# Unsupervised Learning

## Propagation on Factor Graphs

**Zoubin Ghahramani**
zoubin@gatsby.ucl.ac.uk

**Gatsby Computational Neuroscience Unit, and**
**MSc in Intelligent Systems, Dept Computer Science**
**University College London**

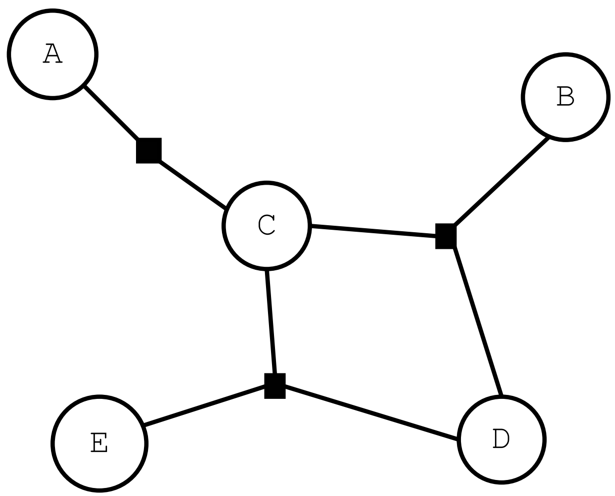
**Autumn 2003**

# Factor Graphs

In a factor graph, the joint probability distribution is written as a product of factors. Consider a vector of variables $\mathbf{x} = (x_1, \ldots, x_n)$

$$p(\mathbf{x}) = p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_j f_j(\mathbf{x}_{S_j})$$

where $Z$ is the normalisation constant, $S_j$ denotes the subset of $\{1, \ldots, n\}$ which participate in factor $f_j$ and $\mathbf{x}_{S_j} = \{x_i : i \in S_j\}$.




**variables nodes**: we draw open circles for each variable $x_i$ in the distribution.
**function nodes**: we draw filled dots for each function $f_j$ in the distribution.

# Propagation in Factor Graphs

Let $\mathrm{n}(x)$ denote the set of function nodes that are neighbors of $x$.
Let $\mathrm{n}(f)$ denote the set of variable nodes that are neighbors of $f$.

We can compute probabilities in a factor graph by propagating messages from variable nodes to function nodes and viceversa.

**message from variable $x$ to local function $f$:**

$$\mu_{x \to f}(x) = \prod_{h \in \mathrm{n}(x) \setminus \{f\}} \mu_{h \to x}(x)$$

**message from local function $f$ to variable $x$:**

$$\mu_{f \to x}(x) = \sum_{\mathbf{x} \setminus x} \left( f(\mathbf{x}) \prod_{y \in \mathrm{n}(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

# Propagation in Factor Graphs

$\mathrm{n}(x)$ denotes the set of function nodes that are neighbors of $x$.
$\mathrm{n}(f)$ denotes the set of variable nodes that are neighbors of $f$.

**message from variable $x$ to local function $f$:**

$$\mu_{x\to f}(x) = \prod_{h\in \mathrm{n}(x)\backslash\{f\}} \mu_{h\to x}(x)$$

**message from local function $f$ to variable $x$:**

$$\mu_{f\to x}(x) = \sum_{\mathbf{x}\backslash x} \left( f(\mathbf{x}) \prod_{y\in \mathrm{n}(f)\backslash\{x\}} \mu_{y\to f}(y) \right)$$

Once a variable has received all messages from its neighboring function nodes we can compute the probability of that variable by multiplying all the messages and renormalising:

$$p(x) \propto \prod_{h\in \mathrm{n}(x)} \mu_{h\to x}(x)$$

# Elimination Rules for Factor Graphs

- **eliminating observed variables**

  If a variable $x_i$ is **observed**, i.e. its value is given, then it is a *constant* in all functions that include $x_i$.

  We can **eliminate** $x_i$ from the graph by removing the corresponding node and modifying all neighboring functions to treat it as a constant.

# Elimination Rules for Factor Graphs

- **eliminating hidden variables**

  If a variable $x_i$ is **hidden** and we are not interested in it we can eliminate it from the graph by summing over all its values.

$$
\begin{aligned}
\sum_{x_i} p(\mathbf{x}) &= \frac{1}{Z} \sum_{x_i} \prod_{j} f_j(\mathbf{x}_{S_j}) \\
&= \frac{1}{Z} \prod_{j \notin \mathrm{n}(x_i)} f_j(\mathbf{x}_{S_j}) \left( \sum_{x_i} \prod_{k \in \mathrm{n}(x_i)} f_k(\mathbf{x}_{S_k}) \right) \\
&= \frac{1}{Z} \prod_{j \notin \mathrm{n}(x_i)} f_j(\mathbf{x}_{S_j}) \quad f_{\mathrm{new}}(\mathbf{x}_{S_{\mathrm{new}}})
\end{aligned}
$$

where $f_{\mathrm{new}}(\mathbf{x}_{S_{\mathrm{new}}}) = \sum_{x_i} \prod_{k \in \mathrm{n}(x_i)} f_k(\mathbf{x}_{S_k})$ and $S_{\mathrm{new}} = \bigcup_{k \in \mathrm{n}(x_i)} S_k \setminus \{i\}$.

This causes all its neighboring function nodes to merge into one new function node.