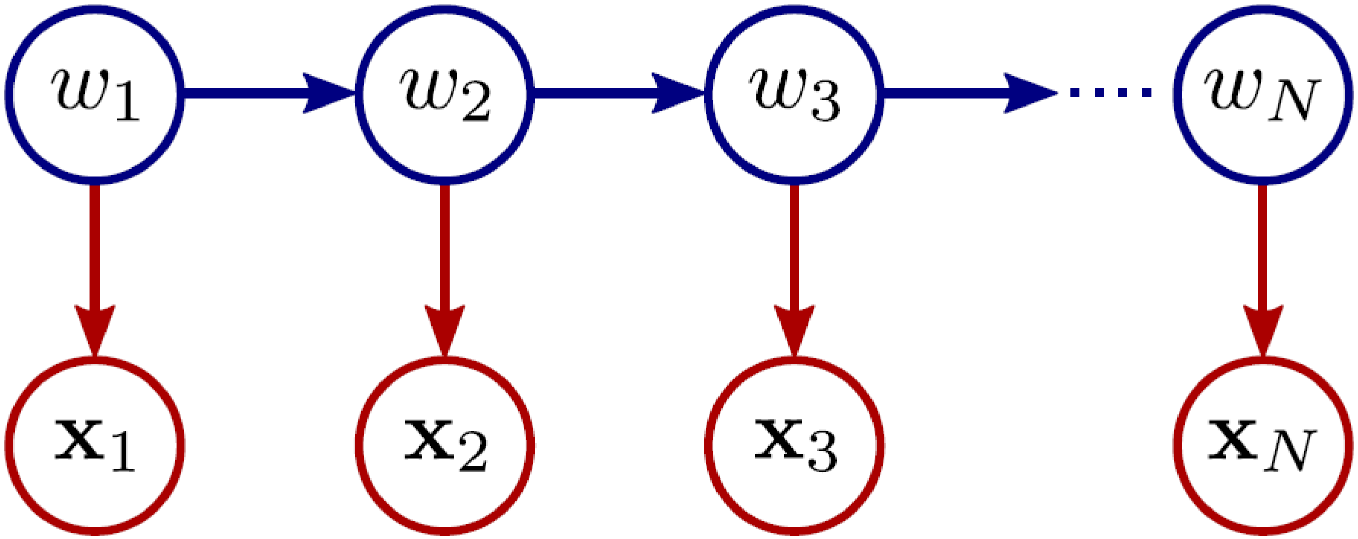
- Observed data variables $\mathbf{x}_n$ for each world state

- The nth data variable $\mathbf{x}_n$ is conditionally independent of all of other data variables and world states, given associated world state

Gesture Tracking

Figure 10.1 Interpreting sign language. We observe a sequence of images of a person using sign language. In each frame we extract a vector $\mathbf{x}$ describing the shape and position of the hands. The goal is to infer the sign w_n that is present. Unfortunately, the visual data in a single frame may be ambiguous. We improve matters by describing probabilistic connections between adjacent states w_n and w_{n-1}; we impose knowledge about the likely sequence of signs and this helps disambiguate any individual frame. Frames from Purdue RVL-SLLL ASL database (Wilbur & Kak 2006).

Directed model for chains (Hidden Markov model)

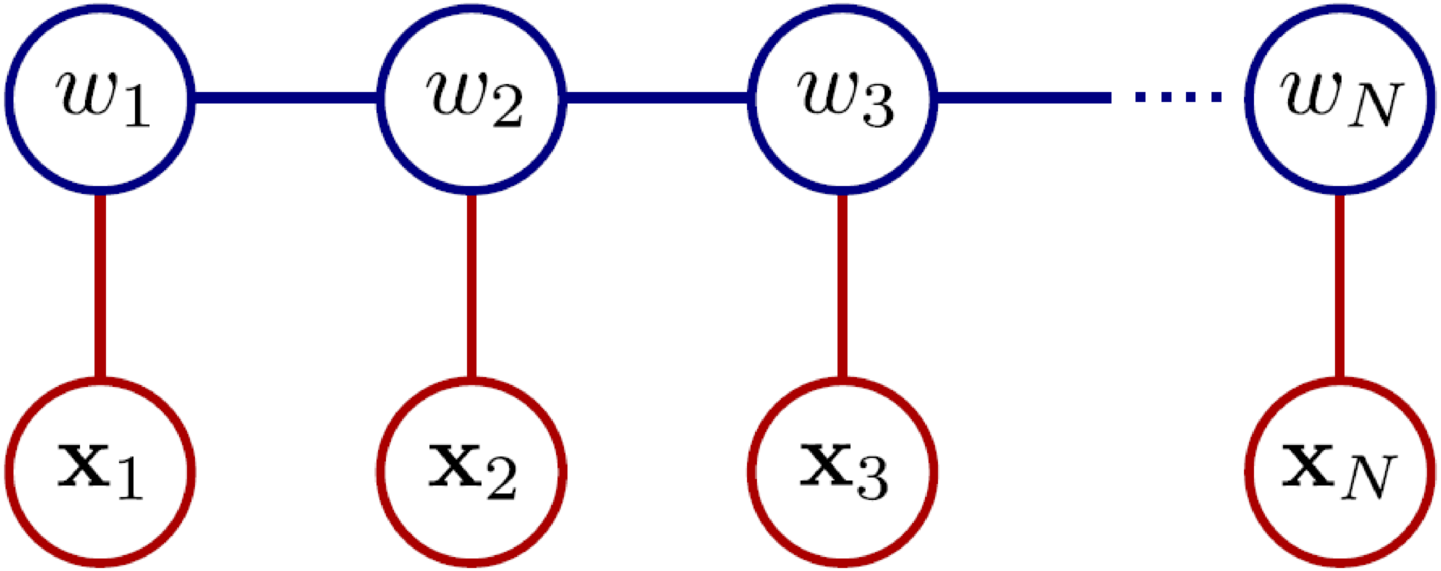


$$Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N}) = \left(\prod_{n=1}^{N} Pr(\mathbf{x}_n|w_n)\right)\left(\prod_{n=2}^{N} Pr(w_n|w_{n-1})\right)$$

Compatibility of measurement and world state

Compatibility of world state and previous world state

Undirected model for chains



$$Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N}) = \frac{1}{Z}\left(\prod_{n=1}^{N}\phi[\mathbf{x}_n, w_n]\right)\left(\prod_{n=2}^{N}\zeta[w_n, w_{n-1}]\right)$$

Compatibility of measurement and world state

Compatibility of world state and previous world state

Equivalence of chain models

Directed:

$$Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N}) = \left(\prod_{n=1}^{N} Pr(\mathbf{x}_n|w_n)\right)\left(\prod_{n=2}^{N} Pr(w_n|w_{n-1})\right)$$

Undirected:

$$Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N}) = \frac{1}{Z}\left(\prod_{n=1}^{N} \phi[\mathbf{x}_n, w_n]\right)\left(\prod_{n=2}^{N} \zeta[w_n, w_{n-1}]\right)$$

Equivalence:

$$\begin{aligned} Pr(\mathbf{x}_n|w_n) &= \frac{1}{z_n}\phi[\mathbf{x}_n, w_n] \\ Pr(w_n|w_{n-1}) &= \frac{1}{z'_n}\zeta[w_n, w_{n-1}] \end{aligned}$$

$$Z = \left(\prod_{n=1}^{N} z_n\right)\left(\prod_{n=2}^{N} z'_n\right)$$

Chain model for sign language application

Observations are normally distributed but depend on sign k

$$Pr(\mathbf{x}_n|w_n\!=\!k) = \text{Norm}_{\mathbf{x}_n}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]$$

World state is categorically distributed, parameters depend on previous world state

$$Pr(w_n|w_{n-1}\!=\!k) = \text{Cat}_{w_n}[\boldsymbol{\lambda}_k]$$

Structure

- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

MAP inference in chain model

Directed model:

$$Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N}) = \left(\prod_{n=1}^{N} Pr(\mathbf{x}_n|w_n)\right)\left(\prod_{n=2}^{N} Pr(w_n|w_{n-1})\right)$$

MAP inference:

$$\begin{aligned}\hat{w}_{1\ldots N} &= \underset{w_{1\ldots N}}{\operatorname{argmax}}\left[Pr(w_{1\ldots N}|\mathbf{x}_{1\ldots N})\right] \\ &= \underset{w_{1\ldots N}}{\operatorname{argmax}}\left[Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N})\right] \\ &= \underset{w_{1\ldots N}}{\operatorname{argmin}}\left[-\log\left[Pr(\mathbf{x}_{1\ldots N}, w_{1\ldots N})\right]\right]\end{aligned}$$

Substituting in :

$$\hat{w}_{1\ldots N} = \underset{w_{1\ldots N}}{\operatorname{argmin}}\left[-\sum_{n=1}^{N}\log\left[Pr(\mathbf{x}_n|w_n)\right] - \sum_{n=2}^{N}\log\left[Pr(w_n|w_{n-1})\right]\right]$$

MAP inference in chain model

$$\hat{w}_{1\ldots N} = \underset{w_{1\ldots N}}{\operatorname{argmin}}\left[-\sum_{n=1}^{N}\log\left[Pr(\mathbf{x}_n|w_n)\right] - \sum_{n=2}^{N}\log\left[Pr(w_n|w_{n-1})\right]\right]$$

Takes the general form:

$$\hat{w}_{1\ldots N} = \underset{w_{1\ldots N}}{\operatorname{argmin}}\left[\sum_{n=1}^{N}U_n(w_n) + \sum_{n=2}^{N}P_n(w_n, w_{n-1})\right]$$

Unary term: $U_n(w_n) = -\log[Pr(\mathbf{x}_n|w_n)]$

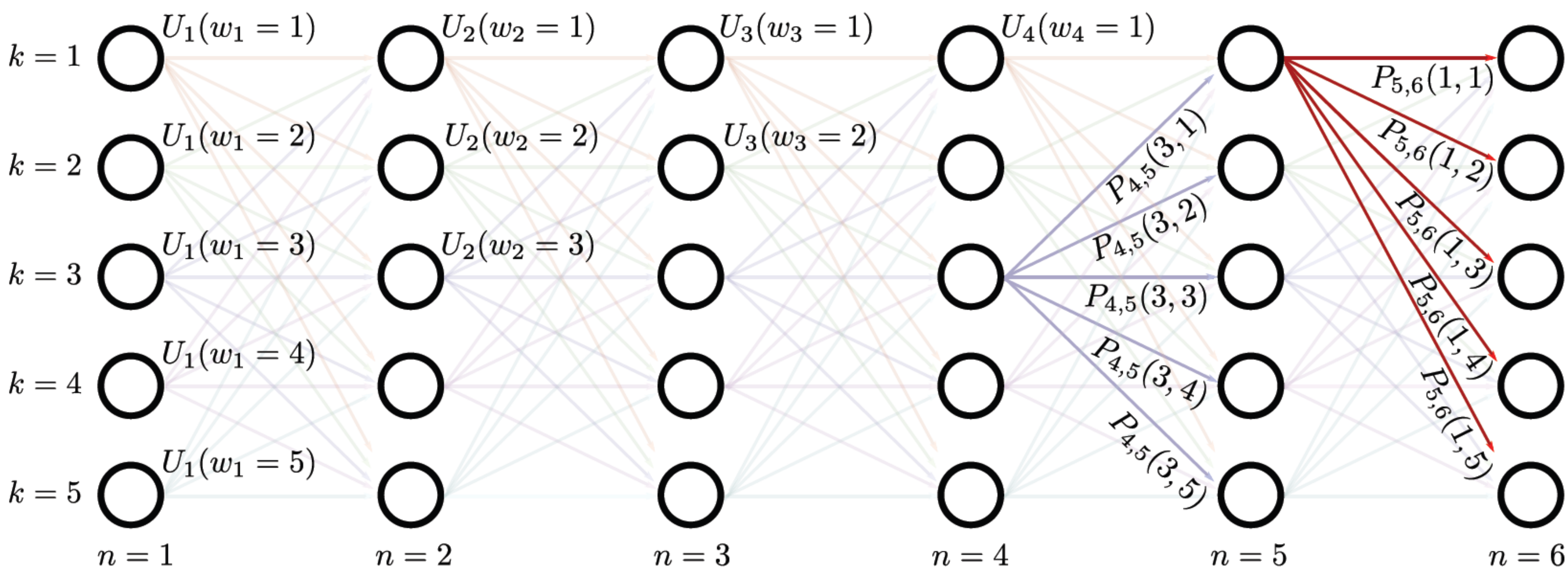
Pairwise term: $P_n(w_n, w_{n-1}) = -\log[Pr(w_n|w_{n-1})]$

Dynamic programming

Maximizes functions of the form:

$$\hat{w}_{1\ldots N} = \underset{w_{1\ldots N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{n=2}^{N} P_n(w_n, w_{n-1}) \right]$$

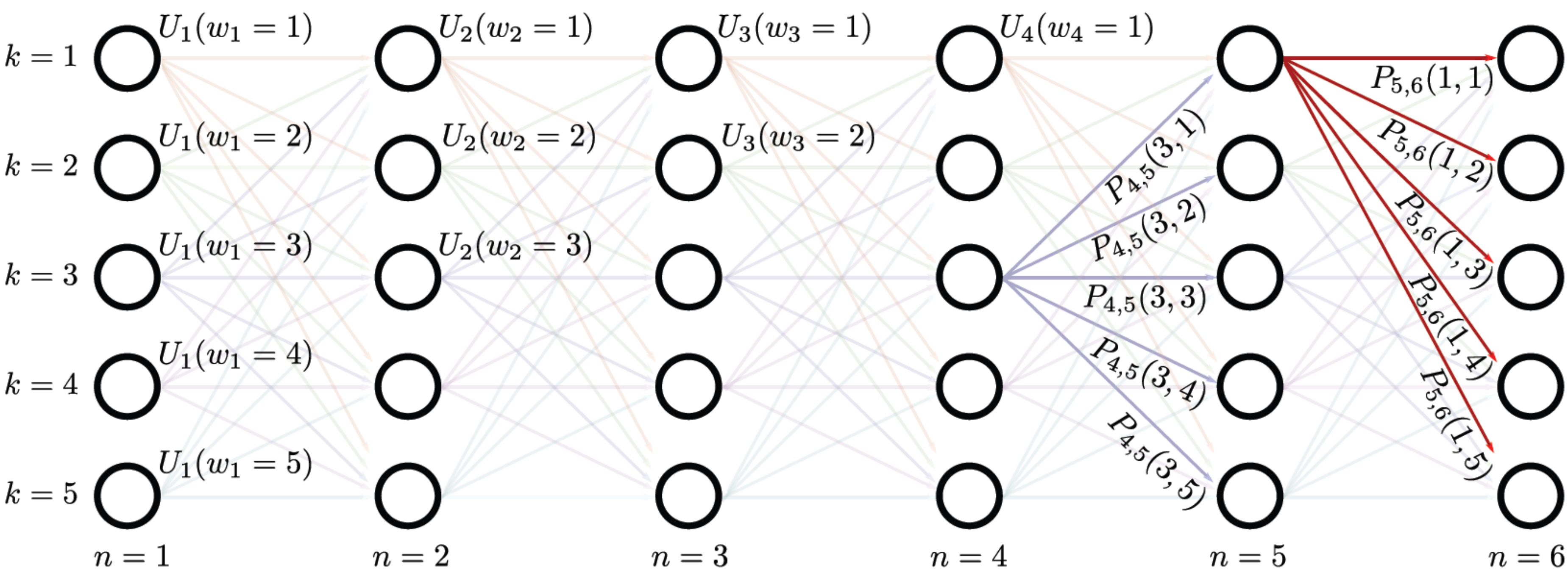
Set up as cost for traversing graph – each path from left to right is one possible configuration of world states



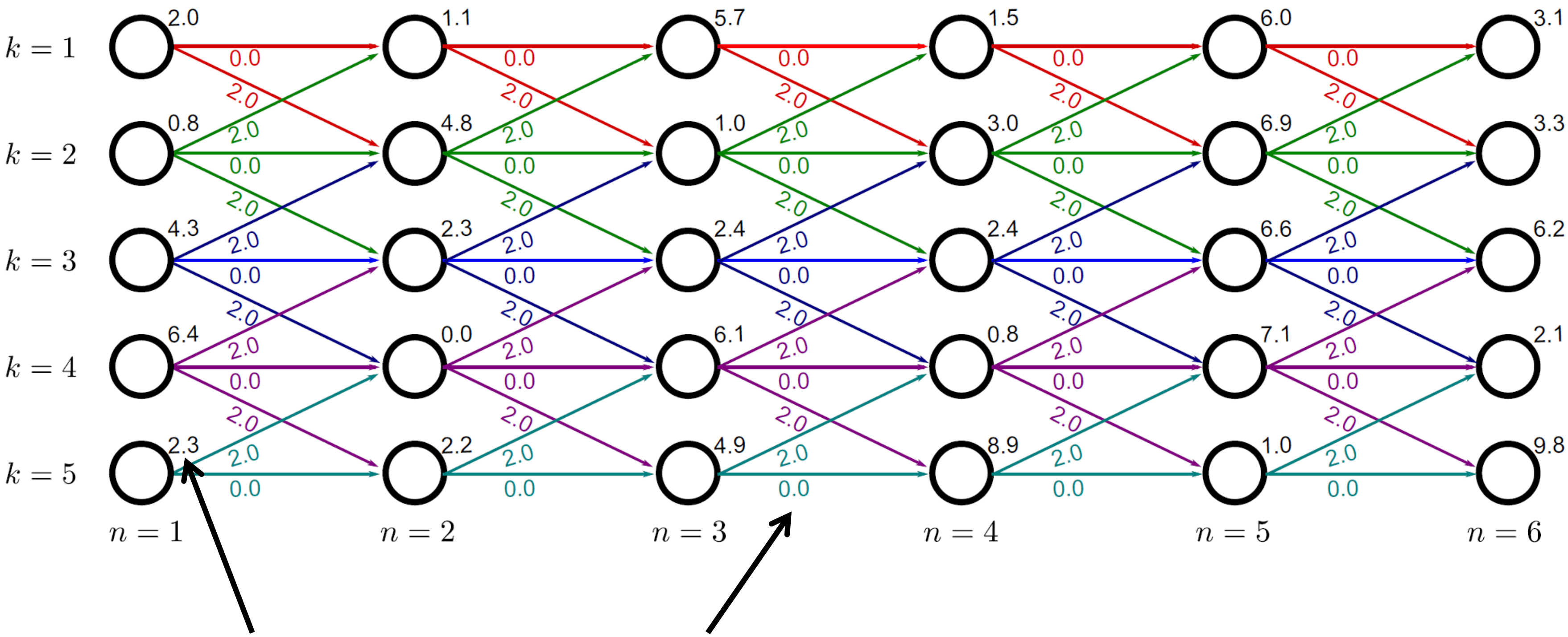
Dynamic programming

Algorithm:

1. Work through graph computing minimum possible cost $S_{n,k}$ to reach each node
2. When we get to last column, find minimum
3. Trace back to see how we got there



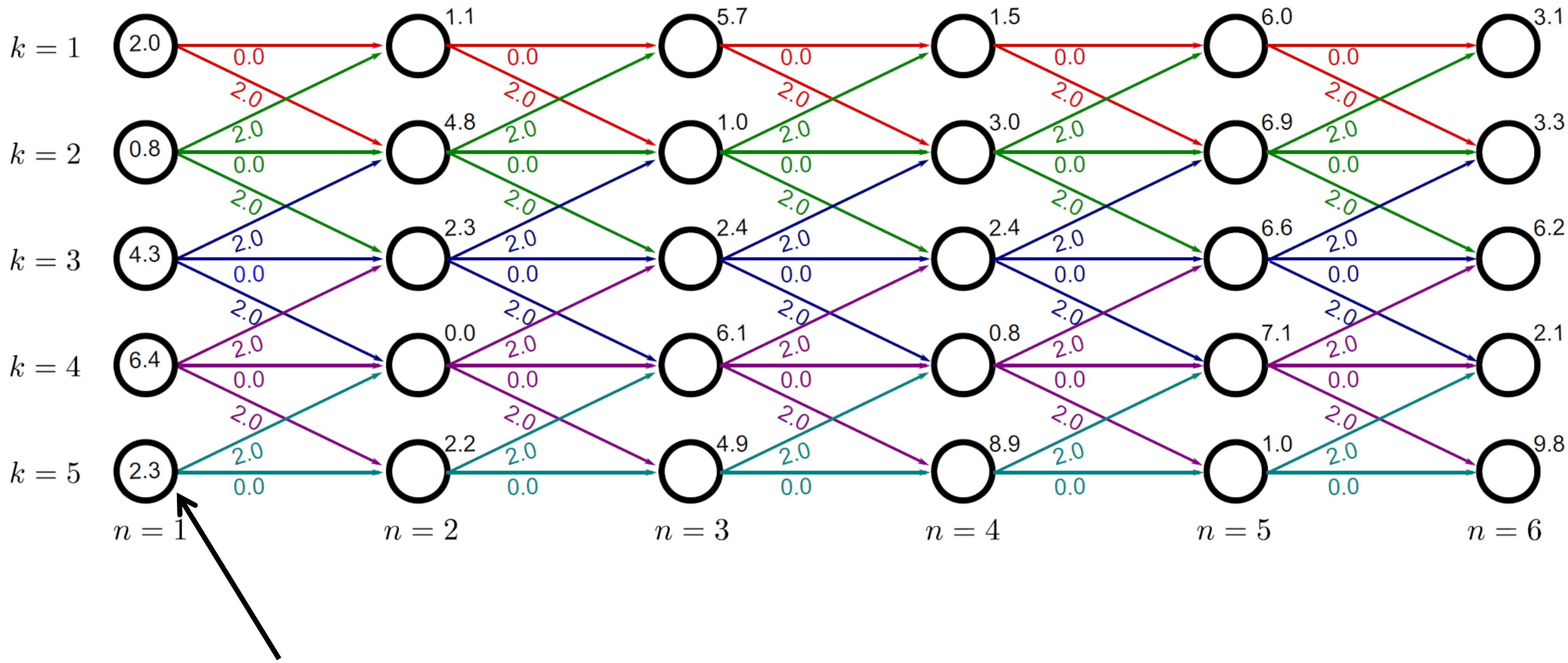
Worked example



Unary cost

Pairwise costs:
- Zero cost to stay at same label
- Cost of 2 to change label by 1
- Infinite cost for changing by more than one (not shown)

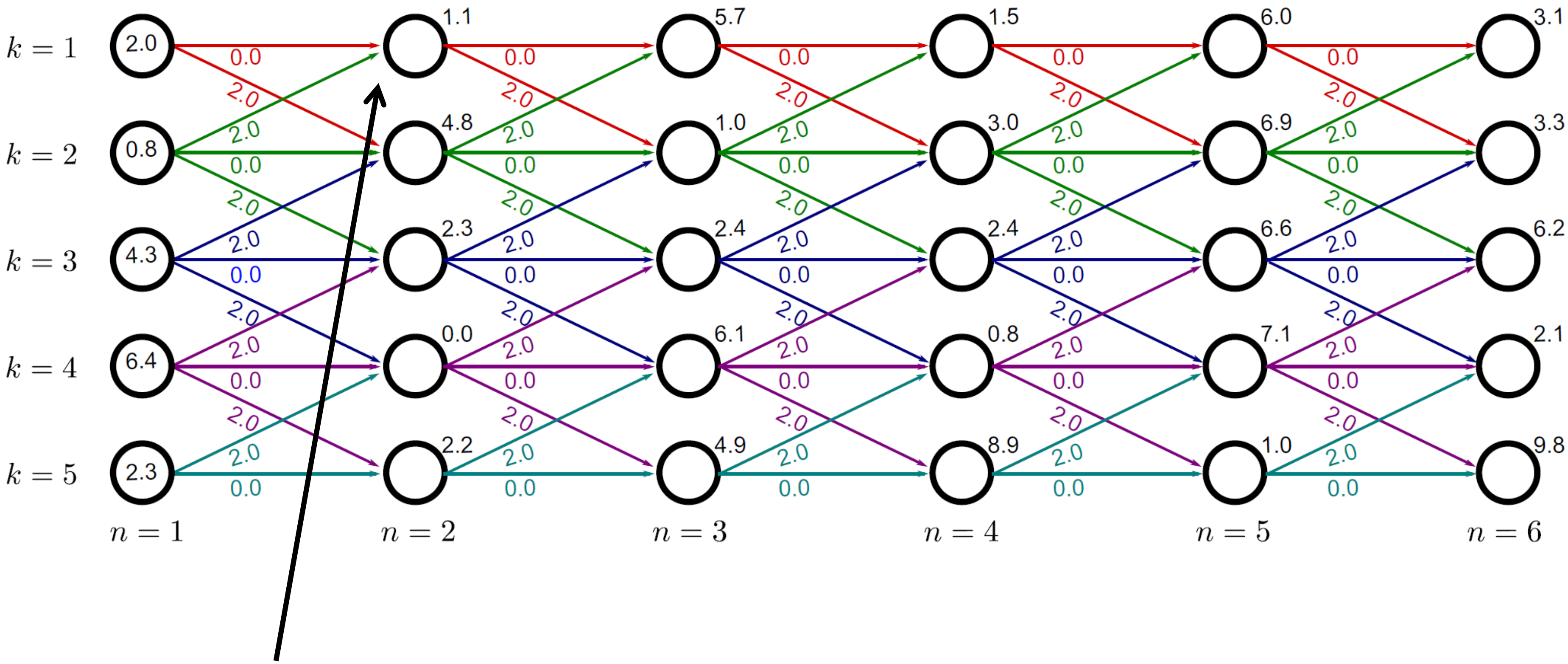
Worked example



Minimum cost $S_{1,1} \ldots S_{1,5}$
to reach first node is just unary cost

$$S_{1,k} = U_1(w_1 = k)$$

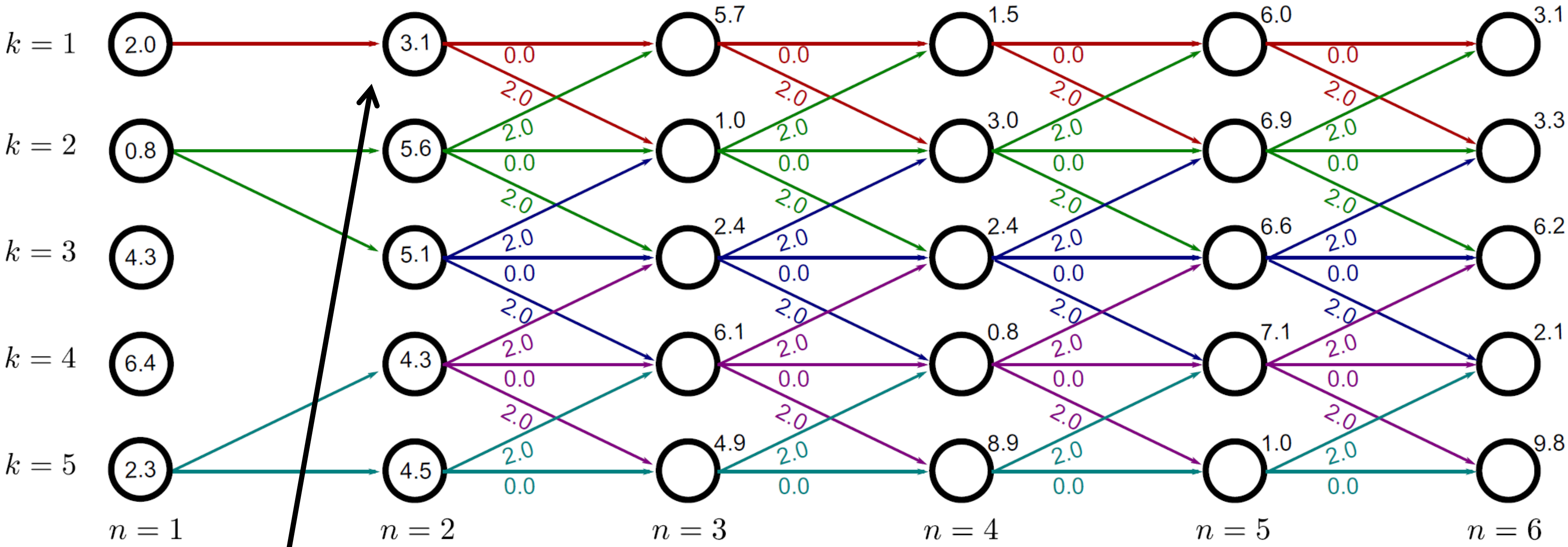
Worked example



Minimum cost $S_{2,1}$ is minimum of two possible routes to get here

Route 1: 2.0+0.0+1.1 = 3.1
Route 2: 0.8+2.0+1.1 = 3.9

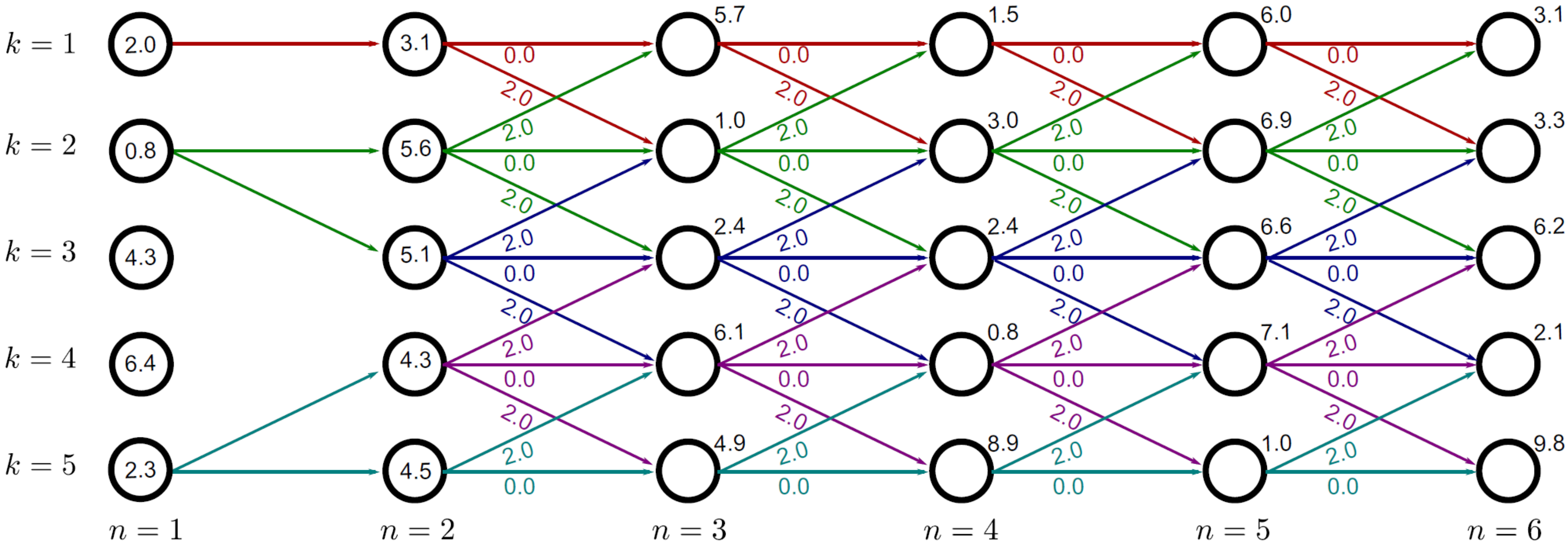
Worked example



Minimum cost $S_{2,1}$ is minimum of two possible routes to get here

Route 1: 2.0+0.0+1.1 = 3.1 -- this is the minimum – note this down
Route 2: 0.8+2.0+1.1 = 3.9

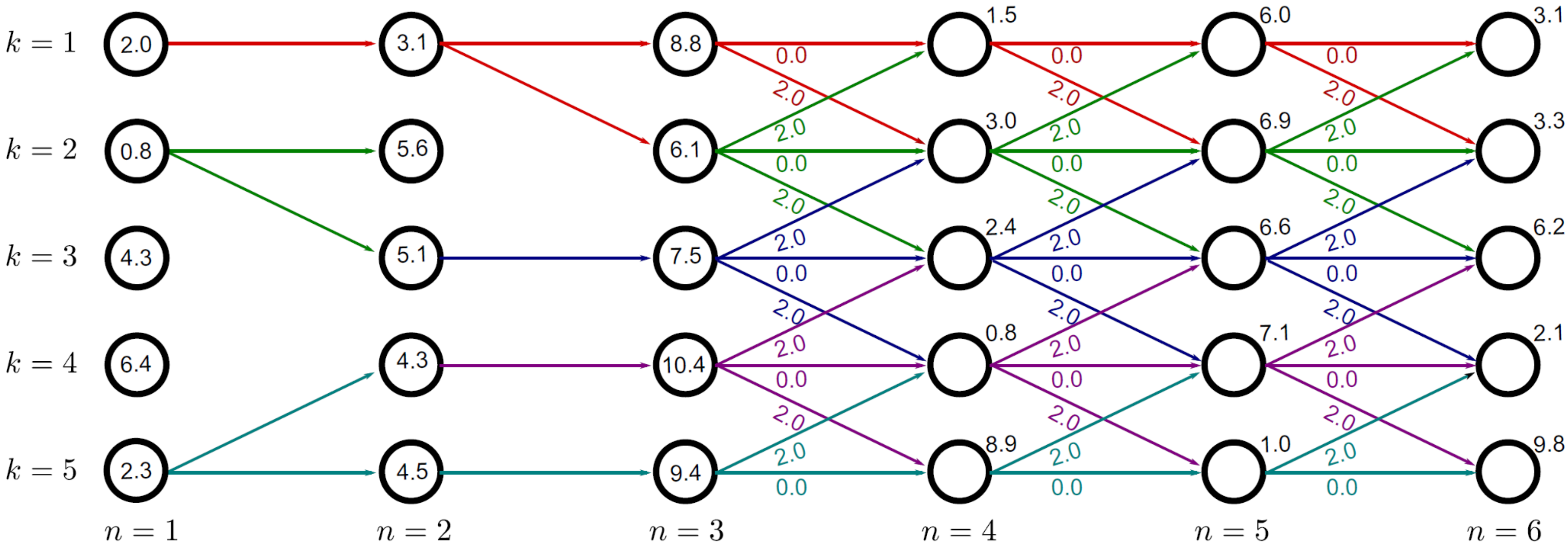
Worked example



General rule:

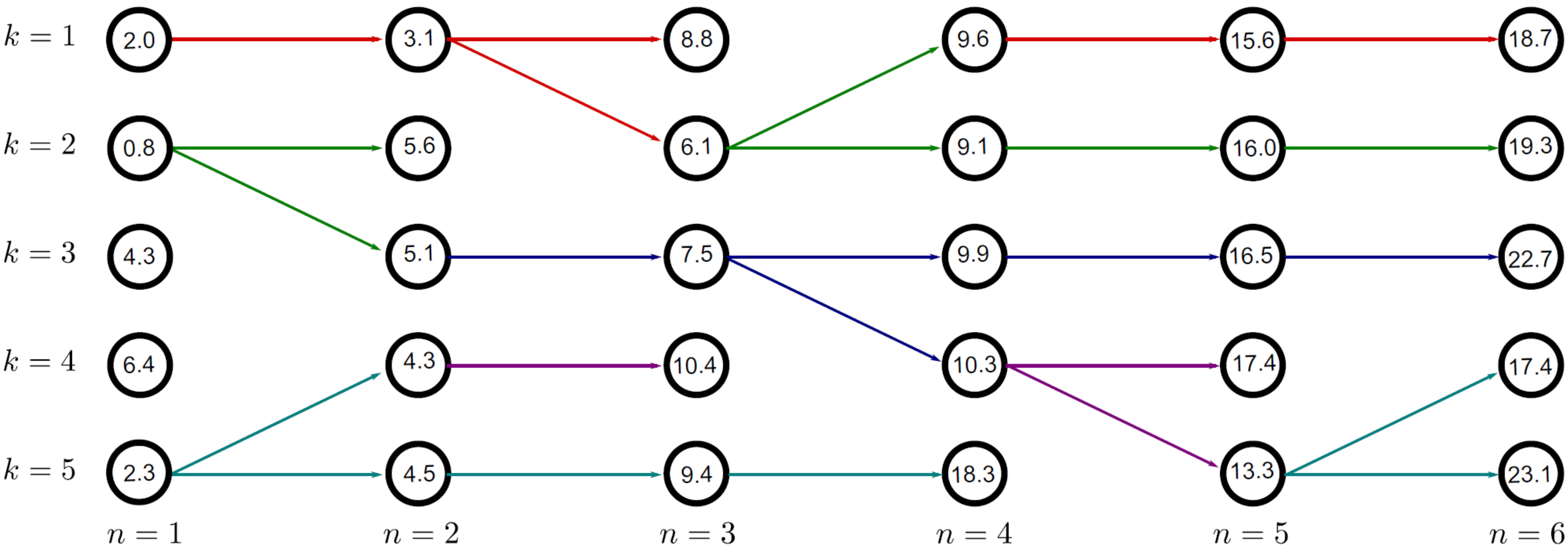
$$S_{n,k} = U_n(w_n = k) + \min_l \left[S_{n-1,l} + P_n(w_n = k, w_{n-1} = l)\right]$$

Worked example



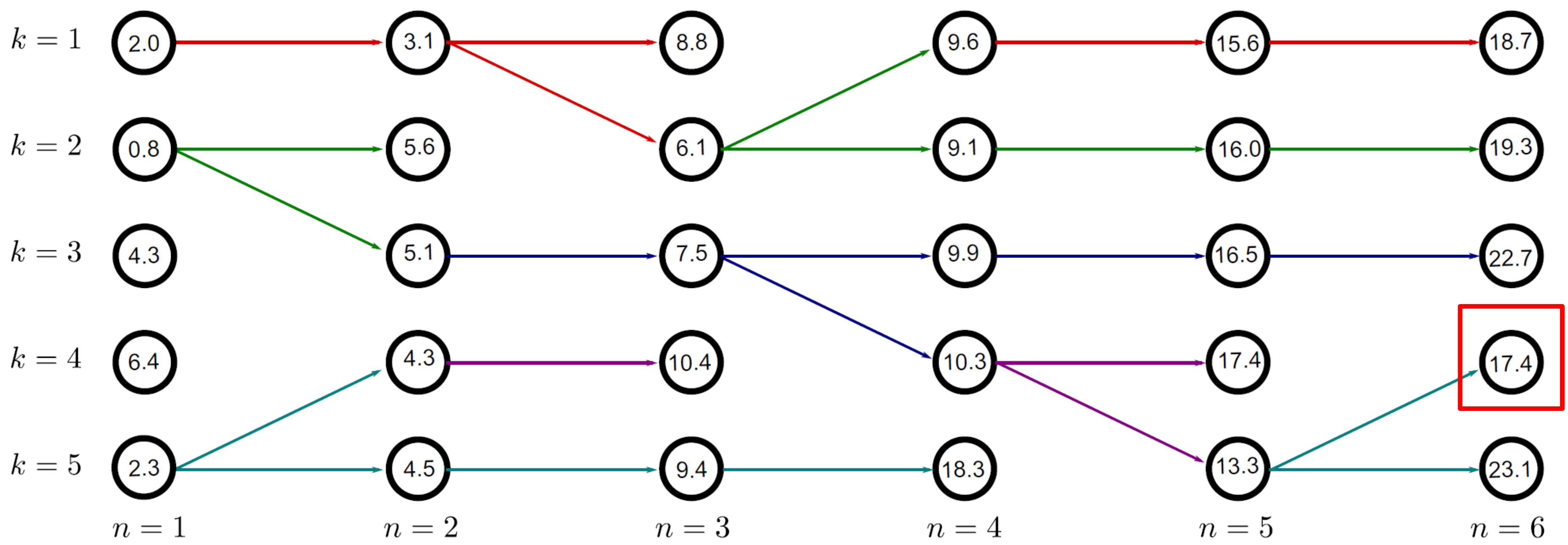
Work through the graph, computing the minimum cost to reach each node

Worked example



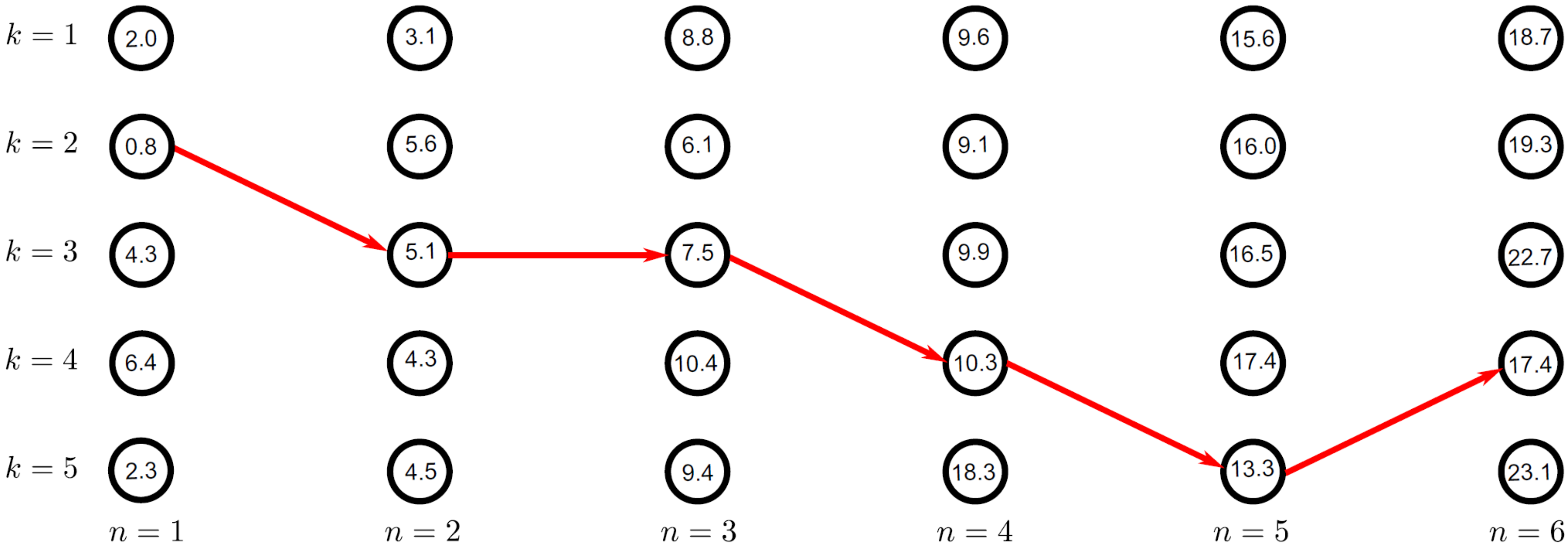
Keep going until we reach the end of the graph

Worked example



Find the minimum possible cost to reach the final column

Worked example

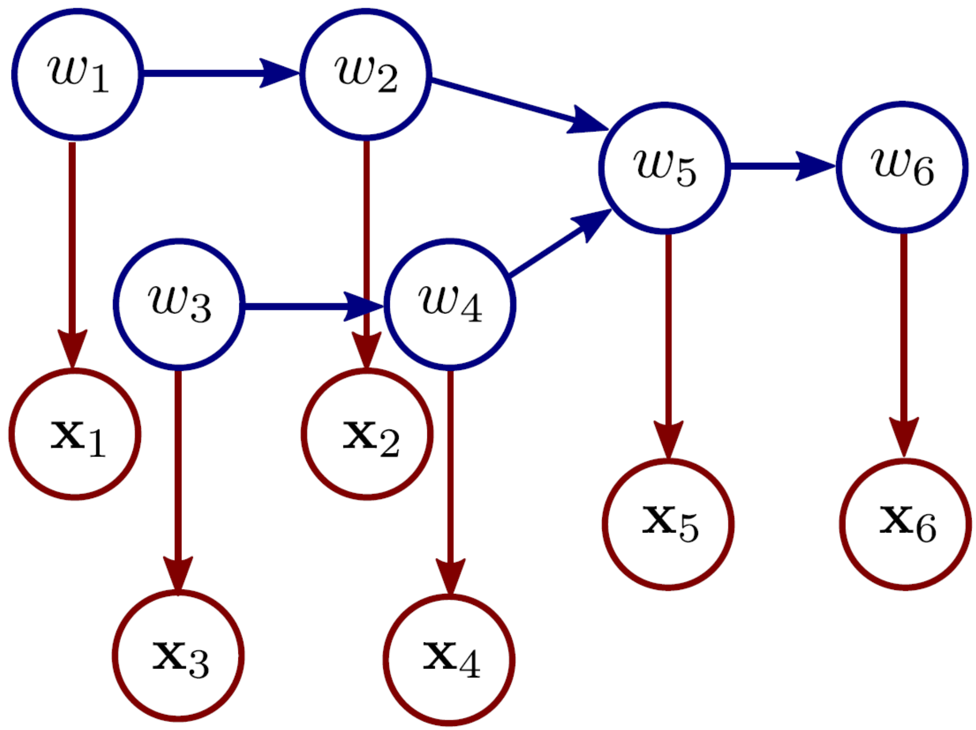


Trace back the route that we arrived here by – this is the minimum configuration

Structure

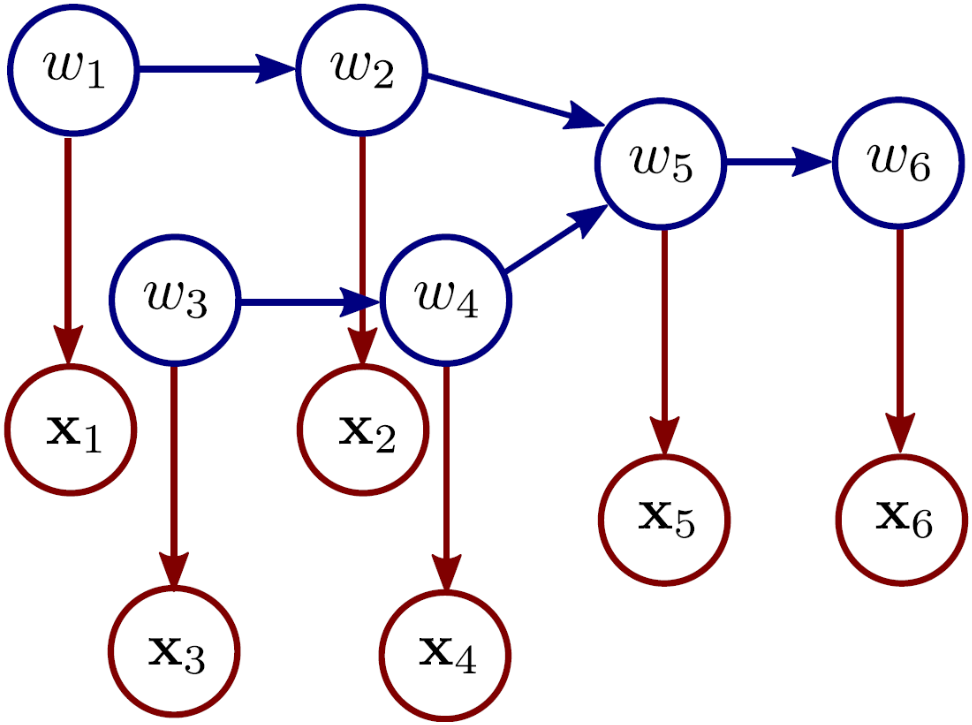
- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

MAP inference for trees



$$Pr(w_{1\ldots 6}) = Pr(w_1)Pr(w_3)Pr(w_2|w_1)Pr(w_4|w_3)Pr(w_5|w_2,w_4)Pr(w_6|w_5)$$

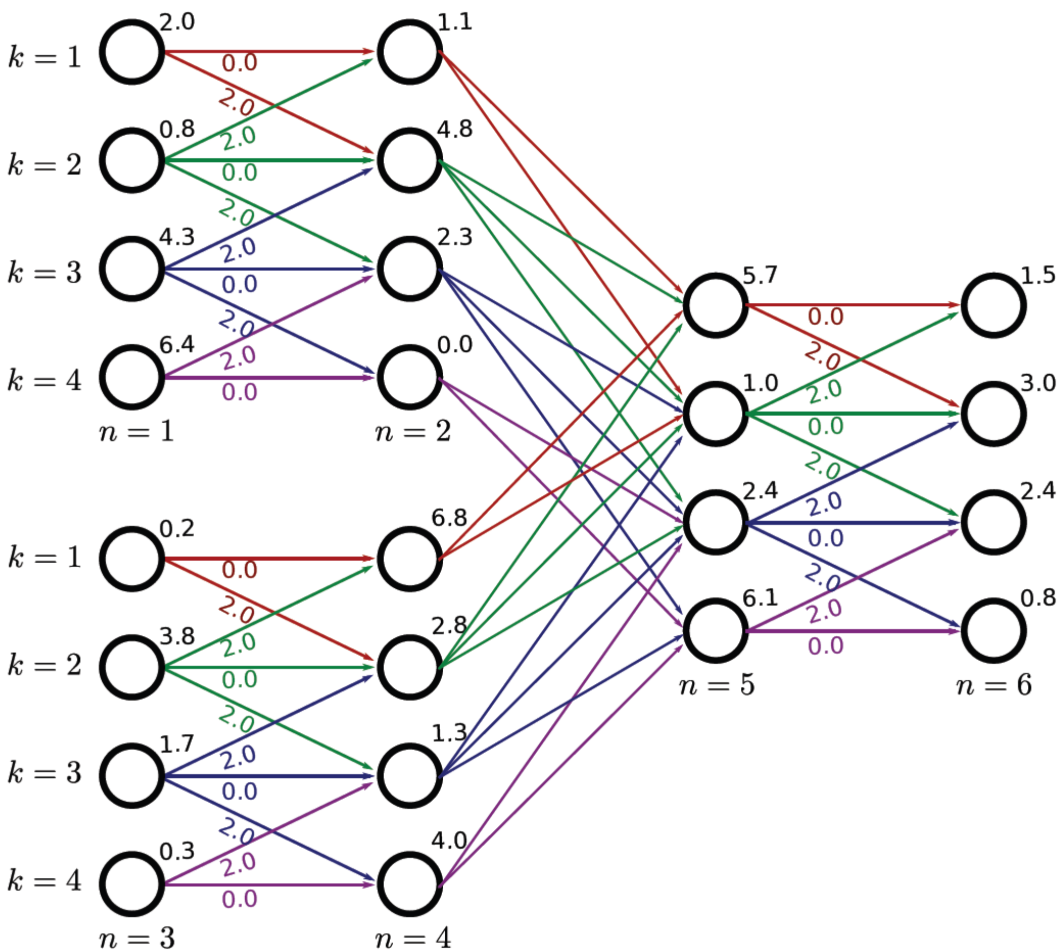
MAP inference for trees



$$\hat{w}_{1\ldots6} = \underset{w_{1\ldots6}}{\operatorname{argmax}}\left[\sum_{n=1}^{6}\log[Pr(\mathbf{x}_n|w_n)] + \log[Pr(w_{1\ldots6})]\right]$$

$$\hat{w}_{1\ldots6} = \underset{w_{1\ldots6}}{\operatorname{argmin}}\left[\sum_{n=1}^{6}U_n(w_n)+P_2(w_2,w_1)+P_4(w_4,w_3)+P_6(w_6,w_5)+T_5(w_5,w_2,w_4)\right]$$

Worked example



$T_5(w_5 = 1, w_2, w_4)$

w_2 \ w_4	1	2	3	4
1	0.4	1.7	∞	∞
2	2.3	0.0	∞	∞
3	∞	∞	∞	∞
4	∞	∞	∞	∞

$T_5(w_5 = 2, w_2, w_4)$

w_2 \ w_4	1	2	3	4
1	0.0	1.7	3.7	∞
2	2.7	0.3	1.0	∞
3	2.3	1.4	0.2	∞
4	∞	∞	∞	∞

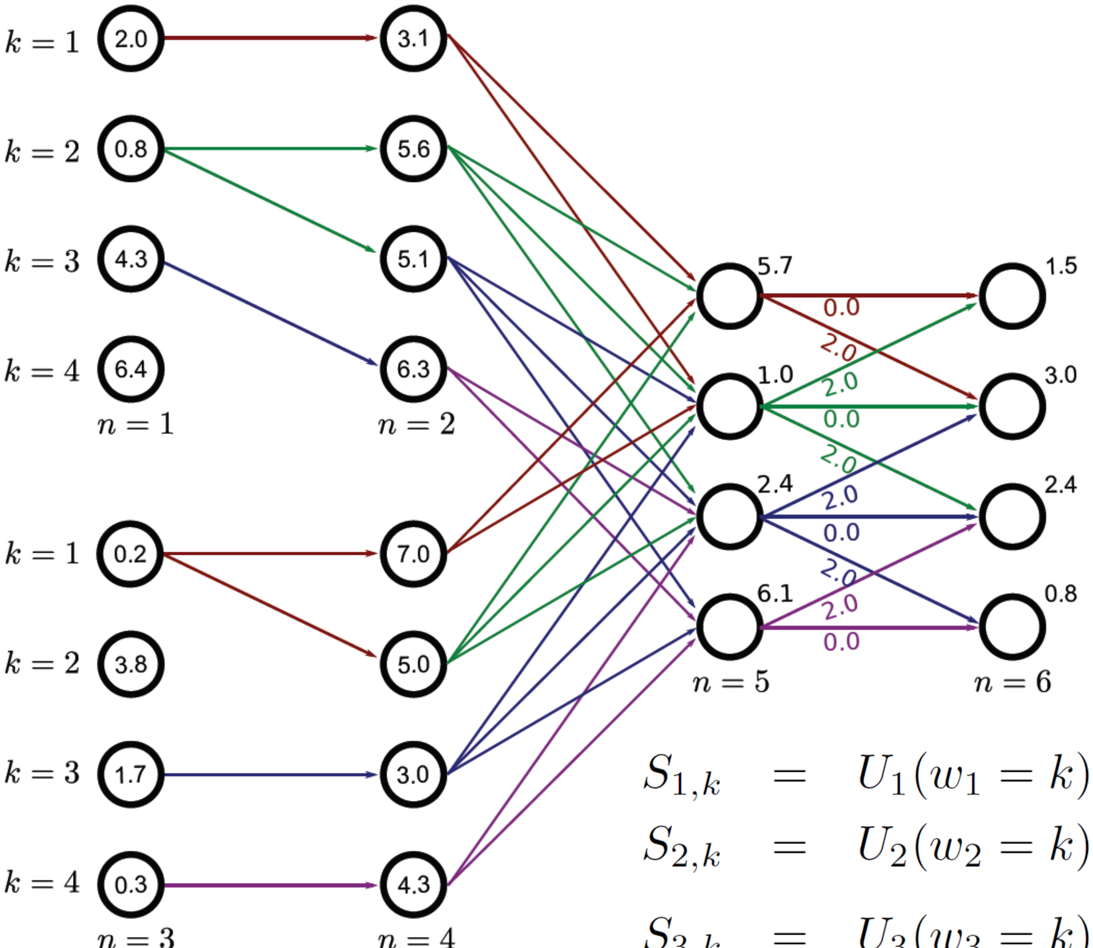
$T_5(w_5 = 3, w_2, w_4)$

w_2 \ w_4	1	2	3	4
1	∞	∞	∞	∞
2	∞	0.3	1.2	2.3
3	∞	1.7	0.4	1.4
4	∞	1.7	5.4	0.4

$T_5(w_5 = 4, w_2, w_4)$

w_2 \ w_4	1	2	3	4
1	∞	∞	∞	∞
2	∞	∞	∞	∞
3	∞	∞	0.4	1.7
4	∞	∞	2.3	0.0

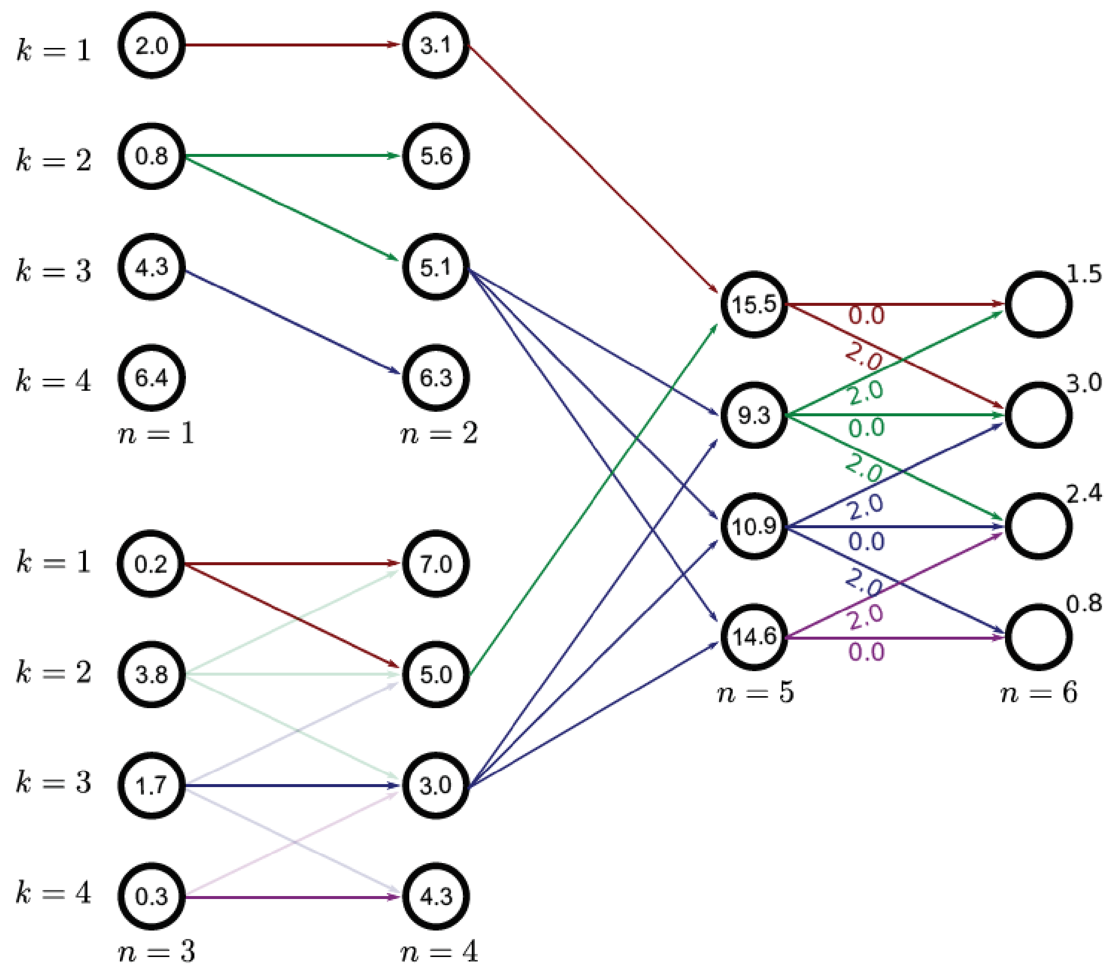
Worked example



Variables 1-4 proceed as for the chain example.

$$
\begin{aligned}
S_{1,k} &= U_1(w_1 = k) \\
S_{2,k} &= U_2(w_2 = k) + \min_l \left[S_{1,l} + P_2(w_2 = k, w_1 = l)\right] \\
S_{3,k} &= U_3(w_3 = k) \\
S_{4,k} &= U_4(w_4 = k) + \min_l \left[S_{3,1} + P_4(w_4 = k, w_3 = l)\right]
\end{aligned}
$$

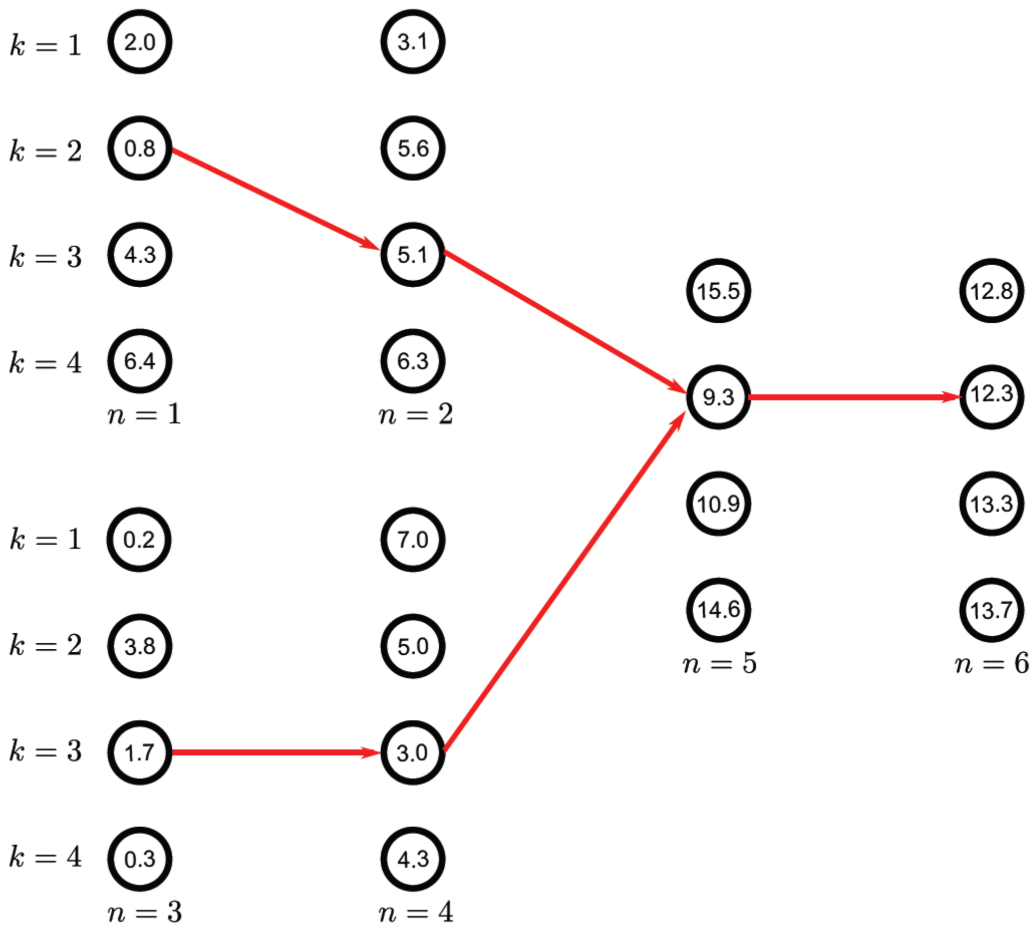
Worked example



At variable n=5 must consider all pairs of paths from into the current node.

$$S_{5,k} = U_5(w_5 = k) + \min_{l,m}\left[S_{2,l} + S_{4,m} + T_5(w_5 = k, w_2 = l, w_4 = m)\right]$$

Worked example



Variable 6 proceeds as normal.

Then we trace back through the variables, splitting at the junction.

Structure

- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

Marginal posterior inference

- Start by computing the marginal distribution $Pr(w_N|\mathbf{x}_{1\ldots N})$ over the N^{th} variable

- Then we`ll consider how to compute the other marginal distributions

Computing one marginal distribution

Compute the posterior using Bayes` rule:

$$Pr(w_N|\mathbf{x}_{1\ldots N}) = \frac{Pr(w_N, \mathbf{x}_{1\ldots N})}{Pr(\mathbf{x}_{1\ldots N})} \propto \Pr(w_N, \mathbf{x}_{1\ldots N})$$

We compute this expression by writing the joint probability :

$$\begin{aligned} Pr(w_N|\mathbf{x}_{1\ldots N}) &\propto \sum_{w_1}\sum_{w_2}\ldots\sum_{w_{N-1}} Pr(w_{1\ldots N}, \mathbf{x}_{1\ldots N}) \\ &\propto \sum_{w_1}\sum_{w_2}\ldots\sum_{w_{N-1}} \left(\prod_{n=1}^{N} Pr(\mathbf{x}_n|w_n)\right) Pr(w_1) \left(\prod_{n=2}^{N} Pr(w_n|w_{n-1})\right) \end{aligned}$$

Computing one marginal distribution

$$Pr(w_N|\mathbf{x}_{1\ldots N}) \propto \sum_{w_1}\sum_{w_2}\ldots\sum_{w_{N-1}}\left(\prod_{n=1}^{N} Pr(\mathbf{x}_n|w_n)\right) Pr(w_1)\left(\prod_{n=2}^{N} Pr(w_n|w_{n-1})\right)$$

Problem: Computing all N^K states and marginalizing explicitly is intractable.

Solution: Re-order terms and move summations to the right

$$\begin{aligned}Pr(w_N|\mathbf{x}_{1\ldots N}) &\propto \\ &Pr(\mathbf{x}_N|w_N)\sum_{w_{N-1}}\ldots\sum_{w_2} Pr(w_3|w_2)Pr(\mathbf{x}_2|w_2)\sum_{w_1} Pr(w_2|w_1)Pr(\mathbf{x}_1|w_1)Pr(w_1)\end{aligned}$$

Computing one marginal distribution

$$Pr(w_N|\mathbf{x}_{1\ldots N}) \propto$$
$$Pr(\mathbf{x}_N|w_N)\sum_{w_{N-1}}\ldots\sum_{w_2}Pr(w_3|w_2)Pr(\mathbf{x}_2|w_2)\sum_{w_1}Pr(w_2|w_1)Pr(\mathbf{x}_1|w_1)Pr(w_1)$$

Define function of variable w_1 (two rightmost terms)

$$\mathbf{f}_1[w_1] = Pr(\mathbf{x}_1|w_1)Pr(w_1)$$

Then compute function of variables w_2 in terms of previous function

$$\mathbf{f}_2[w_2] = Pr(\mathbf{x}_2|w_2)\sum_{w_1}Pr(w_2|w_1)\mathbf{f}_1[w_1]$$

Leads to the recursive relation

$$\mathbf{f}_n[w_n] = Pr(\mathbf{x}_n|w_n)\sum_{w_{n-1}}Pr(w_n|w_{n-1})\mathbf{f}_{n-1}[w_{n-1}]$$

Computing one marginal distribution

$$\mathbf{f}_n[w_n] = Pr(\mathbf{x}_n|w_n)\sum_{w_{n-1}} Pr(w_n|w_{n-1})\mathbf{f}_{n-1}[w_{n-1}]$$

We work our way through the sequence using this recursion.

At the end we normalize the result to compute the posterior $Pr(w_N|\mathbf{x}_{1\ldots N})$

Total number of summations is (N-1)K as opposed to K^N for brute force approach.

Forward-backward algorithm

- We could compute the other N-1 marginal posterior distributions using a similar set of computations
- However, this is inefficient as much of the computation is duplicated
- The forward-backward algorithm computes all of the marginal posteriors at once

$$\begin{aligned} Pr(w_n|\mathbf{x}_{1\ldots N}) &\propto Pr(w_n, \mathbf{x}_{1\ldots N}) \\ &= Pr(w_n, \mathbf{x}_{1\ldots n}) Pr(\mathbf{x}_{n+1\ldots N}|w_n, \mathbf{x}_{1\ldots n}) \\ &= Pr(w_n, \mathbf{x}_{1\ldots n}) Pr(\mathbf{x}_{n+1\ldots N}|w_n) \end{aligned}$$

Solution:

... and take products

Compute all first term using a recursion

Compute all second terms using a recursion

Forward recursion

Conditional probability rule

$$
\begin{aligned}
&Pr(w_n, \mathbf{x}_{1\ldots n})\\
&\quad = \sum_{w_{n-1}} Pr(w_n, w_{n-1}, \mathbf{x}_{1\ldots n})\\
&\quad = \sum_{w_{n-1}} Pr(w_n, \mathbf{x}_n | w_{n-1}, \mathbf{x}_{1\ldots n-1}) Pr(w_{n-1}, \mathbf{x}_{1\ldots n-1})\\
&\quad = \sum_{w_{n-1}} Pr(\mathbf{x}_n | w_n, w_{n-1}, \mathbf{x}_{1\ldots n-1}) Pr(w_n | w_{n-1}, \mathbf{x}_{1\ldots n-1}) Pr(w_{n-1}, \mathbf{x}_{1\ldots n-1})\\
&\quad = \underbrace{\sum_{w_{n-1}} Pr(\mathbf{x}_n | w_n) Pr(w_n | w_{n-1}) Pr(w_{n-1}, \mathbf{x}_{1\ldots n-1})}_{\text{This is the same recursion as before}},
\end{aligned}
$$

Using conditional independence relations

$$
\mathbf{f}_n[w_n] = Pr(\mathbf{x}_n | w_n) \sum_{w_{n-1}} Pr(w_n | w_{n-1}) \mathbf{f}_{n-1}[w_{n-1}]
$$

Backward recursion

$$
\begin{aligned}
Pr(\mathbf{x}_{n\ldots N}|w_{n-1}) \\
&= \sum_{w_n} Pr(\mathbf{x}_{n\ldots N}, w_n|w_{n-1}) \\
&= \sum_{w_n} Pr(\mathbf{x}_{n\ldots N}|w_n, w_{n-1})Pr(w_n|w_{n-1}) \\
&= \sum_{w_n} Pr(\mathbf{x}_{n+1\ldots N}|\mathbf{x}_n, w_n, w_{n-1})Pr(\mathbf{x}_n|w_n, w_{n-1})Pr(w_n|w_{n-1}) \\
&= \sum_{w_n} Pr(\mathbf{x}_{n+1\ldots N}|w_n)Pr(\mathbf{x}_n|w_n)Pr(w_n|w_{n-1}).
\end{aligned}
$$

Conditional probability rule

Using conditional independence relations

This is another recursion of the form

$$
\mathbf{b}_{n-1}[w_{n-1}] = \sum_{w_n} Pr(\mathbf{x}_n|w_n)Pr(w_n|w_{n-1})\mathbf{b}_n[w_n]
$$

Forward backward algorithm

Compute the marginal posterior distribution as product of two terms

$$
\begin{aligned}
Pr(w_n|\mathbf{x}_{1\ldots N}) &\propto Pr(w_n, \mathbf{x}_{1\ldots n})Pr(\mathbf{x}_{n+1\ldots N}|w_n) \\
&= \mathbf{f}_n[w_n]\mathbf{b}_n[w_n]
\end{aligned}
$$

Forward terms:

$$
\mathbf{f}_n[w_n] = Pr(\mathbf{x}_n|w_n)\sum_{w_{n-1}} Pr(w_n|w_{n-1})\mathbf{f}_{n-1}[w_{n-1}]
$$

Backward terms:

$$
\mathbf{b}_{n-1}[w_{n-1}] = \sum_{w_n} Pr(\mathbf{x}_n|w_n)Pr(w_n|w_{n-1})\mathbf{b}_n[w_n]
$$

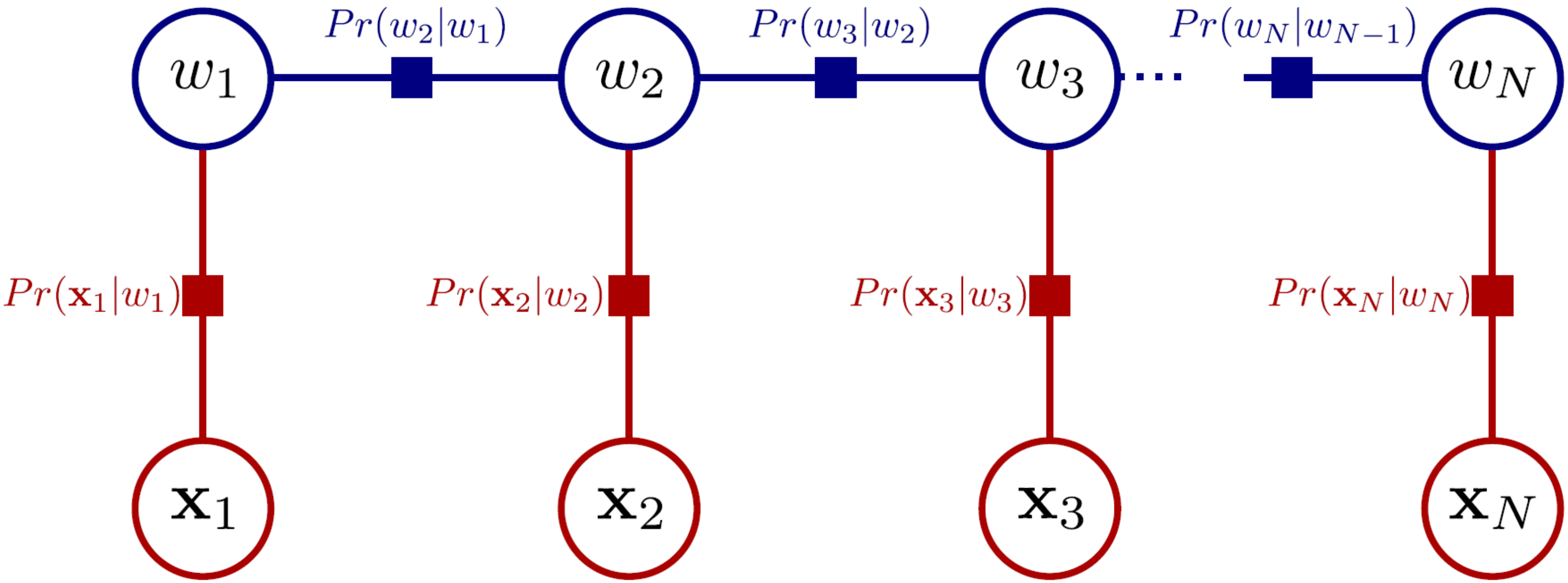
Belief propagation

- Forward backward algorithm is a special case of a more general technique called belief propagation
- Intermediate functions in forward and backward recursions are considered as messages conveying beliefs about the variables.
- We’ll examine the Sum-Product algorithm.
- The sum-product algorithm operates on factor graphs.

Sum product algorithm

- Forward backward algorithm is a special case of a more general technique called belief propagation
- Intermediate functions in forward and backward recursions are considered as messages conveying beliefs about the variables.
- We'll examine the Sum-Product algorithm.
- The sum-product algorithm operates on factor graphs.

Factor graphs



- One node for each variable
- One node for each function relating variables

Sum product algorithm

Forward pass

- Distribute evidence through the graph

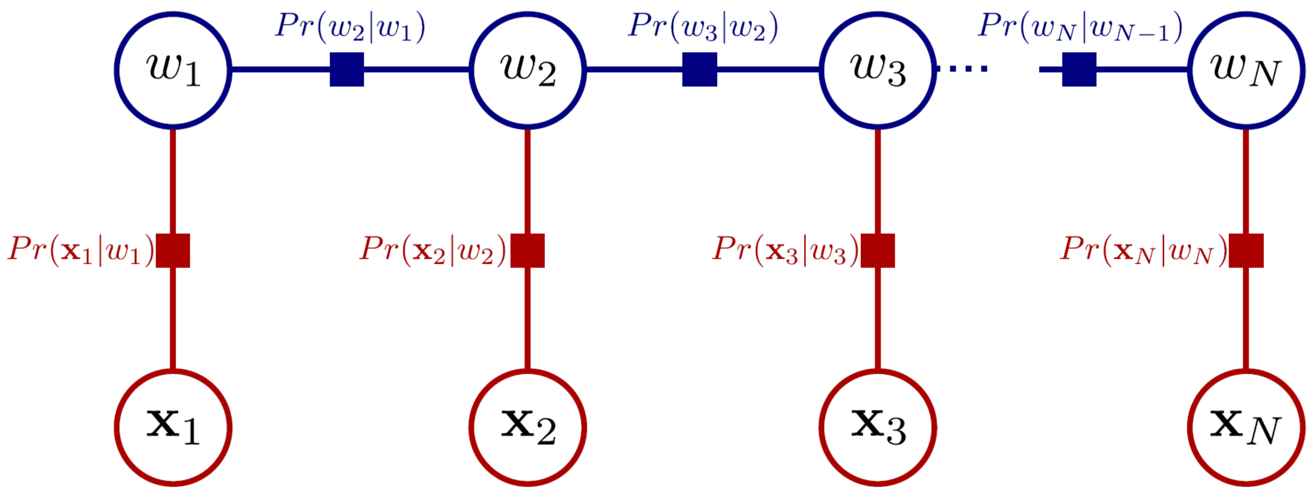
Backward pass

- Collates the evidence

Both phases involve passing messages between nodes:

- The forward phase can proceed in any order as long as the outgoing messages are not sent until all incoming ones received
- Backward phase proceeds in reverse order to forward

Sum product algorithm



Three kinds of message

- Messages from unobserved variables to functions
- Messages from observed variables to functions
- Messages from functions to variables

Sum product algorithm

Message type 1:

- Messages from unobserved variables z to function g

$$\mathbf{m}_{\mathbf{z}_p \to g_q} = \prod_{r \in \mathrm{Ne}[p] \setminus q} \mathbf{m}_{g_r \to \mathbf{z}_p}$$

- Take product of incoming messages
- Interpretation: combining beliefs

Message type 2:

- Messages from observed variables z to function g

$$\mathbf{m}_{z_p \to g_q} = \delta[\mathbf{z}_p^*]$$

- Interpretation: conveys certain belief that observed values are true

Sum product algorithm

Message type 3:

- Messages from a function g to variable z

$$\mathbf{m}_{g_p\rightarrow\mathbf{z}_q} = \sum_{\text{Ne}[p]\setminus q} g_p[\text{Ne}[p]] \prod_{r\in\text{Ne}[p]\setminus q} m_{\mathbf{z}_r\rightarrow g_p}$$

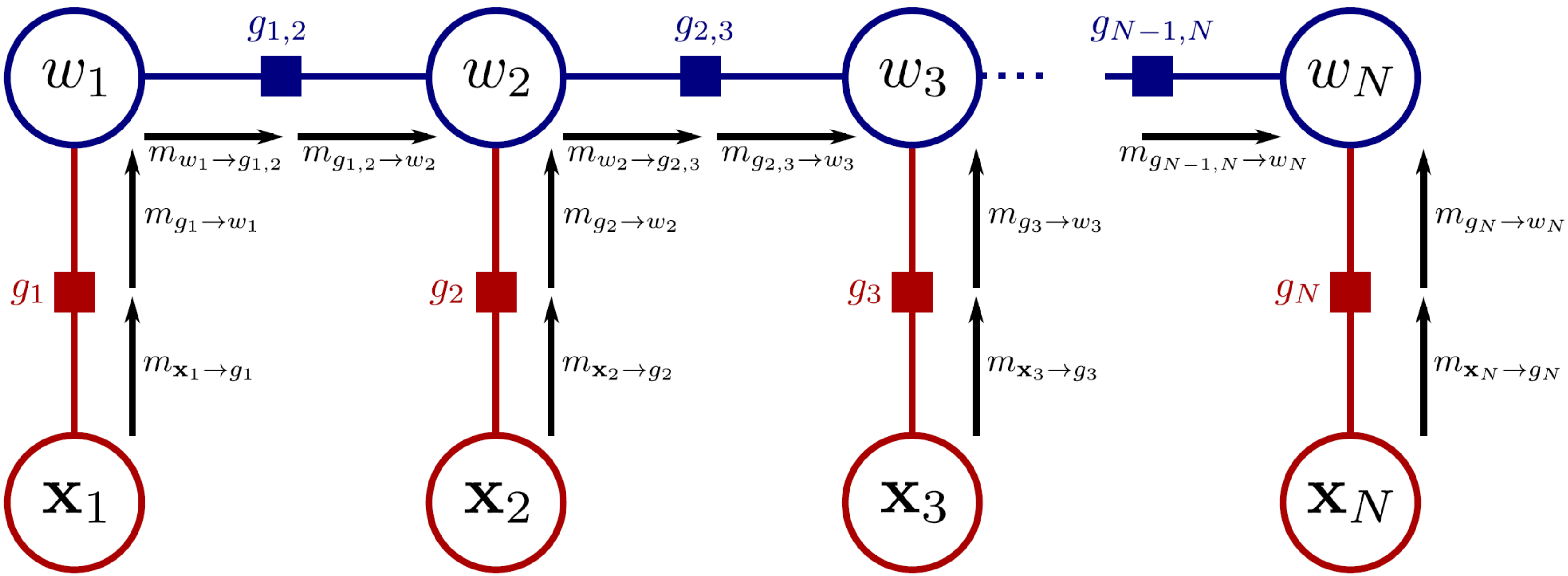
- Takes beliefs from all incoming variables except recipient and uses function g to a belief about recipient

Computing marginal distributions:

- After forward and backward passes, we compute the marginal dists as the product of all incoming messages

$$Pr(\mathbf{z}_p) \propto \prod_{r\in\text{Ne}[p]} \mathbf{m}_{g_r\rightarrow\mathbf{z}_p}$$

Sum product: forward pass

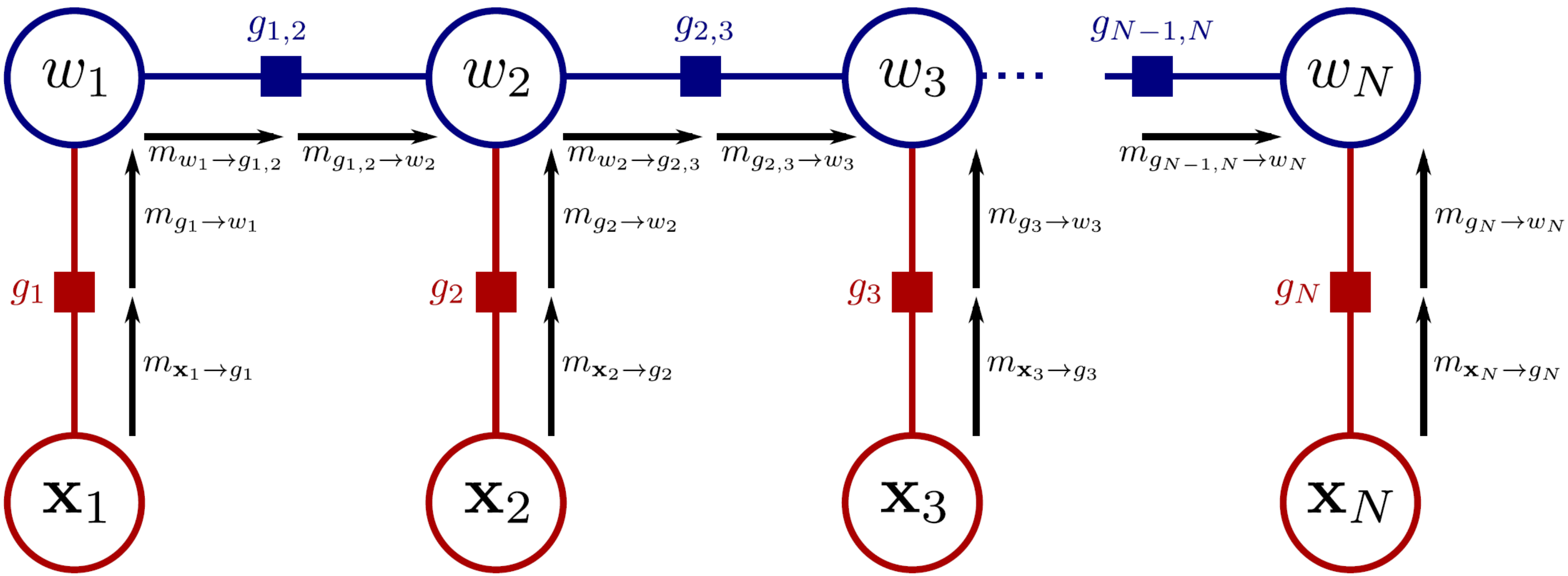


Message from x_1 to g_1:

By rule 2:

$$\mathbf{m}_{\mathbf{x}_1\to g_1} = \delta[\mathbf{x}_1^*]$$

Sum product: forward pass

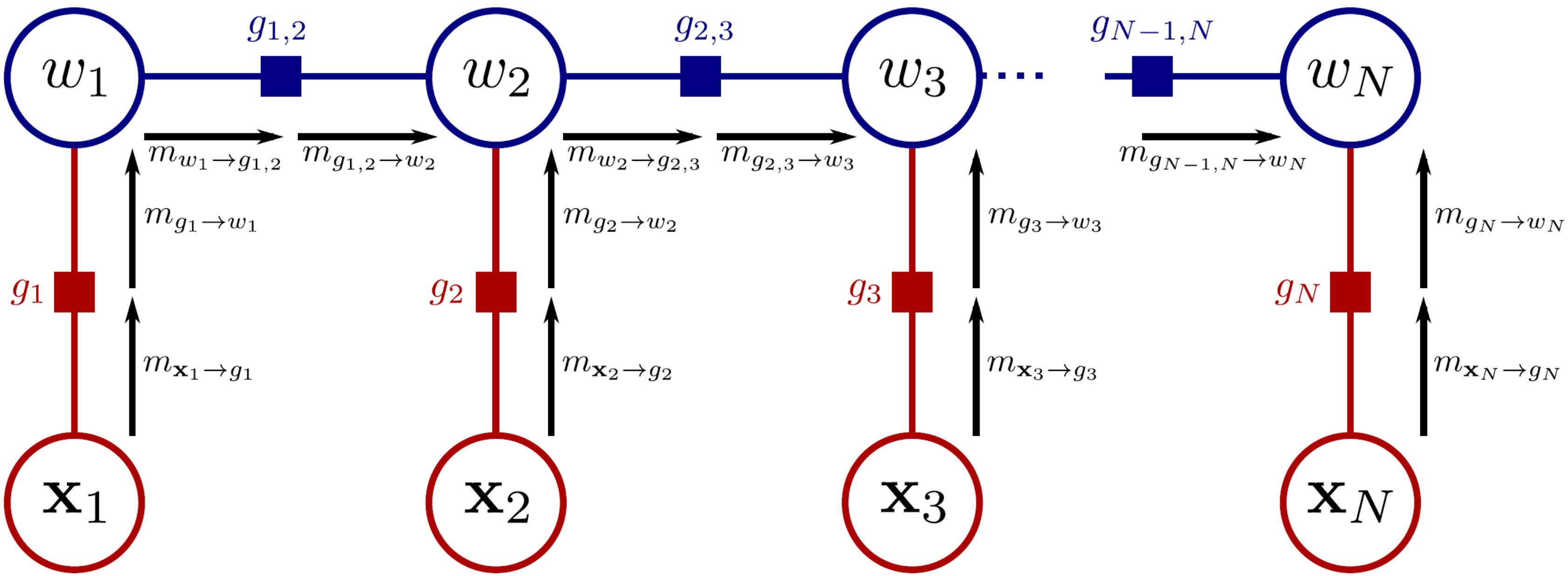


Message from g_1 to w_1:

By rule 3:

$$\mathbf{m}_{g_1\to w_1} = \int Pr(\mathbf{x}_1|w_1)\delta[\mathbf{x}_1^*]d\mathbf{x}_1 = Pr(\mathbf{x}_1 = \mathbf{x}_1^*|w_1)$$

Sum product: forward pass

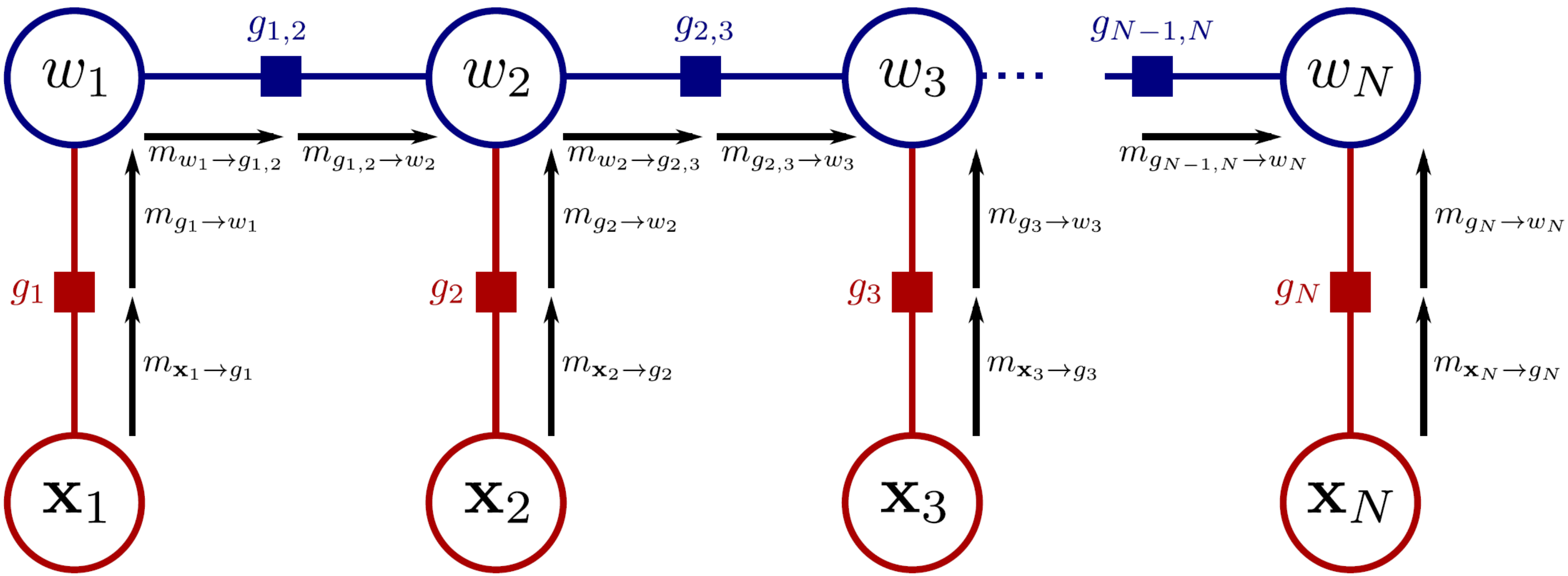


Message from w_1 to $g_{1,2}$:

By rule 1: $$\mathbf{m}_{w_1\rightarrow g_{12}} = Pr(\mathbf{x}_1 = \mathbf{x}_1^* | w_1)$$

(product of all incoming messages)

Sum product: forward pass

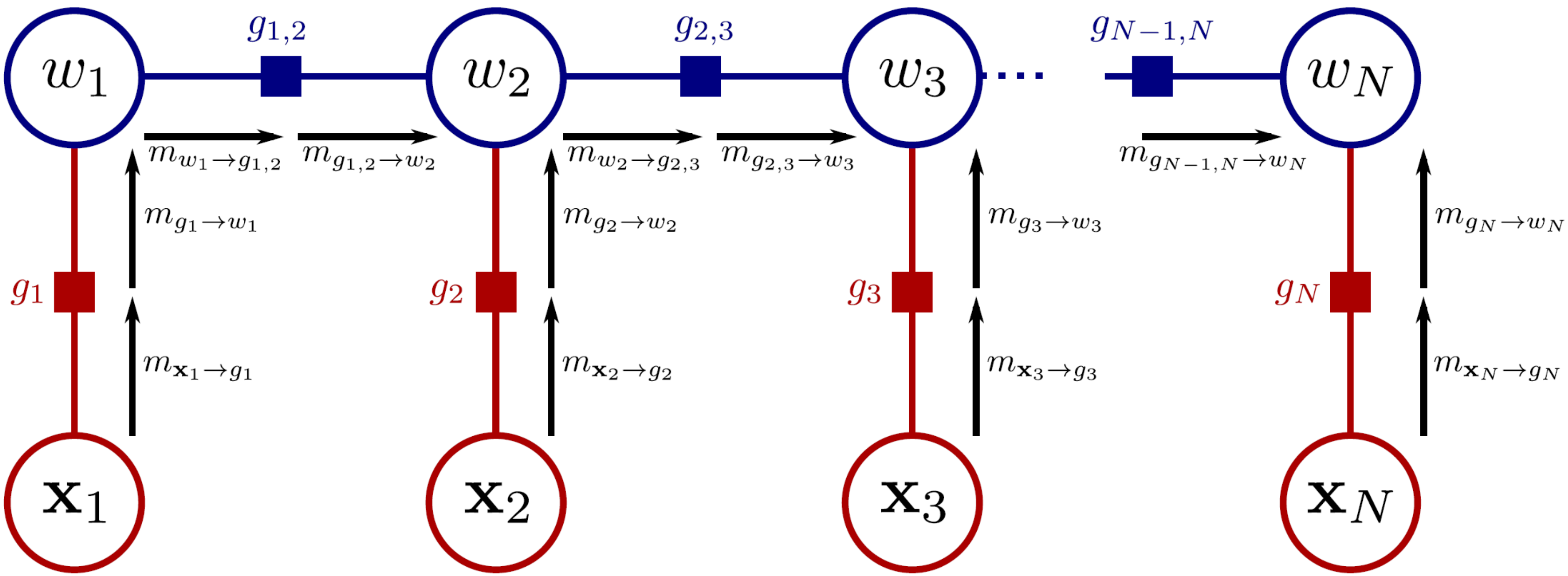


Message from $g_{1,2}$ from w_2:

By rule 3:

$$\mathbf{m}_{g_{12}\to w_2} = \sum_{w_1} Pr(w_2|w_1)Pr(\mathbf{x}_1 = \mathbf{x}_1^*|w_1)$$

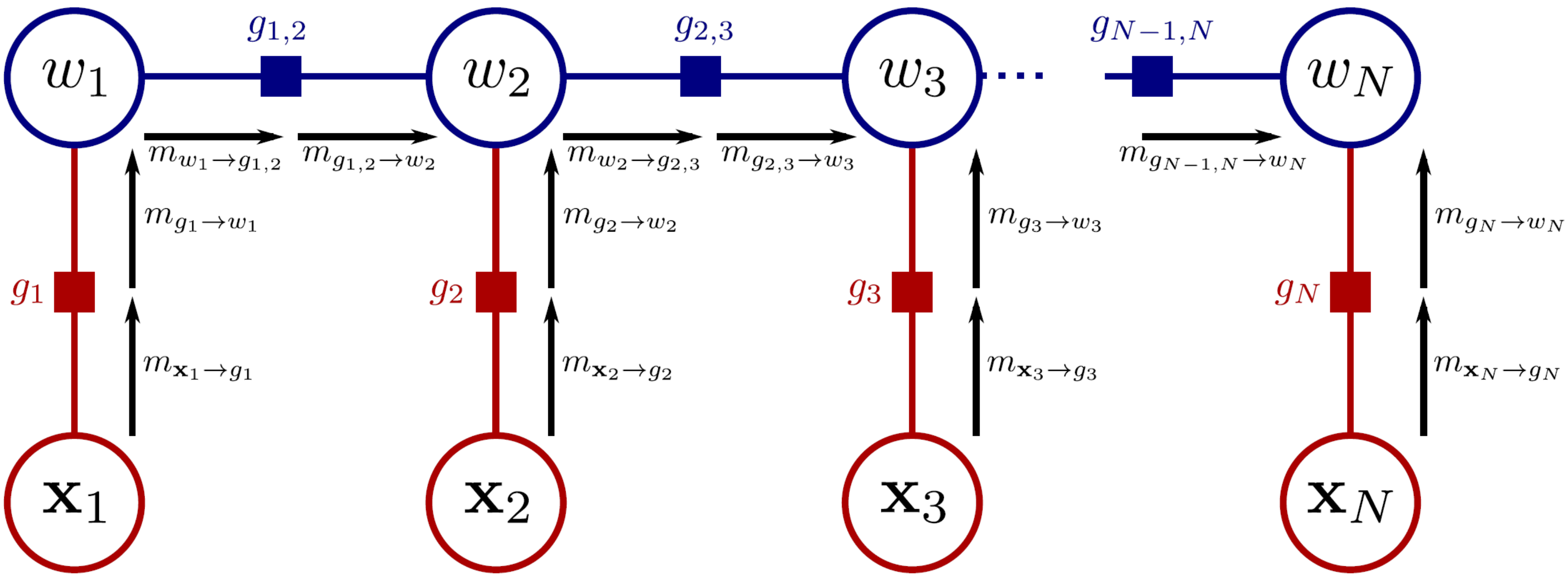
Sum product: forward pass



Messages from x_2 to g_2 and g_2 to w_2:

$$
\begin{aligned}
\mathbf{m}_{\mathbf{x}_2\to g_2} &= \delta[\mathbf{x}_2^*] \\
\mathbf{m}_{g_2\to w_2} &= Pr(\mathbf{x}_2=\mathbf{x}_2^*|w_2)
\end{aligned}
$$

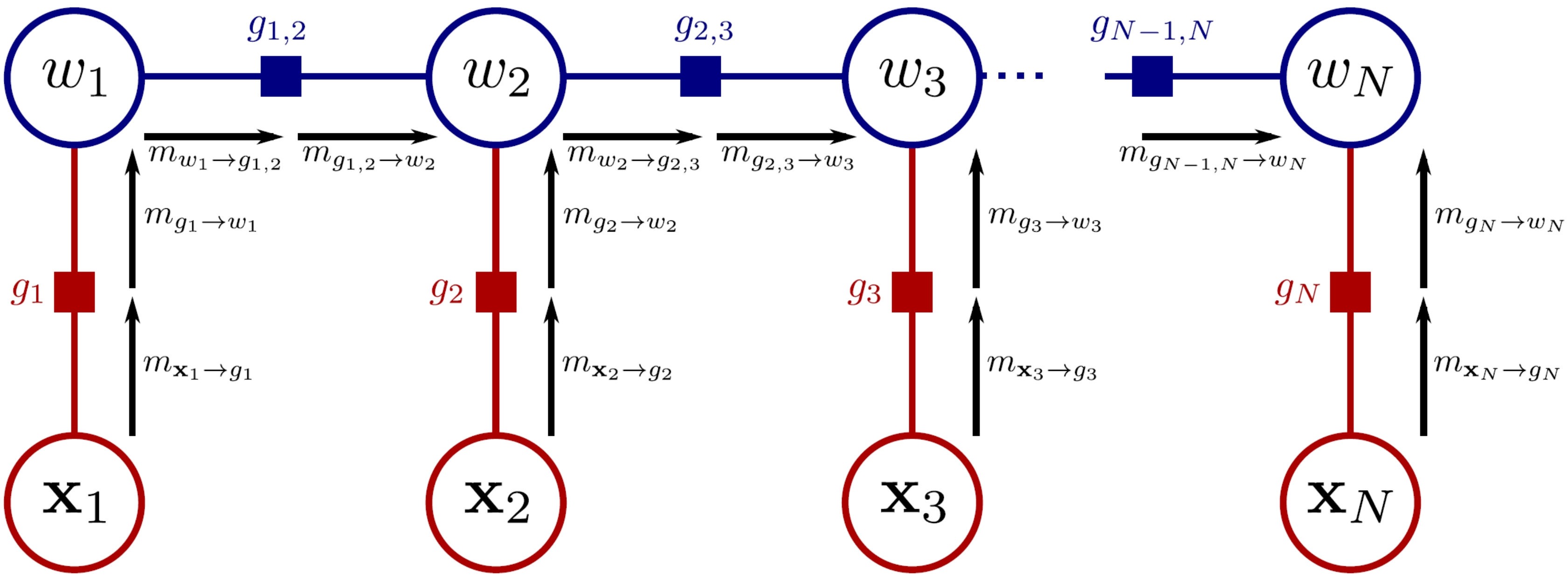
Sum product: forward pass



Message from w_2 to $g_{2,3}$:

$$\mathbf{m}_{w_2\to g_{23}} = Pr(\mathbf{x}_2=\mathbf{x}_2^*|w_2)\underbrace{\sum_{w_1} Pr(w_2|w_1)Pr(\mathbf{x}_1=\mathbf{x}_1^*|w_1)}_{\text{The same recursion as in the forward backward algorithm}}$$

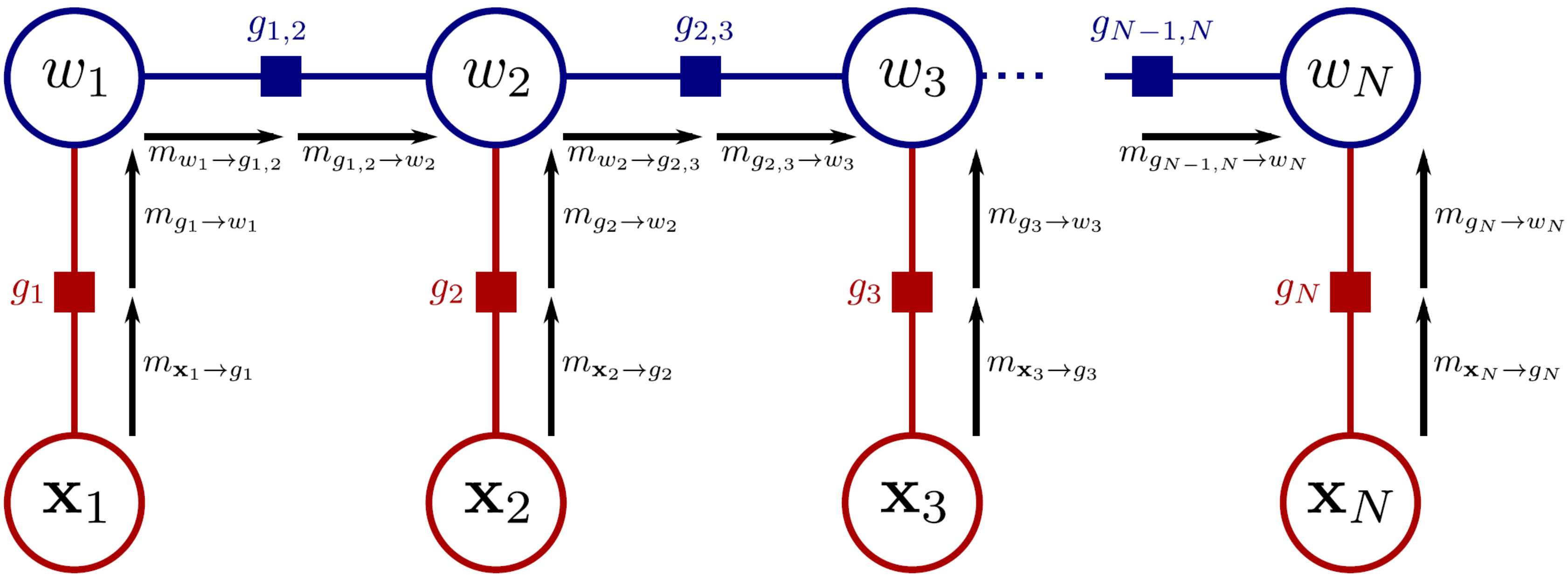
Sum product: forward pass



Message from w_2 to $g_{2,3}$:

$$\mathbf{m}_{w_n\rightarrow g_{n,n+1}} = \mathbf{f}_n[w_n] = Pr(w_n|\mathbf{x}_{1\ldots n})$$

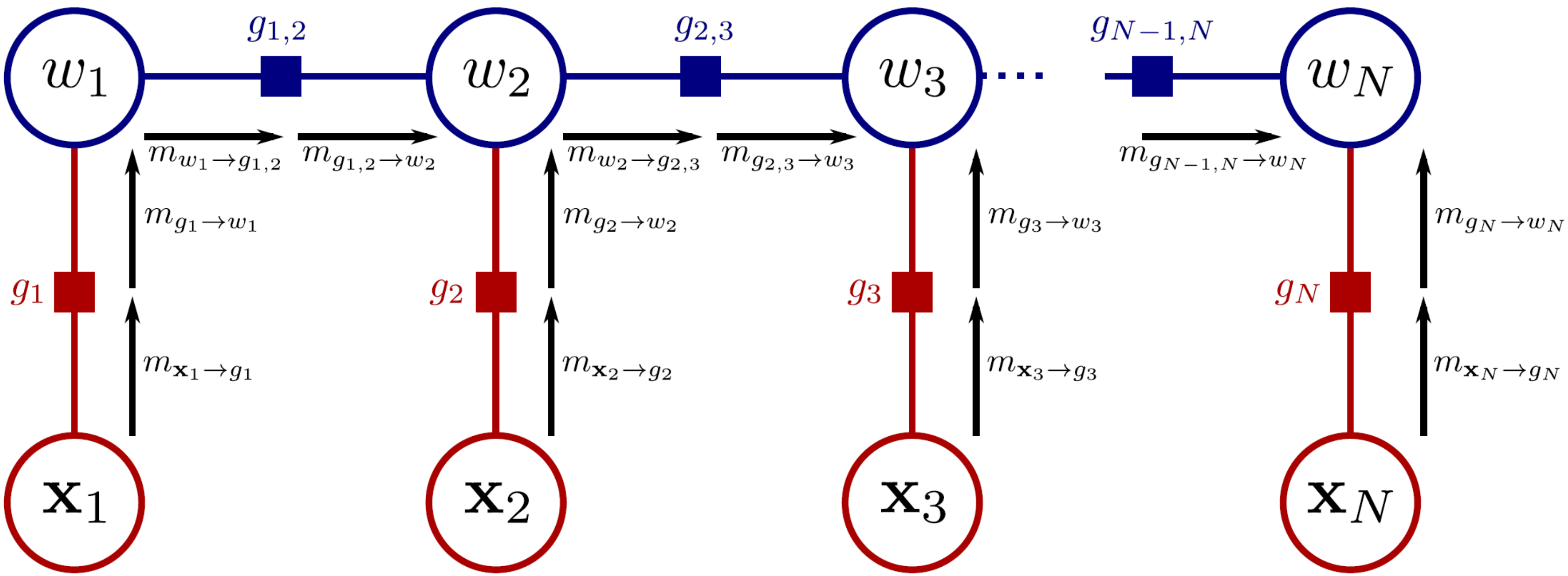
Sum product: backward pass



Message from w_N to $g_{N,N-1}$:

$$\mathbf{m}_{w_N\to g_{N,N-1}} = Pr(\mathbf{x}_N = \mathbf{x}_N^* | w_N)$$

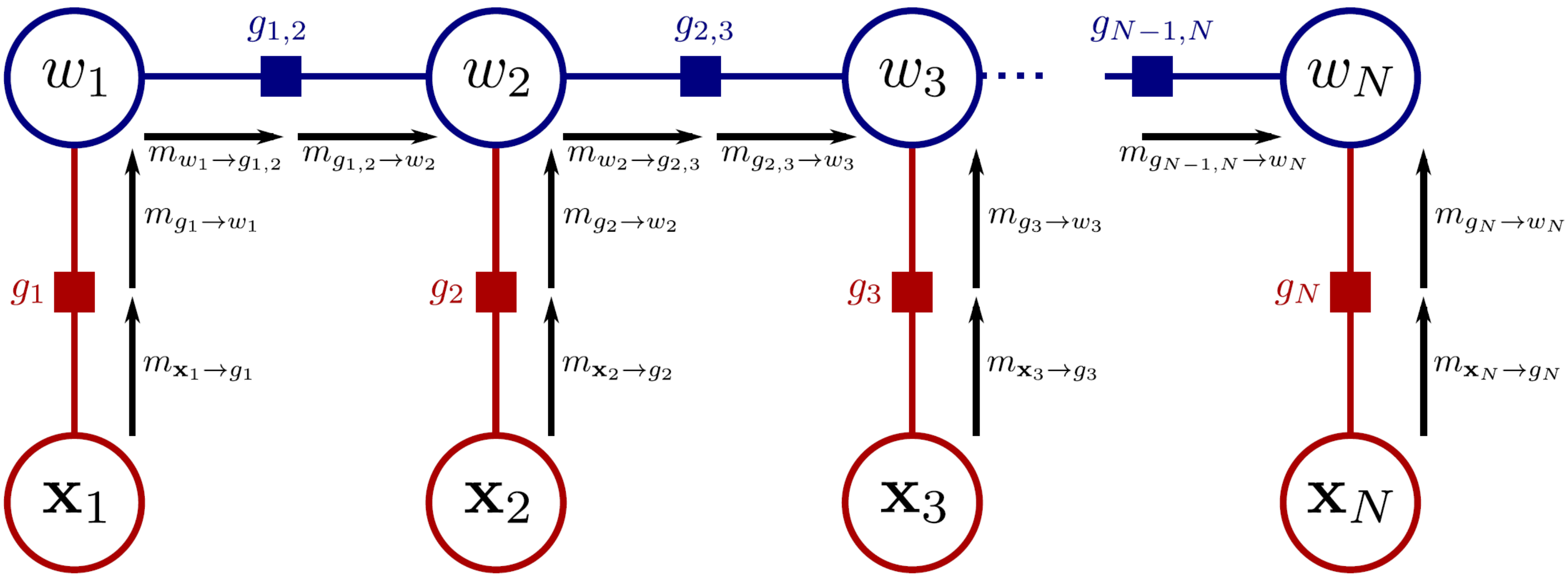
Sum product: backward pass



Message from $g_{N,N-1}$ to w_{N-1}:

$$\mathbf{m}_{g_{N,N-1}\rightarrow w_{N-1}} = \sum_{w_N} Pr(w_N|w_{N-1})Pr(\mathbf{x}_N = \mathbf{x}_N^*|w_N)$$

Sum product: backward pass



Message from $g_{n,n-1}$ to w_{n-1}:

$$\underbrace{\mathbf{m}_{g_{n,n-1} \to w_{n-1}} = \sum_{w_n} Pr(w_n | w_{n-1}) \mathbf{m}_{g_{n+1,n} \to w_n}}_{\text{The same recursion as in the forward backward algorithm}} = \mathbf{b}_{n-1}[w_{n-1}]$$

Sum product: collating evidence

- Marginal distribution is products of all messages at node

$$Pr(\mathbf{w}_n|\mathbf{x}_{1\ldots N}) \propto \prod_{m\in\mathrm{Ne}[n]} \mathbf{m}_{g_m\rightarrow\mathbf{w}_n}$$

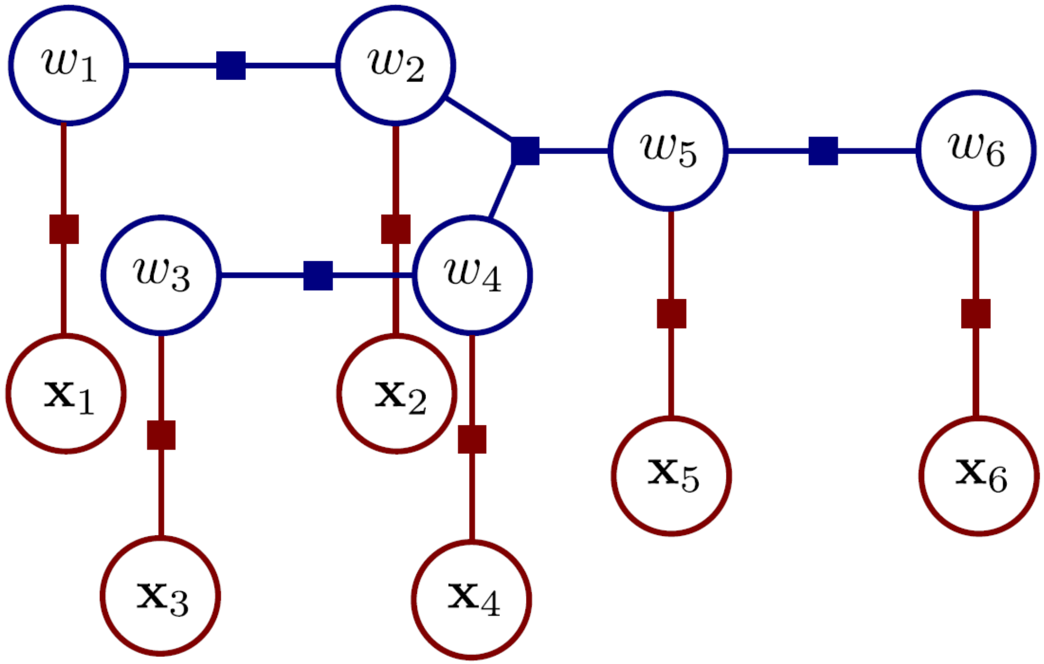
- Proof:

$$\begin{aligned} Pr(w_n|\mathbf{x}_{1\ldots N}) &\propto \mathbf{m}_{g_{n-1,n}\rightarrow w_n}\mathbf{m}_{g_n\rightarrow w_n}\mathbf{m}_{g_{n,n+1}\rightarrow w_n} \\ &= Pr(w_n|\mathbf{x}_{1\ldots n-1})Pr(w_n|\mathbf{x}_n)Pr(w_n|\mathbf{x}_{n+1\ldots N}) \\ &= Pr(w_n|\mathbf{x}_{1\ldots N}) \end{aligned}$$

Structure

- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

Marginal posterior inference for trees

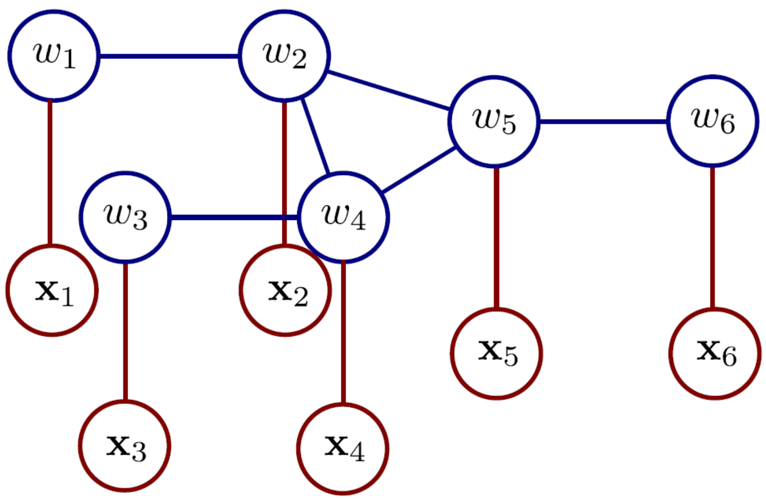


Apply sum-product algorithm to the tree-structured graph.

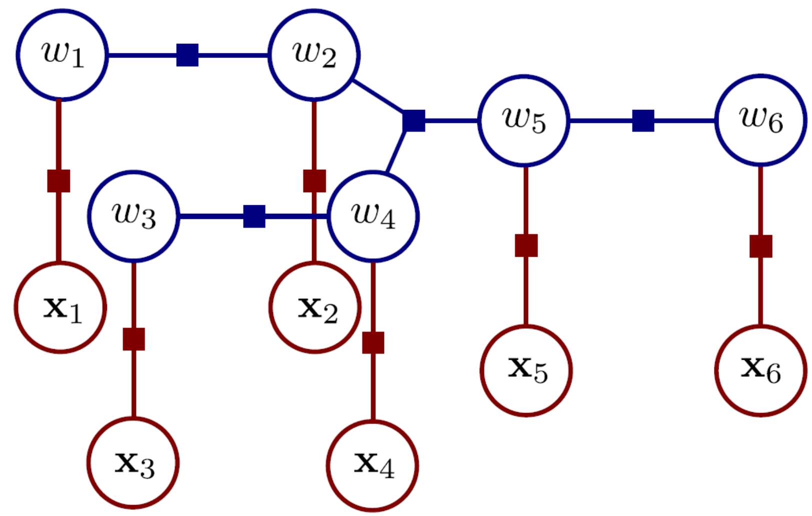
Structure

- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

Tree structured graphs



This graph contains loops



But the associated factor graph has structure of a tree

Can still use Belief Propagation

Learning in chains and trees

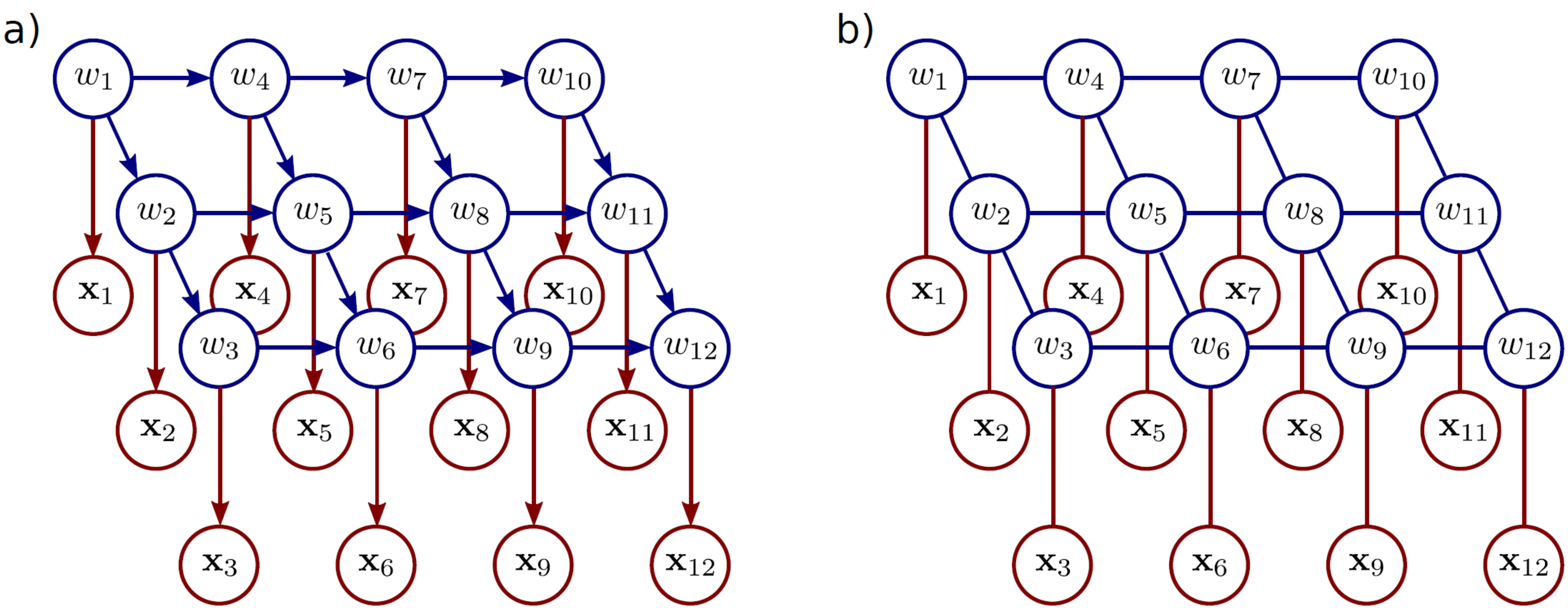
Supervised learning (where we know world states w_n) is relatively easy.

Unsupervised learning (where we do not know world states w_n) is more challenging. Use the EM algorithm:

- E-step – compute posterior marginals over states
- M-step – update model parameters

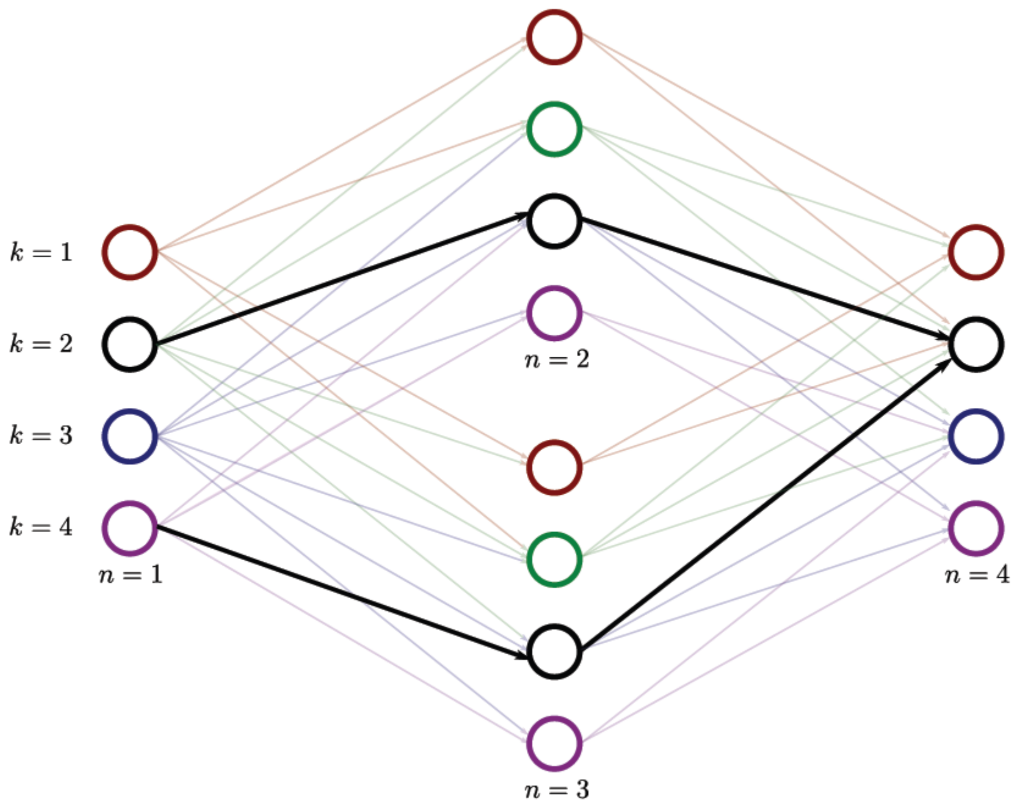
For the chain model (hidden Markov model) this is known as the Baum-Welch algorithm.

Grid-based graphs



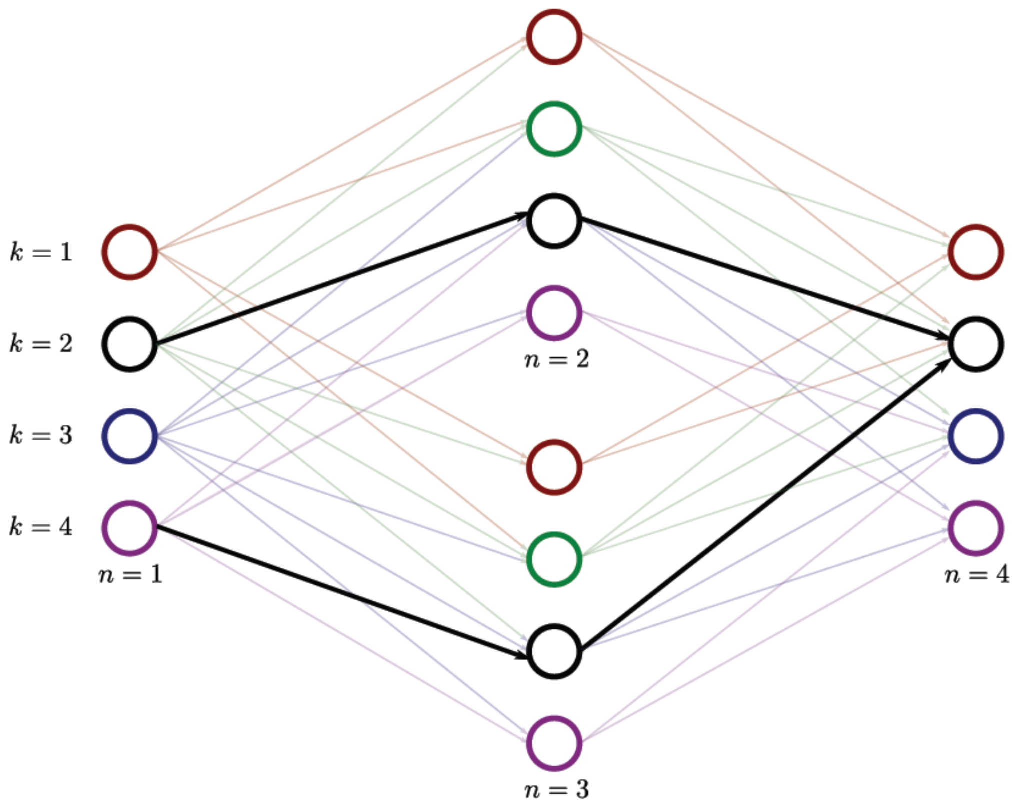
Often in vision, we have one observation associated with each pixel in the image grid.

Why not dynamic programming?



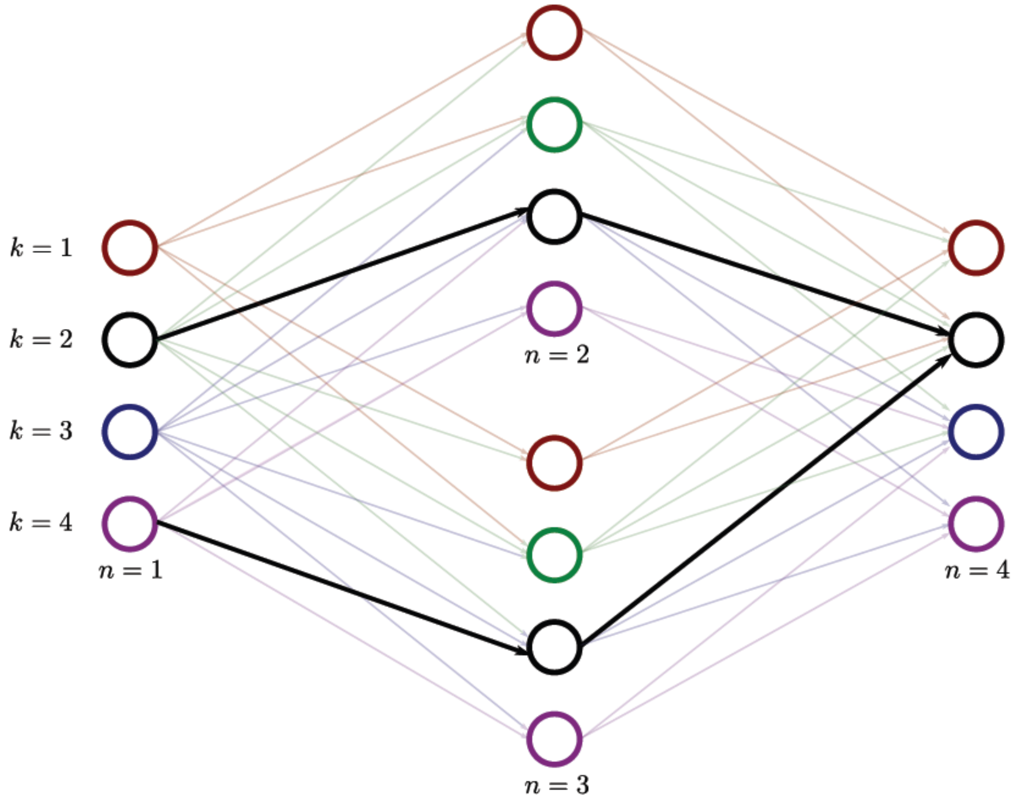
When we trace back from the final node, the paths are not guaranteed to converge.

Why not dynamic programming?



$$
\begin{aligned}
S_{1,k} &= U_1(w_1=k) \\
S_{2,k} &= U_2(w_2=k) + \min_l \left[S_1(w_1=l) + P_2(w_2=k, w_1=l)\right] \\
S_{3,k} &= U_3(w_3=k) + \min_l \left[S_1(w_1=l) + P_2(w_3=k, w_1=l)\right]
\end{aligned}
$$

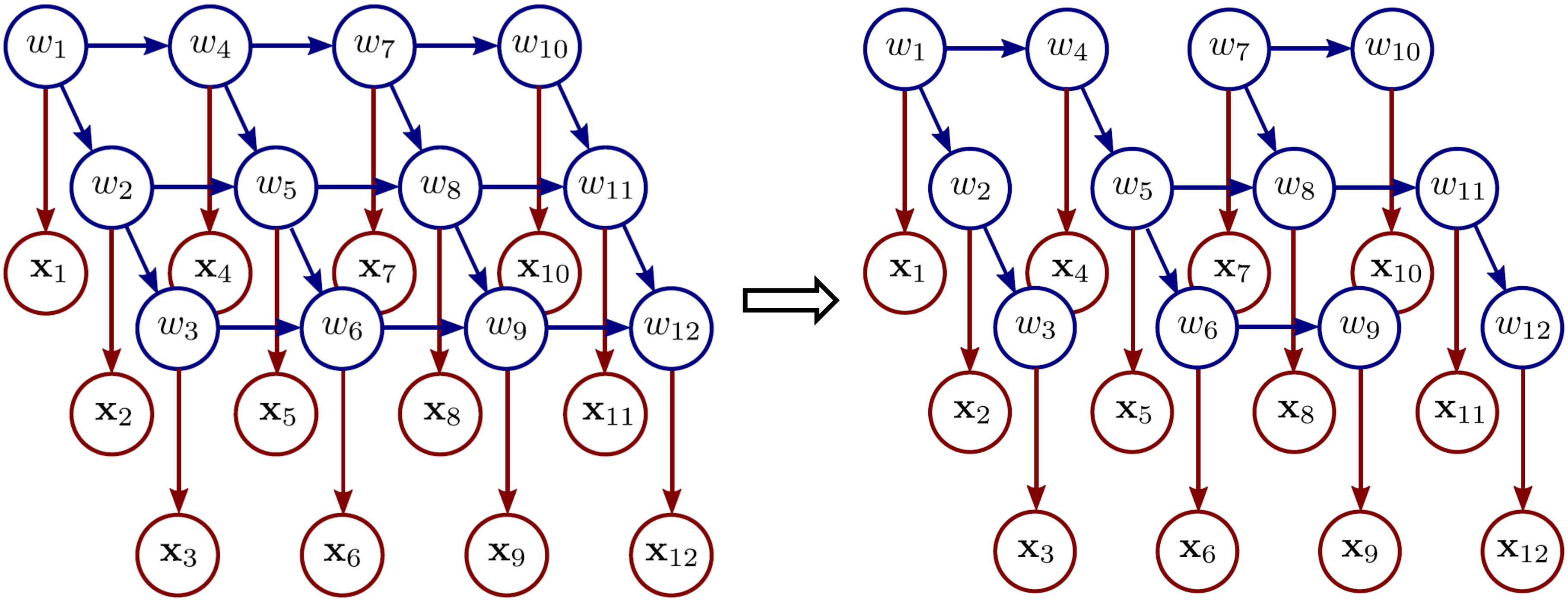
Why not dynamic programming?



But:

$$S_{4,k} \neq U_4(w_k=4) + \min_{l,m}\left[S_2(w_2=l) + S_3(w_3=m) + T(w_4=k, w_2=l, w_3=m)\right]$$

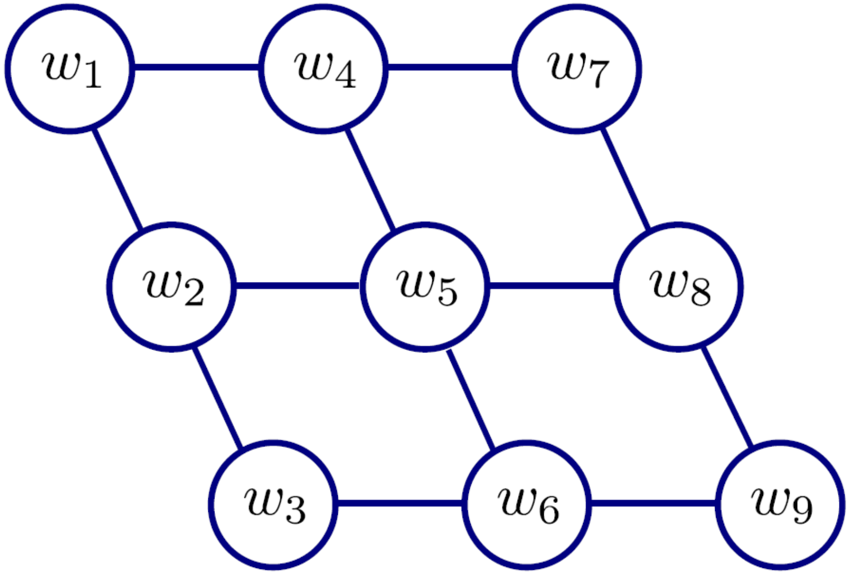
Approaches to inference for grid-based models



1. Prune the graph.

Remove edges until an edge remains

Approaches to inference for grid-based models



2. Combine variables.

Merge variables to form compound variable with more states until what remains is a tree.
Not practical for large grids

Approaches to inference for grid-based models

3. Loopy belief propagation.

   Just apply belief propagation. It is not guaranteed to converge, but in practice it works well.

4. Sampling approaches

   Draw samples from the posterior (easier for directed models)

5. Other approaches

   - Tree-reweighted message passing
   - Graph cuts

Structure

- Chain and tree models
- MAP inference in chain models
- MAP inference in tree models
- Maximum marginals in chain models
- Maximum marginals in tree models
- Models with loops
- Applications

Gesture Tracking

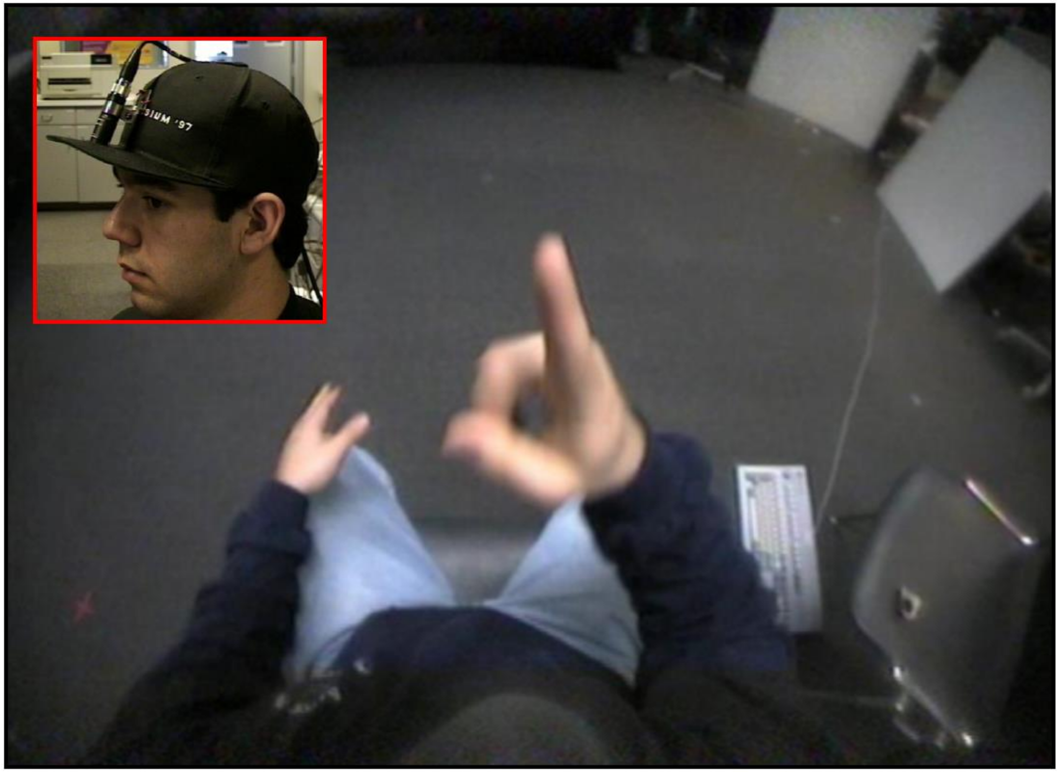


Figure 10.16 Gesture tracking from Starner *et al.* (1998). A camera was mounted on a baseball cap looking down at the users hands (inset). The camera image (main figure) was used to track the hands in a HMM based system that could accurately classify a 40 word lexicon and worked in real time. Each word was associated with four states in the HMM. The system was based on a compact description of the hand position and orientation within each frame. Adapted from Starner *et al.* (1998) ©1998 Springer.

Stereo vision

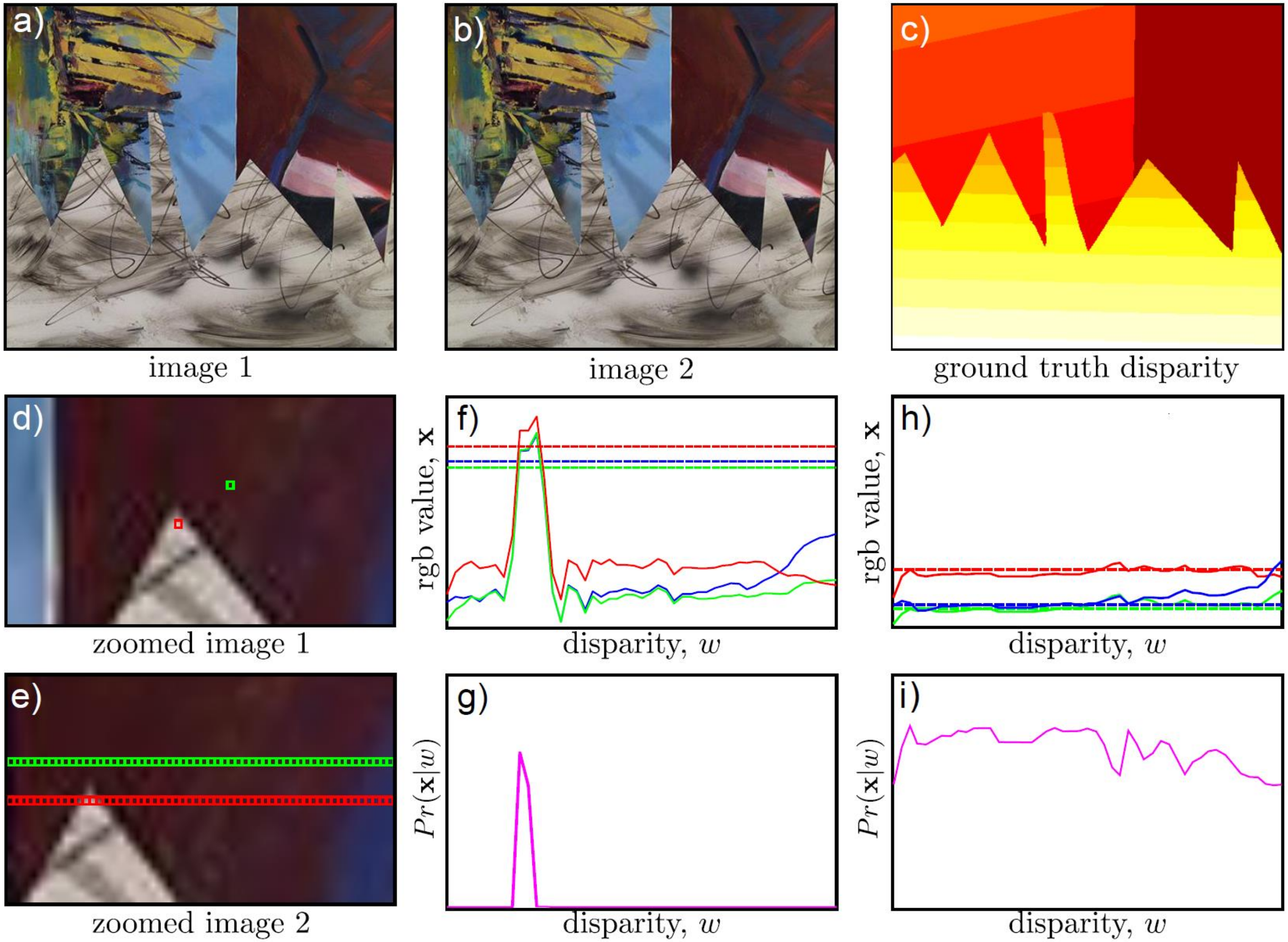
image 1 image 2 ground truth disparity

- Two image taken from slightly different positions
- Matching point in image 2 is on same scanline as image 1
- Horizontal offset is called disparity
- Disparity is inversely related to depth
- Goal – infer disparities $w_{m,n}$ at pixel m,n from images $x^{(1)}$ and $x^{(2)}$

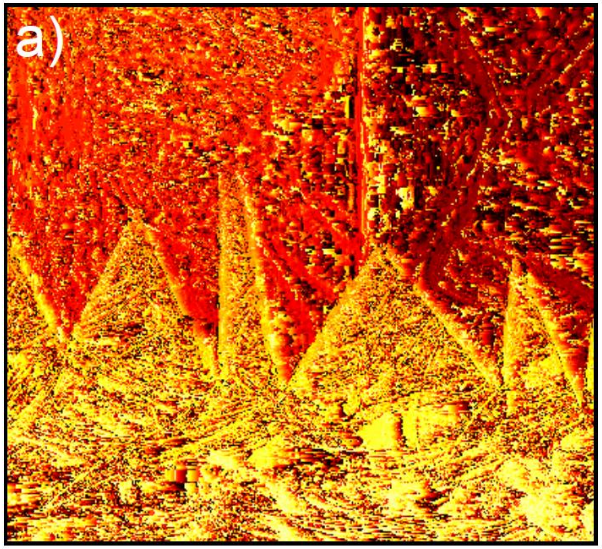
Use likelihood:

$$Pr(\mathbf{x}^{(1)}_{m,n}|w_{m,n}=k) = \text{Norm}_{\mathbf{x}^{(1)}_{m,n}}\left[\mathbf{x}^{(2)}_{m,n+k}, \sigma^2\mathbf{I}\right]$$

Stereo vision



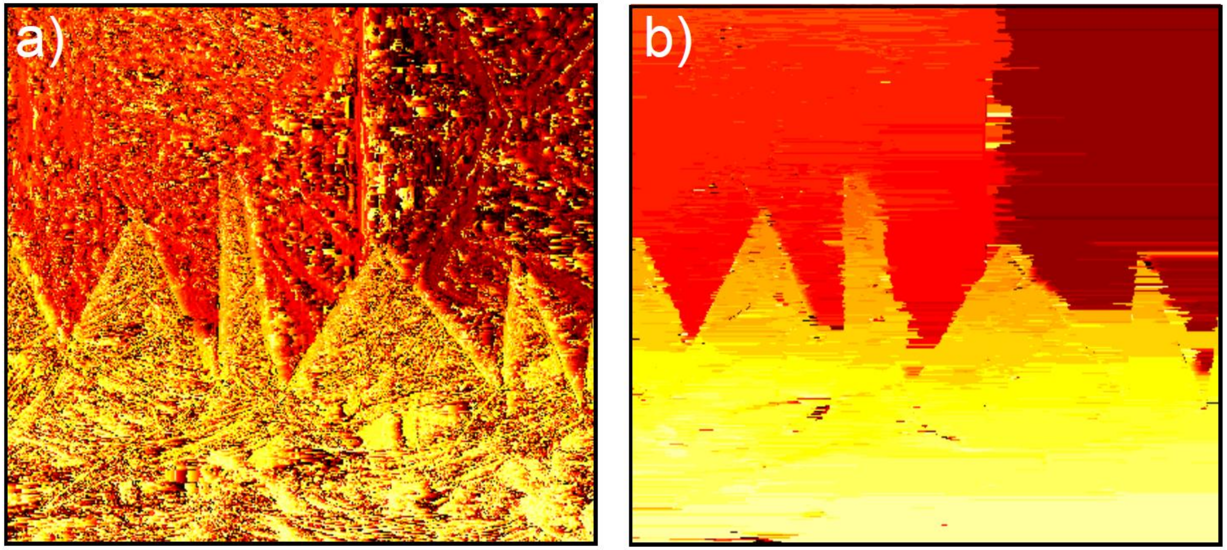
Stereo vision



1. Independent pixels

$$Pr(\mathbf{w}) = \prod_{m=1}^{M} Pr(\mathbf{w}_m)$$

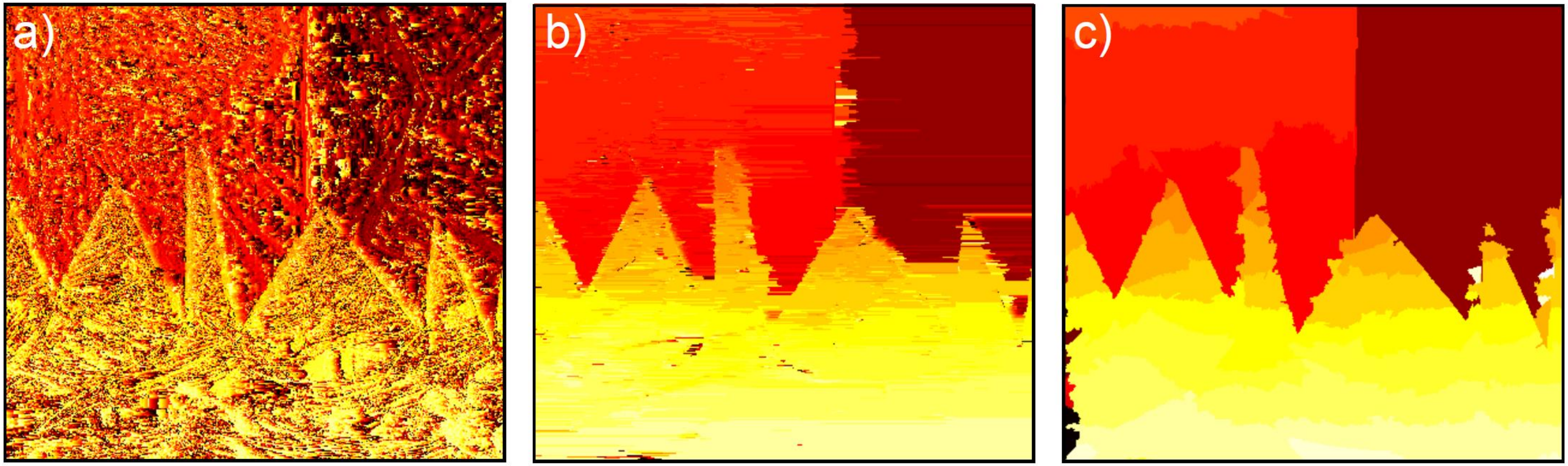
Stereo vision



2. Scanlines as chain model (hidden Markov model)

$$Pr(\mathbf{w}_m) = Pr(w_{m,1}) \prod_{n=1}^{N} Pr(w_{m,n}|w_{m,n-1})$$

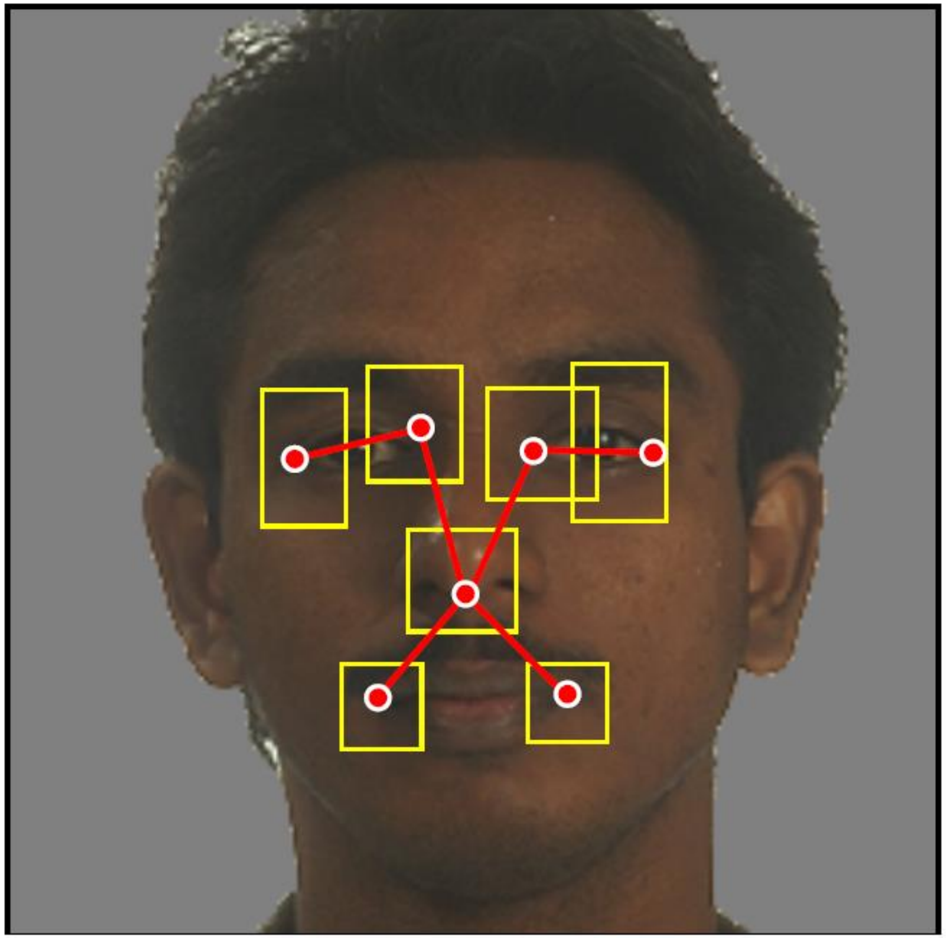
Stereo vision



3. Pixels organized as tree (from Veksler 2005)

Pictorial Structures

Figure 10.19 Pictorial structure. This face model consists of seven parts (red dots) which are connected together in a tree-like structure (red lines). The possible positions of each part are indicated by the yellow boxes. Although each part can take several hundred pixel positions, the MAP positions can be inferred efficiently by exploiting the tree-structure of the graph using a dynamic programming approach. Localizing facial features is a common element of many face recognition pipelines.



Pictorial Structures

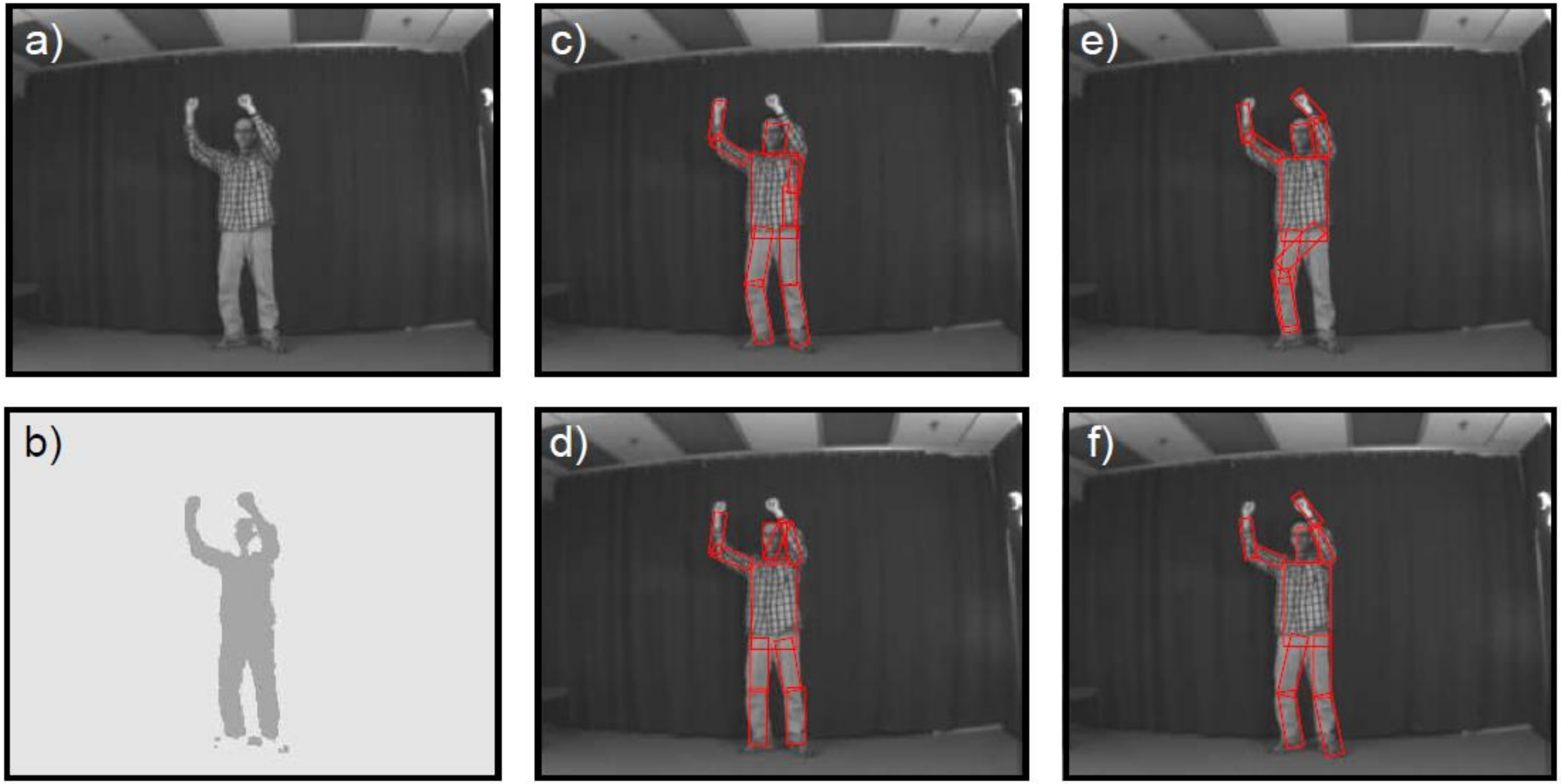


Figure 10.20 Pictorial structure for human body. a) Original image. b) After background subtraction. c-f) Four samples from the posterior distribution over part positions. Each part position is represented by a rectangle of fixed aspect ration and characterized by its position, size and angle. Adapted from Felzenszwalb & Huttenlocher (2005). ©2005 Springer.

Segmentation

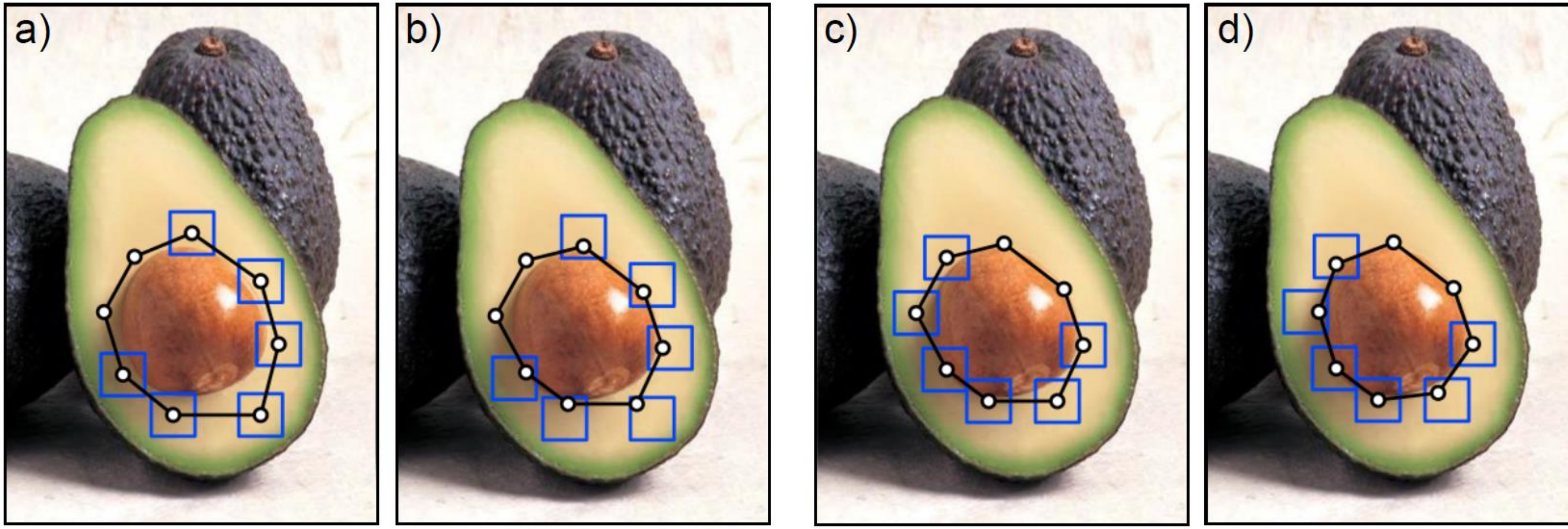


Figure 10.21 Segmentation using snakes. a) Two points are fixed, but the remaining points can take any position within their respective boxes. The posterior distribution favours positions that are on image contours (due to the likelihood term) and positions that are close to other points (due to the pairwise connections). b) Results of inference. c) Two other points are considered fixed. d) Result of inference. In this way, a closed contour in the image is identified. Adapted from Felzenszwalb & Zabih (2011). ©2011 IEEE.

Conclusion

- For the special case of chains and trees we can perform MAP inference and compute marginal posteriors efficiently.

- Unfortunately, many vision problems are defined on pixel grid – this requires special methods