# Probabilistic Graphical Models

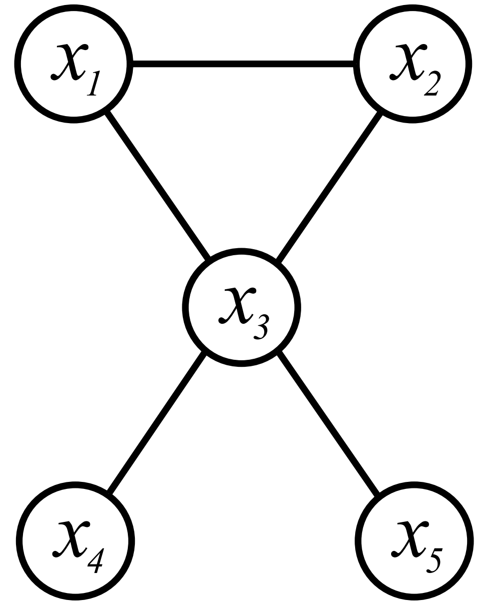
Brown University CSCI 2950-P, Spring 2013
Prof. Erik Sudderth

Lecture 6:
Sum-Product Inference for Factor Graphs,
Learning Directed Graphical Models

Some figures courtesy Michael Jordan's draft textbook,
*An Introduction to Probabilistic Graphical Models*

# Pairwise Markov Random Fields

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




- Simple parameterization, but still expressive and widely used in practice
- Guaranteed Markov with respect to graph

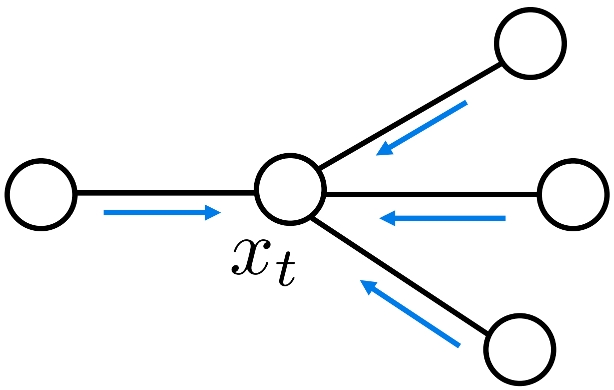
$\mathcal{E} \longrightarrow$ set of undirected edges $(s,t)$ linking pairs of nodes

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

$Z \longrightarrow$ normalization constant (partition function)

# Belief Propagation (Sum-Product)

**BELIEFS:** Posterior marginals

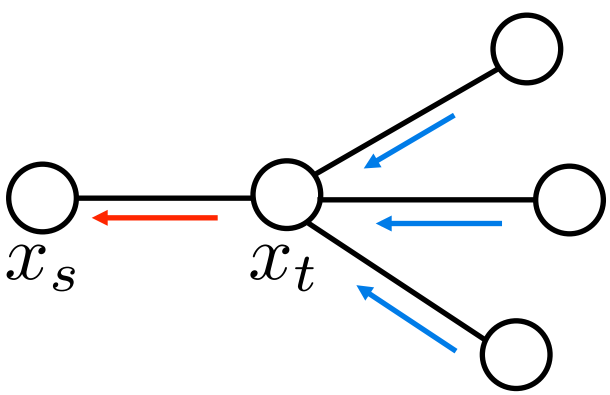



$$\hat{p}_t(x_t) \propto \psi_t(x_t) \prod_{u \in \Gamma(t)} m_{ut}(x_t)$$

$\Gamma(t) \longrightarrow$ neighborhood of node t (adjacent nodes)

**MESSAGES:** Sufficient statistics

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




I) Message Product

II) Message Propagation

# Belief Propagation for Trees

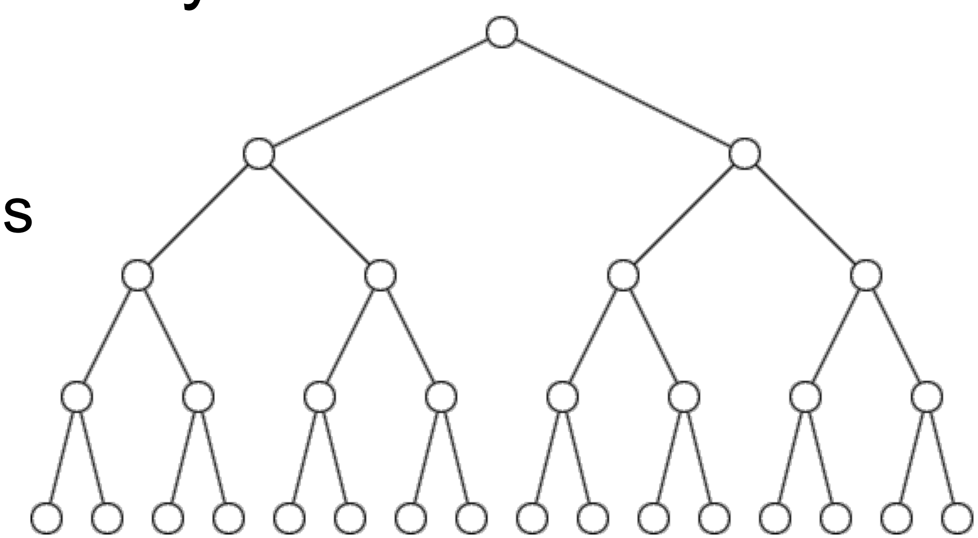
- Dynamic programming algorithm which exactly computes all marginals
- On Markov chains, BP equivalent to alpha-beta or forward-backward algorithms for HMMs
- Sequential message schedules require each message to be updated only once
- Computational cost:

$N \longrightarrow$ number of nodes

$M \longrightarrow$ discrete states for each node

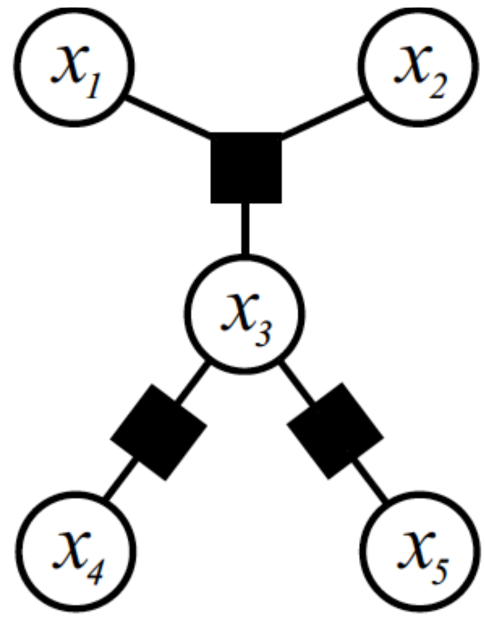
Belief Prop: $\mathcal{O}(NM^2)$

Brute Force: $\mathcal{O}(M^N)$



# Factor Graphs

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




- 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*

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

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

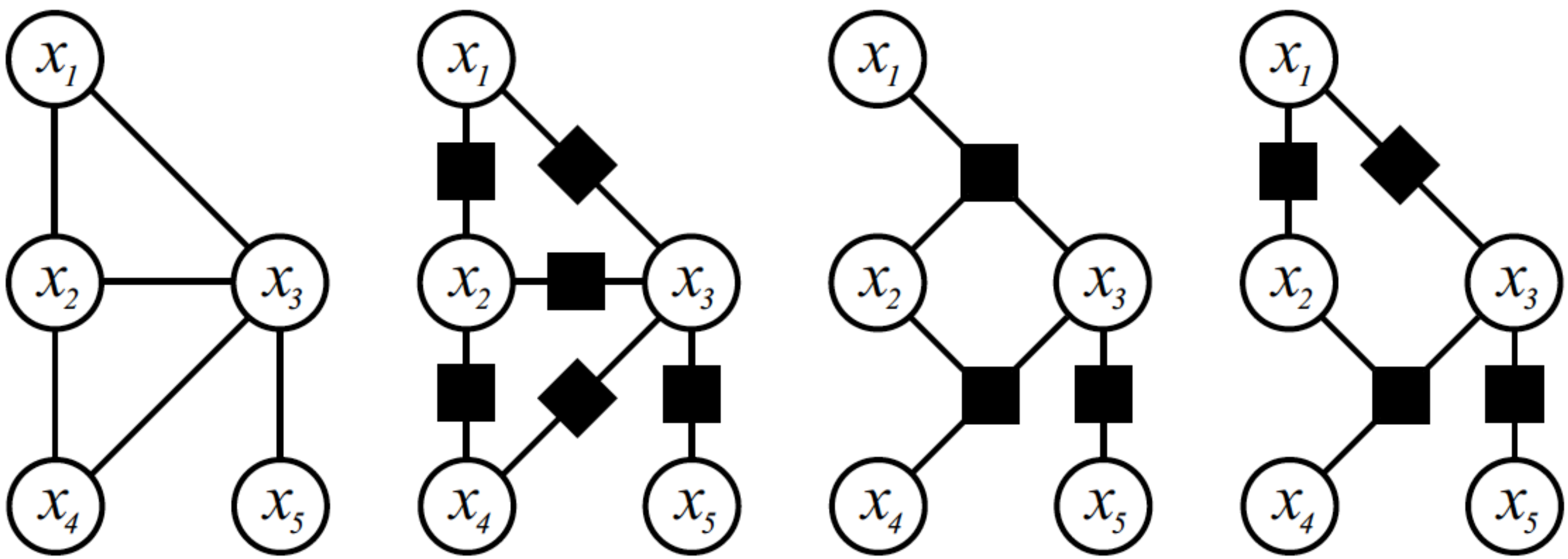
$Z \longrightarrow$ normalization constant (partition function)

# Factor Graphs & Factorization

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

- For a given undirected graph, there exist distributions which have equivalent Markov properties, but different factorizations and different inference/learning complexities:

*Undirected Graphical Model* | *Pairwise (edge) Potentials* | *Potentials on Maximal Cliques* | *Alternative Factorization*



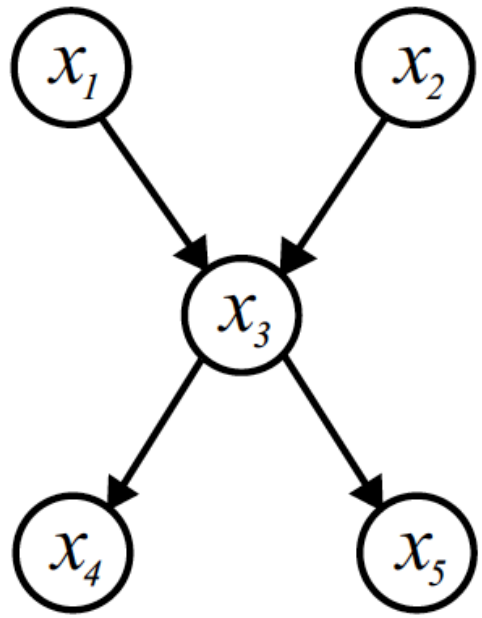
# Directed Graphs as Factor Graphs

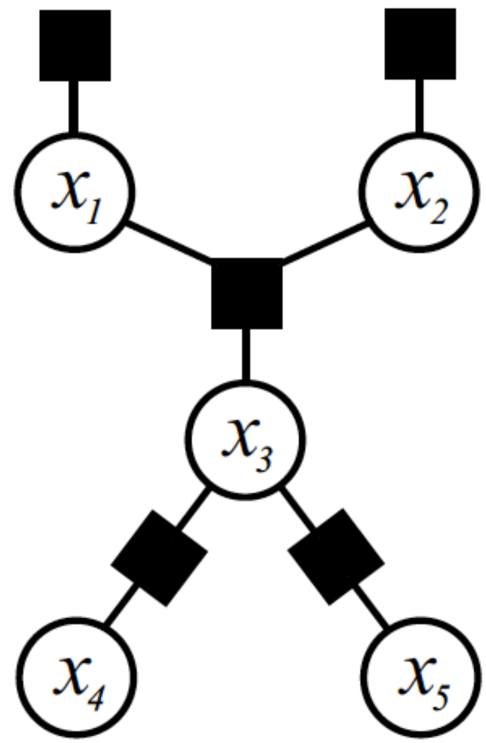
*Directed Graphical Model:*

$$p(x) = \prod_{i=1}^{N} p(x_i \mid x_{\Gamma(i)})$$

*Corresponding Factor Graph:*

$$p(x) = \prod_{i=1}^{N} \psi_i(x_i, x_{\Gamma(i)})$$

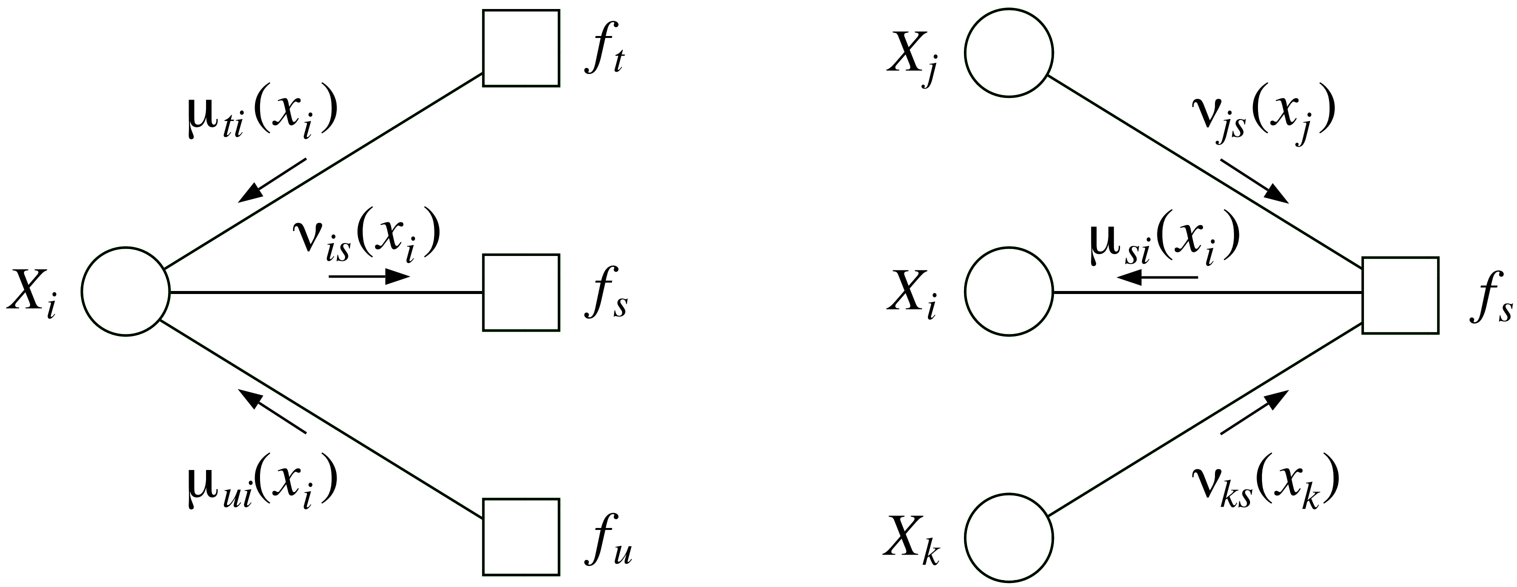



- Associate one factor with each node, linking it to its parents and defined to equal the corresponding conditional distribution
- Information lost: Directionality of conditional distributions, and fact that global partition function $Z = 1$

# Sum-Product Algorithm

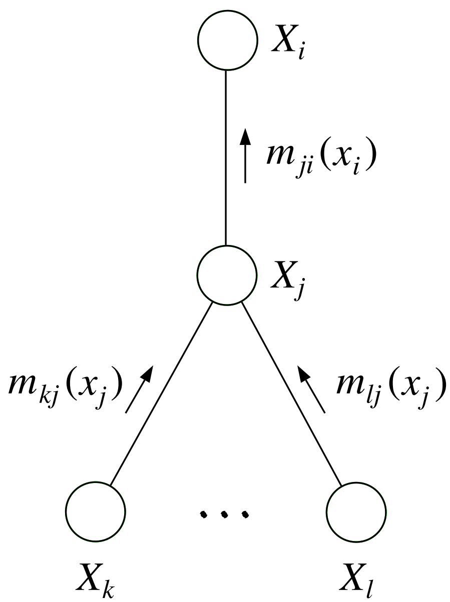
*Belief Propagation for Factor Graphs*

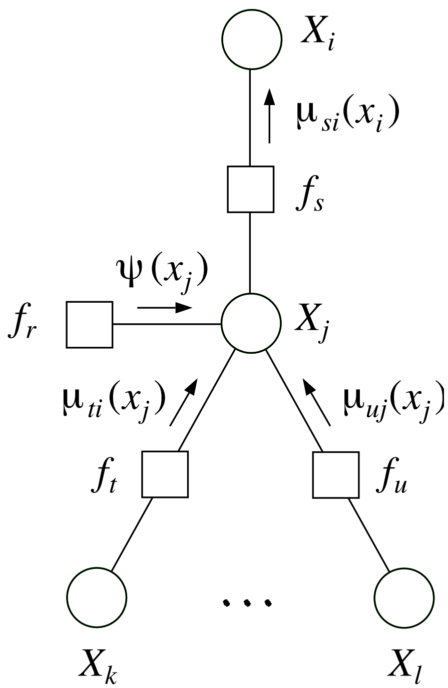



$$\nu_{is}(x_i) = \prod_{t \in \mathcal{N}(i) \setminus s} \mu_{ti}(x_i) \qquad \mu_{si}(x_i) = \sum_{x_{\mathcal{N}(s) \setminus i}} \left( f_s(x_{\mathcal{N}(s)}) \prod_{j \in \mathcal{N}(s) \setminus i} \nu_{js}(x_j) \right)$$

- From each variable node, the incoming and outgoing messages are functions only of that particular variable
- Factor message updates must sum over all *combinations* of the adjacent variable nodes (exponential in degree)

# Comparing Sum-Product Variants



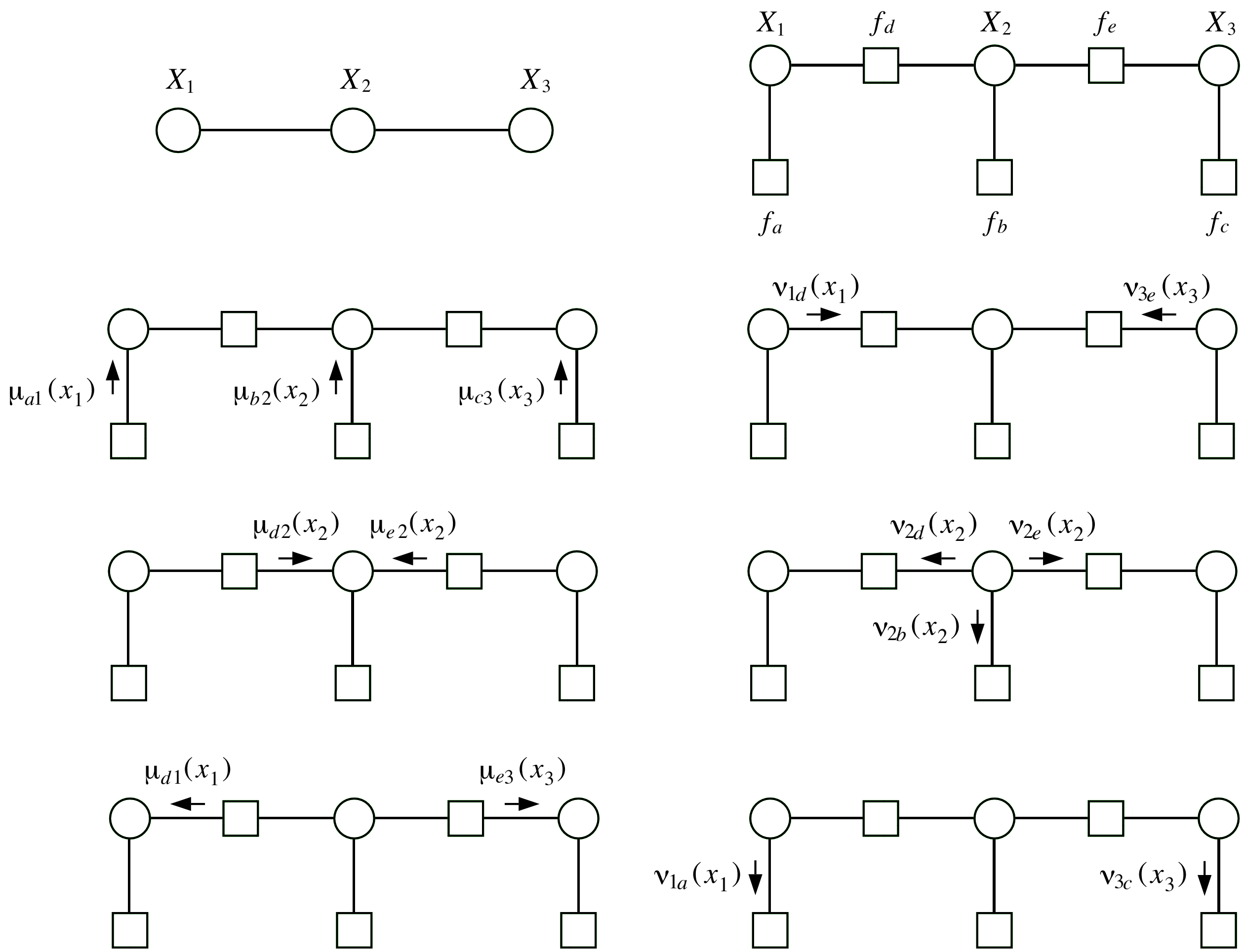

$$
\begin{aligned}
\mu_{si}(x_i) &= \sum_{x_{\mathcal{N}(s)\setminus i}} \left( f_s(x_{\mathcal{N}(s)}) \prod_{j\in\mathcal{N}(s)\setminus i} \nu_{js}(x_j) \right) \\
&= \sum_{x_j} \psi(x_i, x_j)\nu_{js}(x_j) \\
&= \sum_{x_j} \psi(x_i, x_j) \prod_{t\in\mathcal{N}(j)\setminus s} \mu_{tj}(x_j) \\
&= \sum_{x_j} \left( \psi^E(x_j)\psi(x_i, x_j) \prod_{t\in\mathcal{N}'(j)\setminus s} \mu_{tj}(x_j) \right)
\end{aligned}
$$

- For pairwise potentials, there is one "incoming" message for each outgoing factor message, simplifies to earlier algorithm:

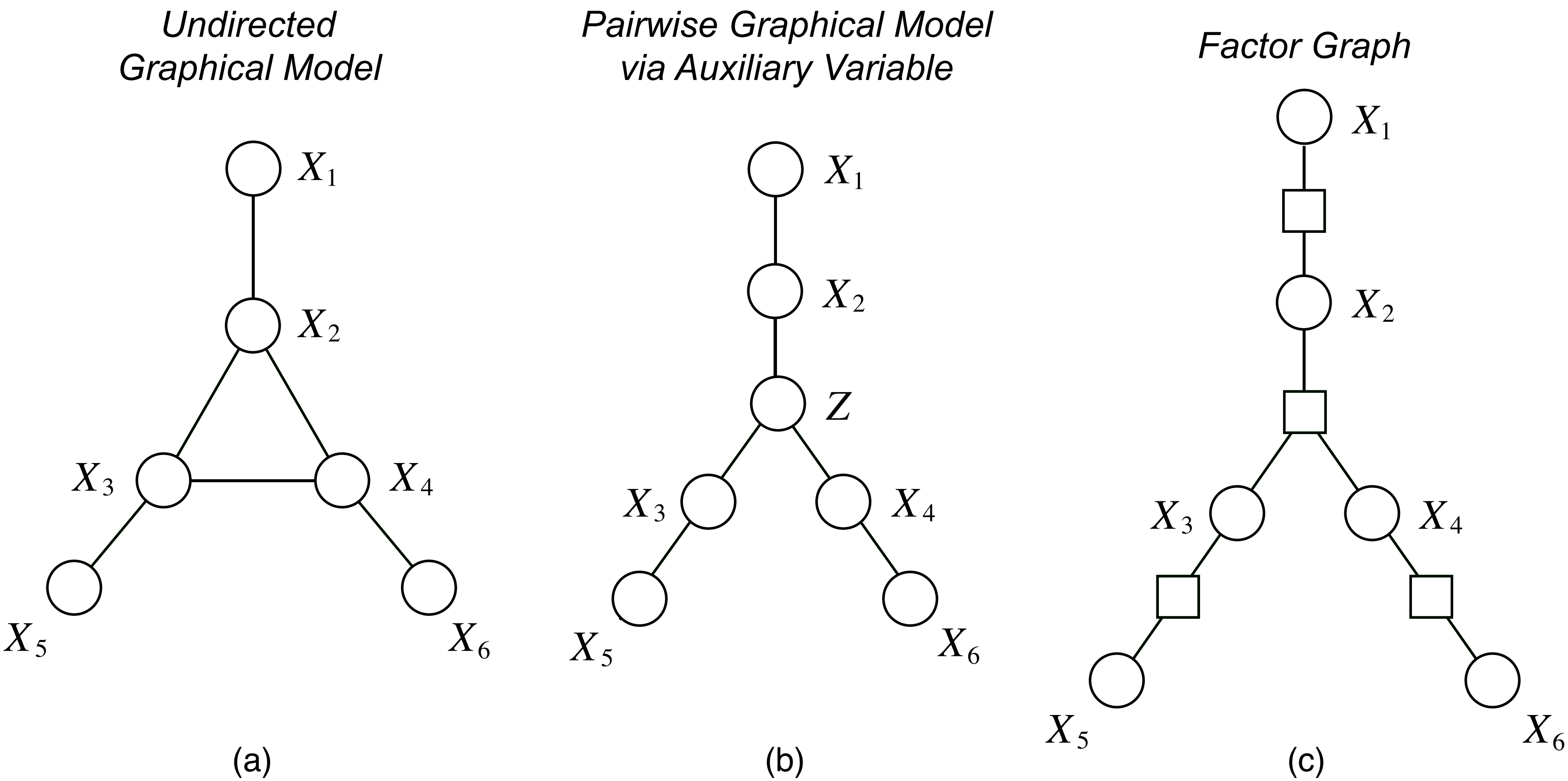
$$
m_{ts}(x_s) \propto \sum_{x_t} \psi_{st}(x_s, x_t)\psi_t(x_t) \prod_{u\in\Gamma(t)\setminus s} m_{ut}(x_t)
$$

# Factor Graph Message Schedules

- All of the previously discussed message schedules are valid
- Here is an example of a synchronous parallel schedule:



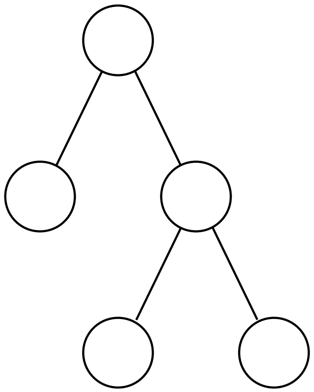
# Sum-Product for “Nearly” Trees



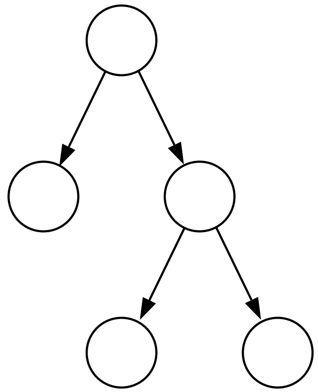

- Sum-product algorithm computes exact marginal distributions for any factor graph which is tree-structured (no cycles)
- This includes some undirected graphs with cycles

# Sum-Product for Polytrees

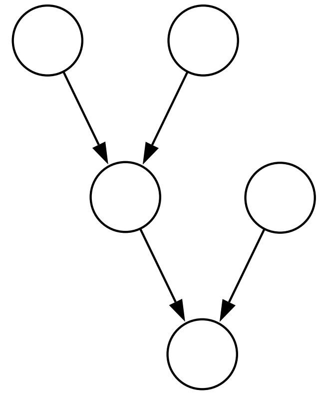
*Undirected Tree*



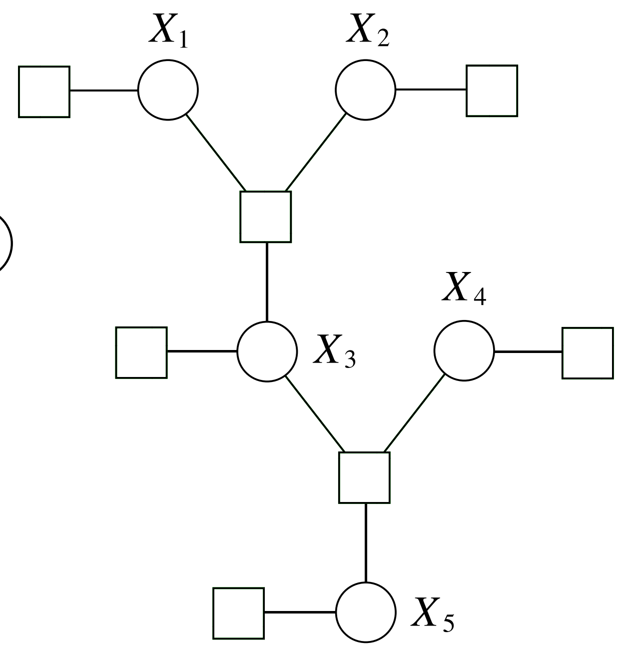
*Directed Tree*



*Directed Polytree*

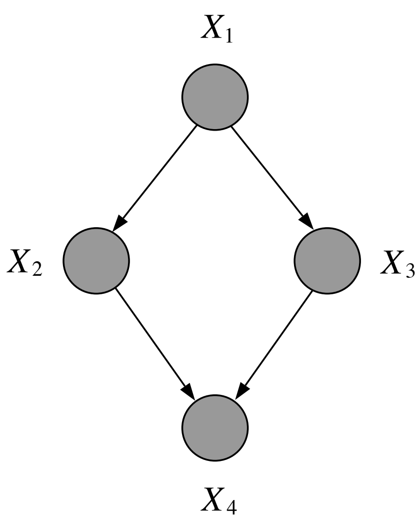


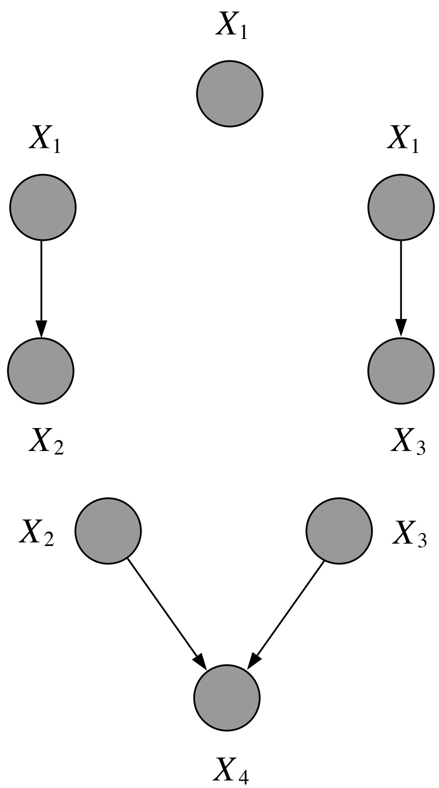
*Factor Graph*




- Early work on belief propagation (Pearl, 1980’s) focused on directed graphical models, and was complicated by directionality of edges and multiple parents (polytrees)
- Factor graph framework makes this a simple special case

# Learning Directed Graphical Models




$$p(x) = \prod_{i \in \mathcal{V}} p(x_i \mid x_{\Gamma(i)}, \theta_i)$$

*Intuition: Must learn a good predictive model of each node, given its parent nodes*

- Directed factorization causes likelihood to locally decompose:

$$p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_2) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_4)$$

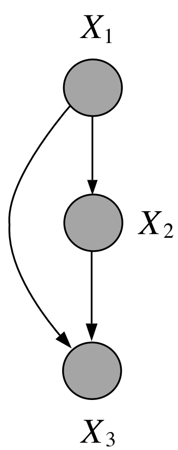
$$\log p(x \mid \theta) = \log p(x_1 \mid \theta_1) + \log p(x_2 \mid x_1, \theta_2) + \log p(x_3 \mid x_1, \theta_3) + \log p(x_4 \mid x_2, x3, \theta_4)$$

- Often paired with a correspondingly factorized prior:

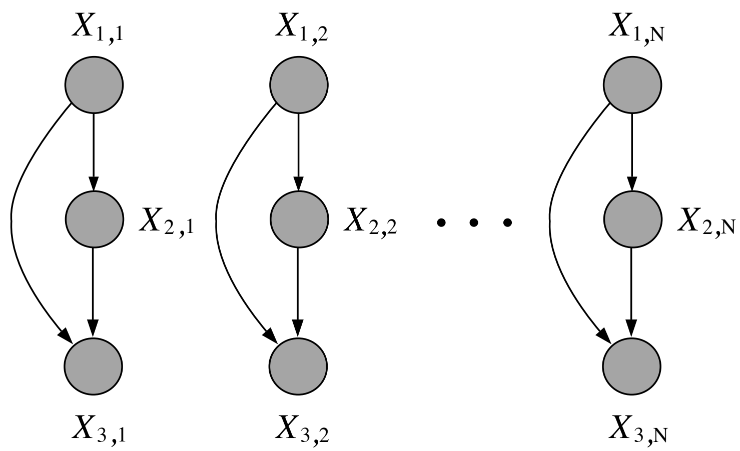
$$p(\theta) = p(\theta_1) p(\theta_2) p(\theta_3) p(\theta_4)$$

$$\log p(\theta) = \log p(\theta_1) + \log p(\theta_2) + \log p(\theta_3) + \log p(\theta_4)$$

# Complete Observations




*Directed Graphical Model*




*N Independent, Identically Distributed Training Examples*

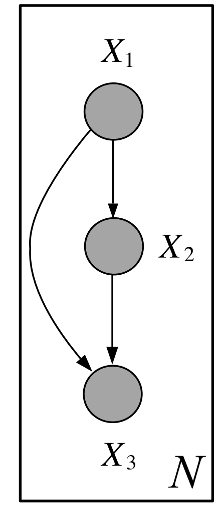



*Plate Notation*

- A directed graphical model encodes assumed statistical dependencies among the different parts of a single training example:

$$p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} \prod_{i \in \mathcal{V}} p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) \qquad \mathcal{D} = \{x_{\mathcal{V},1}, \ldots, x_{\mathcal{V},N}\}$$

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

$$\log p(\mathcal{D} \mid \theta) = \sum_{n=1}^{N} \sum_{i \in \mathcal{V}} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) = \sum_{i \in \mathcal{V}} \left[ \sum_{n=1}^{N} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) \right]$$

# Priors and Tied Parameters

$$\log p(\mathcal{D} \mid \theta) = \sum_{n=1}^{N} \sum_{i \in \mathcal{V}} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) = \sum_{i \in \mathcal{V}} \left[ \sum_{n=1}^{N} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) \right]$$

$$\log p(\theta) = \sum_{i \in \mathcal{V}} p(\theta_i)$$

*A "meta-independent" factorized prior*

- Factorized posterior allows independent learning for each node:

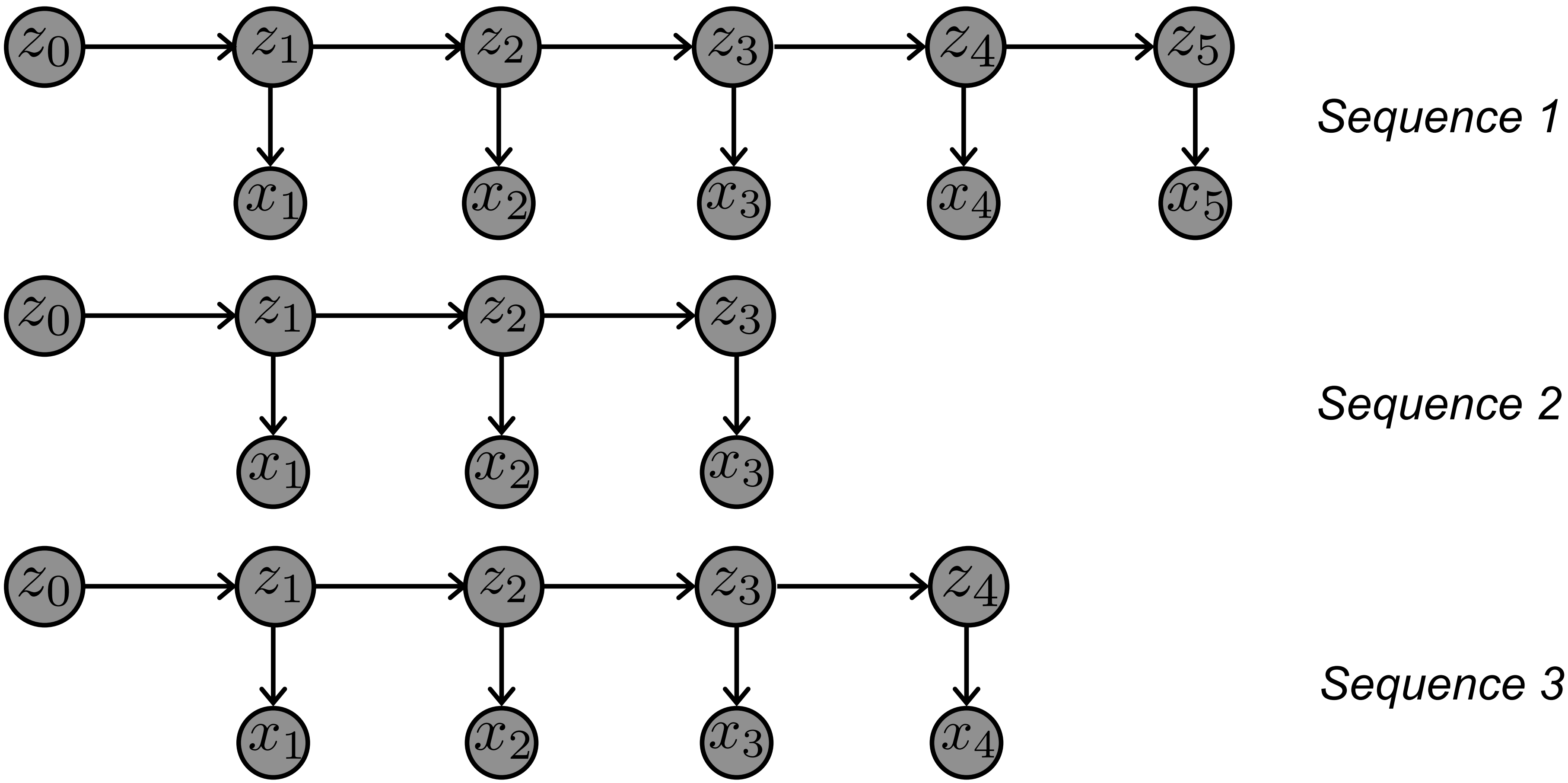
$$\log p(\theta \mid \mathcal{D}) = C + \sum_{i \in \mathcal{V}} \left[ \log p(\theta_i) + \sum_{n=1}^{N} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) \right]$$

- Learning remains tractable when subsets of nodes are "tied" to use identical, shared parameter values:

$$\log p(\mathcal{D} \mid \theta) = \sum_{i \in \mathcal{V}} \left[ \sum_{n=1}^{N} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_{b_i}) \right]$$

$$\log p(\theta_b \mid \mathcal{D}) = C + \log p(\theta_b) + \sum_{i \mid b_i = b} \sum_{n=1}^{N} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_b)$$

# Example: Temporal Models




$$p(x, z \mid \theta) = \prod_{n=1}^{N} \prod_{t=1}^{T_n} p(z_{t,n} \mid z_{t-1,n}, \theta_{\text{time}}) p(x_{t,n} \mid z_{t,n}, \theta_{\text{obs}})$$

# Learning Binary Probabilities

**Bernoulli Distribution:** Single toss of a (possibly biased) coin

$$\mathrm{Ber}(x \mid \theta) = \theta^{\mathbb{I}(x=1)}(1-\theta)^{\mathbb{I}(x=0)} \qquad 0 \leq \theta \leq 1$$

- Suppose we observe N samples from a Bernoulli distribution with unknown mean:

$$X_i \sim \mathrm{Ber}(\theta), i = 1, \ldots, N$$

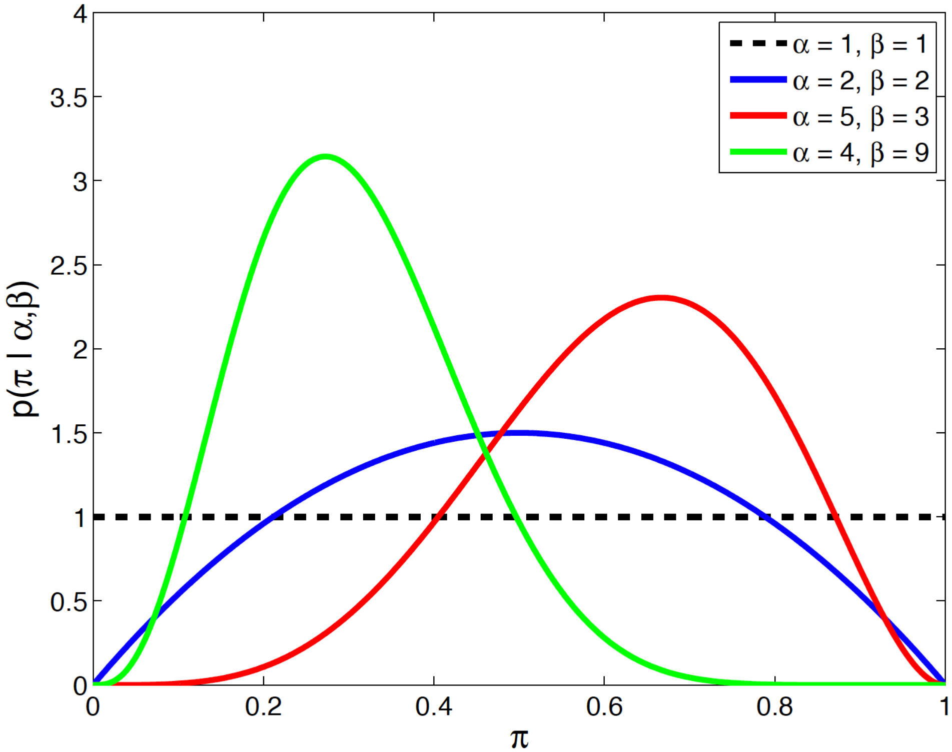
$$p(x_1, \ldots, x_N \mid \theta) = \theta^{N_1}(1-\theta)^{N_0}$$

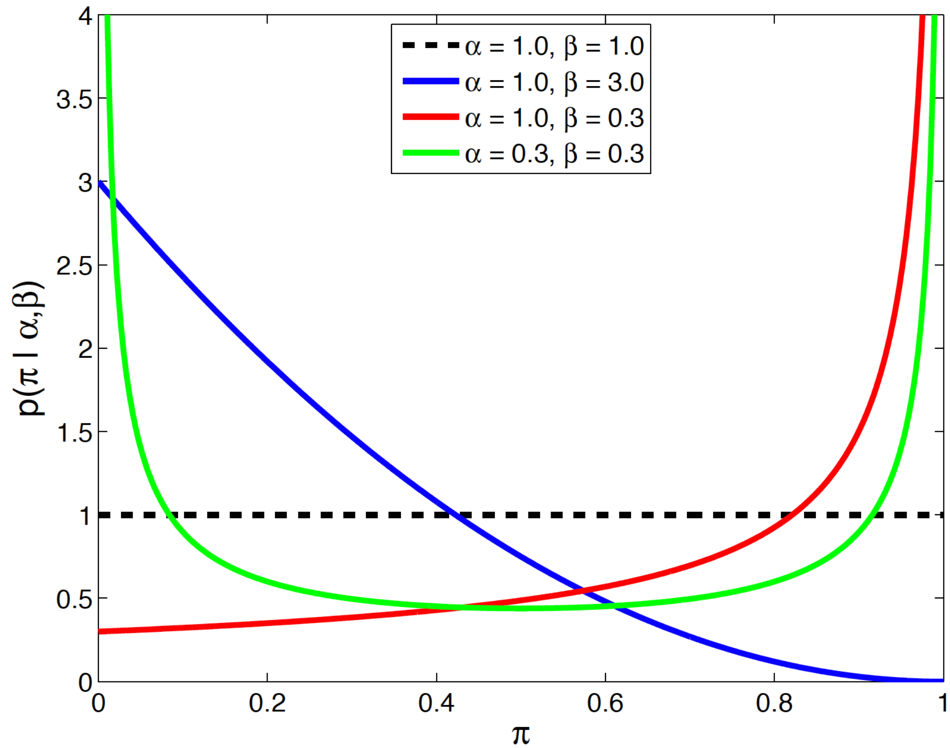
$$N_1 = \sum_{i=1}^{N} \mathbb{I}(x_i = 1) \qquad N_0 = \sum_{i=1}^{N} \mathbb{I}(x_i = 0)$$

- What is the *maximum likelihood* parameter estimate?

$$\hat{\theta} = \arg\max_{\theta} \log p(x \mid \theta) = \frac{N_1}{N}$$

# Beta Distributions

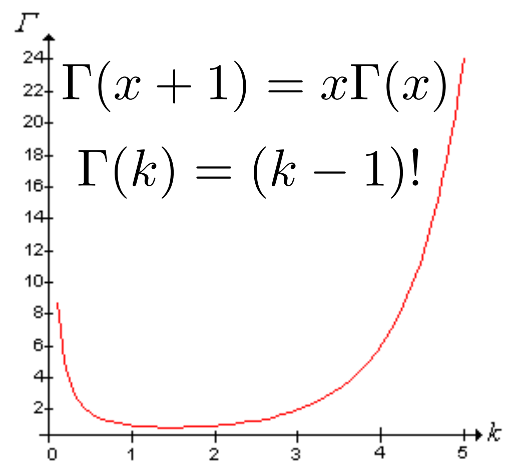



*Probability density function:* $x \in [0, 1]$

$$\mathrm{Beta}(x|a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

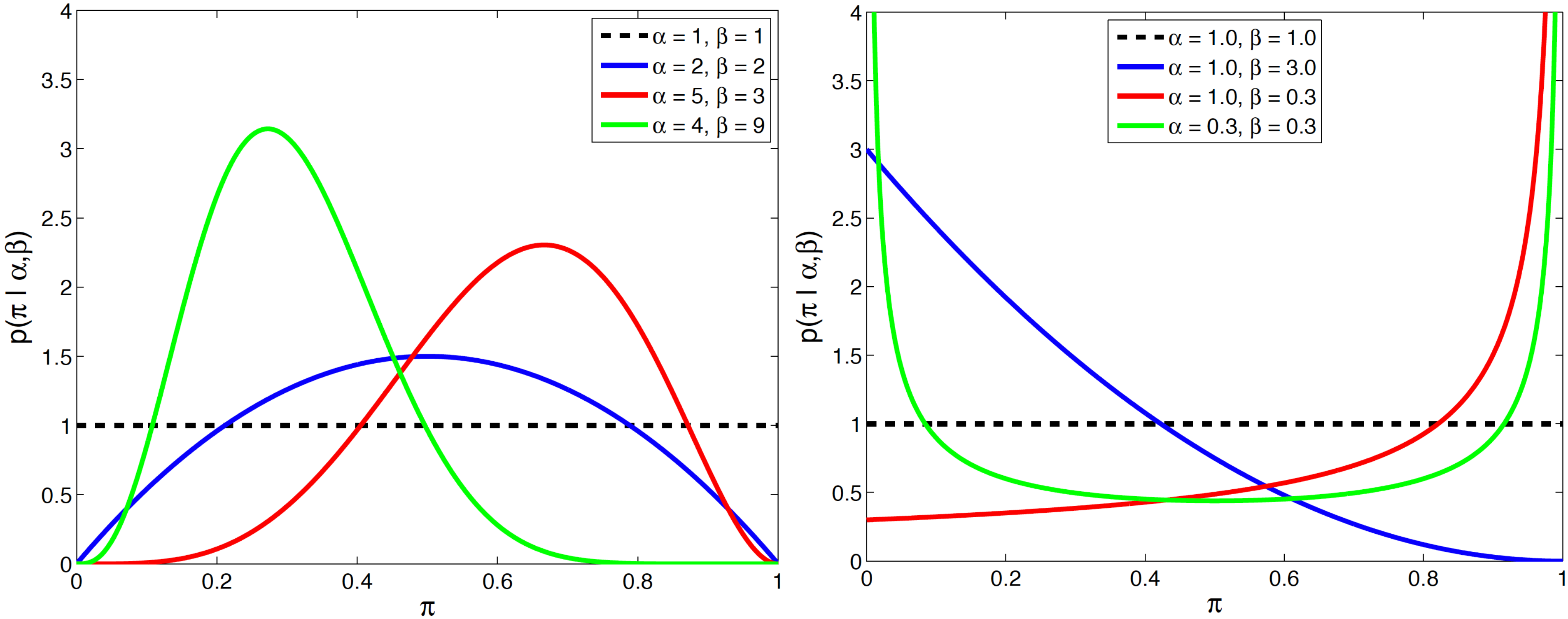
$$B(a,b) \triangleq \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \qquad a, b > 0$$




$$\Gamma(x+1) = x\Gamma(x)$$

$$\Gamma(k) = (k-1)!$$

# Beta Distributions




$$\mathbb{E}[x] = \frac{a}{a+b} \qquad \mathbb{V}[x] = \frac{ab}{(a+b)^2(a+b+1)}$$

$$\mathrm{Mode}[x] = \arg\max_{x\in[0,1]} \mathrm{Beta}(x \mid a, b) = \frac{a-1}{(a-1)+(b-1)}$$

# Bayesian Learning of Probabilities

**Bernoulli Likelihood:** Single toss of a (possibly biased) coin

$$\mathrm{Ber}(x \mid \theta) = \theta^{\mathbb{I}(x=1)}(1-\theta)^{\mathbb{I}(x=0)} \qquad 0 \leq \theta \leq 1$$

$$p(x_1, \ldots, x_N \mid \theta) = \theta^{N_1}(1-\theta)^{N_0}$$

**Beta Prior Distribution:**

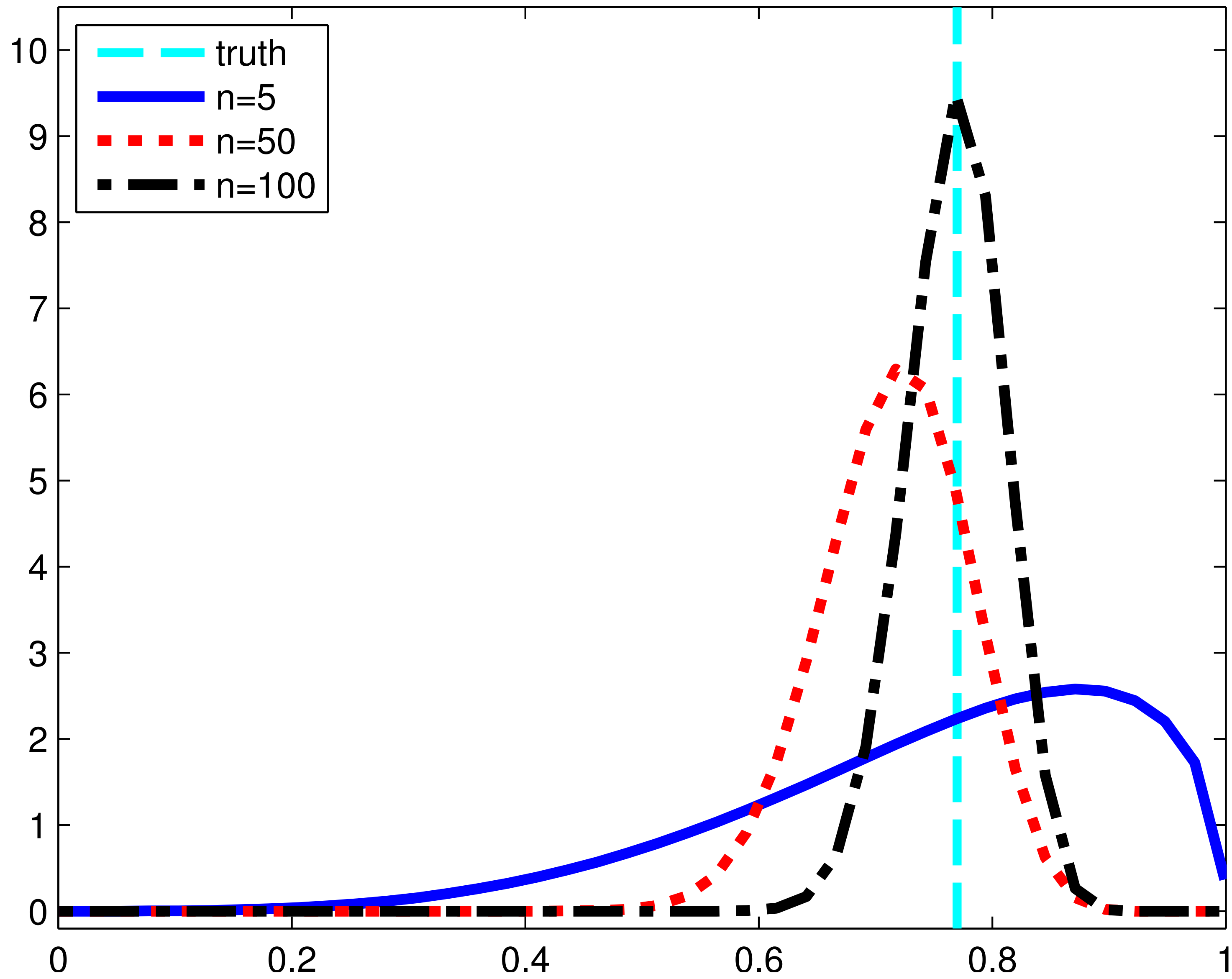
$$p(\theta) = \mathrm{Beta}(\theta \mid a, b) \propto \theta^{a-1}(1-\theta)^{b-1}$$

**Posterior Distribution:**

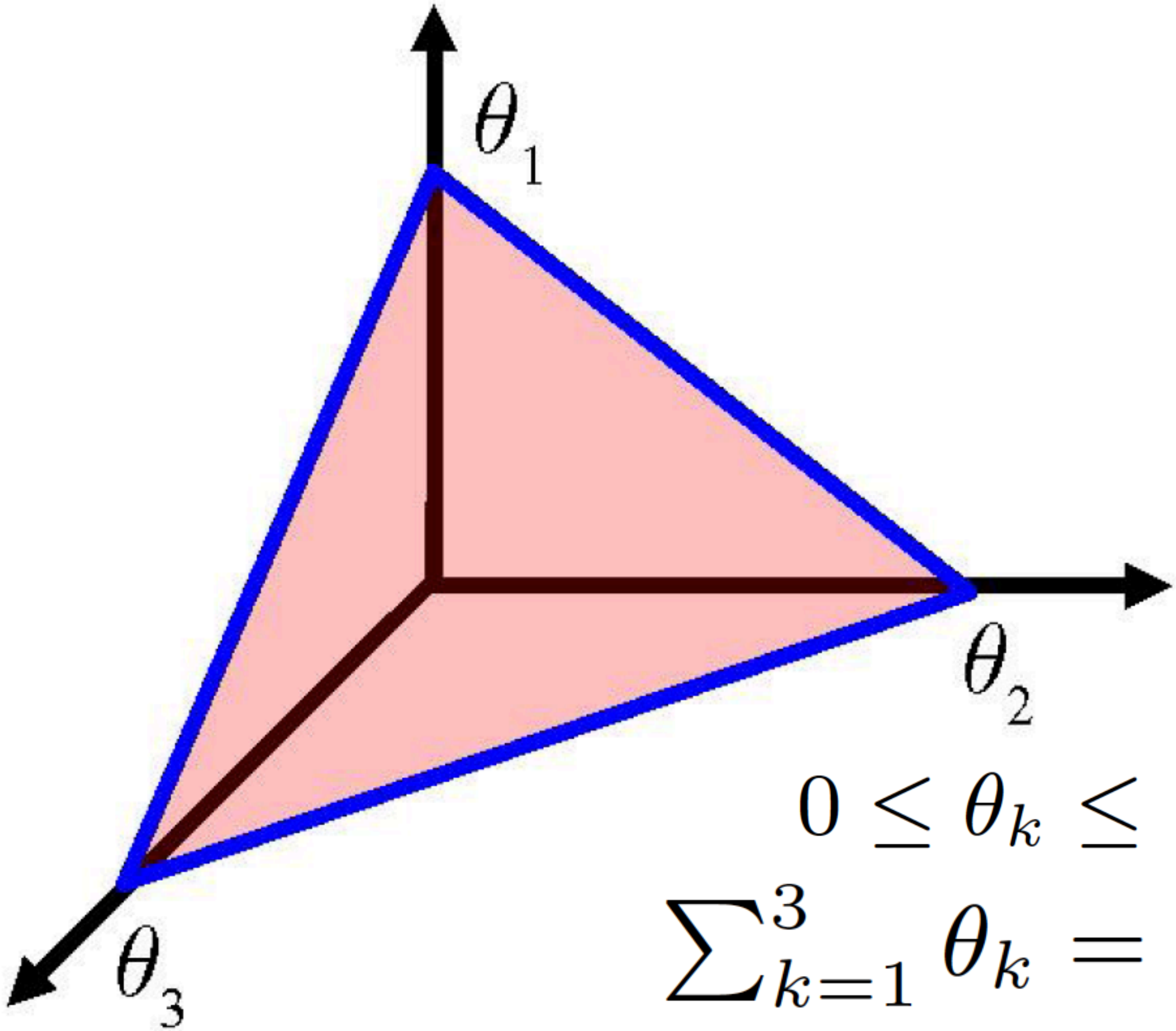
$$p(\theta \mid x) \propto \theta^{N_1+a-1}(1-\theta)^{N_0+b-1} \propto \mathrm{Beta}(\theta \mid N_1 + a, N_0 + b)$$

- *This is a conjugate prior, because posterior is in same family*
- *Estimate by posterior mode (MAP) or mean (preferred)*
- *Here, posterior predictive equivalent to mean estimate*

# Sequence of Beta Posteriors



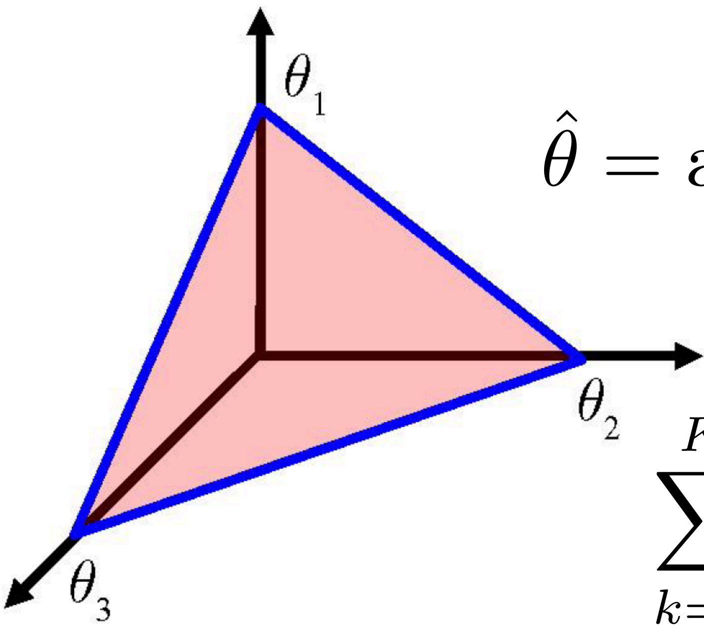
# Multinomial Simplex




$$0 \leq \theta_k \leq 1$$

$$\sum_{k=1}^{3} \theta_k = 1$$

# Constrained Optimization




$$\hat{\theta} = \arg\max_{\theta} \sum_{k=1}^{K} a_k \log \theta_k \qquad a_k \geq 0$$

*subject to*

$$\sum_{k=1}^{K} \theta_k = 1 \qquad \theta_k \geq 0$$

- Solution: $$\hat{\theta}_k = \frac{a_k}{a_0} \qquad a_0 = \sum_{k=1}^{K} a_k$$
- Proof for K=2: Change of variables to unconstrained problem
- Proof for general K: Lagrange multipliers (see textbook)

# Learning Categorical Probabilities

**Multinoulli Distribution:** Single roll of a (possibly biased) die

$$\mathrm{Cat}(x \mid \theta) = \prod_{k=1}^{K} \theta_k^{x_k} \qquad \mathcal{X} = \{0,1\}^K, \sum_{k=1}^{K} x_k = 1$$

- If we have $N_k$ observations of outcome $k$ in $N$ trials:

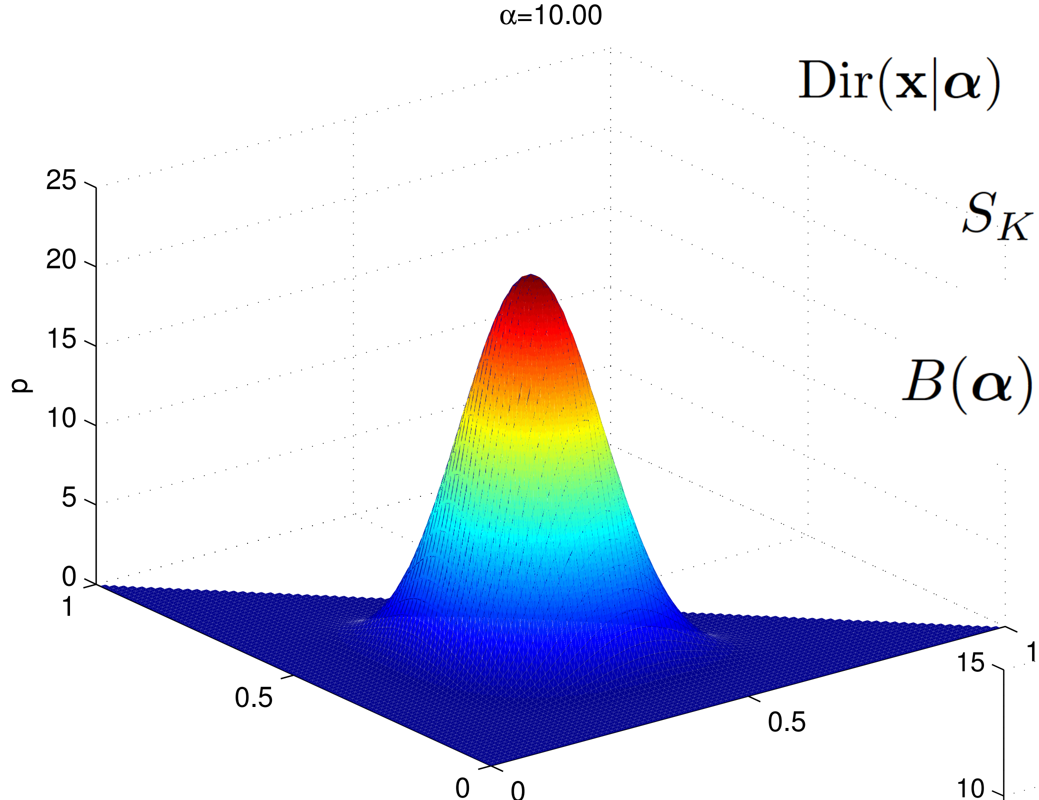
$$p(x_1, \ldots, x_N \mid \theta) = \prod_{k=1}^{K} \theta_k^{N_k}$$

- The *maximum likelihood* parameter estimates are then:

$$\hat{\theta} = \arg\max_{\theta} \log p(x \mid \theta) \qquad \hat{\theta}_k = \frac{N_k}{N}$$

- Will this produce sensible predictions when $K$ is large?

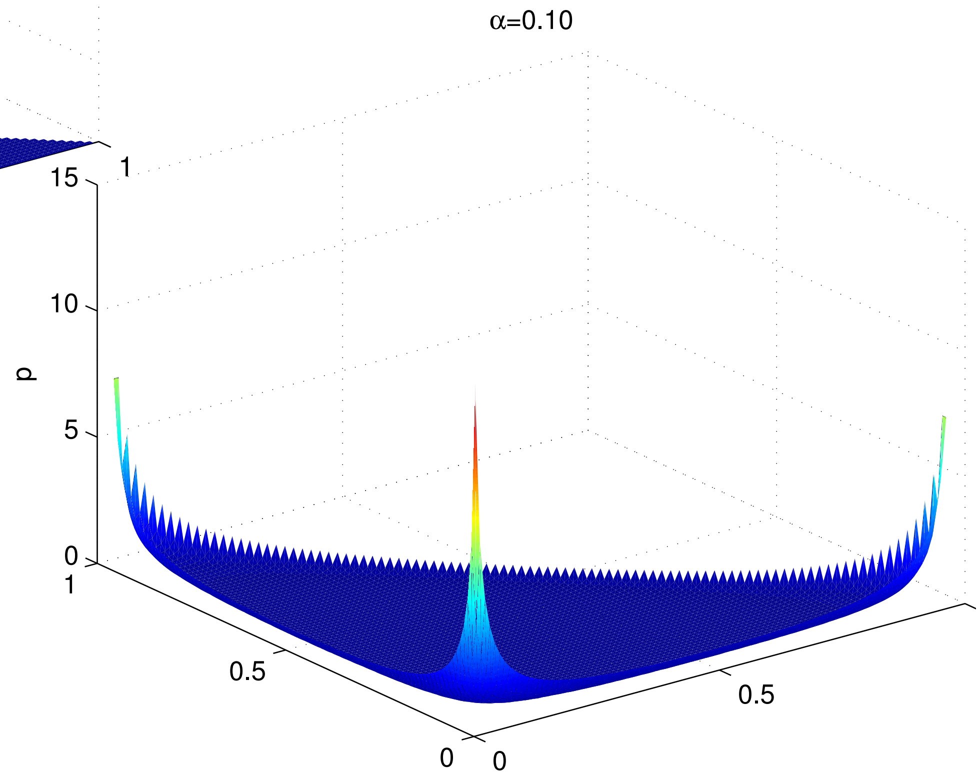
# Dirichlet Probability Densities




$$\mathrm{Dir}(\mathbf{x}|\boldsymbol{\alpha}) \triangleq \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} x_k^{\alpha_k - 1} \mathbb{I}(\mathbf{x} \in S_K)$$

$$S_K = \{\mathbf{x} : 0 \leq x_k \leq 1, \sum_{k=1}^{K} x_k = 1\}$$

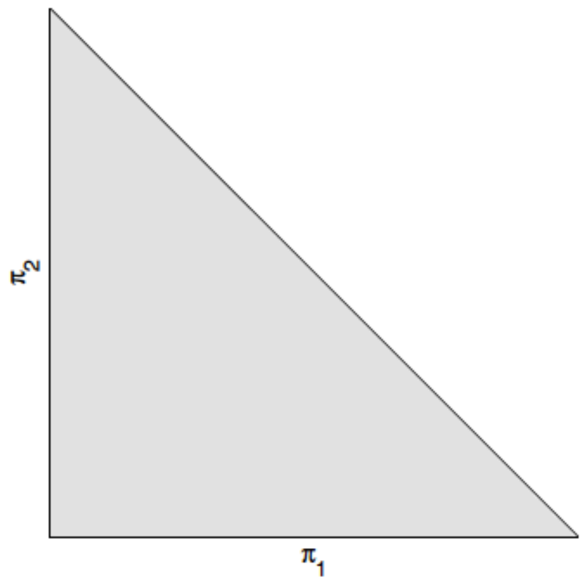
$$B(\boldsymbol{\alpha}) \triangleq \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\alpha_0)} \qquad \alpha_0 \triangleq \sum_{k=1}^{K} \alpha_k$$



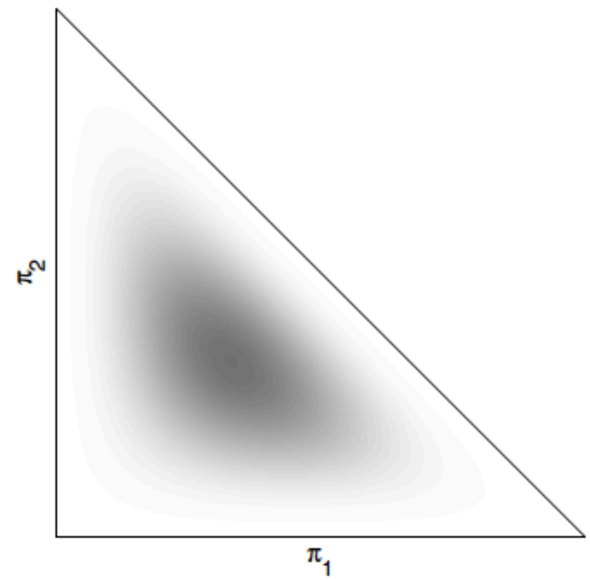

*Mean:* $$\mathbb{E}[x_k] = \frac{\alpha_k}{\alpha_0}$$

*Mode:* $$\hat{x}_k = \frac{\alpha_k - 1}{\alpha_0 - K}$$

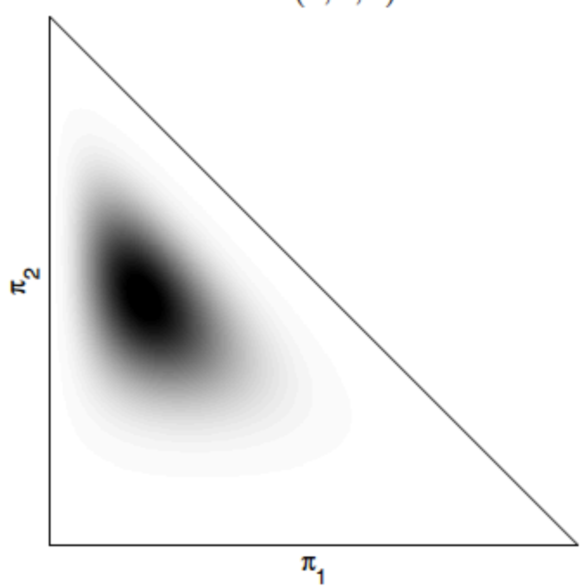
# Dirichlet Probability Densities



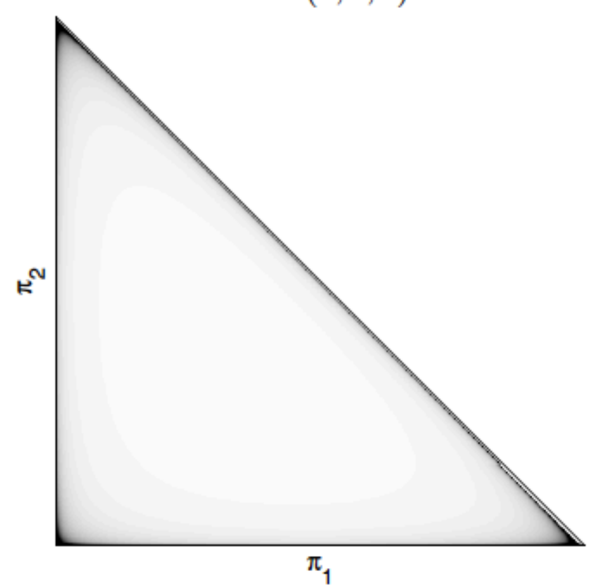

$\pi \sim \mathrm{Dir}(1, 1, 1)$




$\pi \sim \mathrm{Dir}(4, 4, 4)$

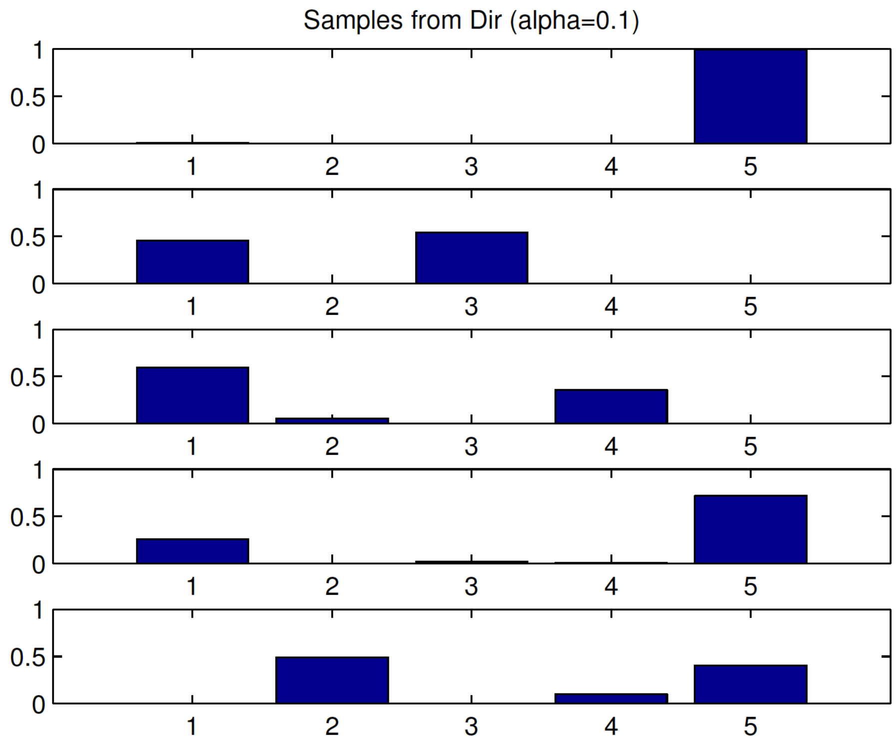



$\pi \sim \mathrm{Dir}(4, 9, 7)$

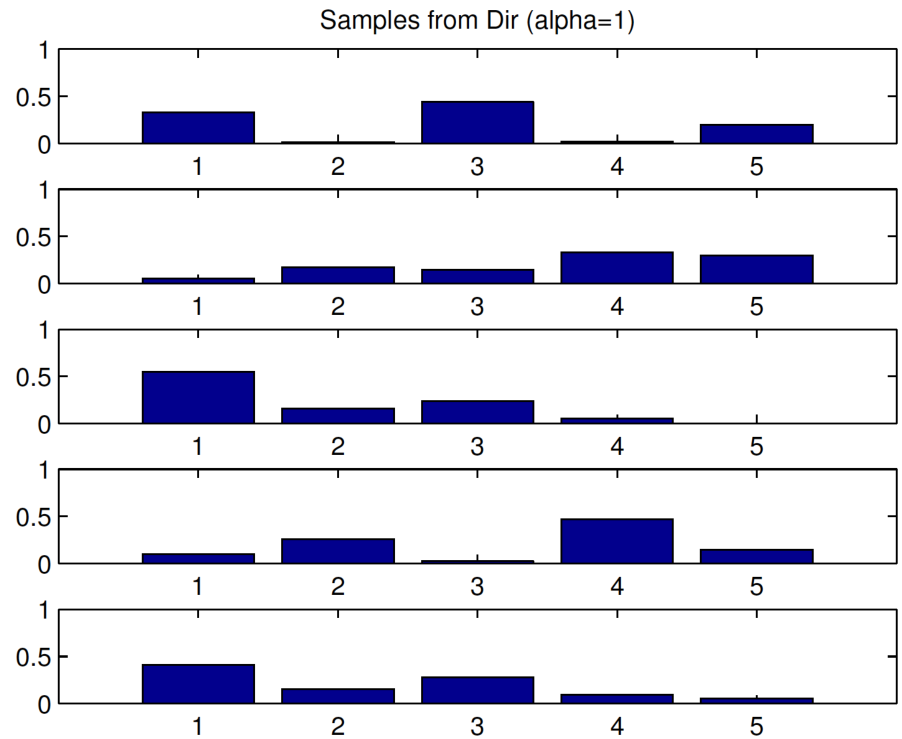



$\pi \sim \mathrm{Dir}(0.2, 0.2, 0.2)$

# Dirichlet Samples




$$\mathrm{Dir}(\theta \mid 0.1, 0.1, 0.1, 0.1, 0.1)$$




$$\mathrm{Dir}(\theta \mid 1.0, 1.0, 1.0, 1.0, 1.0)$$

# Bayesian Learning of Probabilities

**Multinoulli Distribution:** Single roll of a (possibly biased) die

$$\mathrm{Cat}(x \mid \theta) = \prod_{k=1}^{K} \theta_k^{x_k} \qquad \mathcal{X} = \{0,1\}^K, \sum_{k=1}^{K} x_k = 1$$

$$p(x_1, \ldots, x_N \mid \theta) = \prod_{k=1}^{K} \theta_k^{N_k}$$

**Dirichlet Prior Distribution:**

$$p(\theta) = \mathrm{Dir}(\theta \mid \alpha) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

**Posterior Distribution:**

$$p(\theta \mid x) \propto \prod_{k=1}^{K} \theta_k^{N_k + \alpha_k - 1} \propto \mathrm{Dir}(\theta \mid N_1 + \alpha_1, \ldots, N_K + \alpha_K)$$

- *This is a conjugate prior, because posterior is in same family*

# Learning Directed Graphical Models

$$\log p(\theta \mid \mathcal{D}) = C + \sum_{i \in \mathcal{V}} \left[ \log p(\theta_i) + \sum_{n=1}^{N} \log p(x_{i,n} \mid x_{\Gamma(i),n}, \theta_i) \right]$$

- For nodes with no parents, parameters define a single Bernoulli or categorical distribution
  - Bayesian or ML learning as in previous slides
- More generally, there are multiple categorical distributions per node, one for every *combination* of parent variables
  - Learning objective decomposes into multiple terms, one for subset of training data with each parent configuration
  - Apply independent Bayesian or ML learning to each
- Concerns for nodes with many parents:
  - *Computation:* Large number of parameters to estimate
  - *Sparsity:* May have little (or even no) data for some configurations of the parent variables
  - *Priors* can help, but may still be inadequate…

# Naïve Bayes: ML & Bayes

$$p(\mathbf{x}_i, y_i|\boldsymbol{\theta}) = p(y_i|\boldsymbol{\pi})\prod_j p(x_{ij}|\boldsymbol{\theta}_j) = \prod_c \pi_c^{\mathbb{I}(y_i=c)} \prod_j \prod_c p(x_{ij}|\boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i=c)}$$

$$\log p(\mathcal{D}|\boldsymbol{\theta}) \quad = \quad \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D}\sum_{c=1}^{C}\sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\theta}_{jc})$$

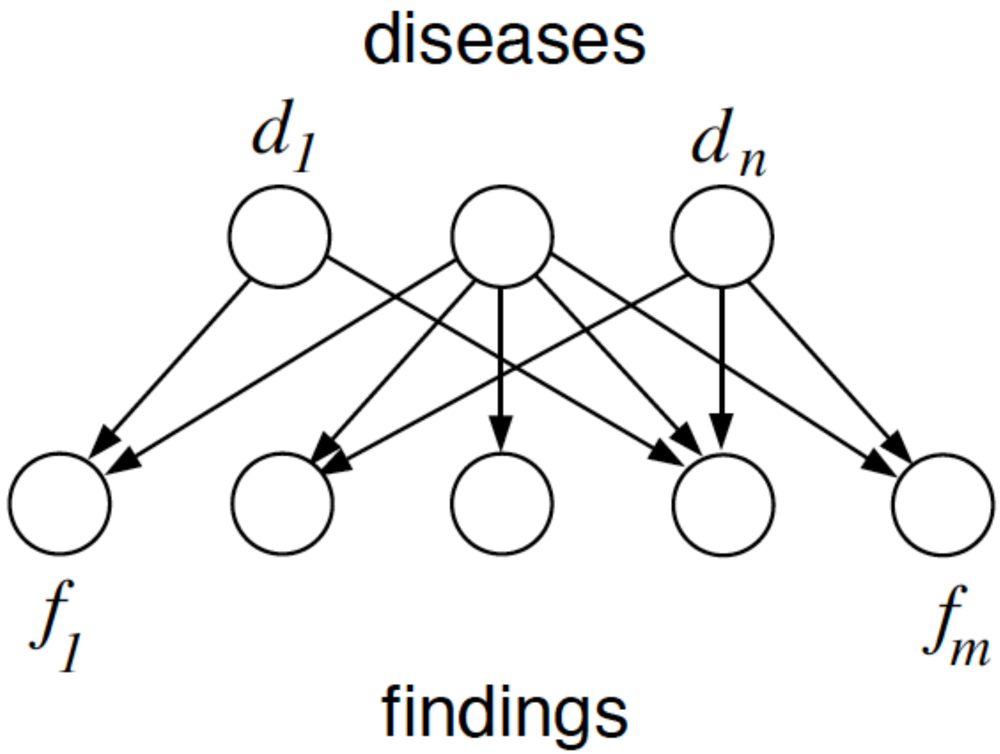
$N_c \longrightarrow$ number of examples of training class *c*

- Maximizing the sum of functions of independent parameters can be done by maximizing them independently:

*Maximum Likelihood* $\hat{\pi}_c \quad = \quad \frac{N_c}{N}$ $\qquad \hat{\theta}_{jc} = \frac{N_{jc}}{N_c}$ *if* $x_j|y = c \sim \text{Ber}(\theta_{jc})$

- Similarly, if the parameters for different features are independent under the prior, they remain independent under the posterior, and Bayesian analysis decomposes

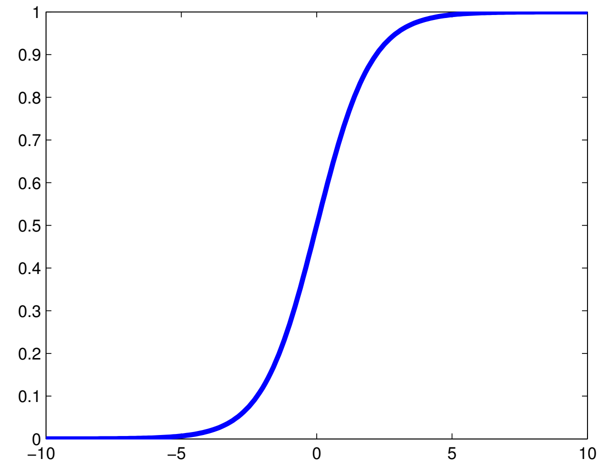
# Example: Medical Diagnosis




- Learning independent finding distribution for every combination of diseases may be computationally intractable and lead to poor statistical generalization
- Instead assume restricted parameterizations, in which child distributions depend on some features of parents. Example:

# Logistic Regression

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




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

- Linear discriminant analysis:

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

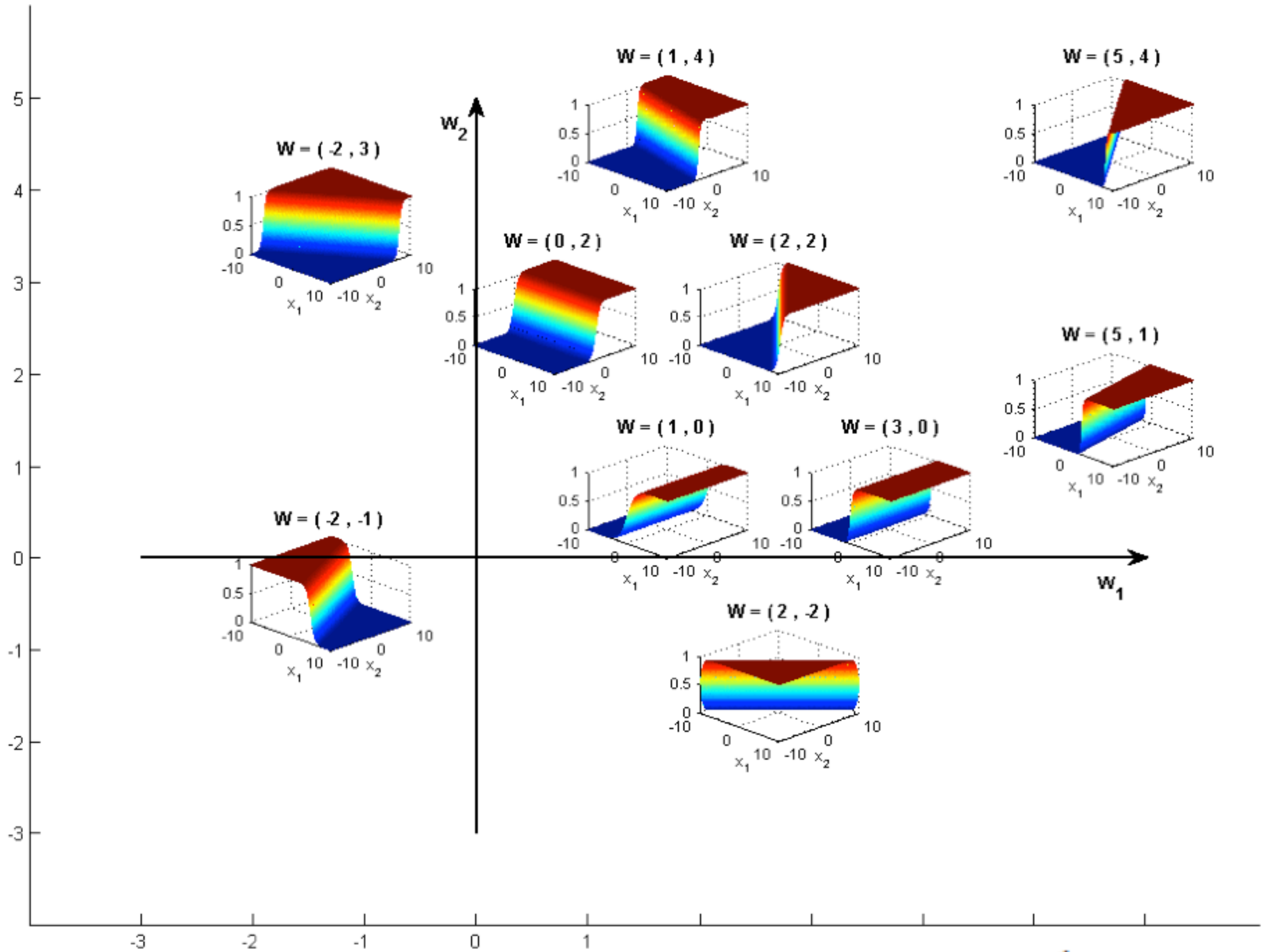
- Quadratic discriminant analysis:

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

- 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:

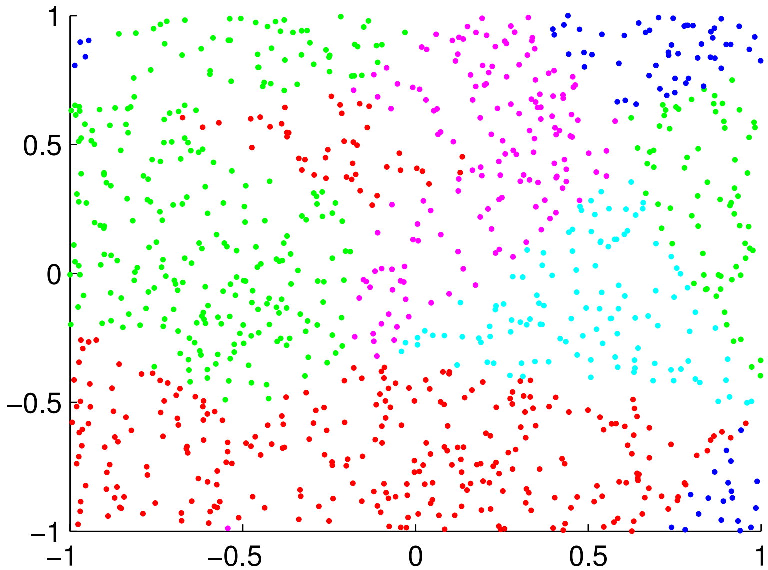
$$p(w \mid x, y) \propto p(w) \prod_{i=1}^{N} \mathrm{Ber}(y_i \mid \mathrm{sigm}(w^T \phi(x_i)))$$

# Logistic Regression




$$p(y|\mathbf{x},\mathbf{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^T\mathbf{x})) \qquad \mathrm{sigm}(\eta) := \frac{1}{1+\exp(-\eta)} = \frac{e^\eta}{e^\eta+1}$$

# Multinomial Logistic Regression




$$p(y|\mathbf{x}, \mathbf{W}) = \mathrm{Cat}(y|\mathcal{S}(\mathbf{W}^T\mathbf{x}))$$

$$\mathcal{S}(\eta)_c = \frac{e^{\eta_c}}{\sum_{c'=1}^{C} e^{\eta_{c'}}}$$

as $T \rightarrow 0$

$$\mathcal{S}(\eta/T)_c = \begin{cases} 1.0 & \text{if } c = \arg\max_{c'} \eta_{c'} \\ 0.0 & \text{otherwise} \end{cases}$$