Inference and the sum-product algorithm

Guillaume Obozinski

Ecole des Ponts - ParisTech

INIT/AERFAI Summer school on Machine Learning
Benicàssim, June 26th 2017

Probabilistic inference

Given a discrete Gibbs model of the form:

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C),$$

where $\mathcal{C}$ is the set of cliques of the graph, inference is the problem of computation of several related quantities of interest:

- Computation of the marginal $p(x_i)$ or more generally, $p(x_C)$.

Probabilistic inference

Given a discrete Gibbs model of the form:

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C),$$

where $\mathcal{C}$ is the set of cliques of the graph, inference is the problem of computation of several related quantities of interest:

- Computation of the marginal $p(x_i)$ or more generally, $p(x_C)$.
- Computation of the partition function Z

Probabilistic inference

Given a discrete Gibbs model of the form:

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C),$$

where $\mathcal{C}$ is the set of cliques of the graph, inference is the problem of computation of several related quantities of interest:

- Computation of the marginal $p(x_i)$ or more generally, $p(x_C)$.
- Computation of the partition function Z
- Computation of the conditional marginal $p(x_i | X_j = x_j, X_k = x_k)$

Probabilistic inference

Given a discrete Gibbs model of the form:

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C),$$

where $\mathcal{C}$ is the set of cliques of the graph, inference is the problem of computation of several related quantities of interest:

- Computation of the marginal $p(x_i)$ or more generally, $p(x_C)$.
- Computation of the partition function Z
- Computation of the conditional marginal $p(x_i | X_j = x_j, X_k = x_k)$

Inference is actually necessary

- For the computation of the gradient of the likelihood of a model
- Computation of the expected value of the log-likelihood of an exponential family at step E of the EM algorithm (for example for the HMM)

Hardness of inference

⚠ Problem: In general, the inference problem is NP-hard.

Hardness of inference

⚠ Problem: In general, the inference problem is NP-hard.

For trees

the inference problem is efficient: its cost is linear in n.

Hardness of inference

⚠ Problem: In general, the inference problem is NP-hard.

For trees

the inference problem is efficient: its cost is linear in n.

For "tree-like" graphs

we use the *Junction Tree Algorithm* which enables us to bring the situation back to that of a tree.

Hardness of inference

⚠ Problem: In general, the inference problem is NP-hard.

For trees

the inference problem is efficient: its cost is linear in n.

For "tree-like" graphs

we use the *Junction Tree Algorithm* which enables us to bring the situation back to that of a tree.

In the general case

There are several methods for approximate/inexact inference.

Hardness of inference

⚠ Problem: In general, the inference problem is NP-hard.

For trees

the inference problem is efficient: its cost is linear in n.

For "tree-like" graphs

we use the *Junction Tree Algorithm* which enables us to bring the situation back to that of a tree.

In the general case

There are several methods for approximate/inexact inference.

Rest of this part of the lecture

We will focus in the rest of this part to inference for undirected graphical models that have cliques of size 1 and 2.

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\} \in E} \psi_{i,j}(x_i, x_j)$$

Rest of this part of the lecture

We will focus in the rest of this part to inference for undirected graphical models that have cliques of size 1 and 2.

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\} \in E} \psi_{i,j}(x_i, x_j)$$

- The general case is not much more difficult but requires the concept of *factor graph*.

Rest of this part of the lecture

We will focus in the rest of this part to inference for undirected graphical models that have cliques of size 1 and 2.

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\} \in E} \psi_{i,j}(x_i, x_j)$$

- The general case is not much more difficult but requires the concept of *factor graph*.
- To perform inference on a directed graph, we will simply moralize the graph and apply the inference algorithm for undirected graphs

Rest of this part of the lecture

We will focus in the rest of this part to inference for undirected graphical models that have cliques of size 1 and 2.

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\} \in E} \psi_{i,j}(x_i, x_j)$$

- The general case is not much more difficult but requires the concept of *factor graph*.
- To perform inference on a directed graph, we will simply moralize the graph and apply the inference algorithm for undirected graphs
- We will focus on the cases where inference can be done efficiently using *dynamic programming*, that is:
  - in the chain case
  - in the tree case

Inference on a chain

We define X_i a random variable, taking value in $\{1, \dots, K\}$, $i \in V = \{1, \dots, n\}$ with joint distribution

$$p(x) = \frac{1}{Z} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i)$$

Inference on a chain

We define X_i a random variable, taking value in $\{1, \ldots, K\}$, $i \in V = \{1, \ldots, n\}$ with joint distribution

$$p(x) = \frac{1}{Z} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i)$$

Computing

$$p(x_j) = \sum_{x_{V \setminus \{j\}}} p(x_1, \ldots, x_n)$$

has a priori a complexity of $O(K^n)$.

Inference on a chain: a dynamic program

$p(x_j)$

Inference on a chain: a dynamic program

$$
\begin{aligned}
& p(x_j) \\
= \quad & \frac{1}{Z} \sum_{x_{V\setminus\{j\}}} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i)
\end{aligned}
$$

Inference on a chain: a dynamic program

$$
\begin{aligned}
& p(x_j) \\
= \quad & \frac{1}{Z} \sum_{x_{V\setminus\{j\}}} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i) \\
= \quad & \frac{1}{Z} \sum_{x_{V\setminus\{j\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_n(x_n)\, \psi_{n-1,n}(x_{n-1}, x_n)
\end{aligned}
$$

Inference on a chain: a dynamic program

$$
\begin{aligned}
& p(x_j) \\
= \; & \frac{1}{Z} \sum_{x_{V\backslash\{j\}}} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i) \\
= \; & \frac{1}{Z} \sum_{x_{V\backslash\{j\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_n(x_n)\, \psi_{n-1,n}(x_{n-1}, x_n) \\
= \; & \frac{1}{Z} \sum_{x_{V\backslash\{j,n\}}} \sum_{x_n} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n)
\end{aligned}
$$

Inference on a chain: a dynamic program

$$
\begin{aligned}
& p(x_j) \\
= \;& \frac{1}{Z} \sum_{x_{V\setminus\{j\}}} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i) \\
= \;& \frac{1}{Z} \sum_{x_{V\setminus\{j\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_n(x_n)\, \psi_{n-1,n}(x_{n-1}, x_n) \\
= \;& \frac{1}{Z} \sum_{x_{V\setminus\{j,n\}}} \sum_{x_n} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n) \\
= \;& \frac{1}{Z} \sum_{x_{V\setminus\{j,n\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \underbrace{\sum_{x_n} \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n)}_{\mu_{n\to n-1}(x_{n-1})}
\end{aligned}
$$

Inference on a chain: a dynamic program

$$= \frac{1}{Z} \sum_{x_{V\setminus\{j,n\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \underbrace{\sum_{x_n} \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n)}_{\mu_{n\to n-1}(x_{n-1})}$$

Inference on a chain: a dynamic program

$$= \frac{1}{Z}\sum_{x_{V\backslash\{j,n\}}}\prod_{i=1}^{n-1}\psi_i(x_i)\prod_{i=2}^{n-1}\psi_{i-1,i}(x_{i-1},x_i)\underbrace{\sum_{x_n}\psi_n(x_n)\psi_{n-1,n}(x_{n-1},x_n)}_{\mu_{n\to n-1}(x_{n-1})}$$

$$= \frac{1}{Z}\sum_{x_{V\backslash\{j,n,n-1\}}}\prod_{i=1}^{n-2}\psi_i(x_i)\prod_{i=2}^{n-2}\psi_{i-1,i}(x_{i-1},x_i)\times$$

$$\times\underbrace{\sum_{x_{n-1}}\psi_{n-1}(x_{n-1})\psi_{n-2,n-1}(x_{n-2},x_{n-1})\mu_{n\to n-1}(x_{n-1})}_{\mu_{n-1\to n-2}(x_{n-2})}$$

Inference on a chain: a dynamic program

$$= \frac{1}{Z} \sum_{x_{V\setminus\{j,n\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \underbrace{\sum_{x_n} \psi_n(x_n)\psi_{n-1,n}(x_{n-1}, x_n)}_{\mu_{n\to n-1}(x_{n-1})}$$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{j,n,n-1\}}} \prod_{i=1}^{n-2} \psi_i(x_i) \prod_{i=2}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i) \times$$

$$\times \underbrace{\sum_{x_{n-1}} \psi_{n-1}(x_{n-1})\psi_{n-2,n-1}(x_{n-2}, x_{n-1})\mu_{n\to n-1}(x_{n-1})}_{\mu_{n-1\to n-2}(x_{n-2})}$$

Descending messages:

$$\boxed{\mu_{i\to i-1}(x_{i-1}) = \sum_{x_i} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_i(x_i)\, \mu_{i+1\to i}(x_i)}$$

Inference on a chain: ascending messages

Now marginalizing from the left, $p(x_j) =$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{j,n,n-1\}}} \prod_{i=1}^{n-2} \psi_i(x_i) \prod_{i=2}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i)\, \mu_{n-1\to n-2}(x_{n-2})$$

Inference on a chain: ascending messages

Now marginalizing from the left, $p(x_j) =$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{j,n,n-1\}}} \prod_{i=1}^{n-2} \psi_i(x_i) \prod_{i=2}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i)\, \mu_{n-1\to n-2}(x_{n-2})$$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{1,j,n,n-1\}}} \underbrace{\sum_{x_1} \psi_1(x_1)\psi_{1,2}(x_1, x_2)}_{\mu_{1\to 2}(x_2)} \times \prod_{i=2}^{n-2} \psi_i(x_i) \prod_{i=3}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i) \times$$

$$\times \mu_{n-1\to n-2}(x_{n-2})$$

Inference on a chain: ascending messages

Now marginalizing from the left, $p(x_j) =$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{j,n,n-1\}}} \prod_{i=1}^{n-2} \psi_i(x_i) \prod_{i=2}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i)\, \mu_{n-1 \to n-2}(x_{n-2})$$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{1,j,n,n-1\}}} \underbrace{\sum_{x_1} \psi_1(x_1)\psi_{1,2}(x_1, x_2)}_{\mu_{1 \to 2}(x_2)} \times \prod_{i=2}^{n-2} \psi_i(x_i) \prod_{i=3}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i) \times$$

$$\times \mu_{n-1 \to n-2}(x_{n-2})$$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{1,j,n,n-1\}}} \mu_{1 \to 2}(x_2) \prod_{i=2}^{n-2} \psi_i(x_i) \prod_{i=3}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i)\, \mu_{n-1 \to n-2}(x_{n-2})$$

Inference on a chain: ascending messages

Now marginalizing from the left, $p(x_j) =$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{j,n,n-1\}}} \prod_{i=1}^{n-2} \psi_i(x_i) \prod_{i=2}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i)\, \mu_{n-1\to n-2}(x_{n-2})$$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{1,j,n,n-1\}}} \underbrace{\sum_{x_1} \psi_1(x_1)\psi_{1,2}(x_1, x_2)}_{\mu_{1\to 2}(x_2)} \times \prod_{i=2}^{n-2} \psi_i(x_i) \prod_{i=3}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i) \times$$

$$\times \mu_{n-1\to n-2}(x_{n-2})$$

$$= \frac{1}{Z} \sum_{x_{V\setminus\{1,j,n,n-1\}}} \mu_{1\to 2}(x_2) \prod_{i=2}^{n-2} \psi_i(x_i) \prod_{i=3}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i)\, \mu_{n-1\to n-2}(x_{n-2})$$

Ascending messages:

$$\boxed{\mu_{i\to i+1}(x_{i+1}) = \sum_{x_i} \mu_{i-1\to i}(x_i)\, \psi_i(x_i)\, \psi_{i,i+1}(x_i, x_{i+1})}$$

Ascending and descending messages

$$\boxed{\mu_{i \to i-1}(x_{i-1}) = \sum_{x_i} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_i(x_i)\, \mu_{i+1 \to i}(x_i)}$$

$$\boxed{\mu_{i \to i+1}(x_{i+1}) = \sum_{x_i} \mu_{i-1 \to i}(x_i)\, \psi_i(x_i)\, \psi_{i,i+1}(x_i, x_{i+1})}$$

Ascending and descending messages

$$\boxed{\mu_{i\to i-1}(x_{i-1}) = \sum_{x_i} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_i(x_i)\, \mu_{i+1\to i}(x_i)}$$

$$\boxed{\mu_{i\to i+1}(x_{i+1}) = \sum_{x_i} \mu_{i-1\to i}(x_i)\, \psi_i(x_i)\, \psi_{i,i+1}(x_i, x_{i+1})}$$

Note that node i always receives a message which is a function of x_i.

Ascending and descending messages

$$\boxed{\mu_{i\to i-1}(x_{i-1}) = \sum_{x_i} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_i(x_i)\, \mu_{i+1\to i}(x_i)}$$

$$\boxed{\mu_{i\to i+1}(x_{i+1}) = \sum_{x_i} \mu_{i-1\to i}(x_i)\, \psi_i(x_i)\, \psi_{i,i+1}(x_i, x_{i+1})}$$

Note that node i always receives a message which is a function of x_i.

After propagating all the messages, we have

$$p(x_j) = \frac{1}{Z}\, \mu_{j-1\to j}(x_j)\, \psi_j(x_j)\, \mu_{j+1\to j}(x_j).$$

Ascending and descending messages

$$\boxed{\mu_{i \to i-1}(x_{i-1}) = \sum_{x_i} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_i(x_i)\, \mu_{i+1 \to i}(x_i)}$$

$$\boxed{\mu_{i \to i+1}(x_{i+1}) = \sum_{x_i} \mu_{i-1 \to i}(x_i)\, \psi_i(x_i)\, \psi_{i,i+1}(x_i, x_{i+1})}$$

Note that node i always receives a message which is a function of x_i.

After propagating all the messages, we have

$$p(x_j) = \frac{1}{Z}\, \mu_{j-1 \to j}(x_j)\, \psi_j(x_j)\, \mu_{j+1 \to j}(x_j).$$

How do we compute Z?

Ascending and descending messages

$$\boxed{\mu_{i\to i-1}(x_{i-1}) = \sum_{x_i} \psi_{i-1,i}(x_{i-1}, x_i)\, \psi_i(x_i)\, \mu_{i+1\to i}(x_i)}$$

$$\boxed{\mu_{i\to i+1}(x_{i+1}) = \sum_{x_i} \mu_{i-1\to i}(x_i)\, \psi_i(x_i)\, \psi_{i,i+1}(x_i, x_{i+1})}$$

Note that node i always receives a message which is a function of x_i.

After propagating all the messages, we have

$$p(x_j) = \frac{1}{Z}\, \mu_{j-1\to j}(x_j)\ \psi_j(x_j)\ \mu_{j+1\to j}(x_j).$$

How do we compute Z?

$$1 = \sum_{x_j} p(x_j) \quad \Rightarrow \quad Z = \sum_{x_j} \mu_{j-1\to j}(x_j)\ \psi_j(x_j)\ \mu_{j+1\to j}(x_j)$$

Computing all marginals $p(x_1), p(x_2), \dots, p(x_n)$

Computing $p(x_j)$ requires to compute

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \dots \to \boxed{\mu_{j-1\to j}}$$

and

$$\boxed{\mu_{j+1\to j}} \leftarrow \dots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$$

Computing all marginals $p(x_1), p(x_2), \ldots, p(x_n)$

Computing $p(x_j)$ requires to compute

$\boxed{\mu_{1\to 2}} \rightarrow \boxed{\mu_{2\to 3}} \rightarrow \ldots \rightarrow \boxed{\mu_{j-1\to j}}$ and

$\boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$

How do we compute $p(x_{j'})$ for $j' \neq j$?

Computing all marginals $p(x_1), p(x_2), \ldots, p(x_n)$

Computing $p(x_j)$ requires to compute

$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}}$ and

$\boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$

How do we compute $p(x_{j'})$ for $j' \neq j$? Can use the same messages !

Computing all marginals $p(x_1), p(x_2), \ldots, p(x_n)$

Computing $p(x_j)$ requires to compute

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}}$$

and

$$\boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$$

How do we compute $p(x_{j'})$ for $j' \neq j$? Can use the same messages !

If we compute all $n-1$ ascending messages

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}} \to \ldots \to \boxed{\mu_{n-2\to n-1}} \to \boxed{\mu_{n-1\to n}}$$

Computing all marginals $p(x_1), p(x_2), \ldots, p(x_n)$

Computing $p(x_j)$ requires to compute

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}}$$ and

$$\boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$$

How do we compute $p(x_{j'})$ for $j' \neq j$? Can use the same messages !

If we compute all $n-1$ ascending messages

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}} \to \ldots \to \boxed{\mu_{n-2\to n-1}} \to \boxed{\mu_{n-1\to n}}$$

... and all $n-1$ descending messages

$$\boxed{\mu_{2\to 1}} \leftarrow \boxed{\mu_{3\to 1}} \leftarrow \ldots \to \boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$$

Computing all marginals $p(x_1), p(x_2), \ldots, p(x_n)$

Computing $p(x_j)$ requires to compute

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}}$$

and

$$\boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$$

How do we compute $p(x_{j'})$ for $j' \neq j$? Can use the same messages !

If we compute all $n-1$ ascending messages

$$\boxed{\mu_{1\to 2}} \to \boxed{\mu_{2\to 3}} \to \ldots \to \boxed{\mu_{j-1\to j}} \to \ldots \to \boxed{\mu_{n-2\to n-1}} \to \boxed{\mu_{n-1\to n}}$$

... and all $n-1$ descending messages

$$\boxed{\mu_{2\to 1}} \leftarrow \boxed{\mu_{3\to 1}} \leftarrow \ldots \to \boxed{\mu_{j+1\to j}} \leftarrow \ldots \leftarrow \boxed{\mu_{n-1\to n-2}} \leftarrow \boxed{\mu_{n\to n-1}}$$

then all marginals are computed with an additional $O(n)$ computation.

Computing pairwise marginals $p(x_j, x_{j+1})$

Passing

- ascending messages until node j
- descending messages until node $j+1$

one gets:

$$\boxed{p(x_j, x_{j+1}) = \frac{1}{Z}\, \mu_{j-1\to j}(x_j)\, \psi_j(x_j)\, \psi_{j+1}(x_{j+1})\, \psi_{j,j+1}(x_j, x_{j+1})\, \mu_{j+1\to j}(x_j)}$$

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

- Orient the tree by setting node r to be the root

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

- Orient the tree by setting node r to be the root
- Let $\mathcal{C}_j$ be the set of children of node j

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

- Orient the tree by setting node r to be the root
- Let $\mathcal{C}_j$ be the set of children of node j
- Let $\mathcal{D}_j$ be the set of descendants of node j

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

- Orient the tree by setting node r to be the root
- Let $\mathcal{C}_j$ be the set of children of node j
- Let $\mathcal{D}_j$ be the set of descendants of node j
- Let π_j be the parent of node j

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

- Orient the tree by setting node r to be the root
- Let $\mathcal{C}_j$ be the set of children of node j
- Let $\mathcal{D}_j$ be the set of descendants of node j
- Let π_j be the parent of node j

Let

$$F(x_i, x_j, x_{\mathcal{D}_j}) := \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \left[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \,\right]$$

Inference in undirected trees

Let (V, E) be a tree and assume a tree graphical model

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Again we wish to compute $p(x_r)$ for some node r

- Orient the tree by setting node r to be the root
- Let $\mathcal{C}_j$ be the set of children of node j
- Let $\mathcal{D}_j$ be the set of descendants of node j
- Let π_j be the parent of node j

Let

$$F(x_i, x_j, x_{\mathcal{D}_j}) := \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \big[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \,\big]$$

Then we have

$$p(x) = \psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

Exploiting a recursion on the tree

$$F(x_i, x_j, x_{\mathcal{D}_j}) \quad := \quad \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \left[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \,\right]$$

Exploiting a recursion on the tree

$$\begin{aligned} F(x_i, x_j, x_{\mathcal{D}_j}) &:= \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \big[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \,\big] \\ F(x_i, x_j, x_{\mathcal{D}_j}) &= \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k}) \end{aligned}$$

Exploiting a recursion on the tree

$$F(x_i, x_j, x_{\mathcal{D}_j}) := \psi_{i,j}(x_i, x_j)\,\psi_j(x_j) \prod_{k \in \mathcal{D}_j} \left[\,\psi_k(x_k)\,\psi_{\pi_k,k}(x_{\pi_k}, x_k)\,\right]$$

$$F(x_i, x_j, x_{\mathcal{D}_j}) = \psi_{i,j}(x_i, x_j)\,\psi_j(x_j) \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k})$$

As a consequence

$$\sum_{x_j, x_{\mathcal{D}_j}} F(x_i, x_j, x_{\mathcal{D}_j}) = \sum_{x_j} \left[\psi_{i,j}(x_i, x_j)\,\psi_j(x_j) \sum_{x_{\mathcal{D}_j}} \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k})\right]$$

Exploiting a recursion on the tree

$$F(x_i, x_j, x_{\mathcal{D}_j}) := \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \left[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \,\right]$$

$$F(x_i, x_j, x_{\mathcal{D}_j}) = \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k})$$

As a consequence

$$\begin{aligned} \sum_{x_j, x_{\mathcal{D}_j}} F(x_i, x_j, x_{\mathcal{D}_j}) &= \sum_{x_j} \Big[\psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \sum_{x_{\mathcal{D}_j}} \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k}) \Big] \\ &= \sum_{x_j} \Big[\psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} \sum_{x_k, x_{\mathcal{D}_k}} F(x_j, x_k, x_{\mathcal{D}_k}) \Big] \end{aligned}$$

Exploiting a recursion on the tree

$$
\begin{aligned}
F(x_i, x_j, x_{\mathcal{D}_j}) &:= \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \left[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \right] \\
F(x_i, x_j, x_{\mathcal{D}_j}) &= \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k})
\end{aligned}
$$

As a consequence

$$
\begin{aligned}
\sum_{x_j, x_{\mathcal{D}_j}} F(x_i, x_j, x_{\mathcal{D}_j}) &= \sum_{x_j} \Big[\psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \sum_{x_{\mathcal{D}_j}} \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k}) \Big] \\
&= \sum_{x_j} \Big[\psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} \sum_{x_k, x_{\mathcal{D}_k}} F(x_j, x_k, x_{\mathcal{D}_k}) \Big]
\end{aligned}
$$

So if we define $\mu_{j \to i}(x_i) := \sum_{x_j, x_{\mathcal{D}_j}} F(x_i, x_j, x_{\mathcal{D}_j})$,

Exploiting a recursion on the tree

$$F(x_i, x_j, x_{\mathcal{D}_j}) := \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{D}_j} \left[\, \psi_k(x_k)\, \psi_{\pi_k,k}(x_{\pi_k}, x_k) \,\right]$$

$$F(x_i, x_j, x_{\mathcal{D}_j}) = \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k})$$

As a consequence

$$\sum_{x_j, x_{\mathcal{D}_j}} F(x_i, x_j, x_{\mathcal{D}_j}) = \sum_{x_j} \Big[\psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \sum_{x_{\mathcal{D}_j}} \prod_{k \in \mathcal{C}_j} F(x_j, x_k, x_{\mathcal{D}_k}) \Big]$$

$$= \sum_{x_j} \Big[\psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} \sum_{x_k, x_{\mathcal{D}_k}} F(x_j, x_k, x_{\mathcal{D}_k}) \Big]$$

So if we define $\mu_{j\to i}(x_i) := \sum_{x_j, x_{\mathcal{D}_j}} F(x_i, x_j, x_{\mathcal{D}_j})$, then we have

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{C}_j} \mu_{k\to j}(x_j)}$$

Rooted sum-product algorithm: ascending scheme

The recursion

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{C}_j} \mu_{k\to j}(x_j)}$$

defines an algorithm starting at the leaves.

Rooted sum-product algorithm: ascending scheme

The recursion

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{C}_j} \mu_{k\to j}(x_j)}$$

defines an algorithm starting at the leaves.

1. For any leaf j with parent $i = \pi_j$

$$\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j)$$

Rooted sum-product algorithm: ascending scheme

The recursion

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i,x_j)\,\psi_j(x_j) \prod_{k\in\mathcal{C}_j} \mu_{k\to j}(x_j)}$$

defines an algorithm starting at the leaves.

1. For any leaf j with parent $i = \pi_j$

$$\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i,x_j)\,\psi_j(x_j)$$

2. Then for any node that only has leaves, we can compute the message to their parents
3. etc

Rooted sum-product algorithm: ascending scheme

The recursion

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{C}_j} \mu_{k\to j}(x_j)}$$

defines an algorithm starting at the leaves.

1. For any leaf j with parent $i = \pi_j$

$$\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j)$$

2. Then for any node that only has leaves, we can compute the message to their parents
3. etc

At the end of the algorithm:

Each node i has sent a message $\mu_{i\to\pi_i}(x_{\pi_i})$ to its parent.

Finally computing $p(x_r)$

Remember

$$p(x) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

Finally computing $p(x_r)$

Remember

$$p(x) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

$$p(x_r) = \sum_{x_{\mathcal{D}_r}} p(x)$$

Finally computing $p(x_r)$

Remember

$$p(x) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

$$p(x_r) = \sum_{x_{\mathcal{D}_r}} p(x) \quad = \quad \frac{1}{Z}\psi_r(x_r) \sum_{x_{\mathcal{D}_r}} \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

Finally computing $p(x_r)$

Remember

$$p(x) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

$$\begin{aligned}
p(x_r) = \sum_{x_{\mathcal{D}_r}} p(x) &= \frac{1}{Z}\psi_r(x_r) \sum_{x_{\mathcal{D}_r}} \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i}) \\
&= \frac{1}{Z}\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} \sum_{x_i, x_{\mathcal{D}_i}} F(x_r, x_i, x_{\mathcal{D}_i})
\end{aligned}$$

Finally computing $p(x_r)$

Remember

$$p(x) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

$$\begin{aligned}
p(x_r) = \sum_{x_{\mathcal{D}_r}} p(x) &= \frac{1}{Z}\psi_r(x_r) \sum_{x_{\mathcal{D}_r}} \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i}) \\
&= \frac{1}{Z}\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} \sum_{x_i, x_{\mathcal{D}_i}} F(x_r, x_i, x_{\mathcal{D}_i}) \\
&= \frac{1}{Z}\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} \mu_{i \to r}(x_r)
\end{aligned}$$

Finally computing $p(x_r)$

Remember

$$p(x) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i})$$

$$\begin{aligned}
p(x_r) = \sum_{x_{\mathcal{D}_r}} p(x) &= \frac{1}{Z}\psi_r(x_r) \sum_{x_{\mathcal{D}_r}} \prod_{i \in \mathcal{C}(r)} F(x_r, x_i, x_{\mathcal{D}_i}) \\
&= \frac{1}{Z}\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} \sum_{x_i, x_{\mathcal{D}_i}} F(x_r, x_i, x_{\mathcal{D}_i}) \\
&= \frac{1}{Z}\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} \mu_{i \to r}(x_r)
\end{aligned}$$

So

$$\boxed{p(x_r) = \frac{1}{Z}\,\psi_r(x_r) \prod_{i \in \mathcal{C}(r)} \mu_{i \to r}(x_r)}$$

Towards the sum-product without root

Towards the sum-product without root

Notice

- if we change the root, the edges whose orientation does not change still transmit the same message !

Towards the sum-product without root

Notice

- if we change the root, the edges whose orientation does not change still transmit the same message !
- the edges whose orientation has changed now send a message in the other direction...

Towards the sum-product without root

Notice

- if we change the root, the edges whose orientation does not change still transmit the same message !
- the edges whose orientation has changed now send a message in the other direction...

This leads to a general scheme in which each edge will at some point be traversed by a message in each direction.

Towards the sum-product without root

Notice

- if we change the root, the edges whose orientation does not change still transmit the same message !
- the edges whose orientation has changed now send a message in the other direction...

This leads to a general scheme in which each edge will at some point be traversed by a message in each direction.

Message passing without orientation of the tree

Let $\mathcal{N}_i$ denote the set of neighbors of node i.
Consider the update:

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{N}_j\setminus\{i\}} \mu_{k\to j}(x_j)}$$

Towards the sum-product without root

Notice

- if we change the root, the edges whose orientation does not change still transmit the same message !
- the edges whose orientation has changed now send a message in the other direction...

This leads to a general scheme in which each edge will at some point be traversed by a message in each direction.

Message passing without orientation of the tree

Let $\mathcal{N}_i$ denote the set of neighbors of node i.
Consider the update:

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\,\psi_j(x_j) \prod_{k\in\mathcal{N}_j\setminus\{i\}} \mu_{k\to j}(x_j)}$$

Key remark: a node can only send a message if it has received messages from **all of its neighbors except one**.

Sum-product algorithm: sequential version

At each iteration:
Any node j that has received messages from **all but one** – call it i – of its neighbors sends a message $\mu_{j\to i}(x_i)$ to i with

$$\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{N}_j\setminus\{i\}} \mu_{k\to j}(x_j)$$

Flooding sum-product algorithm (aka parallel SP)

Initialize all messages at random
At each iteration:

For each node j

For each neighbor $i \in \mathcal{N}_j$

- j sends to i the message $\mu_{j \to i}(x_i)$ with

$$\mu_{j \to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{N}_j \setminus \{i\}} \mu_{k \to j}(x_j)$$

The algorithm converges after a number of iterations equal to the diameter of the tree.

Computing the marginals from the messages

At each node $i \in V$

$$p(x_i) = \frac{1}{Z}\psi_i(x_i) \prod_{k\in\mathcal{N}_i} \mu_{k\to i}(x_i).$$

For each edge,

$$p(x_i, x_j) = \frac{1}{Z}\psi_i(x_i)\psi_j(x_j)\psi_{ij}(x_i, x_j) \left(\prod_{\substack{k\in\mathcal{N}_i \\ k\neq j}} \mu_{k\to i}(x_i) \right) \left(\prod_{\substack{\ell\in\mathcal{N}_j \\ \ell\neq i}} \mu_{\ell\to j}(x_j) \right)$$

Computing *conditional marginals* $p(x_i \mid X_B = x_{B0})$

Example

$$p(x_i|x_5 = 3, x_{10} = 2) \propto p(x_i, x_5 = 3, x_{10} = 2)$$

Computing *conditional marginals* $p(x_i \mid X_B = x_{B0})$

Example

$$p(x_i|x_5 = 3, x_{10} = 2) \propto p(x_i, x_5 = 3, x_{10} = 2)$$

Idea: redefine potentials

Computing *conditional marginals* $p(x_i \mid X_B = x_{B0})$

Example

$$p(x_i|x_5 = 3, x_{10} = 2) \propto p(x_i, x_5 = 3, x_{10} = 2)$$

Idea: redefine potentials

$$\widetilde{\psi}_5(x_5) = \psi_5(x_5)\ \delta(x_5, 3)$$

Computing *conditional marginals* $p(x_i \mid X_B = x_{B0})$

Given observations $X_j = x_{j0}$ for $j \in B$, define modified potentials:

$$\widetilde{\psi}_j(x_j) = \psi_j(x_j)\ \delta(x_j, x_{j0})$$

so that

$$p(x \mid X_B = x_{B0}) = \frac{1}{\widetilde{Z}} \prod_{i \in V} \widetilde{\psi}_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Computing *conditional marginals* $p(x_i \mid X_B = x_{B0})$

Given observations $X_j = x_{j0}$ for $j \in B$, define modified potentials:

$$\widetilde{\psi}_j(x_j) = \psi_j(x_j)\ \delta(x_j, x_{j0})$$

so that

$$p(x \mid X_B = x_{B0}) = \frac{1}{\widetilde{Z}} \prod_{i \in V} \widetilde{\psi}_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Indeed we have

$$p(x \mid X_B = x_{B0})\ p(X_B = x_{B0}) = p(x) \prod_{j \in B} \delta(x_j, x_{j0})$$

Computing *conditional marginals* $p(x_i \mid X_B = x_{B0})$

Given observations $X_j = x_{j0}$ for $j \in B$, define modified potentials:

$$\widetilde{\psi}_j(x_j) = \psi_j(x_j)\ \delta(x_j, x_{j0})$$

so that

$$p(x \mid X_B = x_{B0}) = \frac{1}{\widetilde{Z}} \prod_{i \in V} \widetilde{\psi}_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$$

Indeed we have

$$p(x \mid X_B = x_{B0})\ p(X_B = x_{B0}) = p(x) \prod_{j \in B} \delta(x_j, x_{j0})$$

We then simply apply the SPA to these new potentials.

Inference beyond trees

- The SPA is also called *belief propagation* or *message passing*.
- On trees, it is an exact inference algorithm.

Inference beyond trees

- The SPA is also called *belief propagation* or *message passing*.
- On trees, it is an exact inference algorithm.
- If G is not a tree, the algorithm can be extended to *loopy belief propagation* but

Inference beyond trees

- The SPA is also called *belief propagation* or *message passing*.
- On trees, it is an exact inference algorithm.
- If G is not a tree, the algorithm can be extended to *loopy belief propagation* but
  - It does not converge in general to the correct marginals
  - it sometimes gives reasonable approximations.

Inference beyond trees

- The SPA is also called *belief propagation* or *message passing*.
- On trees, it is an exact inference algorithm.
- If G is not a tree, the algorithm can be extended to *loopy belief propagation* but
  - It does not converge in general to the correct marginals
  - it sometimes gives reasonable approximations.

Junction trees

- The exact SPA can be extended to graphs that are close to trees, in the sense that they can be viewed as trees if nodes can be grouped together into cliques to form a tree.
- Elegant theory, but rarely used in practice.

Generalized distributive law

We needed to compute

$$p(x_{i_0}) = \sum_{x_{V\setminus\{i_0\}}} \left[\prod_{i\in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j) \right]$$

Generalized distributive law

We needed to compute

$$p(x_{i_0}) = \sum_{x_{V\setminus\{i_0\}}} \Big[\prod_{i\in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j) \Big]$$

The only key property we used is that

The addition is *distributive* over the product

$$(a+b)(c+d) = ac + bc + ad + bd$$

Generalized distributive law

We needed to compute

$$p(x_{i_0}) = \sum_{x_{V\setminus\{i_0\}}} \Big[\prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j) \Big]$$

The only key property we used is that

The addition is *distributive* over the product

$$(a+b)(c+d) = ac + bc + ad + bd$$

Max product for decoding (MAP inference)

Decoding consists in computing the most probable configuration

$$p(x^*) = \max_{x_V} \Big[\prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j) \Big]$$

Remark: this is often the most probable configuration of a conditional model (given some input data).

Generalized distributive law

We needed to compute

$$p(x_{i_0}) = \sum_{x_{V\setminus\{i_0\}}} \Big[\prod_{i\in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j) \Big]$$

The only key property we used is that

The addition is *distributive* over the product

$$(a+b)(c+d) = ac + bc + ad + bd$$

Max product for decoding (MAP inference)

Decoding consists in computing the most probable configuration

$$p(x^*) = \max_{x_V} \Big[\prod_{i\in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j) \Big]$$

Remark: this is often the most probable configuration of a conditional model (given some input data).

Decoding (aka MAP inference) with max-product

Apply the same algorithms as for SPA, but replace the update with

$$\check{\mu}_{j \to i}(x_i) = \max_{x_j} \; \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k \in \mathcal{N}_j \setminus \{i\}} \check{\mu}_{k \to j}(x_j)$$

Decoding (aka MAP inference) with max-product

Apply the same algorithms as for SPA, but replace the update with

$$\check{\mu}_{j\to i}(x_i) = \max_{x_j} \; \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{N}_j\backslash\{i\}} \check{\mu}_{k\to j}(x_j)$$

- Known as the Viterbi algorithm in the case of chains.

Decoding (aka MAP inference) with max-product

Apply the same algorithms as for SPA, but replace the update with

$$\check{\mu}_{j\to i}(x_i) = \max_{x_j} \; \psi_{i,j}(x_i, x_j)\, \psi_j(x_j) \prod_{k\in\mathcal{N}_j\setminus\{i\}} \check{\mu}_{k\to j}(x_j)$$

- Known as the Viterbi algorithm in the case of chains.

Max-sum algorithm

Same on log-scale:

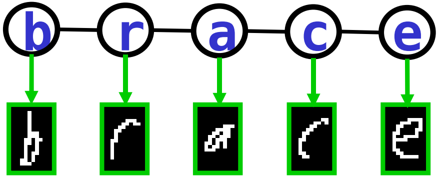
$$\log \check{\mu}_{j\to i}(x_i) = \max_{x_j} \left[\log \psi_{i,j}(x_i, x_j) + \log \psi_j(x_j) + \sum_{k\in\mathcal{N}_j\setminus\{i\}} \log \check{\mu}_{k\to j}(x_j) \right]$$

Other inference methods

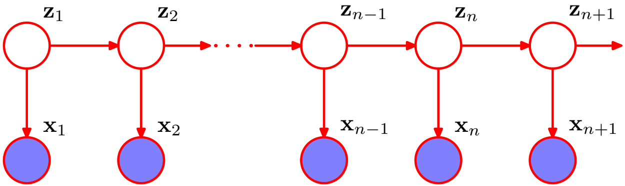
- There are many (exact/approximate/inexact) inference algorithms
- For a general graph exact inference is NP-hard
- The sum-product algorithm (SPA) performs inference on trees, with a complexity linear in the number of nodes.
- SPA can be generalized to graph that are close to trees in some sense using the *junction tree theory.*
- In general, one needs to use approximate or inexact methods
  - Gibbs sampling, and other MCMC sampling methods
  - Variational methods
    - Mean field (/Structured mean field)
    - Loopy belief propagation
    - TRW-entropy based inference

Hidden Markov Model (HMM)

- voice recognition
- natural langage processing
- handwritten character recognition
- modelling biological sequence (protein, DNA)



Hidden Markov Model (HMM)



$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \boldsymbol{z}_1, \dots, \boldsymbol{z}_N) = p(\boldsymbol{z}_1) \prod_{n=2}^{N} p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}) \prod_{n=1} p(\mathbf{x}_n | \boldsymbol{z}_n)$$

Homogeneous Markov chains

- $\boldsymbol{z}_n \in \{0,1\}^K$ state indicator variable $(1, \dots, K)$
- *homogeneous* Markov chain: $\forall n,\ p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_2 | \boldsymbol{z}_1)$
- emitted symbol $\mathbf{x}_n$ $(\{0,1\}^K)$ / observation $(\mathbb{R}^d)$

Hidden Markov Model (HMM)

Parameterization

distribution of the initial state $\quad p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

Hidden Markov Model (HMM)

Parameterization

distribution of the initial state $p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

transition matrix $$p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}; A) = \prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{z_{n-1,j}\, z_{nk}}$$

Hidden Markov Model (HMM)

Parameterization

distribution of the initial state	$p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^K \pi_k^{z_{1k}}$
transition matrix	$p(\boldsymbol{z}_n \vert \boldsymbol{z}_{n-1}; A) = \prod_{j=1}^K \prod_{k=1}^K A_{jk}^{z_{n-1,j}\, z_{nk}}$
emission probabilities	$p(\mathbf{x}_n \vert \boldsymbol{z}_n; \phi)$ e.g. Gaussian Mixture

Hidden Markov Model (HMM)

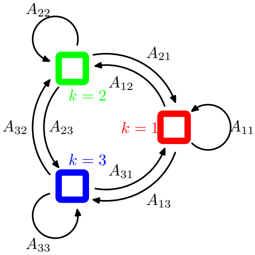
Parameterization

distribution of the initial state $\quad p(\boldsymbol{z}_1;\pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

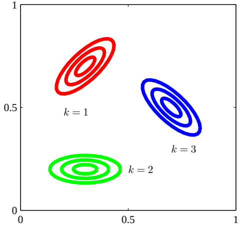
transition matrix $\quad p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1};A) = \displaystyle\prod_{j=1}^{K}\prod_{k=1}^{K} A_{jk}^{z_{n-1,j}\, z_{nk}}$

emission probabilities $\quad p(\mathbf{x}_n|\boldsymbol{z}_n;\phi)$ e.g. Gaussian Mixture

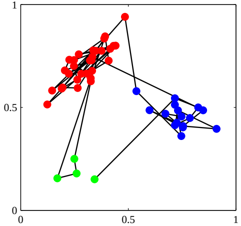
Interpretation



Transistions of $\boldsymbol{z}_n$



$p(\mathbf{x}_n|\boldsymbol{z}_n)$



Path of $\mathbf{x}_n$

Maximum likelihood for HMMs

Application of the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n | \boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n | \boldsymbol{X}, \boldsymbol{\theta}^t)$$

Maximum likelihood for HMMs

Application of the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n | \boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n | \boldsymbol{X}, \boldsymbol{\theta}^t)$$

Expected value of the log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \log p(x_n | \phi$$

Maximum likelihood for HMMs

Application of the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t)$$

Expected value of the log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \log p(x_n|\phi_k$$

Maximizing with respect to the parameters $\{\pi, A\}$, one gets

$$\boxed{\pi_k^{t+1} = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}} \qquad \boxed{A_{jk}^{t+1} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}}$$

Maximum likelihood for HMMs

Application of the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t)$$

Expected value of the log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \log p(x_n|\phi_k$$

Maximizing with respect to the parameters $\{\pi, A\}$, one gets

$$\boxed{\pi_k^{t+1} = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}} \qquad \boxed{A_{jk}^{t+1} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}}$$

If the emissions are Gaussian then we also have:

$$\boldsymbol{\mu}_k^{t+1} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \qquad \boldsymbol{\Sigma}_k^{t+1} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

Maximum likelihood for HMMs

Baum-Welch algorithm

The Baum-Welch algorithm is a special instance of the sum-product algorithm. It is also known under the name *forward-backward*.

One propagates the messages

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1}) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$

Maximum likelihood for HMMs

Baum-Welch algorithm

The Baum-Welch algorithm is a special instance of the sum-product algorithm. It is also known under the name *forward-backward*.

One propagates the messages

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1})p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$
- backward $\beta(\boldsymbol{z}_n) = \sum_{\boldsymbol{z}_{n+1}} \beta(\boldsymbol{z}_{n+1})p(\mathbf{x}_{n+1}|\boldsymbol{z}_{n+1})p(\boldsymbol{z}_{n+1}|\boldsymbol{z}_n)$

Maximum likelihood for HMMs

Baum-Welch algorithm

The Baum-Welch algorithm is a special instance of the sum-product algorithm. It is also known under the name *forward-backward*.

One propagates the messages

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1}) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$
- backward $\beta(\boldsymbol{z}_n) = \sum_{\boldsymbol{z}_{n+1}} \beta(\boldsymbol{z}_{n+1}) p(\mathbf{x}_{n+1}|\boldsymbol{z}_{n+1}) p(\boldsymbol{z}_{n+1}|\boldsymbol{z}_n)$

that satisfy the following properties:

$$\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{z}_n) \qquad \beta(\boldsymbol{z}_n) = p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\boldsymbol{z}_n)$$

Maximum likelihood for HMMs

Baum-Welch algorithm

The Baum-Welch algorithm is a special instance of the sum-product algorithm. It is also known under the name *forward-backward.*

One propagates the messages

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1}) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$
- backward $\beta(\boldsymbol{z}_n) = \sum_{\boldsymbol{z}_{n+1}} \beta(\boldsymbol{z}_{n+1}) p(\mathbf{x}_{n+1}|\boldsymbol{z}_{n+1}) p(\boldsymbol{z}_{n+1}|\boldsymbol{z}_n)$

that satisfy the following properties:

$$\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{z}_n) \qquad \beta(\boldsymbol{z}_n) = p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\boldsymbol{z}_n)$$

Finally one gets the marginals:

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) = \frac{\alpha(\boldsymbol{z}_n)\beta(\boldsymbol{z}_n)}{p(\boldsymbol{X}|\boldsymbol{\theta}^t)}$$

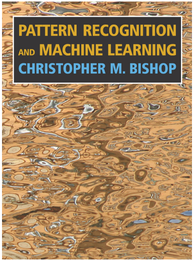
et

$$\xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = \frac{\alpha(\mathbf{x}_{n-1}) p(\mathbf{x}_n|\boldsymbol{z}_n) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \beta(\mathbf{x}_n)}{p(\boldsymbol{X}|\boldsymbol{\theta}^t)}$$

Conclusions

- Graphical models offer a language to design structured probabilistic models
- The graph encodes both
  - A factorization of the probability distribution
  - Implicitly associated conditional independences between the variables
- Three key operations on graphical model are decoding, inference and learning.
- Probabilistic inference and decoding and can be done efficiently using sum-product and max-product algorithm if the graph is a tree.
- Learning with the maximum likelihood principle in exponential families is a convex optimization problem requires to solve the probabilistic inference problem.
- Latent variables are easily handled using the Expectation-Maximization algorithm, which again requires to solve the probabilistic inference problem.

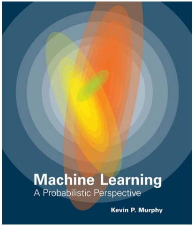
Books



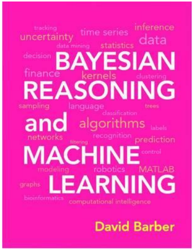
Pattern Recognition and Machine Learning, Christopher Bishop, Springer (2006).
`http://research.microsoft.com/~cmbishop/PRML/`

Many of the figures of these lectures are figures from Bishop's book that are kindly made available for teaching purposes.

Books

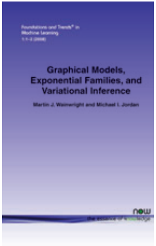


Machine Learning, a probabilistic perspective
Kevin Murphy, MIT Press (2012).

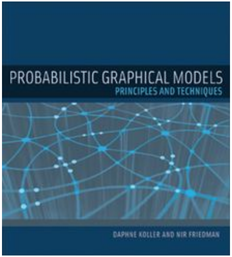


Bayesian reasoning and machine learning,
David Barber,
Cambridge University Press (2012).
`http://www.cs.ucl.ac.uk/staff/d.barber/brml/`

Books



Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning.



Daphne Koller and Nir Friedman, Probabilistic Graphical Models - Principles and Techniques, 2009, MIT Press.

Going further

- There are many approximate/inexact variational inference algorithms (Wainwright, 2008)
- Graphical models can be learned in the max-margin setting (Taskar et al., 2003; Tsochantaridis et al., 2005; Pletscher et al., 2010)
- Graphical models techniques are used in deep learning: variational auto-encoders, RBMs, generative adversarial networks, etc.

References I

Meshi, O., Sontag, D., Globerson, A., and Jaakkola, T. S. (2010). Learning efficiently with approximate inference via dual losses. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 783–790.

Meshi, O., Srebro, N., and Hazan, T. (2015). Efficient training of structured SVMs via soft constraints. In *AISTATS*.

Pletscher, P., Ong, C. S., and Buhmann, J. M. (2010). Entropy and margin maximization for structured output learning. In *ECML*.

Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *NIPS*.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484.

Wainwright, M. J. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*.