Probabilistic Graphical Models

Brown University CSCI 2950-P, Spring 2013
Prof. Erik Sudderth

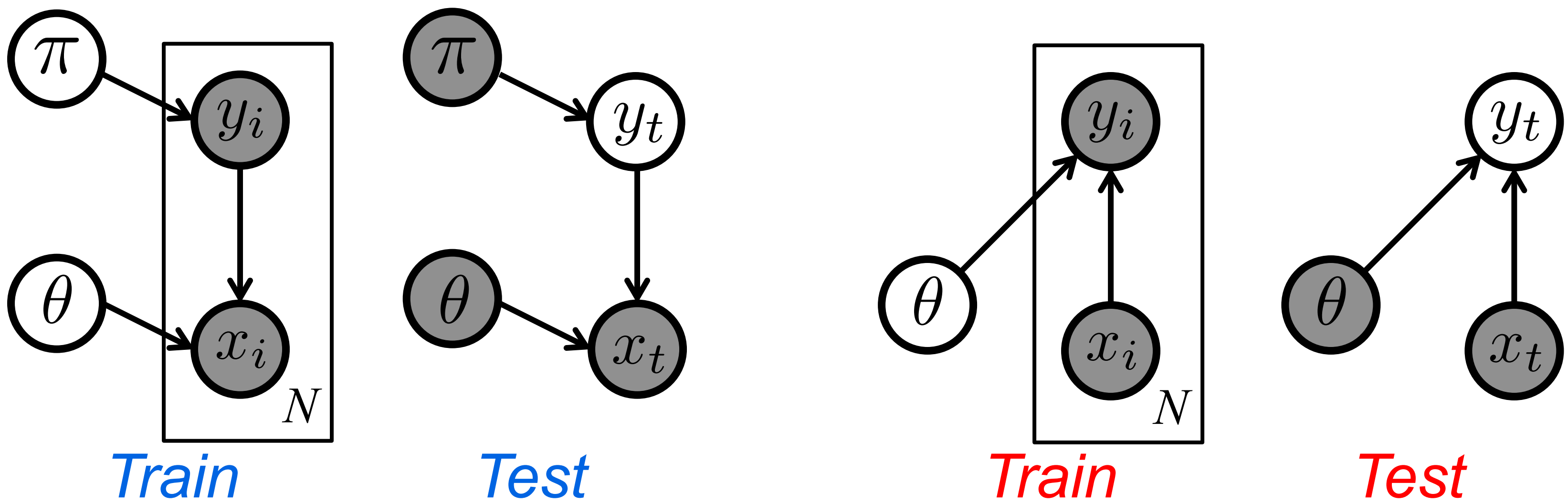
Lecture 24:
Conditional Random Fields, MAP Estimation & Max-Product BP

Some figures and examples courtesy C. Sutton and A. McCallum,
An Introduction to Conditional Random Fields, 2012.

Supervised Learning

Generative ML or MAP Learning: *Naïve Bayes*

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

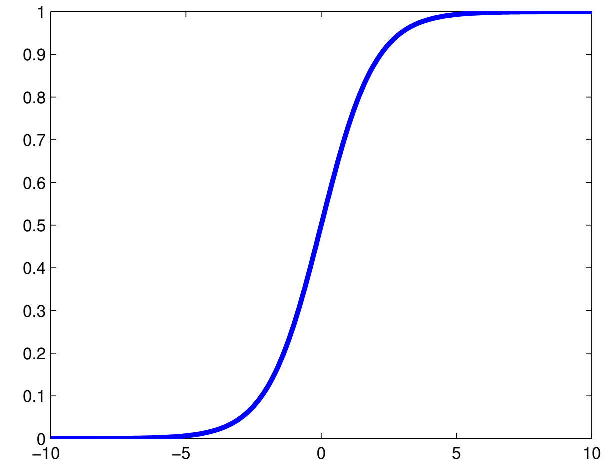


Discriminative ML or MAP Learning: *Logistic regression*

$$\max_{\theta} \; \log p(\theta) + \sum_{i=1}^{N} \log p(y_i \mid x_i, \theta)$$

Binary Logistic Regression

$$p(y_i \mid x_i, \theta) = \mathrm{Ber}(y_i \mid \mathrm{sigm}(\theta^T \phi(x_i)))$$



$$\mathrm{sigm}(\eta) := \frac{1}{1+\exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}$$

- Linear discriminant analysis:

$$\phi(x_i) = [1, x_{i1}, x_{i2}, \ldots, x_{id}]$$

- Quadratic discriminant analysis:

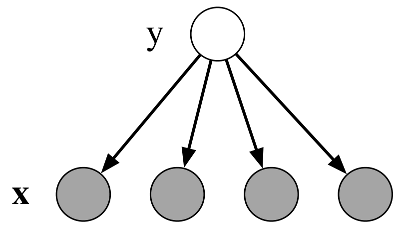
$$\phi(x_i) = [1, x_{i1}, \ldots, x_{id}, x_{i1}^2, x_{i1}x_{i2}, x_{i2}^2, \ldots]$$

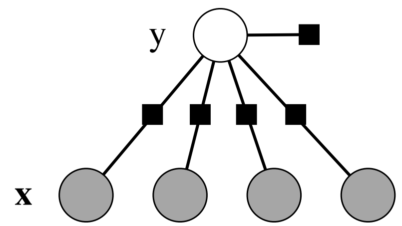
- Can derive weights from Gaussian generative model if that happens to be known, but more generally:
  - Choose any convenient feature set $\phi(x)$
  - Do discriminative Bayesian learning:

$$p(\theta \mid x, y) \propto p(\theta) \prod_{i=1}^{N} \mathrm{Ber}(y_i \mid \mathrm{sigm}(\theta^T \phi(x_i)))$$

Generative versus Discriminative

Generative ML or MAP Learning: *Naïve Bayes*

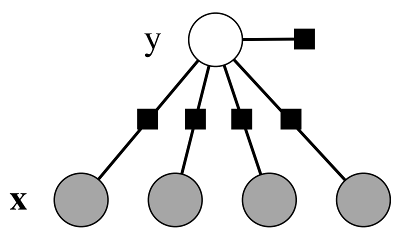




$$p(y, x) = p(y) \prod_{m=1}^{M} p(x_m \mid y)$$

- Class-specific distributions for each of *M* features

Discriminative ML or MAP Learning: *Logistic regression*



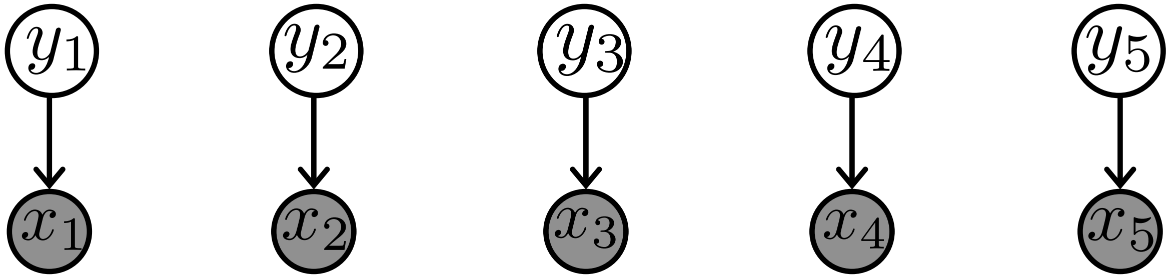
$$p(y = k \mid x, \theta) = \frac{1}{Z(x, \theta)} \prod_{m=1}^{M} \exp\left\{\theta_k^T \phi(x_m)\right\}$$

$$Z(x, \theta) = \sum_{k=1}^{K} \prod_{m=1}^{M} \exp\left\{\theta_k^T \phi(x_m)\right\}$$

- Exponential family distribution (maximum entropy classifier)
- Different distribution, and normalization constant, for each x

Mixture Models versus HMMs

Mixture Model

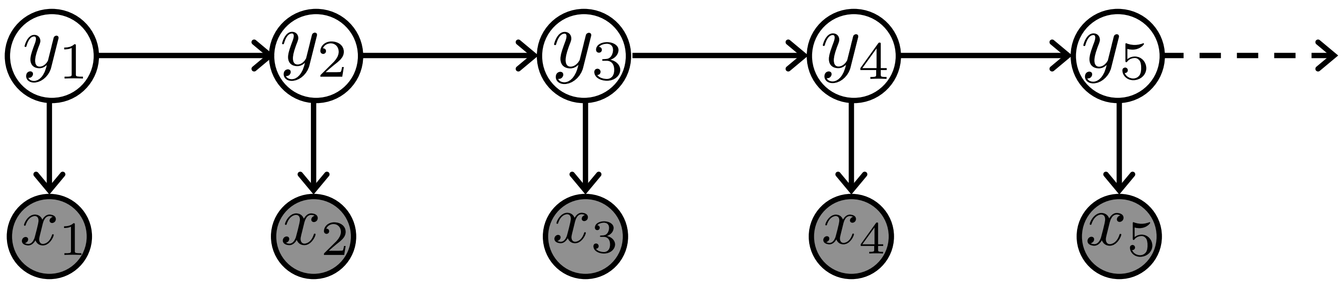


$y_i \in \{1, \ldots, K\}$

$$p(y_i \mid \pi, \theta) = \text{Cat}(y_i \mid \pi)$$

$$p(x_i \mid y_i = k, \pi, \theta) = \exp\{\theta_k^T \phi(x_i) - A(\theta_k)\}$$

Hidden Markov Model



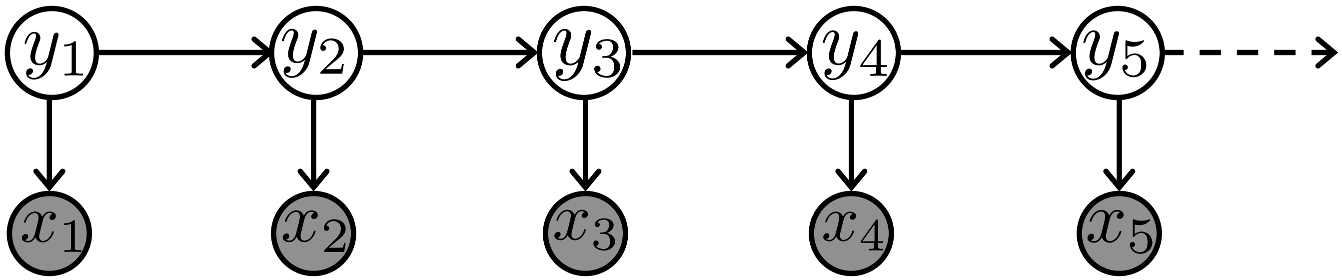
$$p(y_t \mid \pi, \theta, y_{t-1}, y_{t-2}, \ldots) = \text{Cat}(y_t \mid \pi_{y_{t-1}})$$

$$p(x_t \mid y_t = k, \pi, \theta) = \exp\{\theta_k^T \phi(x_t) - A(\theta_k)\}$$

Recover mixture model when all rows of state transition matrix are equal.

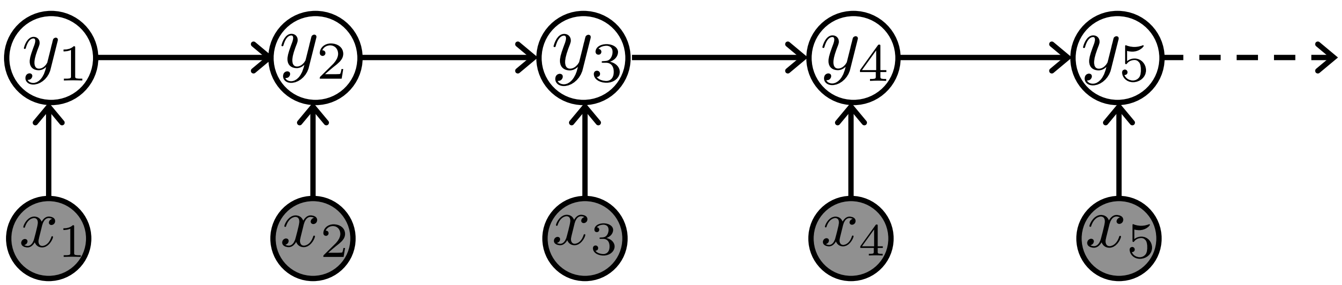
Modeling Sequential Data

Hidden Markov Model



$$p(y_t \mid \pi, \theta, y_{t-1}, y_{t-2}, \ldots) = \mathrm{Cat}(y_t \mid \pi_{y_{t-1}})$$
$$p(x_t \mid y_t = k, \pi, \theta) = \exp\{\theta_k^T \phi(x_t) - A(\theta_k)\}$$

"Maximum Entropy Markov Model"



$$p(y_t = k \mid y_{t-1}, x_t) = \exp\{\theta_k^T \phi(y_{t-1}, x_t) - A(y_{t-1}, x_t, \theta)\}$$

Most graphical models have equal claim to max-entropy terminology…

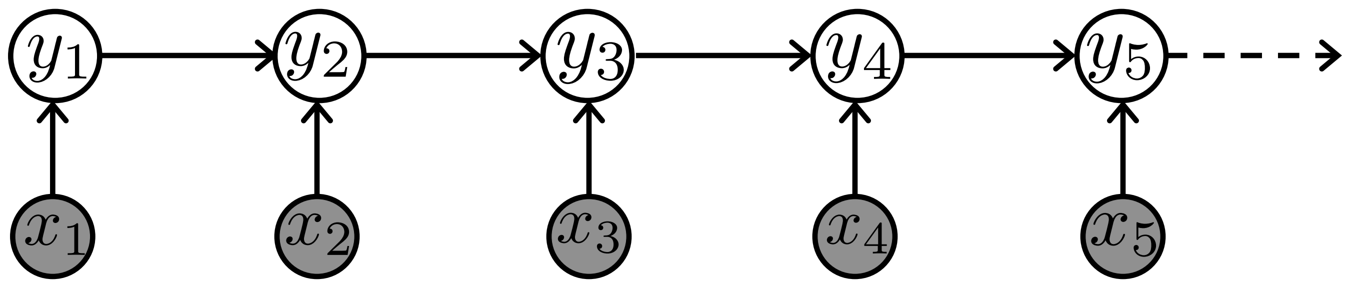
The "Label Bias" Problem

- Directed MEMM structure gives simple local learning problem: "classifier" for next state, given current state & observations
- But the MEMM structure has a major modeling weakness: *future observations provide no information about current state*

$$p(y_1 \mid x_1, x_2, x_3) = \sum_{y_2} \sum_{y_3} p(y_1 \mid x_1) p(y_2 \mid y_1, x_2) p(y_3 \mid y_2, x_3)$$

$$= p(y_1 \mid x_1) \left[\sum_{y_2} p(y_2 \mid y_1, x_2) \left[\sum_{y_3} p(y_3 \mid y_2, x_3) \right] \right]$$

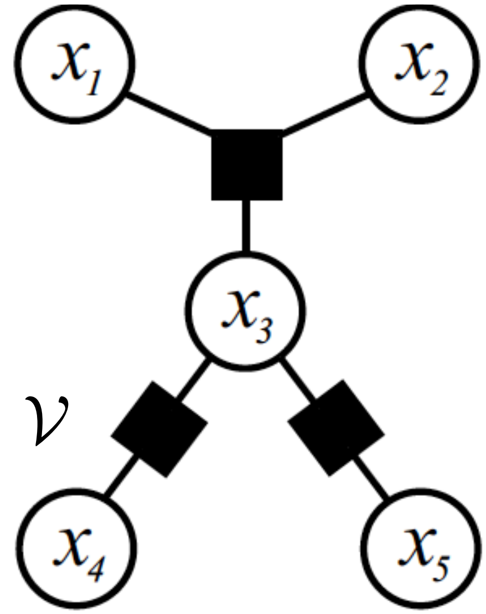
"Maximum Entropy Markov Model"



$$p(y_t = k \mid y_{t-1}, x_t) = \exp\{\theta_k^T \phi(y_{t-1}, x_t) - A(y_{t-1}, x_t, \theta)\}$$

Sum-product algorithm has uninformative backward messages

(Generative) Markov Random Fields



$$p(x \mid \theta) = \frac{1}{Z(\theta)} \prod_{f \in \mathcal{F}} \psi_f(x_f \mid \theta_f)$$

$$Z(\theta) = \sum_x \prod_{f \in \mathcal{F}} \psi_f(x_f \mid \theta_f)$$

$\mathcal{F} \longrightarrow$ set of hyperedges linking subsets of nodes $f \subseteq \mathcal{V}$

$\mathcal{V} \longrightarrow$ set of N nodes or vertices, $\{1, 2, \ldots, N\}$

- Assume an exponential family representation of each factor:

$$p(x \mid \theta) = \exp \left\{ \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(x_f) - A(\theta) \right\}$$

$$\psi_f(x_f \mid \theta_f) = \exp\{\theta_f^T \phi_f(x_f)\} \qquad A(\theta) = \log Z(\theta)$$

- Partition function *globally* couples the local factor parameters

(Discriminative) Conditional Random Fields

$$p(y \mid x, \theta) = \frac{1}{Z(\theta, x)} \prod_{f \in \mathcal{F}} \psi_f(y_f \mid x, \theta_f)$$

$$Z(\theta, x) = \sum_y \prod_{f \in \mathcal{F}} \psi_f(y_f \mid x, \theta_f)$$

- Assume an exponential family representation of each factor:

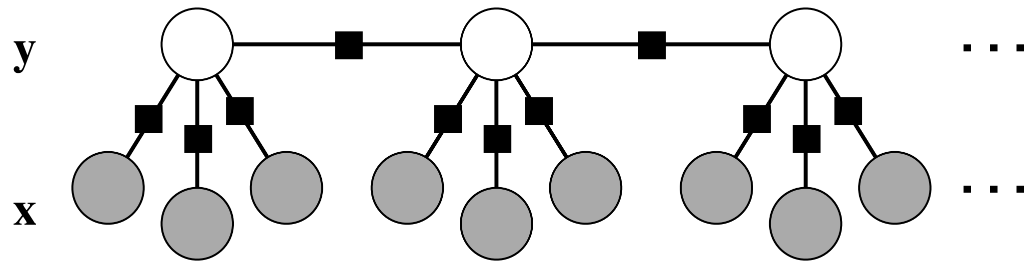
$$p(y \mid x, \theta) = \exp \left\{ \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(y_f, x) - A(\theta, x) \right\}$$

$$\psi_f(y_f \mid x, \theta_f) = \exp\{\theta_f^T \phi_f(y_f, x)\} \quad A(\theta, x) = \log Z(\theta, x)$$

- Log-probability is a linear function of fixed, possibly very complex features of input *x* and output *y*
- Partition function *globally* couples the local factor parameters, and every training example has a different normalizer

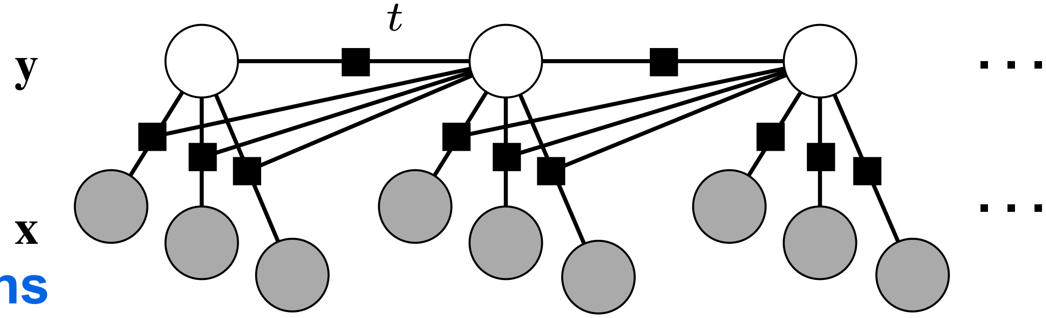
CRF Models for Sequential Data

Direct Extension of HMM



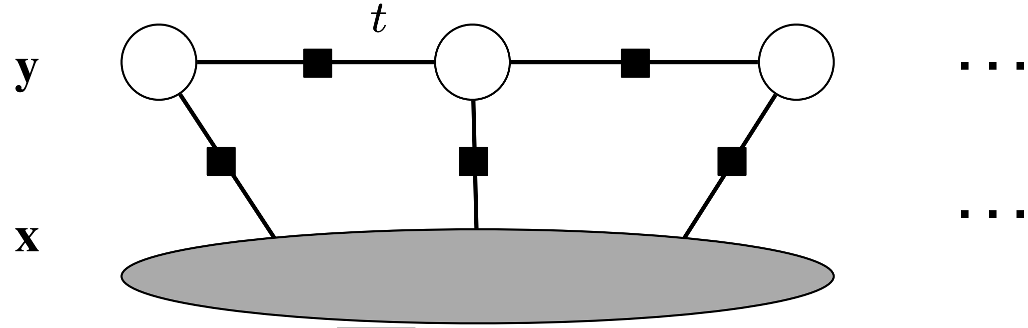
$$p(y \mid x) \propto \prod_t \psi_t(y_t, x_t)\psi_{t,t+1}(y_t, y_{t+1})$$

State Transitions depend on Observations



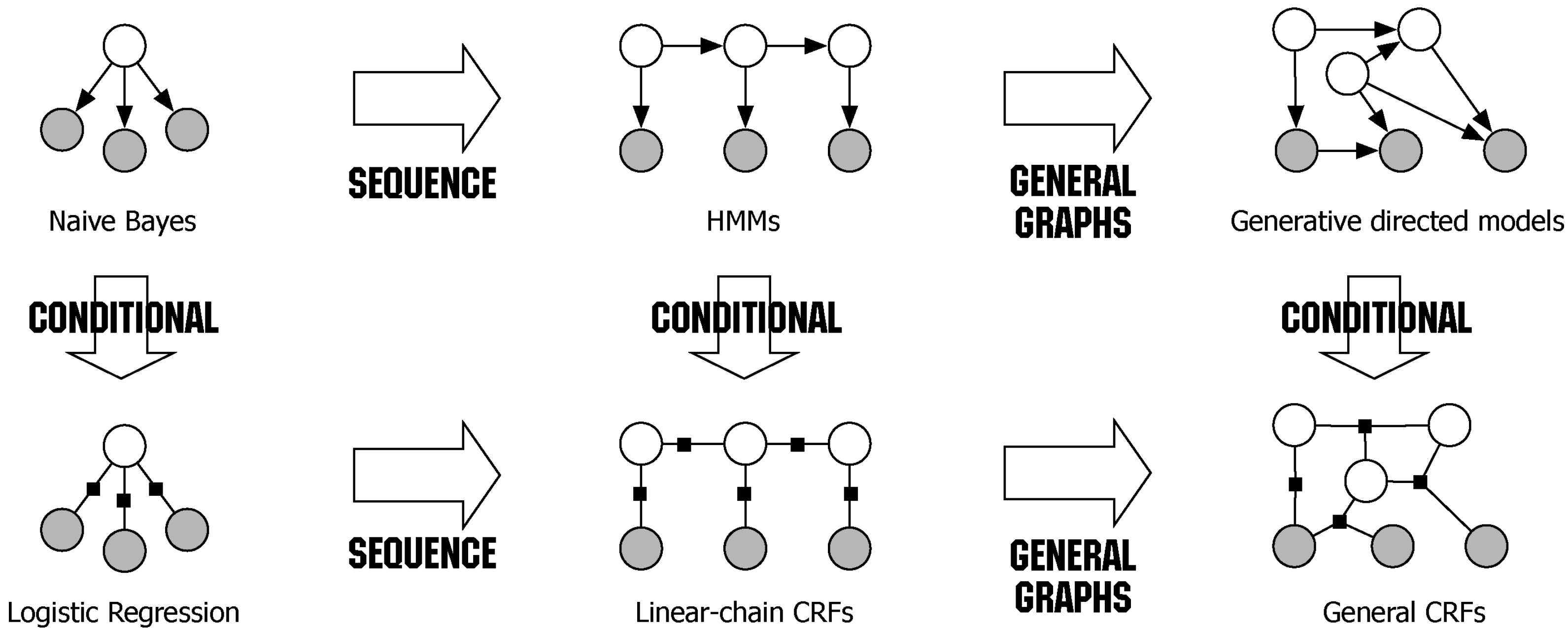
$$p(y \mid x) \propto \prod_t \psi_t(y_t, x_t)\psi_{t,t+1}(y_t, y_{t+1}, x_t)$$

Arbitrary Non-Local Features



$$p(y \mid x) \propto \prod_t \psi_t(y_t, x)\psi_{t,t+1}(y_t, y_{t+1}, x)$$

Families of Graphical Models



$$p(y \mid \theta) = \exp\left\{ \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(y_f) - A(\theta) \right\}$$

$$p(y \mid x, \theta) = \exp\left\{ \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(y_f, x) - A(\theta, x) \right\}$$

With informative observations (good features), the posterior may have a simpler graph (Markov) structure than the prior

Generative Learning for MRFs

- Undirected graph encodes dependencies within a single training example:

$$p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} \frac{1}{Z(\theta)} \prod_{f \in \mathcal{F}} \psi_f(x_{f,n} \mid \theta_f) \qquad \mathcal{D} = \{x_{\mathcal{V},1}, \ldots, x_{\mathcal{V},N}\}$$

- Given N independent, identically distributed, completely observed samples:

$$\log p(\mathcal{D} \mid \theta) = \left[\sum_{n=1}^{N} \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(x_{f,n}) \right] - N A(\theta)$$

- Take gradient with respect to parameters for a single factor:

$$\nabla_{\theta_f} \log p(\mathcal{D} \mid \theta) = \left[\sum_{n=1}^{N} \phi_f(x_{f,n}) \right] - N \mathbb{E}_\theta[\phi_f(x_f)]$$

- Must be able to compute *prior marginal distributions* for factors
- At each gradient step, must solve a single inference problem to find the *marginal statistics of the prior distribution*

Discriminative Learning for CRFs

- Undirected graph encodes dependencies within a single training example:

$$p(y \mid x, \theta) = \prod_{n=1}^{N} \frac{1}{Z(\theta, x_{\mathcal{V},n})} \prod_{f \in \mathcal{F}} \psi_f(y_{f,n} \mid x_{\mathcal{V},n}, \theta_f)$$

- Given N independent, identically distributed, completely observed samples:

$$\log p(y \mid x, \theta) = \sum_{n=1}^{N} \left[\sum_{f \in \mathcal{F}} \theta_f^T \phi_f(y_{f,n}, x_{\mathcal{V},n}) - A(\theta, x_{\mathcal{V},n}) \right]$$

- Take gradient with respect to parameters for a single factor:

$$\nabla_{\theta_f} \log p(y \mid x, \theta) = \sum_{n=1}^{N} \left[\phi_f(y_{f,n}, x_{\mathcal{V},n}) - \mathbb{E}_\theta[\phi_f(y_f, x_{\mathcal{V},n}) \mid x_{\mathcal{V},n}] \right]$$

- Must be able to compute *conditional marginal distributions* for factors
- At each gradient step, must solve a N inference problems to find the *conditional marginal statistics of the posterior for every training example*

Convex Conditional Likelihood Surrogates

- To train CRF models on graphs with cycles:

$$\log p(y \mid x, \theta) = \sum_{n=1}^{N} \left[\sum_{f \in \mathcal{F}} \theta_f^T \phi_f(y_{f,n}, x_{\mathcal{V},n}) - A(\theta, x_{\mathcal{V},n}) \right]$$

$$\log p(y \mid x, \theta) \geq \sum_{n=1}^{N} \left[\sum_{f \in \mathcal{F}} \theta_f^T \phi_f(y_{f,n}, x_{\mathcal{V},n}) - B(\theta, x_{\mathcal{V},n}) \right]$$

  for a convex bound satisfying $A(\theta, x) \leq B(\theta, x)$

- Apply tree-reweighted Bethe variational bound:

$$\nabla_{\theta_f} \log p(y \mid x, \theta) = \sum_{n=1}^{N} \left[\phi_f(y_{f,n}, x_{\mathcal{V},n}) - \mathbb{E}_{\tau}[\phi_f(y_f, x_{\mathcal{V},n}) \mid x_{\mathcal{V},n}] \right]$$

- Gradients depend on expectations of *pseudo-marginals* produced by applying tree-reweighted BP with current model parameters, for features of *every training example*

Inference in Graphical Models

$x_E \longrightarrow$ observed *evidence* variables (subset of nodes)

$x_F \longrightarrow$ unobserved *query* nodes we'd like to infer

$x_R \longrightarrow$ remaining variables, *extraneous* to this query but part of the given graphical representation

$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R) \qquad R = V \setminus \{E, F\}$$

Maximum a Posteriori (MAP) Estimates

$$\hat{x}_F = \arg\max_{x_F} p(x_F \mid x_E) = \arg\max_{x_F} p(x_E, x_F)$$

Posterior Marginal Densities

$$p(x_F \mid x_E) = \frac{p(x_E, x_F)}{p(x_E)} \qquad p(x_E) = \sum_{x_F} p(x_E, x_F)$$

Provides Bayesian estimators, confidence measures, and sufficient statistics for iterative parameter estimation

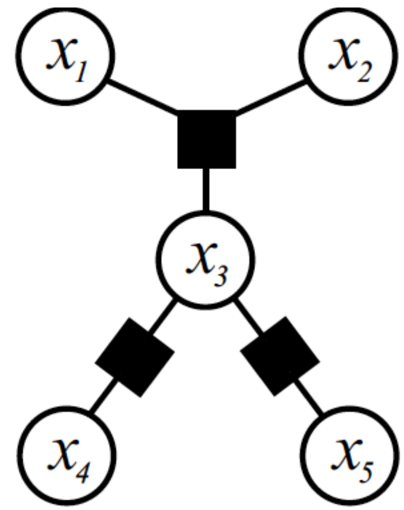
Global versus Local MAP Estimation

Maximum a Posteriori (MAP) Estimates

$$\hat{x}_F = \arg\max_{x_F} p(x_F \mid x_E) = \arg\max_{x_F} p(x_E, x_F)$$

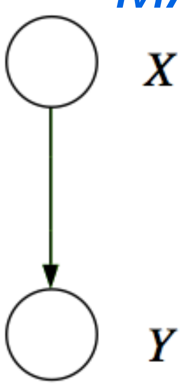
Maximizer of Posterior Marginals (MPM) Estimates

$$p(x_s \mid x_E) = \sum_{x_{F \setminus s}} p(x_F \mid x_E)$$



$$\hat{x}_s = \arg\max_{x_s} p(x_s \mid x_E)$$

MAP and MPM estimators are not equivalent



1	.6
2	.2
3	.2

$p(x)$

	1	2	3
1	0	.6	.4
2	1	0	0
3	1	0	0

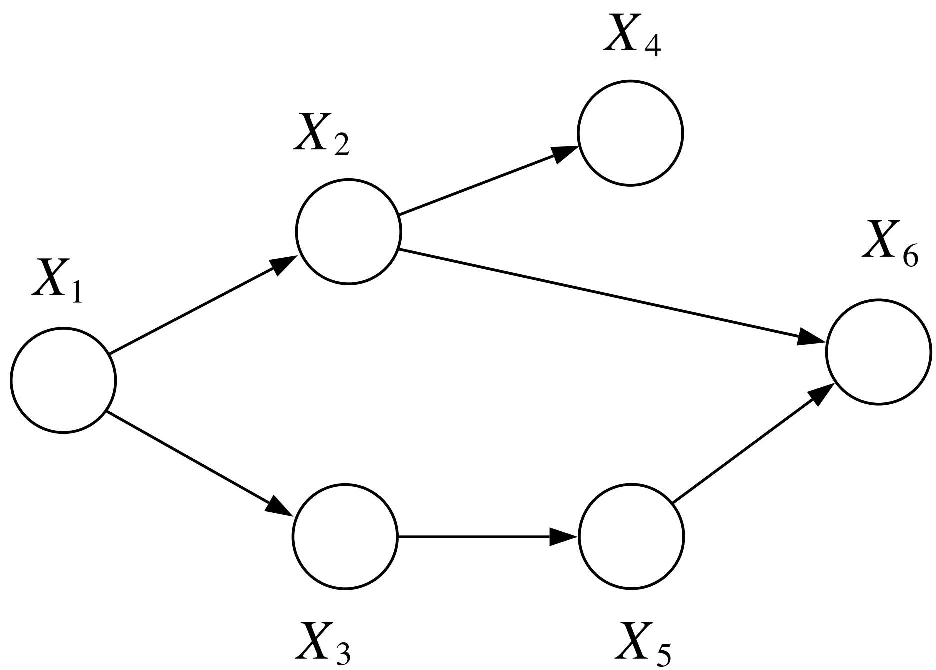
$p(y \mid x)$

1	2	3
.4	.36	.24

$p(y)$

MPM: (1,1)
MAP: (1,2)

MAP in Directed Graphs



$$p(x) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$$

$$\max_x p(x) = \max_{x_1} \max_{x_2} \max_{x_3} \max_{x_4} \max_{x_5} \max_{x_6} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$$

$$= \max_{x_1} p(x_1) \max_{x_2} p(x_2 \mid x_1) \max_{x_3} p(x_3 \mid x_1) \max_{x_4} p(x_4 \mid x_2) \max_{x_5} p(x_5 \mid x_3) \max_{x_6} p(x_6 \mid x_2, x_5)$$

A MAP Elimination Algorithm

Algebraic Maximization Operations

- Determine maximal setting of variable being eliminated, *for every possible configuration of its neighbors*
- Compute a new potential table involving all other variables which depend on the just-marginalized variable

Graph Manipulation Operations

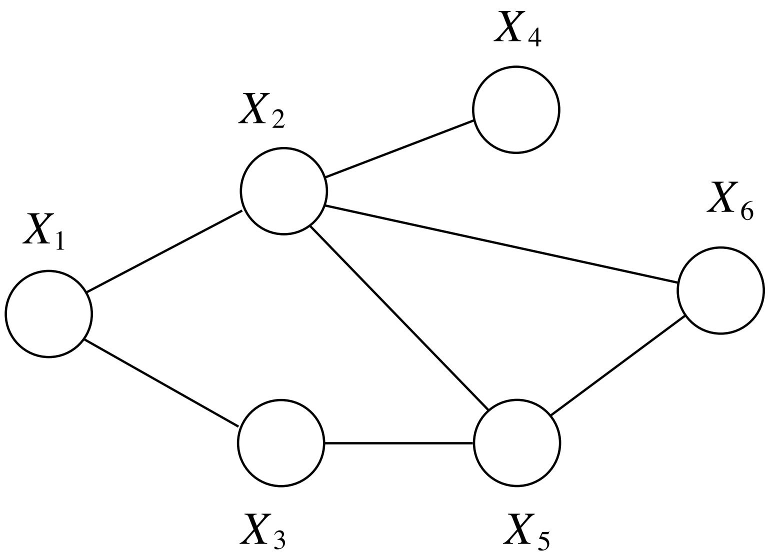
- Remove, or *eliminate*, a single node from the graph
- Add edges (if they don't already exist) between all pairs of nodes who were neighbors of the just-removed node

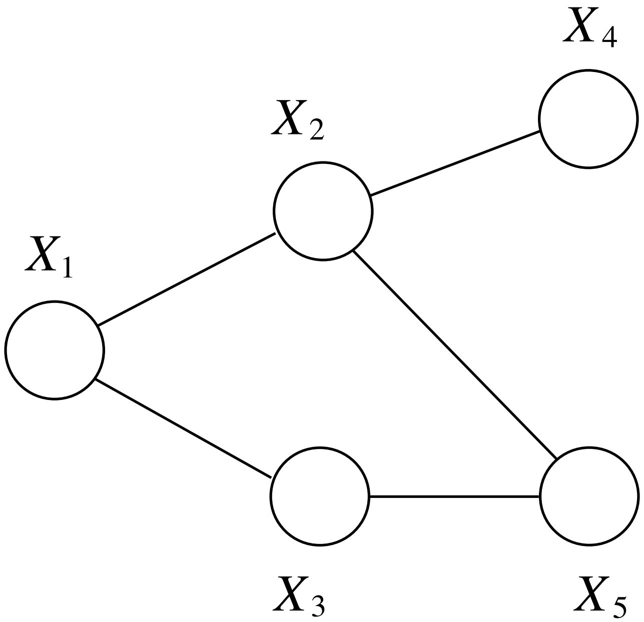
A Graph Elimination Algorithm

- Choose an elimination ordering
- Eliminate a node, remove its incoming edges, add edges between all pairs of its neighbors
- Iterate until all non-observed nodes are eliminated

Graph Elimination Example

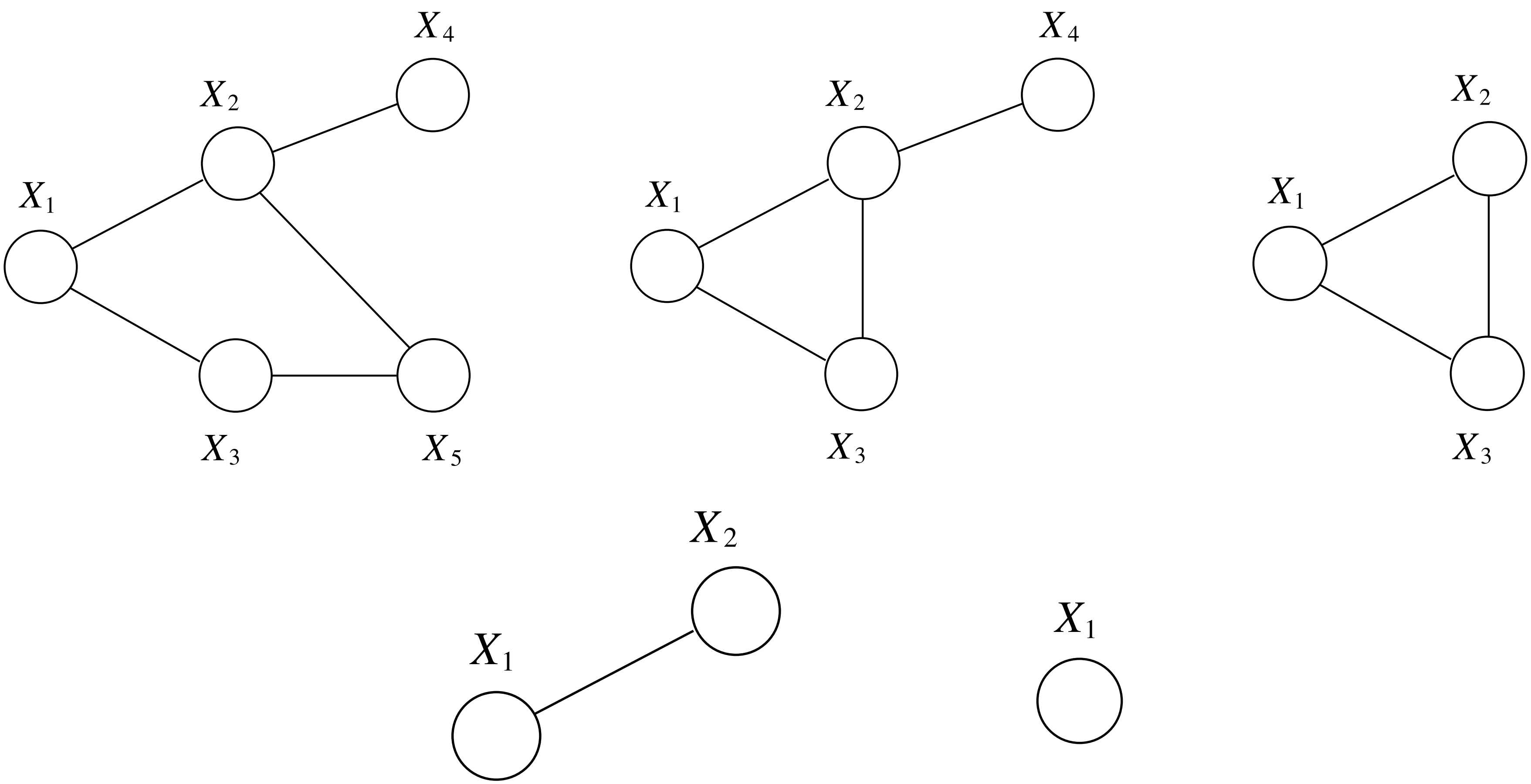
Elimination Order: (6,5,4,3,2,1)



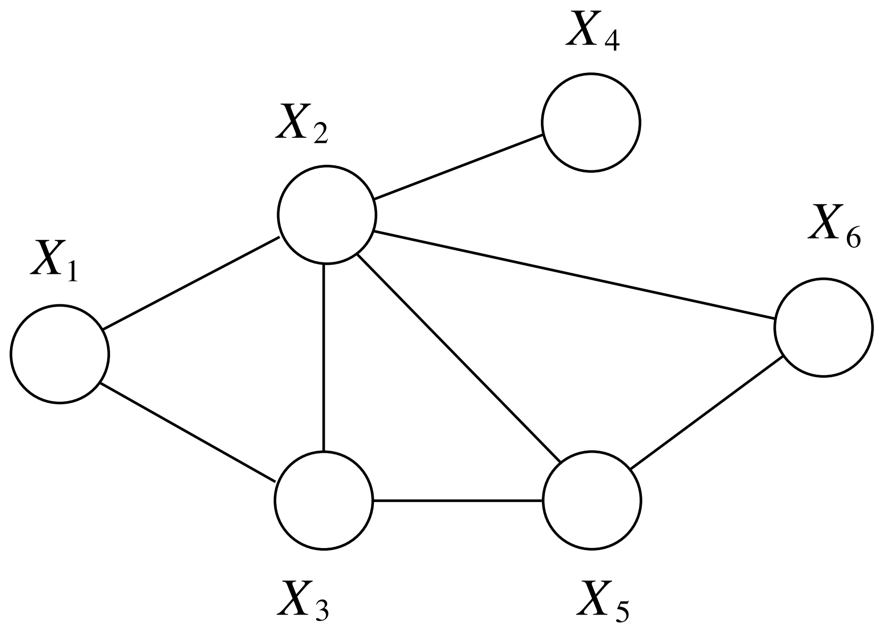


Graph Elimination Example

Elimination Order: (6,5,4,3,2,1)



Elimination Algorithm Complexity



- *Elimination cliques:* Sets of neighbors of eliminated nodes
- *Maximization cost:* Exponential in number of variables in each elimination clique (dominated by largest clique)
- *Treewidth of graph*: Over all possible elimination orderings, the smallest possible max-elimination-clique size, minus one
- *NP-Hard:* Finding the best elimination ordering for an arbitrary input graph (but heuristic algorithms often effective)

The Generalized Distributive Law

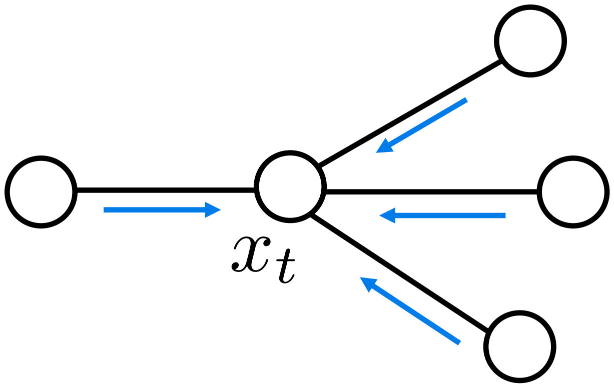
- A commutative semiring is a pair of generalized *"multiplication"* and *"addition"* operations which satisfy:
  Commutative: $a + b = b + a$ $\quad a \cdot b = b \cdot a$
  Associative: $a + (b + c) = (a + b) + c$ $\quad a \cdot (b \cdot c) = (a \cdot b) \cdot c$
  Distributive: $a \cdot (b + c) = a \cdot b + a \cdot c$
  (Why not a *ring*? May be no additive/multiplicative inverses.)
- Examples:

Addition	**Multiplication**
sum	product
max	product
max	sum
min	sum

- For each of these cases, our factorization-based dynamic programming derivation of belief propagation is still valid
- Leads to *max-product and min-sum belief propagation* algorithms for exact MAP estimation in trees

Belief Propagation (Max-Product)

Max-Marginals:

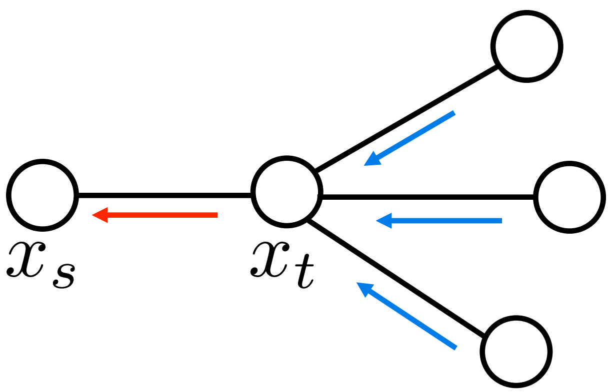


$$\hat{p}_t(x_t) \propto \psi_t(x_t) \prod_{u \in \Gamma(t)} m_{ut}(x_t)$$

$$\hat{p}_t(x_t) \propto \arg\max_x p(x) \mathbb{I}(X_t = x_t)$$

MESSAGES:

$$m_{ts}(x_s) \propto \max_{x_t} \psi_{st}(x_s, x_t) \psi_t(x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t)$$



- If MAP is unique, find via arg-max of each max-marginal independently
- Otherwise, must backtrack from some chosen root node