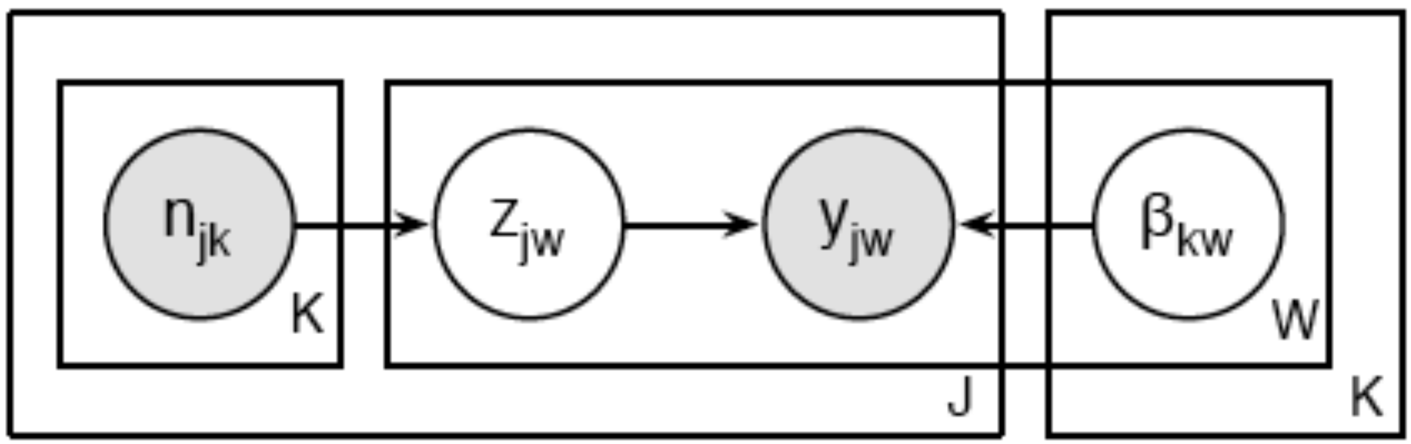
A Hopefully-Simple Introduction to Probabilistic Graphical Models



Outline

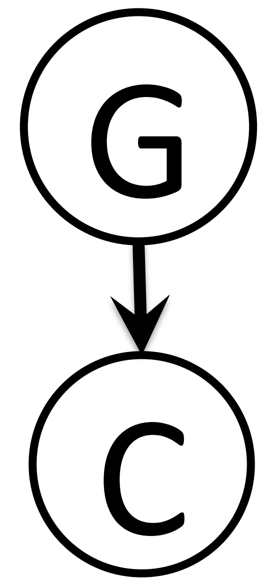
- What are they?
- Intro to necessary probability theory & corresponding notation
- A few simple probabilistic models
  - Naïve Bayes, latent variable models and GMMs, HMMs
- Inference overview
- Example MIR paper
- State of the art in MIR

What are probabilistic graphical models?

- Compact representation of joint probability distributions – factored joint distributions
- Tools for reasoning about conditional independence and dependence
- "a marriage between probability theory and graph theory" (Jordan 1998)
  - Graph theory used to design efficient algorithms to work with models
- Sub-types:
  - Markov Random Fields (undirected), **Bayesian Belief Networks** (directed)

How can graphical models be used?

- Propose a model that could explain a real-world phenomenon (build a model)
- *Infer parameters* of the proposed model from available data
- Given parameters and structure, *make predictions* for new data
  - Classification, regression
  - Causal explanations, diagnosis
  - Temporal predictions or smoothing
- Given several candidate models, pick the "best"



$C_i \sim \mathcal{N}(\mu_{gi}, \sigma^2)$

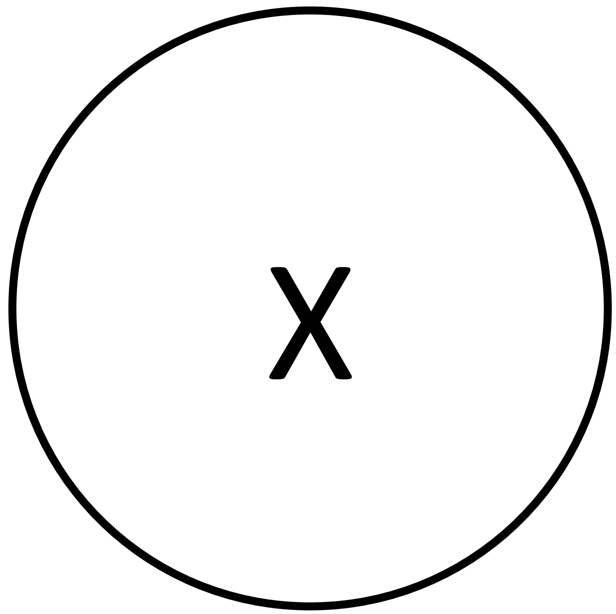
$\mu_{pop} = .23$

Necessary probability and corresponding notation

Random variables

- A random variable is a *function* that maps events to numbers
  - e.g., event = drawing a card from a standard deck
    - r.v. X = 1 iff card is 2 of spades, 0 otherwise
    - r.v. Y = 1 iff card is a 2 of any suit, 0 otherwise
  - Or, event = measuring height of an 18-24 year old female in the USA
    - r.v. Z = height measured in inches

Notation: A random variable



Distributions, parameters, and priors

A **distribution** assigns probability to regions of sample space

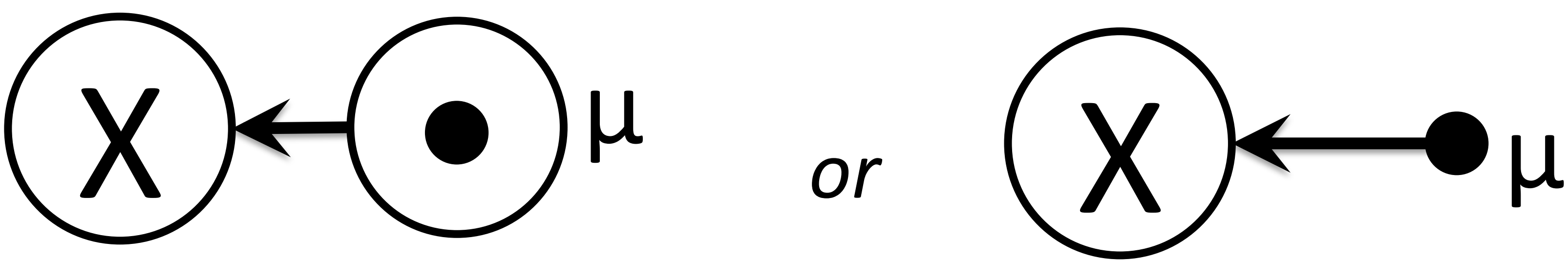
- Space for X: p(1) = 1/52, p(0) = 51/52
- Space for Y: p(1) = 4/52 = 1/13; p(0) = 12/13
- Space for Z: Z ~ $\mathcal{N}(65.5, 6.25)$
  - e.g., p(Z < 68) = 0.841

- Common distributions in the literature: Gaussian/Normal, binomial, beta, gamma, multinomial, Bernoulli, Dirichlet, uniform
- For a particular problem, the distribution is specified by the distribution type and its **parameters**
  - e.g., mean and variance for Gaussian
  - In Bayesian framework, there may be a **prior** distribution over these parameters (e.g., prior on mean is a uniform distribution over [0, 22.5])

Notation: the distribution and parameters

- The distribution is normally not explicitly represented in the graphical notation, but it will be described in the text.
- Occasionally, you will see parameters of the distribution represented in the notation
- θ or β commonly used for *unknown* parameter values; π for prior distributions
- hat commonly used to represent *estimate* of a parameter: $\hat{\theta}$

$X \sim \mathcal{N}(\mu, 1.0)$



Outcomes / samples

- An **sample** is an outcome or observation from a probability distribution (typically a number or vector of numbers)
- E.g., $x_i \sim \mathcal{N}(0, 50)$:
  - Each x_i is an **outcome** of sampling from the distribution, typically assumed i.i.d.
  - x_i, i = 1 to 5 might look like:
    x_1=-17.5, x_2=80.31, x_3=38.49, x_4=-30.49, x_5=55.90

Notation: sample

- Specific outcomes **not** represented in model graph
- Outcomes are lower case (X vs. x)
- Be careful: Random variable may take value of a vector
  - is x_i the i^{th} outcome or the i^{th} element?

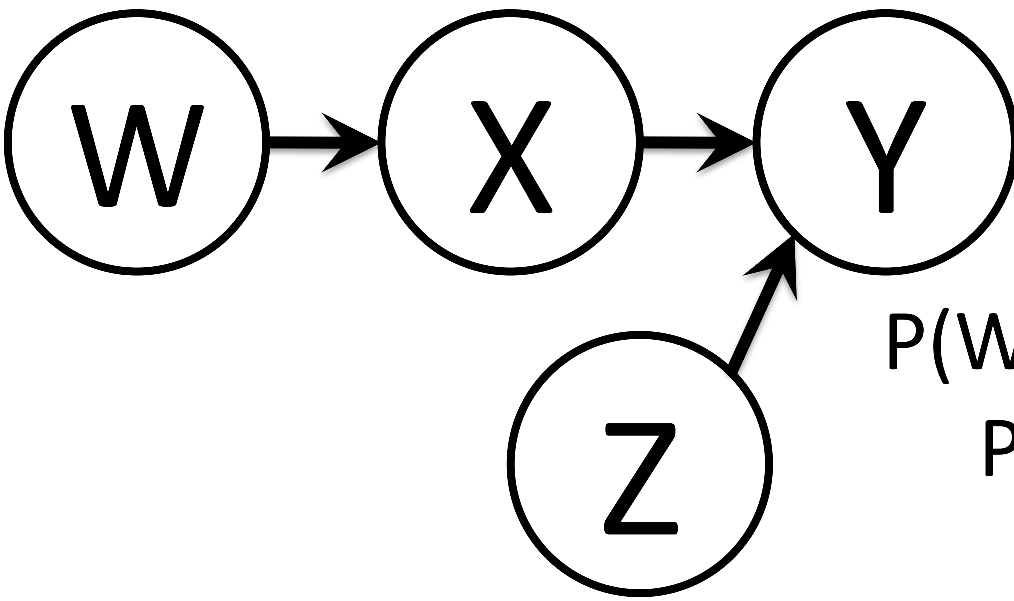
Conditional distribution

- The conditional distribution P(X | Y = y) is the probability distribution of X when the value of r.v. Y is known to be a particular value, y.
  - e.g., P(X|Y) can be specified as:

    $P(X|Y=.3) = \mathcal{N}(.2, 1.0)$ and $P(X|Y \mathrel{!=} .3) = \mathcal{N}(.5, 1.0)$

Notation: Conditional relationships

- Distribution of r.v. X is specified conditioned on its parents in the graph
  - Parents of X defined as all nodes P s.t. exists a directed **arc** from P to X



$$P(W,X,Y,Z) = P(W)P(Z)P(X|W)P(Y|X,Z)$$

Independence and dependence

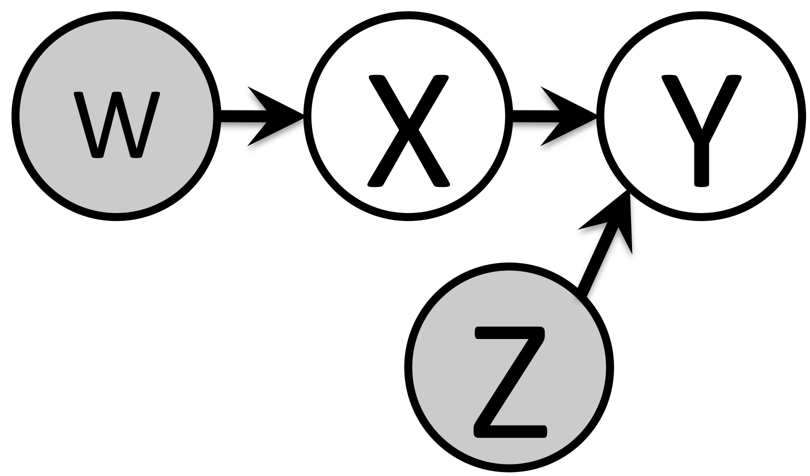
- Conditionally independent events:
  - A = "My favorite color is green"
  - B = "it's going to rain in Bali tomorrow"
  - Knowledge of one != knowledge of probability of the other
    - P(A | B) = P(A), P (B| A) = P(B)
    - P(A and B) = P(A)P(B)
- Conditionally dependent events:
  - A = "It's raining in San Francisco today"
  - B = "Jay's lawn is wet"
  - Knowing A changes our knowledge of likelihood of B, and vice versa
  - P(A | B) != P(A)

Independence & observation

- Independence may change when events are observed (known & fixed)
- Add a 3rd event:
  - A = "It's raining in San Francisco today"
  - B = "Jay's lawn is wet"
  - **C = "Jay turned on his sprinkler before leaving this morning."**
  - A and B are **independent** if we observe C to be true.

Notation: Observation

- Observed variables (whose values are known) are shaded
- **NOT** restricted to particular locations in graph
- Graphical models provide a way to reason about which variables marginally independent given the observed data
  – e.g., "Bayes Ball" algorithm

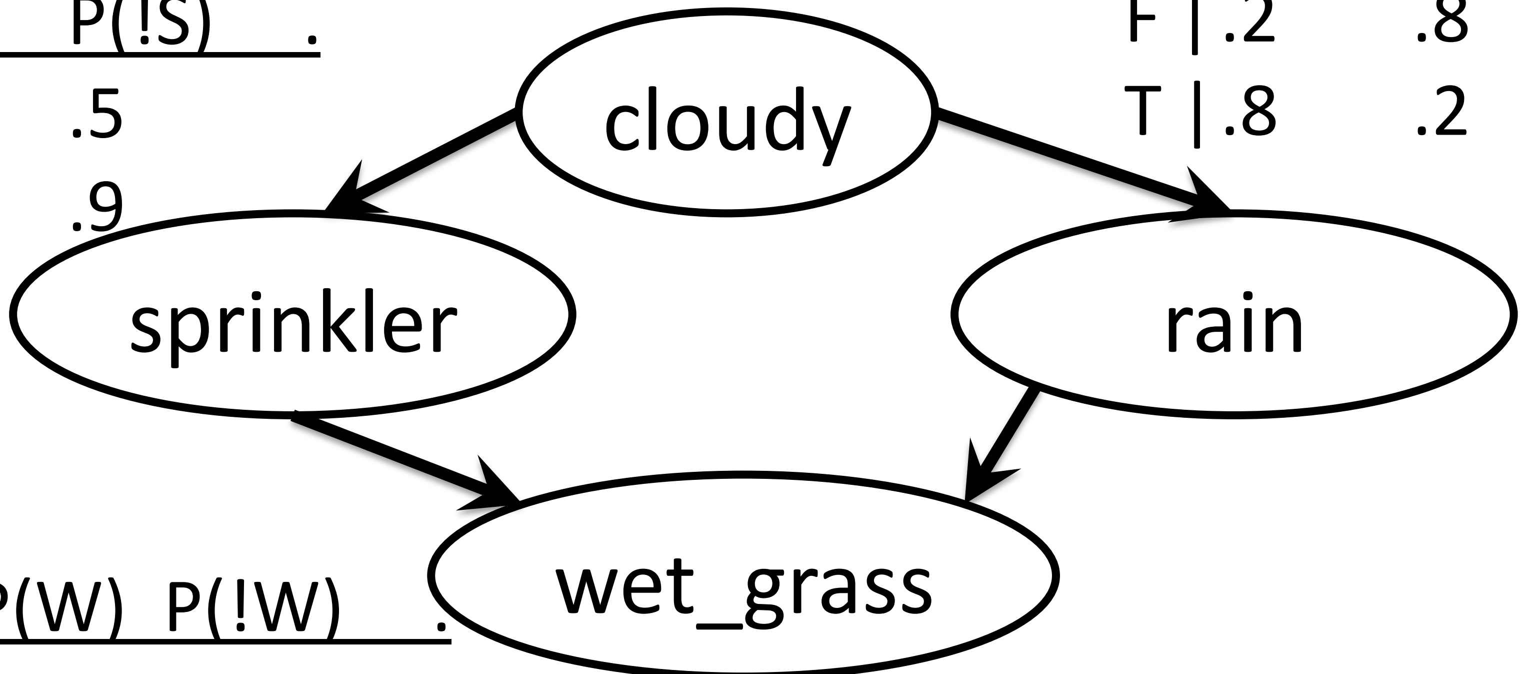


Classic example

P(C)	P(!C)
.5	.5

C	P(S)	P(!S)
F	.5	.5
T	.1	.9

C	P(R)	P(!R)
F	.2	.8
T	.8	.2



S	R	P(W)	P(!W)
F	F	0.0	1.0
F	T	.9	.1
T	F	.9	.1
T	T	.01	.99

What do we get from graphical representation?

- Local structure = factorization of full joint distribution = more efficient *representation* of full joint probability distribution
  - Size of joint is O(2^n) for *n* binary variables; Here only O(n * 2^k) for k max fan-in.
- Ability to *simulate* drawing from joint distribution (*generation*)
- A basis for many exact and approximate *inference* algorithms, using graph theory
  - e.g., infer values of unobserved nodes from observed nodes
  - or infer parameters of the model
- A set of [visually distinguishable] "design patterns" for reasoning about problem structures

A	B	C	D	P(A,B,C,D)
F	F	F	F	0.17
F	F	F	T	0.02
F	F	T	F	0.11
F	F	T	T	0.09
F	T	F	F	0.01
F	T	F	T	0.04
F	T	T	F	0.01
F	T	T	T	0.08
T	F	F	F	0.11
T	F	F	T	0.00
T	F	T	F	0.15
T	F	T	T	0.02
T	T	F	F	0.08
T	T	F	T	0.01
T	T	T	F	0.06
T	T	T	T	0.04

Some common probabilistic models

Naïve Bayes

The Naïve Bayes assumption

- Generative model:
  - Class C drawn from a prior distribution (e.g., Bernoulli for binary classification)
  - Each feature F_i drawn from distribution conditional on C (e.g., a Normal distribution whose parameters depend on c)
- $P(F_1, F_2, \ldots F_N \mid C) = P(F_1|C)P(F_2|C)\ldots P(F_N|C)$
  - Value of each feature independent of other features, given the class.

Naïve Bayes

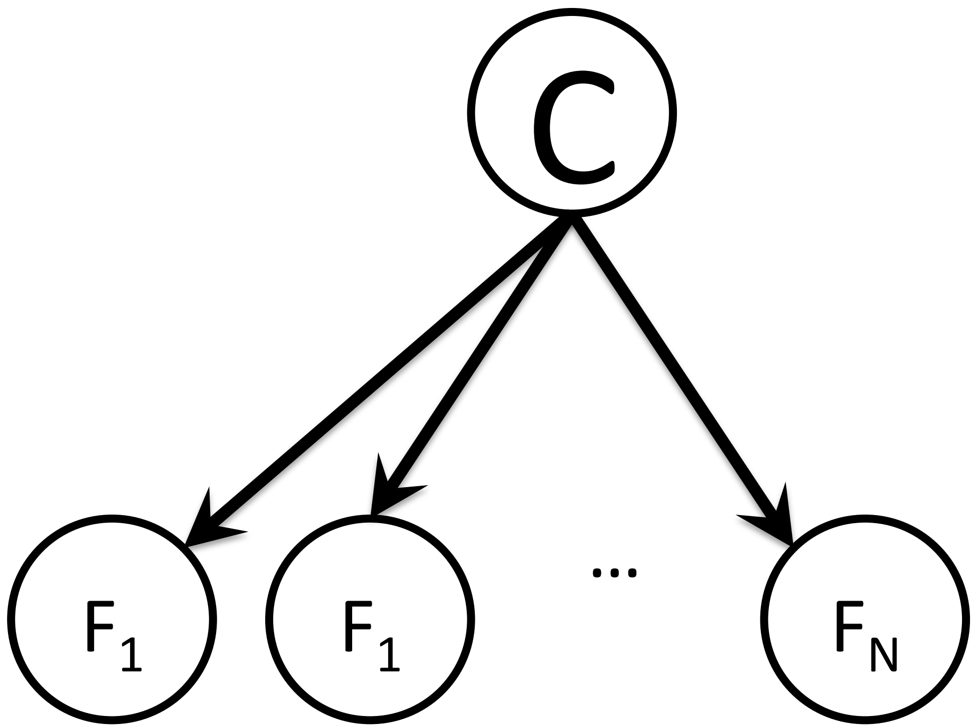
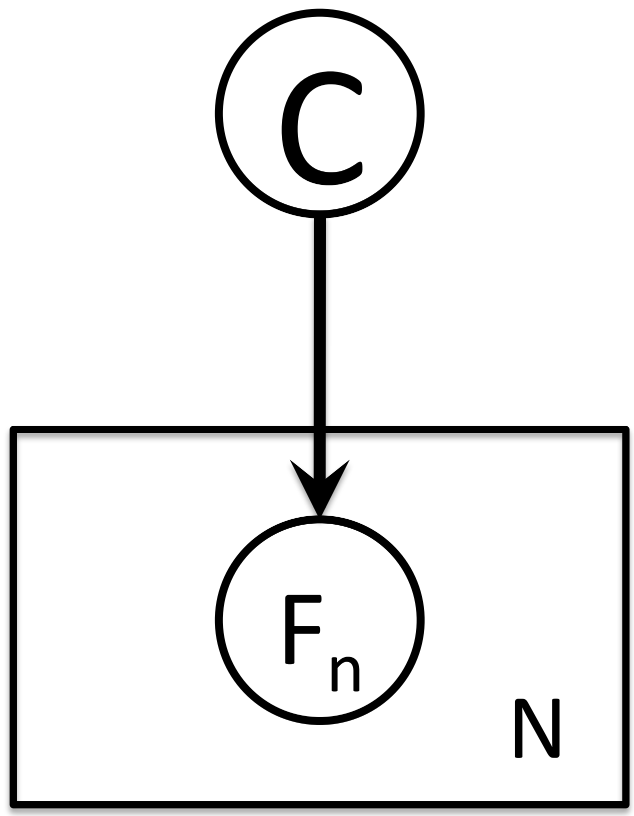


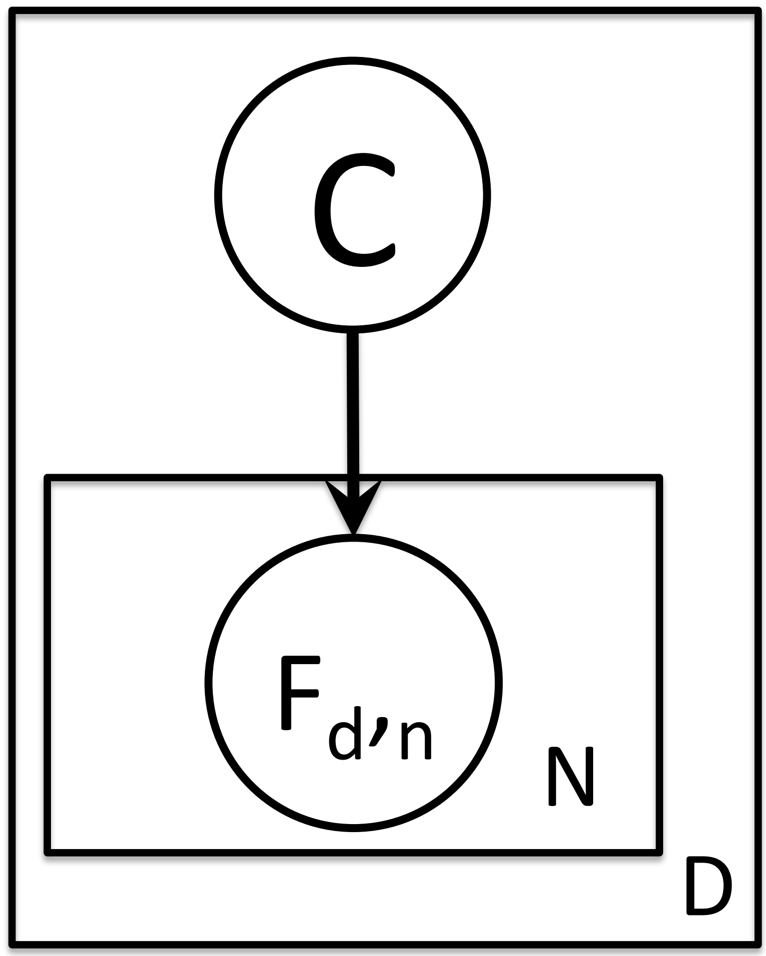
Plate notation

- Plate denotes structural repetitions, e.g. N features



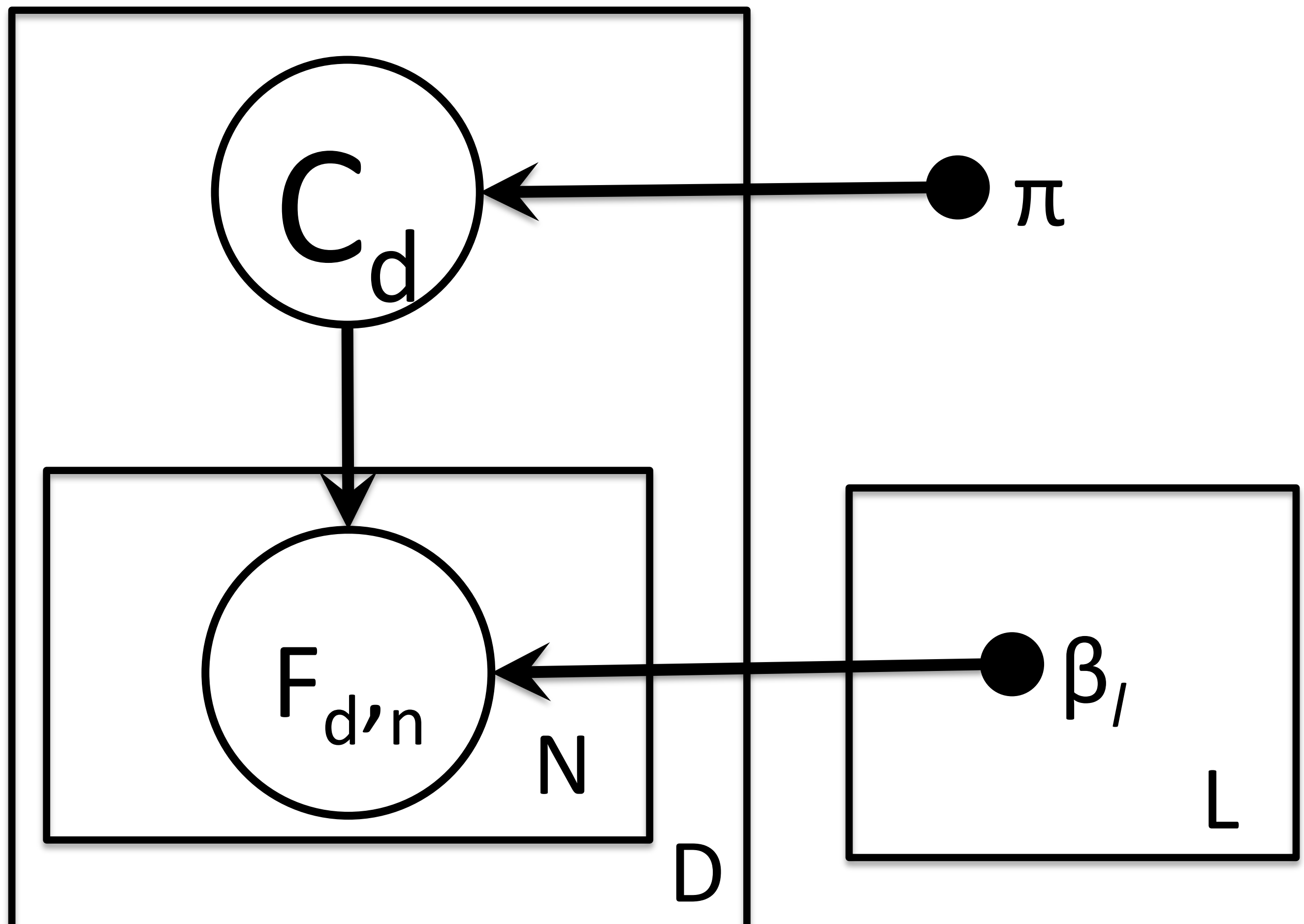
More plates

- Can represent D datapoints



Making *parameters* explicit

(β_l is chosen according to document class, $c_{d)}$

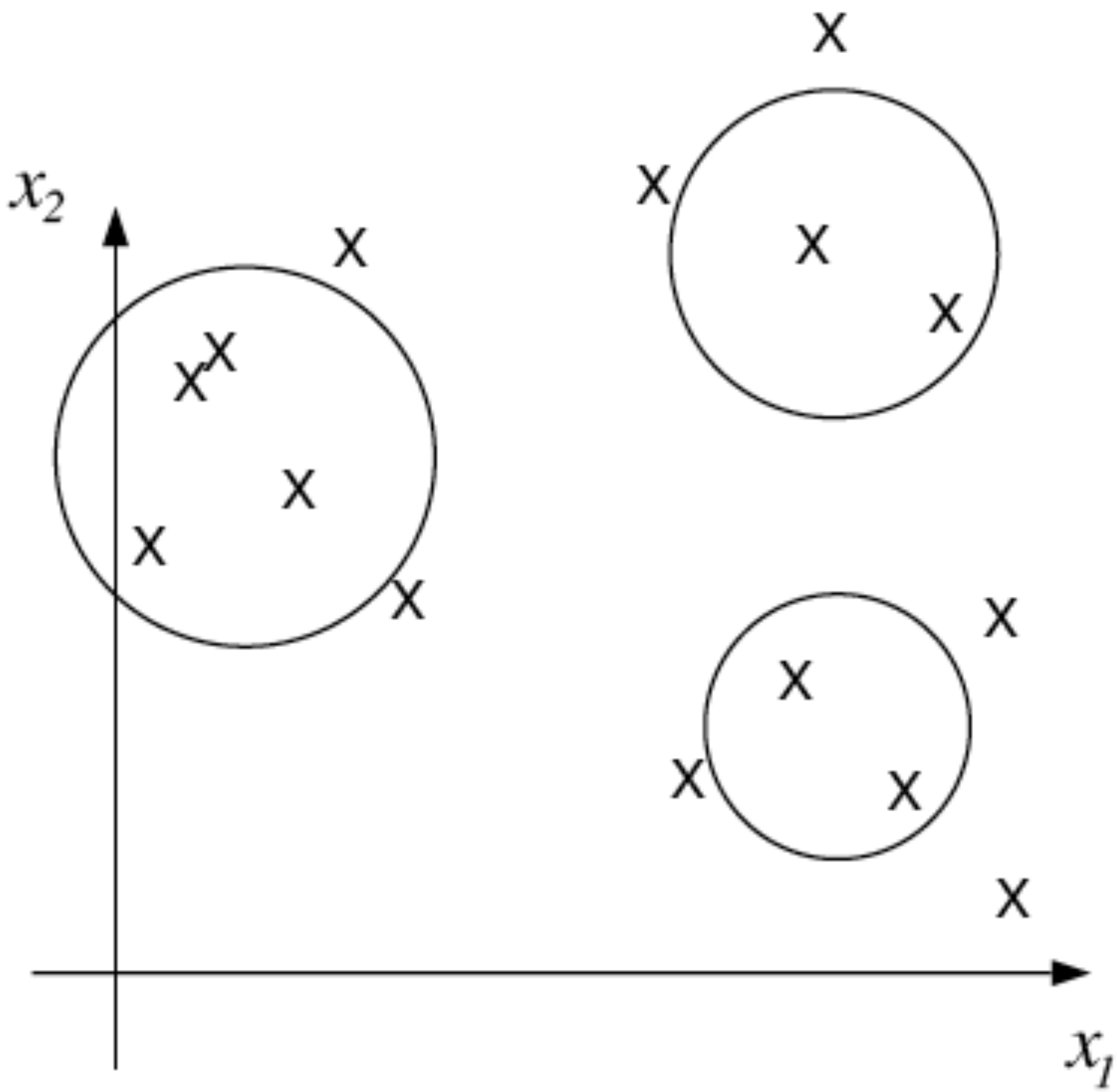


Using Naïve Bayes

- Classification: which class is most likely, given the observed features and model parameters?
  - Maximize $p(C \mid F_d, \pi, \beta)$
- Requires first knowing π, β
  - Training/inference: find the values of π, β that *maximize the likelihood* of the training data

Gaussian Mixture Models

A mixture of Gaussians

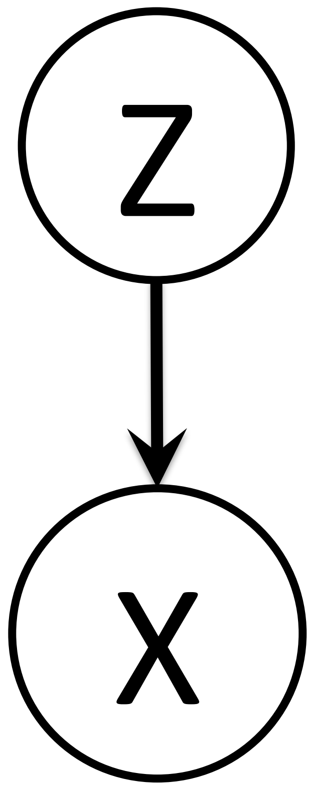


The GMM generative model

- There exist some K "clusters" or "hidden categories"; a prior π assigns probabilities to choosing cluster k
  - $z_i \sim \pi$
- Once the cluster identity k_i has been chosen, the observed vector $\mathbf{x_i}$ is chosen by sampling from Gaussian k_i
  - $\mathbf{x_i} \sim \mathcal{N}(\mu_{ki},\Sigma_{ki})$

Graphical model notation for GMM

Z is a **latent variable:** useful in formulating model but not observed.



Using GMMs

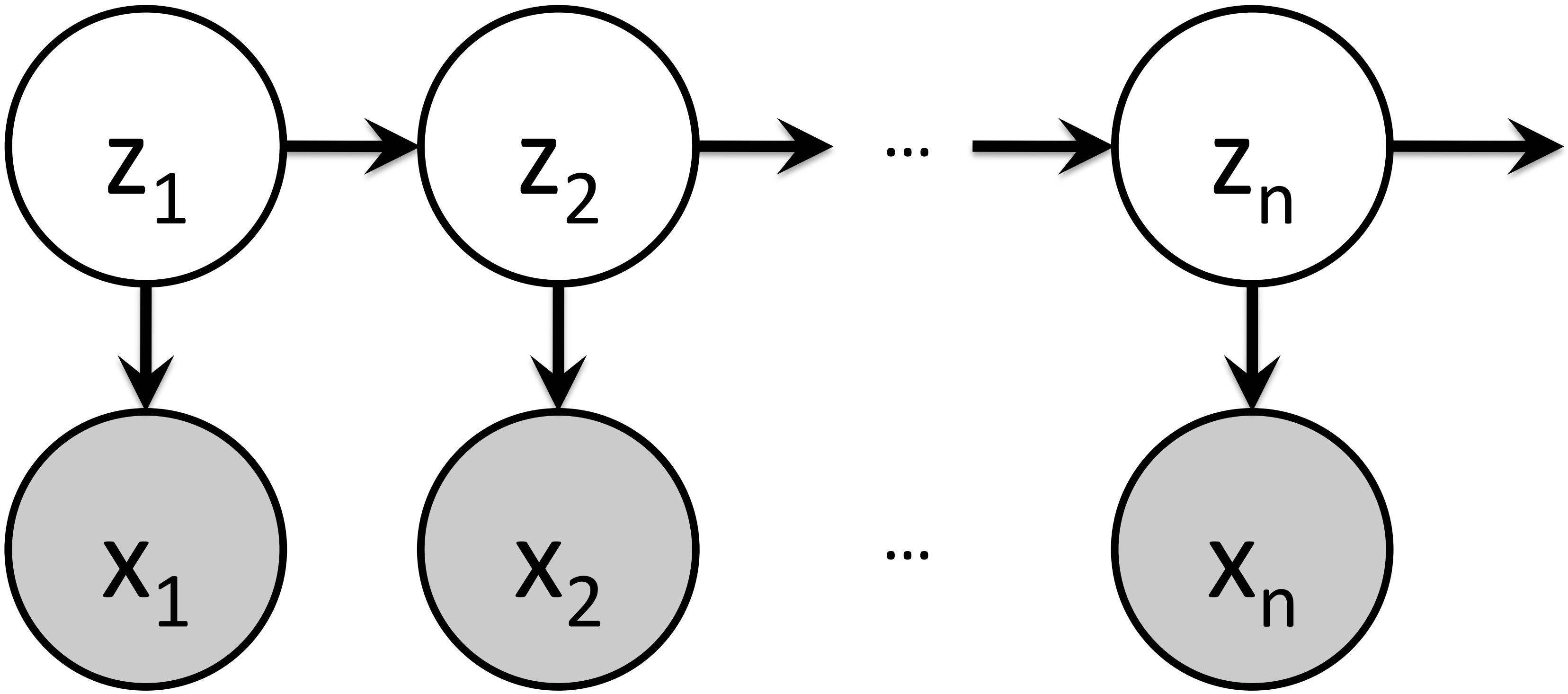
- Could compute "best" cluster ID (Z) for some data
- Could compute likelihood of some data under a GMM
- Classification: Train GMM for each class, then choose GMM that maximizes the likelihood of the observed data
- Vector Quantization (VQ): Represent features with class ID only
- Possible problem: must know or assume an appropriate # of mixture components
  - Non-parametric methods, e.g. latent Dirichlet allocation, allow to infer # of hidden categories

Hidden Markov Models

Hidden Markov Models

- Models a sequence of observations in time
- Assumes an underlying time sequence of hidden states

An HMM



HMM applications

- Infer the most likely hidden sequence
- Predict most likely next observation(s)
- Infer single most likely hidden state at time t
- Generate likely sequences (e.g. Mark V Shaney)
- Choose most likely model for an observed sequence (e.g., word spoken, pitches played)

Inference

Inference: How to estimate model parameters from the data

- Find parameters that maximize the likelihood of the data
  - i.e., find $\boldsymbol{\theta}$ to maximize p(D | $\boldsymbol{\theta}$)
  - Often maximize **log** likelihood (multiplication of terms -> summation of terms)
- Sometimes can compute directly: e.g., Naïve Bayes, HMM
- Sometimes must approximate: e.g., GMMs

Inference algorithms

- Exact and approximate
  - Simple MLE computation for Naïve Bayes
  - Message passing / dynamic programming methods (e.g., forward/backward for HMMs)
  - Expectation-Maximization (EM) for GMMs
  - Variational inference, Gibbs sampling, Markov chain Monte Carlo (MCMC) for other model types
- Challenges
  - If you design a new model architecture, you have to come up with an inference method
  - Certain algorithms can get stuck in local maxima
  - Inference can be computationally intensive

Reading an MIR paper

M. Hoffman, D. Blei, P. Cook, "Easy as CBA: A Simple Probabilistic Model for Tagging Music," in Proceedings of the 10th International Conference on Music Information Retrieval, Kobe, 2009. pdf

Goals

- How to assign semantic tags to audio?
  - Build a binary classifier per tag?
  - Build a GMM for each tag?
    - What does it mean to have multiple tags?
- Hoffman, Blei & Cook: Build a model for joint distribution of tags and audio features
- Use model to compute probability that a tag applies to a song
  - Annotation
  - Retrieval

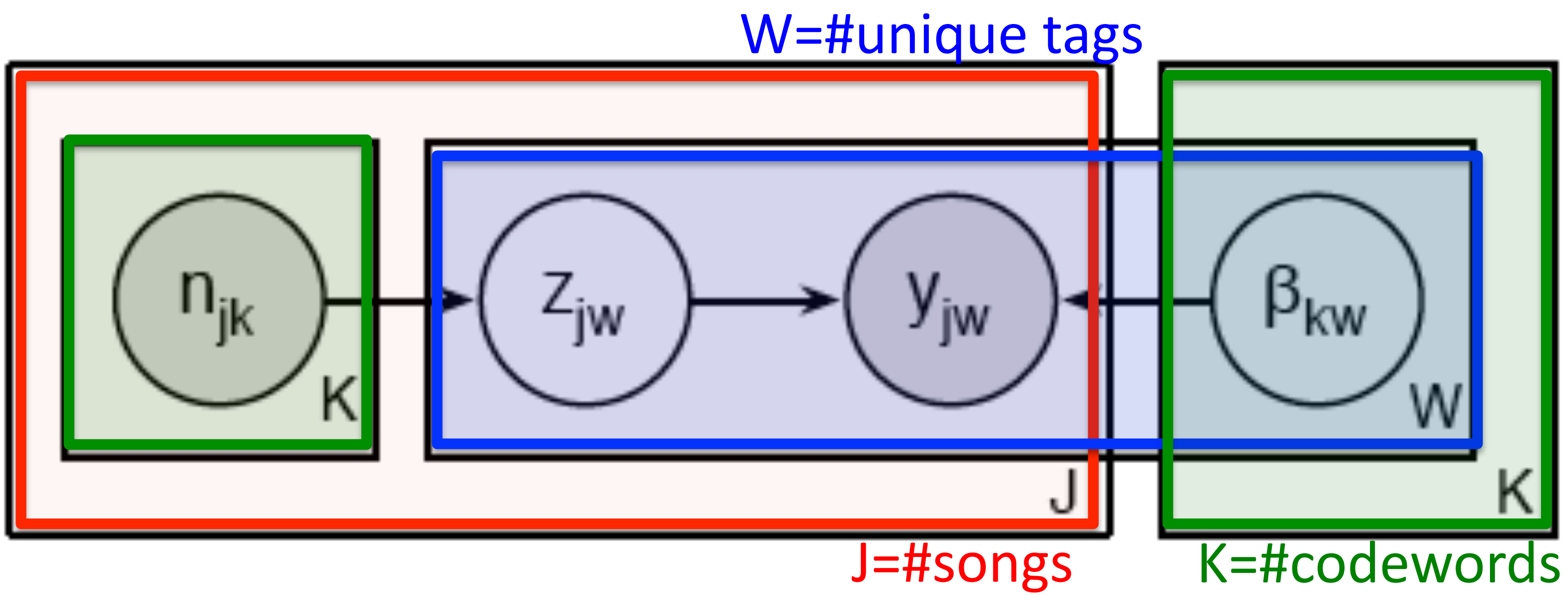
A compact feature representation

- Start with 39-dimensional MFCC-Deltas
  - CAL500: 10,000 unordered feature vectors **per song** (!)
- Vector quantize all feature vectors
  - VQ space: K codewords total (K=5 to 2500)
- Represent song as K-dimensional vector of codeword counts (K features, 1 datapoint per song)

Building a model

- “All models are wrong, but some are useful.” – George E. P. Box

Model



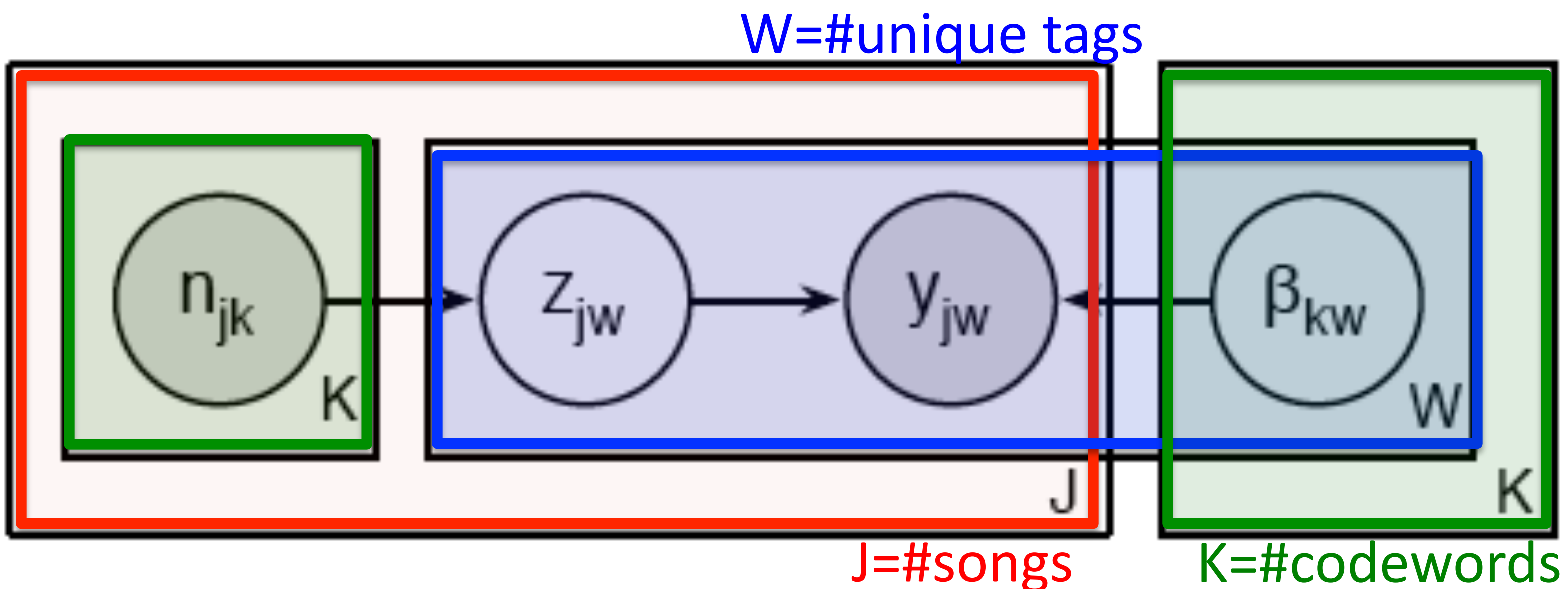
n_{jk} = # times codeword k appears in song j

z_{jw}= the w-th codeword in song j, a value 1:K

b_{kw} = parameter for Bernoulli for codeword k and tag w

y_{jw} = true iff tag w appears in song j

Generative process



For each song:
    For each possible tag:
        Draw a codeword from song j, from observed codewords
        Draw y from Bernoulli for that (codeword,tag) pair
        Apply the tag iff y is true.

Inference

- Find maximum likelihood estimators **β** for (codeword, tag) Bernoullis
  - i.e., find **β** to maximize p (y | n, **β**)
- Use EM algorithm to estimate MLEs
  - Latent variable z comes in handy here

Using the model: Annotation

- Can directly compute probability that tag w applies to new song j, using song features & parameters computed in inference step:

$$
\begin{aligned}
p(y_{jw}|\boldsymbol{n}_j, \beta) &= \sum_k p(z_{jw} = k|\boldsymbol{n}_j) p(y_{jw}|z_{jw} = k) \\
p(y_{jw} = 1|\boldsymbol{n}_j, \beta) &= \frac{1}{N_j} \sum_k n_{jk} \beta_{kw} \qquad (11)
\end{aligned}
$$

Using the model: Retrieval

- Compute probability of each tag applying to each song in database
  - => rank songs for each tag
- Return first N of ranked list for a tag query

Evaluation

- Compute IR metrics for annotation and retrieval (we'll discuss these tomorrow)
  - Compare to previously published results, "upper bound," and "random"
- Compute for different values of K (5 to 2500)
- Compute VQ, training and classification time
- Comparable to or better than previously published results
  - P <= .286 (Upper bound = .712)
  - R <= .162 (Upper bound = .375)

Other example applications in MIR

- Raphael: computer accompaniment of a human soloist
- Lanckriet, Turnbull, Barrington (cal @UCSD): **lots** of work, including tagging
- Hoffman @ Princeton: tagging, source separation / transcription
- Many GMM applications
  - e.g., Tzanetakis & Cook 2002

Wrap-up

Probabilistic generative models vs. discriminative classifiers

- Provide full probability model of all variables, **not just** P (class | features)
- Encode your own assumptions about data and "generative" process
- Take advantage of modularity, hierarchy, temporal behavior in data
- Leverage cutting-edge techniques from speech, vision, document analysis research
- Can be less "cookie-cutter" than classification; inference can be long; can require lots of knowledge of probability & statistics to do create new models, BUT it is still possible to read papers and appreciate contributions and assumptions without this.

Good reading

- Intro to graphical models:
  - Short: http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html by Kevin Murphy
  - Long: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.7467&rep=rep1&type=pdf by Michael Jordan
  - Video: http://videolectures.net/mlss07_ghahramani_grafm/ by Zoubin Ghahramani
- Graphical models & music:
  - A. T. Cemgil's ISMIR 2006 tutorial: http://www-sigproc.eng.cam.ac.uk/~atc27/papers/cemgil-ismir-tutorial.pdf
- EM algorithm for GMMs: http://bengio.abracadoudou.com/lectures/old/tex_gmm.pdf by Samy Bengio
- HMMs: http://www.google.com/url?sa=t&source=web&cd=1&ved=0CBIQFjAA&url=http%3A%2F%2Fwww.cs.ubc.ca%2F~murphyk%2FBayes%2Frabiner.pdf&ei=U7g2TKX8AYPCsAPkrvGoBQ&usg=AFQjCNHeXLhTHmuKUXKKCHYSs58TxVGfZg by Rabiner

Good textbooks

- *Pattern Recognition and Machine Learning* by Christopher M. Bishop, 2006
  - Excellent and readable machine learning textbook covering both generative and discriminative methods
- *Bayesian Data Analysis* by Andrew Gelman, 2nd ed, 2003
  - Extremely thorough and lots of information. True to the title.
- *Artificial Intelligence: A Modern Approach* by Stuart Russell and Peter Norvig, 3rd ed., 2009
  - A standard university textbook on AI, accessible to those without any prior knowledge. Focus is broader than machine learning, but good introductory treatment of several classifiers and HMMs.