Fig. 2.9 (a) A factor graph representation of a parity check code of length $m = 6$. On the left, circular white nodes $x_i$ represent the unobserved bits that define the code, whereas circular gray nodes represent the observed values $y_i$, received from the channel. On the right, black squares $\square$ represent the associated factors, or parity checks. This particular code is a (2, 3) code, since each bit is connected to two parity variables, and each parity relation involves three bits. (b) A large factor graph with a "locally tree-like" structure. Random constructions of factor graphs on $m$ vertices with bounded degree have cycles of typical length $\approx \log m$; this tree-like property is essential to the success of the sum-product algorithm [200, 169].

With the specification of a model for channel noise, this decoding problem can be cast as an inference problem. Depending on the loss function, optimal decoding is based either on computing the marginal probability $p(x_s = 1 | y)$ at each node, or computing the most likely codeword (i.e., the mode of the posterior). For the simple code of Figure 2.9(a), optimal decoding is easily achievable via the junction tree algorithm. Of interest in many applications, however, are much larger codes in which the number of bits is easily several thousand. The graphs underlying these codes are not of low treewidth, so that the junction tree algorithm is not viable. Moreover, MCMC algorithms have not been deployed successfully in this domain.

For many graphical codes, the most successful decoder is based on applying the sum-product algorithm, described in Section 2.6. Since the graphical models defining good codes invariably have cycles, the sum-product algorithm is not guaranteed to compute the correct marginals, nor even to converge. Nonetheless, the behavior of this
Learning from Structured Data
Probabilistic Graphical Models

• In all cases, (circular) nodes are random variables (discrete or continuous)
• Graph specifies coarse structure of joint distribution (not probability values)
• Three different conventions for associating (hyper)edges with distributions
• Common set of algorithms for efficient inference and learning
Directed Graphical Models

Chain rule implies that any joint distribution equals:

\[ p(x_{1:D}) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_1, x_2, x_3) \ldots p(x_D|x_{1:D-1}) \]

Directed graphical model implies a restricted factorization:

\[ p(x_{1:D}|G) = \prod_{t=1}^{D} p(x_t|x_{pa(t)}) \]

Nodes \( \rightarrow \) random variables

\( pa(t) \rightarrow \) parents with edges pointing to node \( t \)

Valid for any directed acyclic graph (DAG):

equivalent to dropping conditional dependencies in chain rule

\[
\begin{align*}
    p(x_{1:5}) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4) \\
    &= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_3)
\end{align*}
\]
Representational (storage, learning, computation) Complexity

- **Joint distribution**: Exponential in number of variables
- **Directed graphical model**: Exponential in number of parents ("fan-in") of each node, linear in number of nodes
**Markov Chains**

**Markov Property:** Conditioned on the present, the past and future are independent

\[ p(x) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3) \cdots \]

\[ p(x_{1:T}) = p(x_1, x_2)p(x_3 \mid x_1, x_2)p(x_4 \mid x_2, x_3) \cdots = p(x_1, x_2) \prod_{t=3}^{T} p(x_t \mid x_{t-1}, x_{t-2}) \]

**Markov Property:** Deduce by checking graph connectivity
Naïve Bayes Model

\[ p(y, x) = p(y) \prod_{j=1}^{D} p(x_j | y) \]
Medical Diagnosis

diseases

\[ d_1 \quad d_n \]

findings

\[ f_1 \quad f_m \]
Shading & Plate Notation

Convention: Shaded nodes are observed, open nodes are latent/hidden/unobserved

\[ p(y, x) = p(y) \prod_{j=1}^{D} p(x_j | y) \]
Learning & Unknown Parameters

N training examples described by D features, drawn from one of C classes

\[
p(\pi) \prod_{c=1}^{C} \prod_{j=1}^{D} p(\theta_{cj}) \prod_{i=1}^{N} p(y_i | \pi) \prod_{j=1}^{D} p(x_{ij} | y_i, \theta_{j1}, \ldots, \theta_{jC})
\]
Wavelet Decompositions

- Bandpass decomposition of images into multiple scales & orientations
- Dense features which simplify statistics of natural images
• Hidden *states* model evolution of image patterns across scale and location
• Generalizes temporal structure of Markov chains
Learning Hidden Markov Trees
Undirected Graphical Models

\[ \mathcal{V} \rightarrow \text{set of } N \text{ nodes or vertices, } \{1, 2, \ldots, N\} \]

\[ \mathcal{E} \rightarrow \text{set of undirected edges } (s,t), \text{ or equivalently } (t,s), \text{ linking pairs of nodes. The neighbors of a node are} \]

\[ \Gamma(t) = \{ s \in \mathcal{V} \mid (s, t) \in \mathcal{E} \} \]

\[ \mathcal{X}_s \rightarrow \text{random variable associated with node } s \]

**Pairwise Markov Random Field**

\[ p(x) = \frac{1}{Z} \prod_{(s,t) \in \mathcal{E}} \psi_{st}(x_s, x_t) \prod_{s \in \mathcal{V}} \psi_s(x_s) \]

\[ Z \rightarrow \text{normalization constant (partition function)} \]
Undirected Graphical Models

\[ \mathcal{V} \rightarrow \text{set of } N \text{ nodes or vertices, } \{1, 2, \ldots, N\} \]

\[ \mathcal{E} \rightarrow \text{set of undirected edges } (s,t), \text{ or equivalently } (t,s), \text{ linking pairs of nodes. The neighbors of a node are} \]

\[ \Gamma(t) = \{ s \in \mathcal{V} \mid (s, t) \in \mathcal{E} \} \]

\[ x_s \rightarrow \text{random variable associated with node } s \]

\[ C \rightarrow \text{set of cliques (fully connected subsets) of nodes} \]

**General Markov Random Field**

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \]

\[ Z \rightarrow \text{normalization constant (partition function)} \]
Undirected Markov Properties

\[ \mathcal{V} \rightarrow \text{set of } N \text{ nodes } \{1, 2, \ldots, N\} \]

\[ \mathcal{E} \rightarrow \text{set of edges } (s, t) \text{ connecting nodes } s, t \in \mathcal{V} \]

\[ p(x_A, x_C|x_B) = p(x_A|x_B)p(x_C|x_B) \]

Distribution is Markov if and only if preceding factorization holds
**Factor Graphs**

\[ \mathcal{F} \rightarrow \text{set of hyperedges linking subsets of nodes} \quad f \subseteq \mathcal{V} \]

\[ \mathcal{V} \rightarrow \text{set of } N \text{ nodes or vertices, } \{1, 2, \ldots, N\} \]

\[ Z \rightarrow \text{normalization constant (partition function)} \]

- In a **hypergraph**, the **hyperedges** link arbitrary subsets of nodes (not just pairs)
- Visualize by a bipartite graph, with square (usually black) nodes for hyperedges
- A **factor graph** associates a non-negative potential function with each hyperedge
- Motivation: **factorization key to computation**

\[ p(x) = \frac{1}{Z} \prod_{f \in \mathcal{F}} \psi_f(x_f) \]
Pairwise Nearest Neighbor MRF

- Observed nodes: Features of 2D image (intensity, color, texture, ...)
- Hidden nodes: Property of 3D world (depth, motion, object category, ...)
Kinematic Hand Tracking

Kinematic Prior

Structural Prior

Dynamic Prior
Low Density Parity Check (LDPC) Codes

Fig. 2.9 (a) A factor graph representation of a parity check code of length $m = 6$. On the left, circular white nodes represent the unobserved bits $x_i$ that define the code, whereas circular gray nodes represent the observed values $y_i$, received from the channel. On the right, black squares represent the associated factors, or parity checks. This particular code is a $(2, 3)$ code, since each bit is connected to two parity variables, and each parity relation involves three bits. (b) A large factor graph with a "locally tree-like" structure. Random constructions of factor graphs on $m$ vertices with bounded degree have cycles of typical length $\approx \log m$; this tree-like property is essential to the success of the sum-product algorithm [200, 169].

With the specification of a model for channel noise, this decoding problem can be cast as an inference problem. Depending on the loss function, optimal decoding is based either on computing the marginal probability $p(x_s = 1 | y)$ at each node, or computing the most likely codeword (i.e., the mode of the posterior). For the simple code of Figure 2.9(a), optimal decoding is easily achievable via the junction tree algorithm. Of interest in many applications, however, are much larger codes in which the number of bits is easily several thousand. The graphs underlying these codes are not of low treewidth, so that the junction tree algorithm is not viable. Moreover, MCMC algorithms have not been deployed successfully in this domain.

For many graphical codes, the most successful decoder is based on applying the sum-product algorithm, described in Section 2.6. Since the graphical models defining good codes invariably have cycles, the sum-product algorithm is not guaranteed to compute the correct marginals, nor even to converge. Nonetheless, the behavior of this...
Course Evaluation

Homeworks: 60%
- Five equally weighted assignments
- Each assignment available for two weeks before due date
- Combine mathematical derivations, algorithm design, programming, and analysis of real datasets
  - Expert systems for medical diagnosis
  - Particle filters for localization and tracking
  - Relational models of social networks and voter behavior
  - ...

Final Project: 40%
- Proposal: 1-3 pages, due in October (5%)
- Presentation: ~10 minutes, on December 9 (10%)
- Conference-style technical report, due on December 16 (25%)
- Team or individual projects? Details to be determined.
Course Prerequisites

• A course in modern statistical machine learning
  ➢ Brown CS 142: Introduction to Machine Learning
  ➢ Brown APMA 169: Computational Probability & Statistics
  ➢ Brown APMA 174/261: Recent Topics (more than sufficient)
  ➢ Possibly other classes or experience...

• Programming experience (Matlab support code)

• Readings will require “mathematical maturity”

• Insufficient background by themselves:
  ➢ Brown CS 141: Introduction to AI
  ➢ Traditional undergrad statistics (APMA 165/166)
David Barber’s *Bayesian Reasoning and Machine Learning*:
- Introductory machine learning material, as well as graphical models
- Primary source of readings for first half of course (free online)
### CS142: Introduction to Machine Learning

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete</strong></td>
<td><strong>Continuous</strong></td>
</tr>
<tr>
<td>classification or</td>
<td>clustering</td>
</tr>
<tr>
<td>categorization</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td>dimensionality</td>
</tr>
<tr>
<td></td>
<td>reduction</td>
</tr>
</tbody>
</table>

- **Supervised:** Learn to approximate a function from examples
- **Unsupervised:** Learn a representation which compresses data
- **Probabilistic:** Learn by maximizing probability, or minimizing expected loss
Probabilistic Machine Learning

**Classification:** Choose one of $C$ labels $y$ based on features $x$

$$\mathcal{D} \rightarrow \text{Training Data} \quad \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$$

**Models:** Finite set of $M$ candidates with tunable parameters

$$\theta_m \rightarrow \text{parameters for model } m = 1, \ldots, M \quad p_m(x_i, y_i \mid \theta_m)$$

**Estimation:** Learn parameters of each model from training data

- **Bayesian:** Posterior mean or mode (Maximum a Posteriori, MAP)
- **Frequentist:** Maximum likelihood (ML)

$$\hat{\theta}_m = \arg \max_{\theta_m} \left[ \log p(\theta_m) + \sum_{i=1}^{N} \log p(x_i, y_i \mid \theta_m) \right]$$

- Operationally, ML is equivalent to MAP estimation with a uniform prior
- All of these estimators are consistent as $N \rightarrow \infty$

**Evaluation:** Loss for decisions made on

Cannot tune methods on this data!
**Probabilistic Machine Learning**

**Classification:** Choose one of $C$ labels $y$ based on features $x$

\[ \mathcal{D} \rightarrow \text{Training Data} \quad \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \]

**Models:** Finite set of $M$ candidates with tunable parameters

\[ \theta_m \rightarrow \text{parameters for model } m = 1, \ldots, M \]

\[ p_m(x_i, y_i \mid \theta_m) \]

**Estimation:** Learn parameters of each model from training data

- **Bayesian:** Posterior mean or mode (*Maximum a Posteriori, MAP*)
- **Frequentist:** Maximum likelihood (*ML*)

**Model Selection:** Which model fits training data best?

- **Invalid:** Pick model where optimized MAP/ML objective is largest. Overfitting!
- **Invalid:** Pick model which performs best on test data. Overfitting!

**Evaluation:** Loss for decisions made on

*Cannot tune methods on this data!*
Probabilistic Machine Learning

**Classification:** Choose one of $C$ labels $y$ based on features $x$

$\mathcal{D} \rightarrow \text{Training Data} \quad \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$

**Models:** Finite set of $M$ candidates with tunable parameters

$\theta_m \rightarrow \text{parameters for model } m = 1, \ldots, M \quad p_m(x_i, y_i \mid \theta_m)$

**Estimation:** Learn parameters of each model from training data

- **Bayesian:** Posterior mean or mode (Maximum a Posteriori, MAP)
- **Frequentist:** Maximum likelihood (ML)

**Model Selection:** Which model fits training data best?

- **Bayesian:** Marginal likelihood of training data
- **Frequentist:** Validation or cross-validation

**Evaluation:** Loss for decisions made on

Cannot tune methods on this data!
Probabilistic Machine Learning

**Classification:** Choose one of $C$ labels $y$ based on features $x$

$\mathcal{D} \rightarrow \text{Training Data} \quad \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$

**Models:** Finite set of $M$ candidates with tunable parameters

$\theta_m \rightarrow \text{parameters for model } m = 1, \ldots, M \quad p_m(x_i, y_i \mid \theta_m)$

**Estimation:** Learn parameters of each model from training data

- **Bayesian:** Posterior mean or mode (Maximum a Posteriori, MAP)
- **Frequentist:** Maximum likelihood (ML)

**Model Selection:** Which model fits training data best?

- **Bayesian:** Marginal likelihood of training data
- **Frequentist:** Validation or cross-validation

**Inference:** Analyze test examples with best learned model $m$

$p(y_{N+1} \mid x_{N+1}, \hat{\theta}_m) \propto p(y_{N+1} \mid \hat{\theta}_m)p(x_{N+1} \mid y_{N+1}, \hat{\theta}_m)$

**Evaluation:** Loss for decisions made on

*Cannot tune methods on this data!*
Supervised Learning

Generative ML or MAP Learning:  *Naïve Bayes*

\[
\max_{\pi, \theta} \log p(\pi) + \log p(\theta) + \sum_{i=1}^{N} [\log p(y_i | \pi) + \log p(x_i | y_i, \theta)]
\]

Discriminative ML or MAP Learning:  *Logistic regression*

\[
\max_{\theta} \log p(\theta) + \sum_{i=1}^{N} \log p(y_i | x_i, \theta)
\]
Unsupervised Learning

Clustering:
\[
\max_{\pi, \theta} \log p(\pi) + \log p(\theta) + \sum_{i=1}^{N} \log \left[ \sum_{z_i} p(z_i \mid \pi) p(x_i \mid z_i, \theta) \right]
\]

Dimensionality Reduction:
\[
\max_{\pi, \theta} \log p(\pi) + \log p(\theta) + \sum_{i=1}^{N} \log \left[ \int_{z_i} p(z_i \mid \pi) p(x_i \mid z_i, \theta) \, dz_i \right]
\]

- No notion of training and test data: labels are never observed
- As before, maximize posterior probability of model parameters
- For hidden variables associated with each observation, we marginalize over possible values rather than estimating
  - Fully accounts for uncertainty in these variables
  - There is one hidden variable per observation, so cannot perfectly estimate even with infinite data
- Must use generative model (discriminative degenerates)
Unsupervised Learning: EM Algorithm

Discrete Variables
- Independent Observations
- Mixture Models & K-means
- Hidden Markov Model

Continuous Variables
- Factor Analysis & PCA
- State Space Model

Sequential Observations
- Discrete Variables
- Continuous Variables