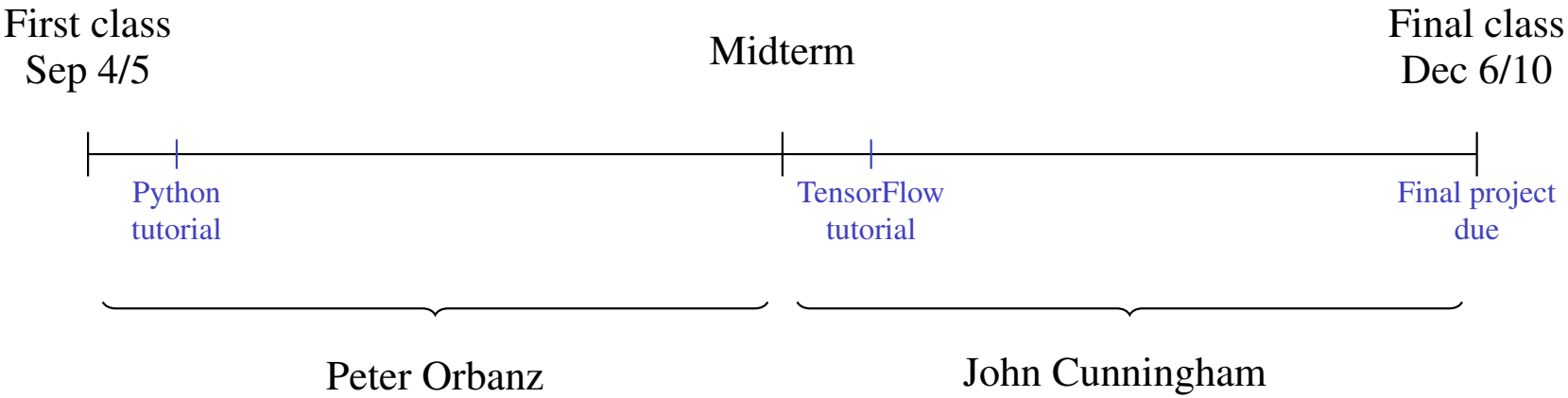
Course Admin

Term Timeline

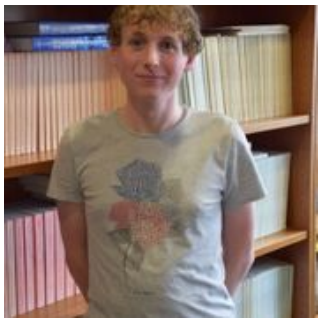


Dates

Python tutorial	6/10 September
Midterm exam	18/22 October
TensorFlow tutorial	24/25 October
Final project due	10 December

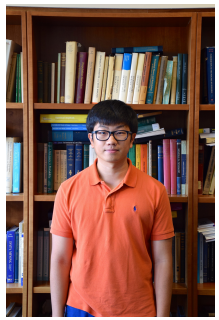
Assistants and Grading

Teaching Assistants



Andrew Davison

Ian Kinsella



Peter Lee

Gabriel Loaiza

Office Hours	Mon/Tue 5:30-7:30pm, Room 1025, Dept of Statistics, 10th floor SSW

Class Homepage

`https://www.adavison.co.uk/teaching/AdvancedML18/`

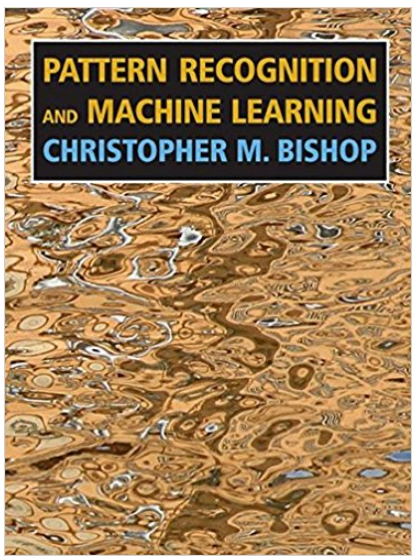
Homework

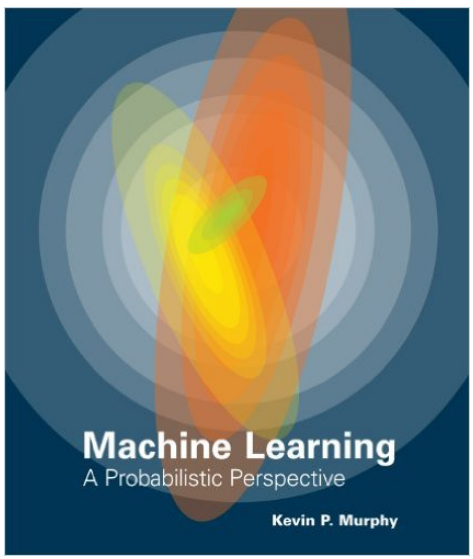
- Some homework problems and final project require coding
- Coding: Python
- Homework due: Tue/Wed at 4pm – no late submissions, please
- You can drop two homeworks from your final score

READING

The relevant course material are the slides.

Books (optional)





See class homepage for references.

TOPICS (TENTATIVE)

Part I (Orbanz)

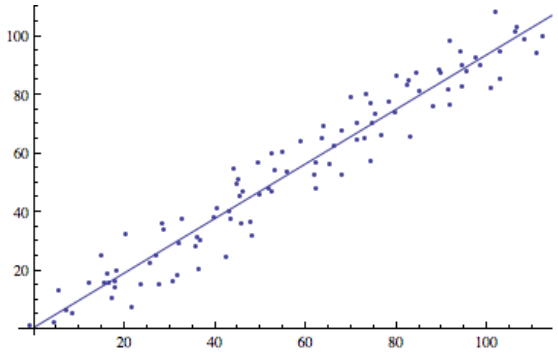
- Markov and hidden Markov models
- Graphical models
- Sampling and MCMC algorithms
- Variational inference
- Neural network basics

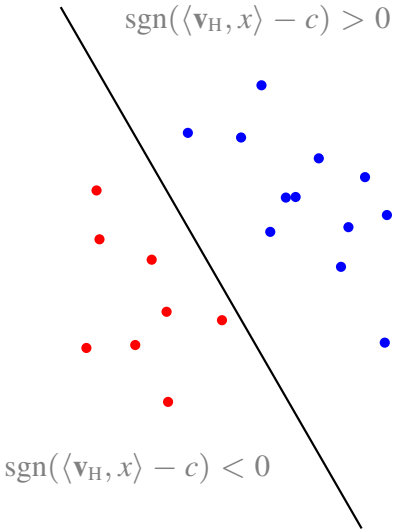
Part II (Cunningham)

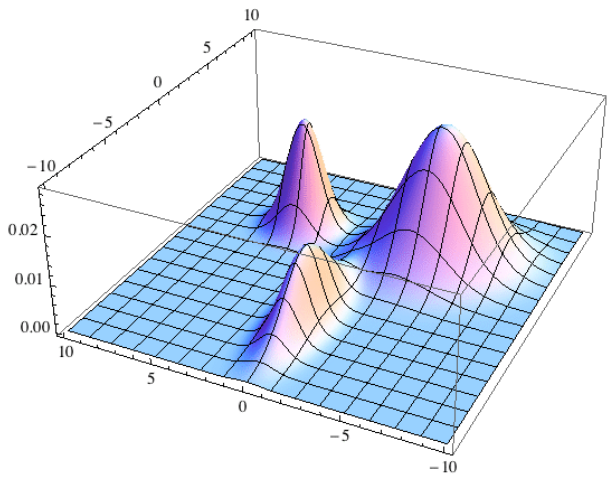
- NN software
- Neural networks: convolutional, recurrent, etc.
- Reinforcement learning
- Dimension reduction and autoencoders

INTRODUCTION

Recall Previous Term





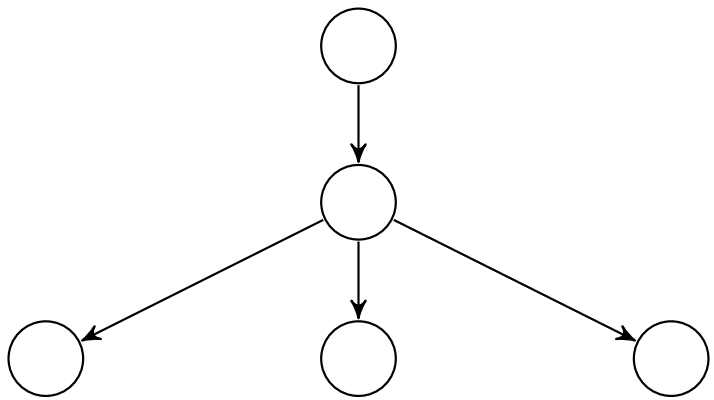


	Supervised learning	Unsupervised learning
Problems	Classification Regression	Clustering (mixture models) HMMs Dimension reduction (PCA)
Solutions	Functions	Distributions

Our main topics

Neural networks

- Define *functions*
- Represented by directed graph



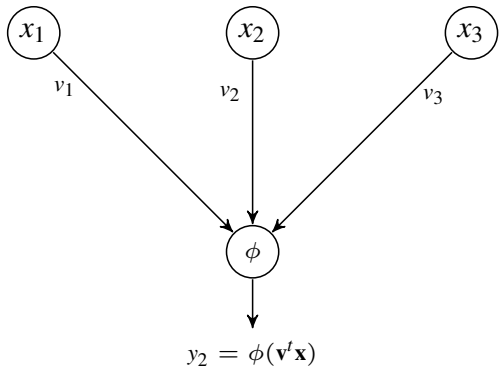
- Each vertex represents a function
- Incoming edges: Function arguments
- Outgoing edges: Function values
- Learning: Differentiation/optimization

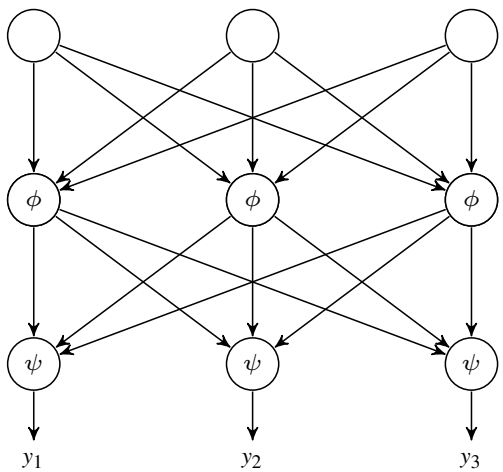
Graphical models

- Define *distributions*
- Represented by directed graph

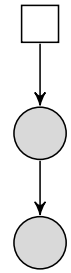
- Each vertex represents a distribution
- Incoming edges: Conditions
- Outgoing edges: Draws from distribution
- Learning: Estimation/inference

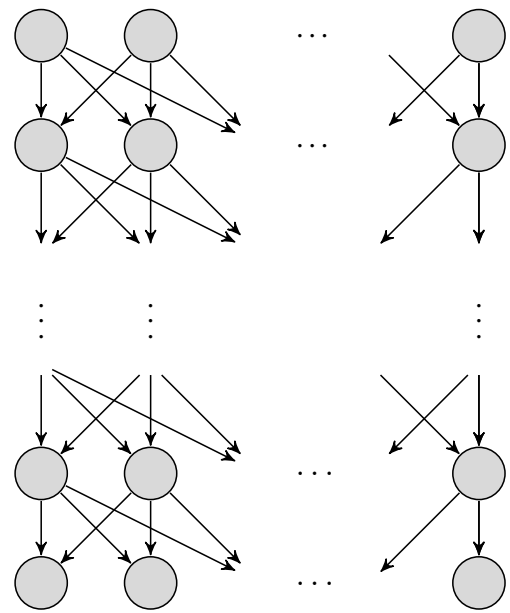
Neural networks





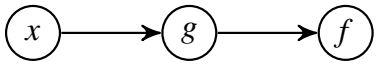
Graphical models





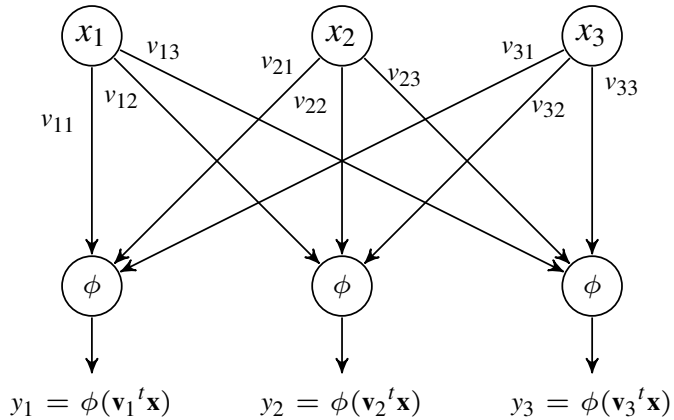
Neural networks

- Representation of function using a graph
- Layers:



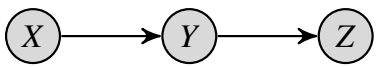
- Symbolizes: $f(g(x))$

"f depends on x only through g"



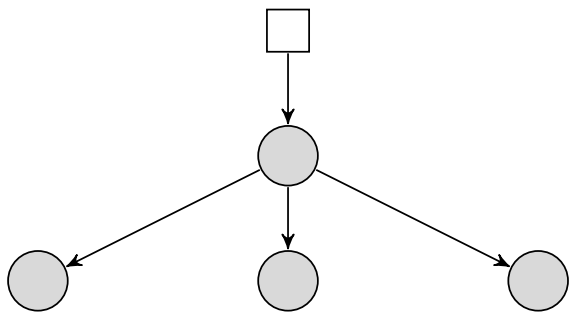
Graphical models

- Representation of a distribution using a graph
- Layers:



- Symbolizes: $p(x,z,y) = p(z|y)p(y|x)p(x)$

"Z is conditionally independent of X given Y"



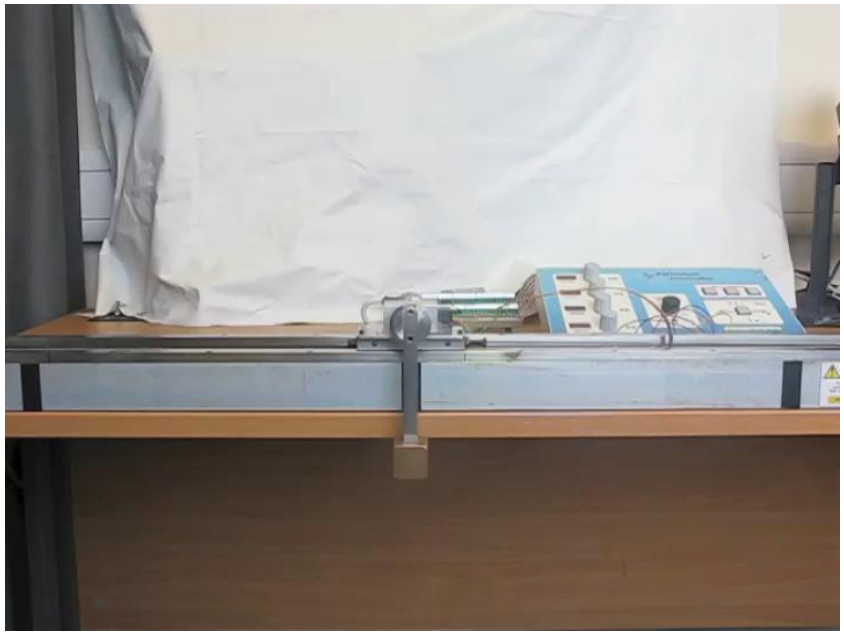
TOPICS (TENTATIVE)

Part I (Orbanz)

- Markov and hidden Markov models
- Graphical models
- Sampling and MCMC algorithms
- Variational inference
- Neural network basics

Part II (Cunningham)

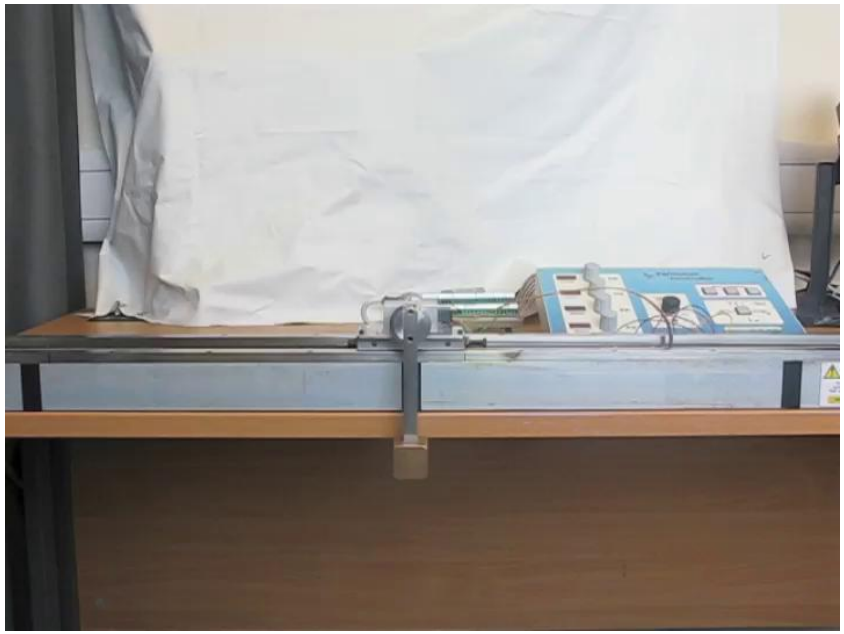
- NN software
- Neural networks: convolutional, recurrent, etc.
- Reinforcement learning
- Dimension reduction and autoencoders



Task

Balance the pendulumn upright by moving the sled left and right.

- The computer can control *only* the motion of the sled.
- Available data: Current state of system (measured 25 times/second).



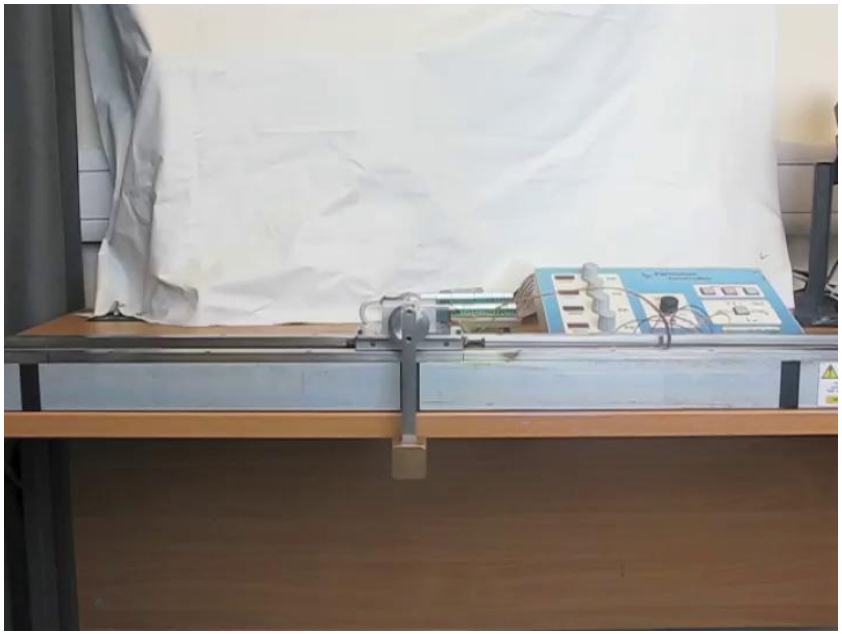
Formalization

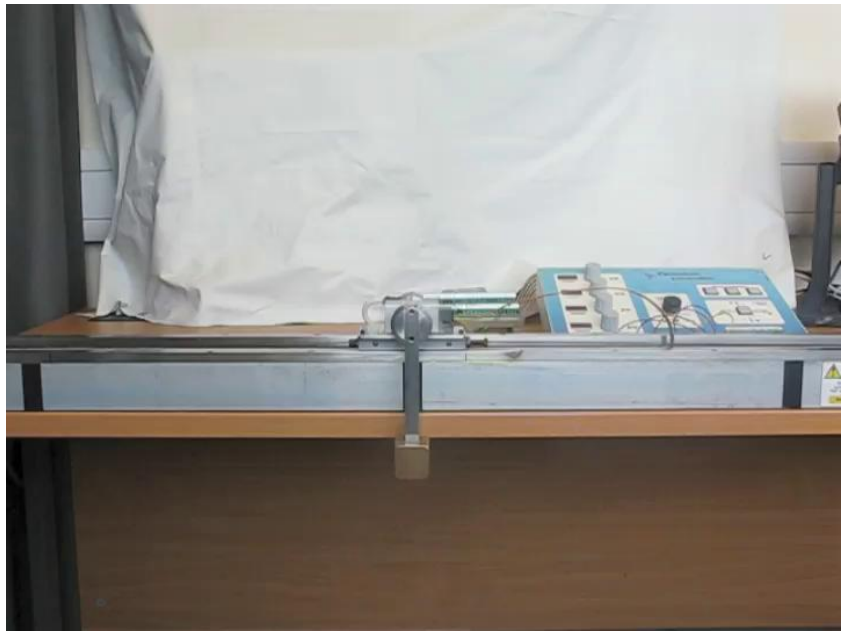
State	=	4 variables (sled location, sled velocity, angle, angular velocity)
Actions	=	sled movements

The system can be described by a function

$$\begin{aligned} f: \quad & \mathcal{S} \times \mathcal{A} \quad \rightarrow \quad \mathcal{S} \\ & (\text{state}, \text{action}) \mapsto \text{state} \end{aligned}$$

Learning and Statistics





After each run

Fit a function

$$\begin{array}{rccc} f: & \mathcal{S} \times \mathcal{A} & \to & \mathcal{S} \\ & (\text{state}, \text{action}) & \mapsto & \text{state} \end{array}$$

to the data obtained in previous runs.

Running the system involves:

1. The function f, which tells the system "how the world works".
2. An optimization method that uses f to determine how to move towards the optimal state.

Note well

Learning how the world works is a regression problem.

SEQUENTIAL DATA AND MARKOV MODELS

Simple random walk

Start with a graph G. Define a random sequence of vertices as follows:

- Choose a vertex X_1 uniformly at random.
- Choose a vertex X_2 uniformly at random from the neighbors of X_1. Move to X_2.
- Iterate: At step n, uniformly sample a neighbor X_n of X_{n-1}, and move to X_n.

This is called *simple random walk* on G.

Google's PageRank Algorithm

To sort the web pages matching a search query by importance, PageRank:

1. Defines a graph G whose vertices are web pages and whose edges are web links.
2. Computes the probability distribution on vertices x in G given by

$$P_n(x) = P(X_n = x) \qquad \text{where} \qquad X_1, \dots X_n \text{ is a simple random walk on } G$$

and n is very large.

We will try to understand (a) why and (b) how P_n can be computed.

So far: I.i.d. sequences

We have assumed that samples are of the form

$$X_1 = x_1, X_2 = x_2, \ldots \qquad \text{where} \qquad X_1, X_2, \ldots \sim_{\text{iid}} P$$

for some distribution P. In particular, the order of observations does not matter.

Now: Dependence on the past

We now consider sequences in which the random variable X_n can be stochastically dependent on $X_1, \ldots, X_{n-1}$, so we have to consider conditional probabilities of the form

$$P(X_n = x_n | X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) \ .$$

Application examples

- Speech and handwriting recognition.
- Time series, e.g. in finance. (These often assume a *continuous* index. Our index n is discrete.)
- Simulation and estimation algorithms (Markov chain Monte Carlo).
- Random walk models (e.g. web search).

Markov models

The sequence $(X_n)_n$ is called a **Markov chain of order** r if X_n depends only on a fixed number r of previous samples, i.e. if

$$P(X_n = x_n | X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) \;=\; P(X_n = x_n | X_{n-r} = x_{n-r}, \ldots, X_{n-1} = x_{n-1})$$

If we simply call $(X_n)_n$ a **Markov chain**, we imply $r = 1$.

Initial state

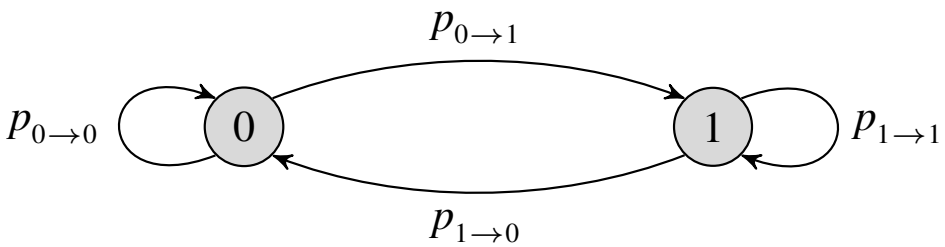
The first state in the sequence is special because it does not have a "past", and is usually denoted X_0.

Example: $r = 2$

$$\underbrace{X_0 = x_0,\ X_1 = x_1,}_{\substack{X_4 \text{ is independent} \\ \text{of these } \textit{given } X_2, X_3}}\ \underbrace{X_2 = x_2,\ X_3 = x_3,}_{X_4 \text{ may depend on these}}\ X_4 = ?$$

A simple binary chain

Suppose $\mathbf{X} = \{0, 1\}$.



- We regard 0 and 1 as possible "states" of X, represented as vertices in a graph.
- Each pair $X_{n-1} = s, X_n = t$ in the sequence is regarded as a "transition" from s to t and represented as an edge in the graph.
- Each edge $s \to t$ is weighted by the probability

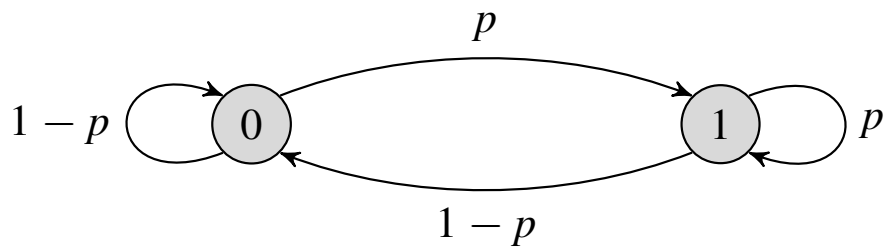
$$p_{s\to t} := P(X_n = t | X_{n-1} = s) \ .$$

State space

The elements of the sample space $\mathbf{X}$ are called the **states** of the chain. $\mathbf{X}$ is often called the **state space**. We generally assume that $\mathbf{X}$ is finite, but Markov chains can be generalized to infinite and even uncountable state spaces.

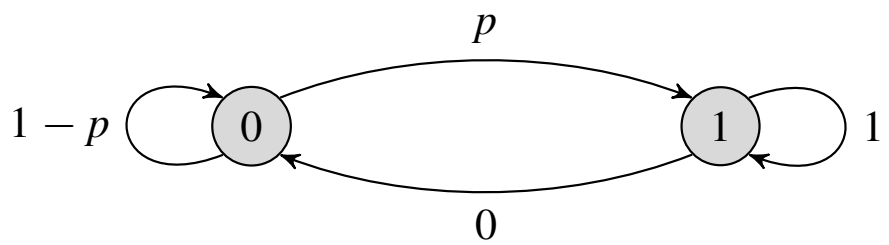
First example: Independent coin flips

Suppose X is a biased coin with $P(X_n = 1) = p$ independently of X_{n-1}. In other words, the sequence (X_n) is iid Bernoulli with parameter p.



Breaking independence

Here is a simple modification to the chain above; only $p_{1\to 0}$ and $p_{1\to 1}$ have changed:



This is still a valid Markov chain, but the elements of the sequence are no longer independent.

Observation

The graph representation is only possible if $p_{s\to t}$ is independent of n. Otherwise we would have to draw a different graph for each n.

If $p_{s\to t}$ does not depend on n, the Markov chain is called **stationary**.

Transition matrix

The probabilities $p_{s\to t}$ are called the **transition probabilities** of the Markov chain. If $|\mathbf{X}| = d$, the $d \times d$-matrix

$$\mathbf{p} := (p_{i\to j})_{j,i\leq d} = \begin{pmatrix} p_{1\to 1} & \cdots & p_{d\to 1} \\ \vdots & & \vdots \\ p_{1\to d} & \cdots & p_{d\to d} \end{pmatrix}$$

is called the **transition matrix** of the chain. This is precisely the adjacency matrix of the graph representing the chain. Each column is a probability distribution on d events.

Complete description of a Markov chain

The transition matrix does not completely determine the chain: It determines the probability of a state given a previous state, but not the probability of the starting state. We have to additionally specify the distribution of the first state.

Initial distribution

The distribution of the first state, i.e. the vector

$$P_{\text{init}} := (P(X_0 = 1), \ldots, P(X_0 = d)) \ ,$$

is called the **initial distribution** of the Markov chain.

Representing stationary Markov chains

Any stationary Markov chain with finite state space can be completely described by a transition matrix $\mathbf{p}$ and an initial distribution P_{init}. That is, the pair $(\mathbf{p}, P_{\text{init}})$ completely determines the joint distribution of the sequence $(X_0, X_1, \ldots)$.

Simple random walk

Suppose we are given a directed graph G (with unweighted edges). We had already mentioned that the **simple random walk** on G is the vertex-valued random sequence $X_0, X_1, \ldots$ defined as:

- We select a vertex X_0 in G uniformly at random.
- For $n = 1, 2, \ldots$, select X_n uniformly at random from the children of X_{n-1} in the graph.

Markov chain representation

Clearly, the simple random walk on a graph with d vertices is a Markov chain with

$$P_{\text{init}} = \left(\frac{1}{d}, \ldots, \frac{1}{d}\right) \qquad \text{and} \qquad p_{i \to j} = \begin{cases} \frac{1}{\#\text{ edges out of } i} & \text{if } i \text{ links to } j \\ 0 & \text{otherwise} \end{cases}$$

Generalizing simple random walk

We can generalize the idea of simple random walk by substituting the uniform distributions by other distributions. To this end, we can weight each edge in the graph by a probability of following that edge.

Adjacency matrix

If the edge weights are proper probabilities, each row of the adjacency matrix must sum to one. In other words, the matrix is the transition matrix of a Markov chain.

Random walks and Markov chains

If we also choose a general distribution for the initial state of the random walk, we obtain a completely determined Markov chain. Hence:

Any Markov chain on a finite state space is a random walk on a weighted graph and vice versa.

Queries

The first step in internet search is query matching:

1. The user enters a search query (a string of words).
2. The search engine determines all web pages indexed in its database which match the query.

This is typically a large set. For example, Google reports ca 83 million matches for the query "random walk".

The ranking problem

- For the search result to be useful, the most useful link should with high probability be among the first few matches shown to the user.
- That requires the matching results to be *ranked*, i.e. sorted in order of decreasing "usefulness".

Available data

Using a web crawler, we can (approximately) determine the link structure of the internet. That is, we can determine:

- Which pages there are.
- Which page links which.

A web crawler cannot determine:

- How often a link is followed.
- How often a page is visited.

Web graph

The link structure can be represented as a graph with

vertices = web pages and edges = links.

Key idea

The popularity of a page x is proportional to the probability that a "random web surfer" ends up on page x after a n steps.

Probabilistic model

The path of the surfer is modeled by a random walk on the web graph.

Modeling assumptions

Two assumptions are implicit in this model:

1. Better pages are linked more often.
2. A link from a high-quality page is worth more than one from a low-quality page.

Remarks

- We will find later that the choice of n does not matter.
- To compute the popularity score, we first have to understand Markov chains a bit better.

Probability after $n = 1$ steps

If we *know* the initial state, then

$$P(X_1 = s_1 \,|\, X_0 = s_0) = p_{s_0 \to s_1} \ .$$

P_1 describes the probability of X_1 if we do *not* know the starting state (i.e. the probability before we start the chain):

$$\begin{aligned} P_1(s_1) = P(X_1 = s_1) &= \sum_{s_0 \in \mathbf{X}} P(X_1 = s_1 \,|\, X_0 = s_0) P_{\text{init}}(s_0) \\ &= \sum_{s_0 \in \mathbf{X}} p_{s_0 \to s_1} P_{\text{init}}(s_0) \ . \end{aligned}$$

Matrix representation

Recall that $\mathbf{p}$ is a $d \times d$-matrix and P_{init} a vector of length d. The equation for P_1 above is a matrix-vector product, so

$$P_1 = \mathbf{p} \cdot P_{\text{init}} \ .$$

Probability after $n = 2$ steps

The same argument shows that P_2 is given by

$$P_2(s_2) = \sum_{s_1 \in \mathbf{X}} p_{s_1 \to s_2} P_1(s_1) \ ,$$

hence

$$P_2 = \mathbf{p} \cdot P_1 = \mathbf{p} \cdot \mathbf{p} \cdot P_{\text{init}} \ .$$

For arbitary n

$$P_n = \mathbf{p}^n P_{\text{init}}$$

Limiting distribution

Instead of considering P_n for a specific, large n, we take the limit

$$P_\infty := \lim_{n\to\infty} P_n = \lim_{n\to\infty} \mathbf{p}^n P_{\text{init}} \ ,$$

provided that the limit exists.

Observation

If the limit P_∞ exists, then

$$\mathbf{p} \cdot P_\infty = \mathbf{p} \cdot \lim_{n\to\infty} \mathbf{p}^n P_{\text{init}} = \lim_{n\to\infty} \mathbf{p}^n P_{\text{init}} = P_\infty \ ,$$

which motivates the next definition.

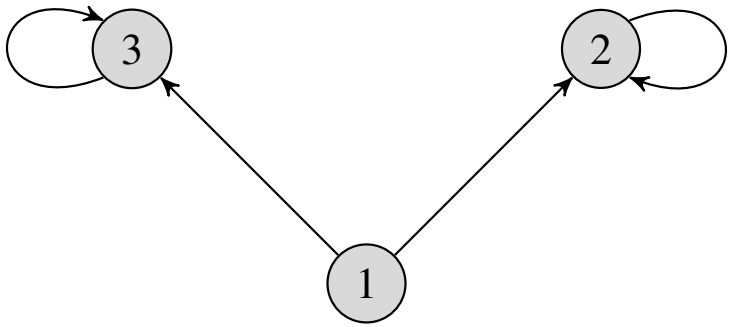
Equilibrium distribution

If $\mathbf{p}$ is the tansition matrix of a Markov chain, a distribution P on $\mathbf{X}$ which is invariant under $\mathbf{p}$ in the sense that

$$\mathbf{p} \cdot P = P$$

is called an **equilibrium distribution** or **invariant distribution** of the Markov chain.

Problem 1: The equilibrium distribution may not be unique



For this chain, both $P = (0, 1, 0)$ and $P' = (0, 0, 1)$ are valid equilibria. Which one emerges depends on the initial state and (if we start in state 1) on the first transition.

Remedy

Require that there is a path in the graph (with non-zero probability) from each state to every other state. A Markov chain satisfying this condition is called **irreducible**.

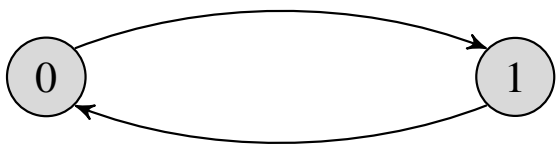
Recall that a sequence in $\mathbb{R}$ does not have a limit if it "oscillates". For example,

$$\lim_n 1^n = 1 \qquad \text{but} \qquad \lim_n (-1)^n \text{ does not exist}$$

Problem 2: The limit may not exist

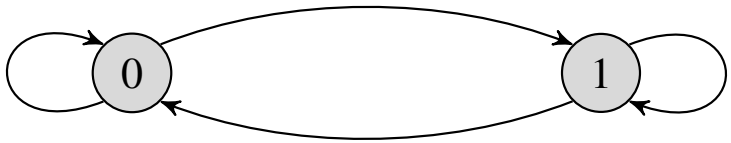
- The chain on the right has no limit distribution.
- If we start e.g. in state 0, then:
  - 0 can only be reached in even steps.
  - 1 only in odd steps.
- The distribution P_n oscillates between

$$P_{\text{even}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad P_{\text{odd}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} .$$



Remedy

To prevent this (particular) problem, we can add two edges:



Now each state is reachable in every step.

The problem (at least this example) is that we have to leave the state before we can return to it. We prevent this, we introduce the following definition.

Aperiodic chains

We call a stationary Markov chain **aperiodic** if, for every state s,

$$P(X_n = s \,|\, X_{n-1} = s) = p_{s \to s} > 0 \ .$$

In short, a stationary chain is aperiodic if the transition matrix has non-zero diagonal.

Equilibrium Distributions

We have introduced two definitions which prevent two rather obvious problems. Surprisingly, these definitions are all we need to guarantee limits.

Theorem

Suppose a Markov chain $(\mathbf{p}, P_{\text{init}})$ is stationary, and for each state $s \in \mathbf{X}$:

1. There is a path (with non-zero probability) from s to every other state (i.e. the chain is irreducible).
2. $p_{s \to s} > 0$ (i.e. the chain is aperiodic).

Then:

- The limit distribution P_∞ exists.
- The limit distribution is also the equlibrium distribution.
- The equilibrium distribution is unique.

Power method

If the the transition matrix $\mathbf{p}$ makes the chain irreducible and aperiodic, we know that

$$\text{equilibrium distribution} = \text{limit distribution} \quad .$$

This means we can approximate the equilibrium P_∞ by P_n: We start with any distribution P_{init} (e.g. uniform) and repeatedly multiply by $\mathbf{p}$:

$$P_{n+1} = \mathbf{p} \cdot P_n$$

We can threshold the change between steps, e.g. by checking $\|P_{n+1} - P_n\| < \tau$ for some small τ.

Remark: Eigenstructure

The power method can be regarded as an eigenvector computation. The definition

$$P = \mathbf{p} \cdot P$$

of the equilibrium means that $P = P_\infty$ is an eigenvector of $\mathbf{p}$ with eigenvalue 1. If $\mathbf{p}$ is irreducible and aperiodic, it can be shown that 1 is the largest eigenvalue.

Constructing the transition matrix

We start with the web graph and construct the transition matrix of simple random walk, i.e.

$$a_{ij} := \begin{cases} \frac{1}{\#\text{ edges out of } i} & \text{if } i \text{ links to } j \\ 0 & \text{otherwise} \end{cases}$$

A chain defined by $A := (a_{ij})$ will almost certainly not be irreducible (think of web pages which do not link anywhere). We therefore regularize A by defining

$$\mathbf{p} := (1-\alpha)A + \frac{\alpha}{d}\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

for some small $\alpha \in (0,1)$.
Clearly, this makes $\mathbf{p}$ both irreducible and aperiodic.

Computing the equilibrium

Given $\mathbf{p}$, the equilibrium distribution is computed using the power method. Since the web changes, the power method can be re-run every few days with the previous equilibrium as initial distribution.

The Random Surfer Again

We can now take a more informed look at the idea of a random web surfer:

- Suppose the surfer is more likely to start on a popular page than on an unpopular one.
- In terms of the popularity model, this means

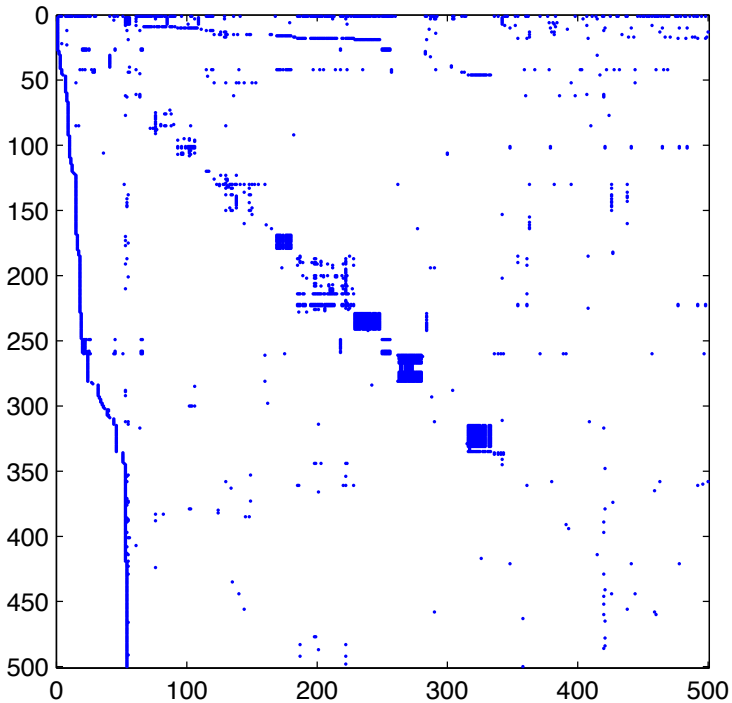
$$X_0 \sim P_{\text{equ}} \ ,$$

  where P_{equ} is the equilibrium distribution of the chain.
- After following any number of links n (with probabilities given by the transition matrix $\mathbf{p}$),

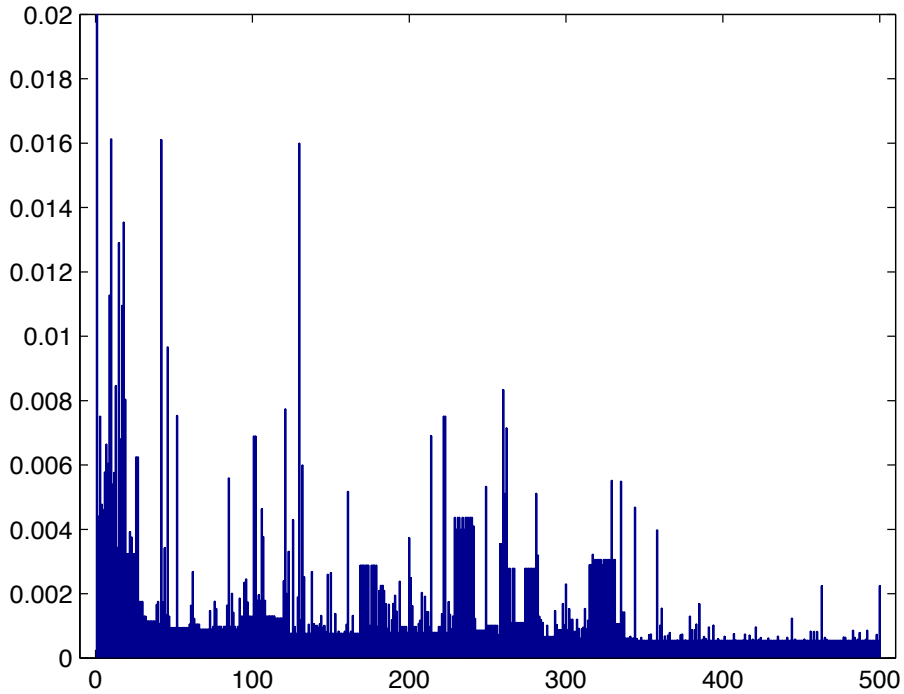
$$P_n = \mathbf{p}^n P_{\text{equ}} = P_{\text{equ}} \ .$$

- In this sense, P_{equ} is really the consistent solution to our problem, even if we *compute* it by starting the random walk from e.g. a uniform initial distribution instead.
- In particular, it does not matter how we choose n in the model.

EXAMPLE



Adjacence matrix of the web graph of 500 web pages. The root (index 0) is www.harvard.edu.



Equilibrium distribution computed by PageRank.

See K. Murphy, "Machine Learning", MIT Press 2012.

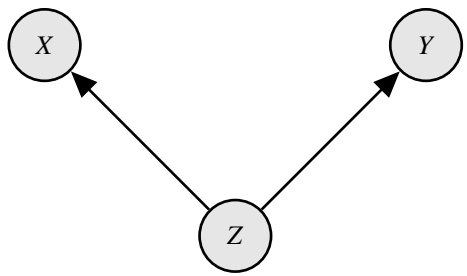
GRAPHICAL MODELS

A graphical model represents the dependence structure within a set of random variables as a graph.

Overview

Roughly speaking:

- Each random variable is represented by vertex.
- If Y depends on X, we draw an edge $X \to Y$.
- For example:



  This says: "X depends on Z, and Y depends on Z".
- We have to be careful: The above does not imply that X and Y are independent. We have to make more precise what *depends on* means.

We will use the notation:

$$\begin{aligned} \mathcal{L}(X) &= \text{ distribution of the random variable } X \\ \mathcal{L}(X|Y) &= \text{ conditional distribution of } X \text{ given } Y \end{aligned}$$

($\mathcal{L}$ means "law".)

Reason

- If X is discrete, $\mathcal{L}(X)$ is usually given by a mass function $P(x)$.
- If it is continuous, $\mathcal{L}(X)$ is usually given by a density $p(x)$.
- With the notation above, we do not have to distinguish between discrete and continuous variables.

Dependence between random variables $X_1, \ldots, X_n$ is a property of their joint distribution $\mathcal{L}(X_1, \ldots, X_n)$.

Recall

Two random variables are *stochastically independent*, or *independent* for short, if their joint distribution factorizes:

$$\mathcal{L}(X, Y) = \mathcal{L}(X)\mathcal{L}(Y)$$

For densities/mass functions:

$$P(x, y) = P(x)P(y) \qquad \text{or} \qquad p(x, y) = p(x)p(y)$$

Dependent means *not independent*.

Intuitively

X and Y are dependent if knowing the outcome of X provides any information about the outcome of Y.

More precisely:

- If someone draws (X, Y) simultanuously, and only discloses $X = x$ to you, does that change your mind about the distribution of Y? (If so: Dependence.)
- Once X is given, the distribution of Y is the conditional $\mathcal{L}(Y|X = x)$.
- If that is still $\mathcal{L}(Y)$, as before X was drawn, the two are independent. If $\mathcal{L}(Y|X = x) \neq \mathcal{L}(Y)$, they are dependent.

Definition

Given random variables X, Y, Z, we say that X is **conditionally independent of** Y **given** Z if

$$\mathcal{L}(X, Y|Z = z) = \mathcal{L}(X|Z = z)\mathcal{L}(Y|Z = z) \ .$$

That is equivalent to

$$\mathcal{L}(X|Y = y, Z = z) = \mathcal{L}(X|Z = z) \ .$$

Notation

$$X \perp\!\!\!\perp_Z Y$$

Intuitively

X and Y are dependent given $Z = z$ if, although Z is known, knowing the outcome of X provides additional information about the outcome of Y.

EXAMPLE: MARKOV CHAINS (OF ORDER 1)

Consider a Markov chain (X_0, X_1, X_2):

$$X_0 = x_0, X_1 = x_1, X_2 = ?$$

X_2 is conditionally independent of X_0 given X_1:

$$\mathcal{L}(X_2|X_1 = x_1, X_0 = x_0) = \mathcal{L}(X_2|X_1 = x_1)$$

However: If we do not condition on X_1, then X_2 need *not* be independent of X_0:

$$\mathcal{L}(X_2|X_0 = x_0) \neq \mathcal{L}(X_2)$$

In general

For a Markov chain of order 1,

$$X_n \perp\!\!\!\perp_{X_{n-1}} X_{n-2}, \ldots, X_0$$

but not

$$X_n \perp\!\!\!\perp X_{n-2}, \ldots, X_0$$

Factorizing a joint distribution

The joint probability of random variables $X_1, \ldots, X_n$ can always be factorized as

$$\mathcal{L}(X_1, \ldots, X_n) = \mathcal{L}(X_n | X_1, \ldots, X_{n-1}) \mathcal{L}(X_{n-1} | X_1, \ldots, X_{n-2}) \cdots \mathcal{L}(X_1) \ .$$

Note that we can re-arrange the variables in any order.

If there are conditional independencies, we can remove some variables from the conditionals:

$$\mathcal{L}(X_1, \ldots, X_n) = \mathcal{L}(X_n | \mathcal{X}_n) \mathcal{L}(X_{n-1} | \mathcal{X}_{n-1}) \cdots \mathcal{L}(X_1) \ ,$$

where $\mathcal{X}_i$ is the subset of $X_1, \ldots, X_n$ on which X_i depends.

Definition

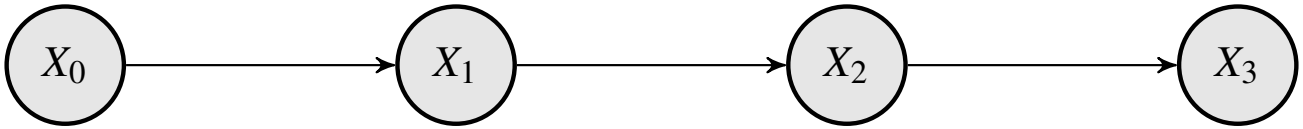
Let $X_1, \ldots, X_n$ be random variables. A **(directed) graphical model** represents a factorization of joint distribution $\mathcal{L}(X_1, \ldots, X_n)$ as follows:

- Factorize $\mathcal{L}(X_1, \ldots, X_n)$.
- Add one vertex for each variable X_i.
- For each variable X_i, add and edge from each variable $X_j \in \mathcal{X}_i$ to X_i.

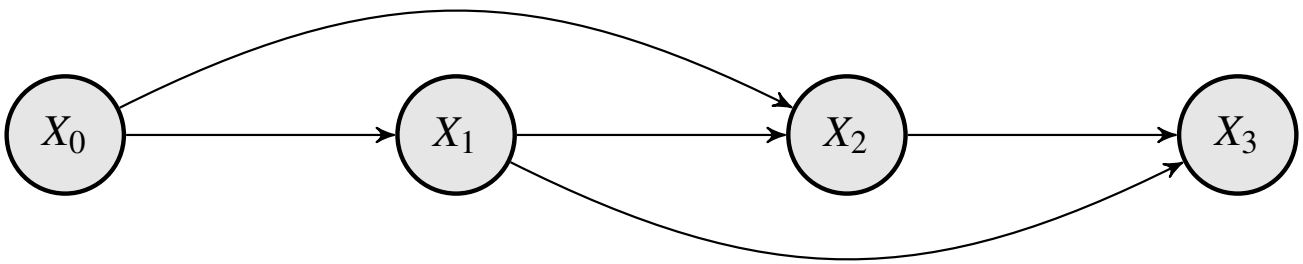
That is: An edge $X_j \to X_i$ is added if $\mathcal{L}(X_1, \ldots, X_n)$ contains the factor $\mathcal{L}(X_i | X_j)$.

EXAMPLE: MARKOV CHAINS

Markov chain of order $r = 1$



Markov chain of order $r = 2$



Note these graphs are graphical models, not Markov chain transition diagrams as those on the Markov chain slides. Here, each node is a random variable. In transition diagrams, each node represents a possible value of these variables.

Lack of uniqueness

The factorization is usually not unique, since e.g.

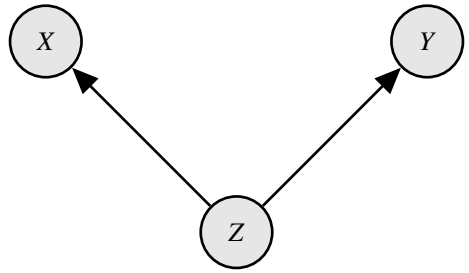
$$\mathcal{L}(X, Y) = \mathcal{L}(X|Y)\mathcal{L}(Y) = \mathcal{L}(Y|X)\mathcal{L}(X) \ .$$

That means the direction of edges is not generally determined.

Remark

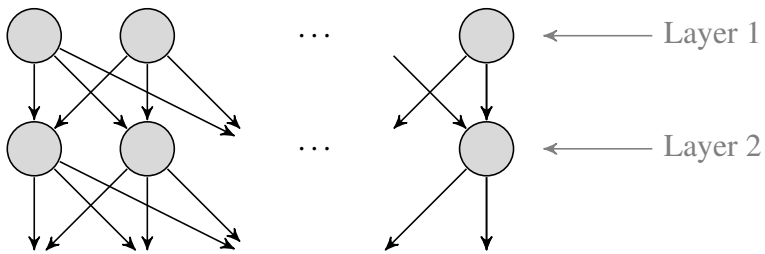
- If we use a graphical model to *define* a model or visualize a model, we decide on the direction of the edges.
- Estimating the direction of edges from data is a very difficult (and very important) problem. This is one of the main subjects of a research field called **causal inference** or **causality**.

A simple example

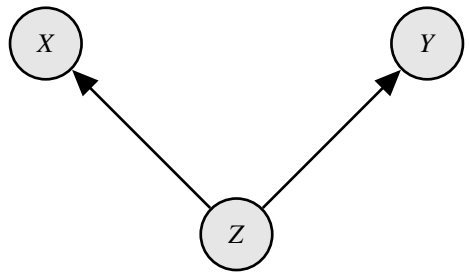


$$X \perp\!\!\!\perp_Z Y$$

An example with layers



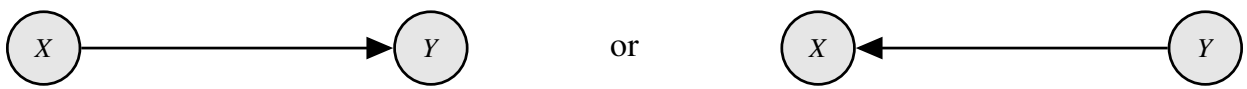
All variables in the $(k+1)$st layer are conditionally independent given the variables in the kth layer.



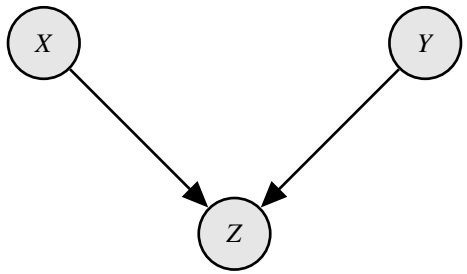
$X \perp\!\!\!\perp_Z Y$

Important

- X and Y are *not* independent, independence holds only conditionally on Z.
- In other words: If we do not observe Z, X and Y are dependent, and we have to change the graph:



Words of Caution II



Conditioning on Z makes X and Y dependent.

Example

- Suppose we start with two indepedent normal variables X and Y.
- $Z = X + Y$.

If we know Z, and someone reveals the value of Y to us, we know everything about X.

This effect is known as *explaining away*. We will revisit it later.

HIDDEN MARKOV MODELS

Motivation

We have already used Markov models to model sequential data. Various important types of sequence data (speech etc) have long-range dependencies that a Markov model does not capture well.

Hidden Markov model

- A hidden Markov model is a latent variable model in which a sequence of latent (or "hidden") variables is generated by a Markov chain.
- These models can generate sequences of *observations* with long-range dependencies, but the *explanatory* variables (the latent variables) are Markovian.
- It turns out that this is a useful way to model dependence for a variety of important problems, including speech recognition, handwriting recognition, and parsing problems in genetics.

Definition

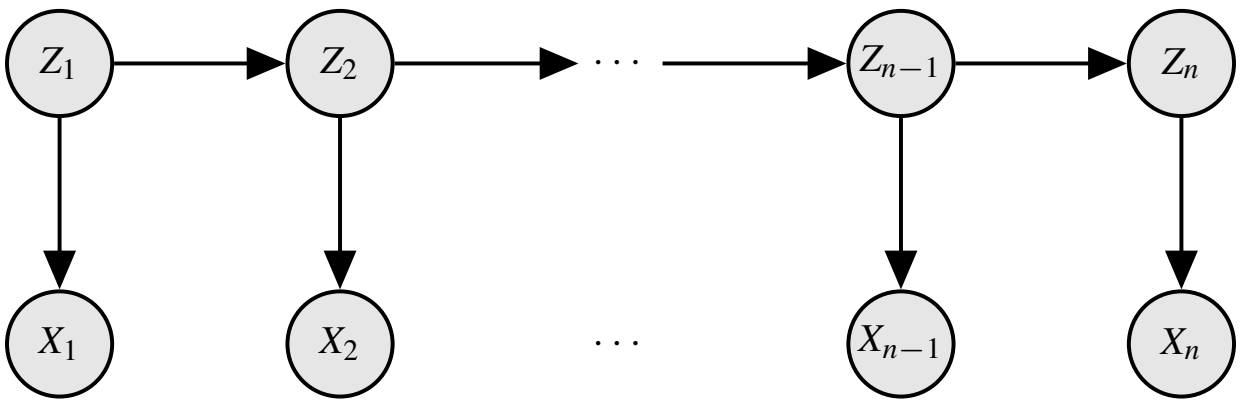
A (**discrete**) **hidden Markov model (HMM)** consists of:

- A stationary Markov chain $(Q_{\text{init}}, \mathbf{q})$ with states $\{1, \ldots, K\}$, initial distribution Q_{init} and transition matrix $\mathbf{q}$.
- A (discrete) **emission distribution**, given by a conditional probability $P(x|z)$.

The model generates a sequence $X_1, X_2, \ldots$ by:

1. Sampling a sequence $Z_1, Z_2, \ldots$ from the Markov chain $(Q_{\text{init}}, \mathbf{q})$.
2. Sampling a sequence $X_1, X_2, \ldots$ by independently sampling $X_i \sim P(\,.\,|Z_i)$.

In a **continuous HMM**, the variables X_i have continuous distributions, and $P(x|z)$ is substituted by a density $p(x|z)$. The Markov chain still has finite state space $[K]$.



NOTATION

We will see a lot of sequences, so we use the "programming" notation

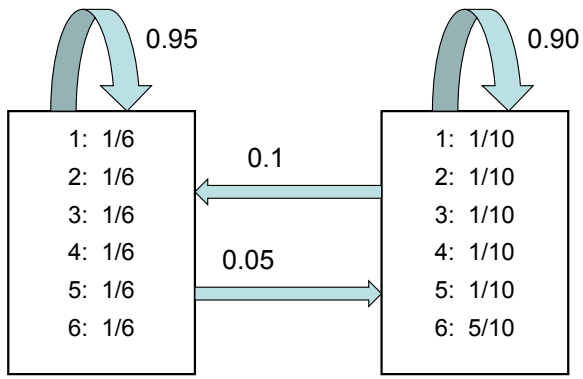
$$x_{1:n} := (x_1, \ldots, x_n)$$

EXAMPLE: DISHONEST CASINO

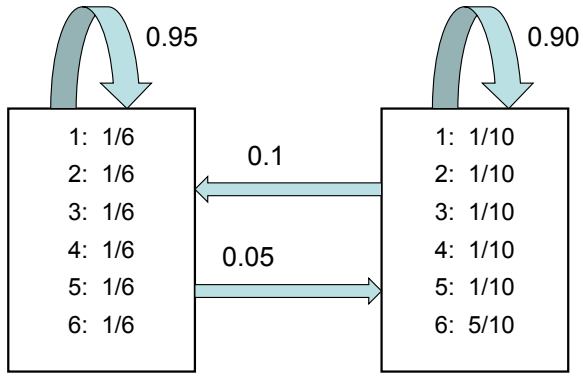
This example is used in many textbooks and is very simple, but it is useful to understand the conditional independence structure.

Problem

- We consider two dice (one fair, one loaded).
- At each roll, we either keep the current dice, or switch to the other one with a certain probability.
- A roll of the chosen dice is then observed.



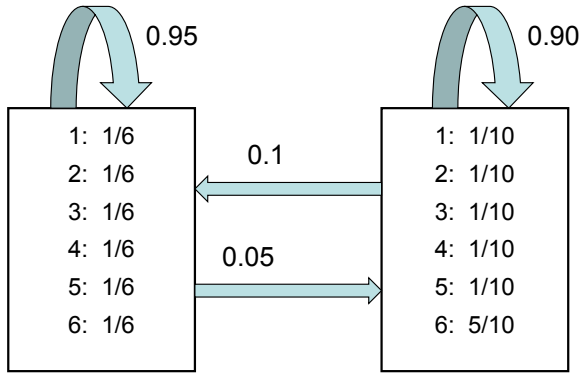
EXAMPLE: DISHONEST CASINO



HMM

- States: $Z_n \in \{\text{fair}, \text{loaded}\}$.
- Sample space: $\mathbf{X} = \{1, \ldots, 6\}$.
- Transition matrix: $\mathbf{q} = \begin{pmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{pmatrix}$
- Emission probabilities:
  $P(x|z = \text{fair}) = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$
  $P(x|z = \text{loaded}) = (1/10, 1/10, 1/10, 1/10, 1/10, 5/10)$

EXAMPLE: DISHONEST CASINO



Conditional independence

- Given the state (=which dice), the outcomes are independent.
- If we do not know the current state, observations are dependent!
- For example: If we observe a sequence of sixes, we are more likely to be in state "loaded" than "fair", which increases the probability of the next observation being a six.

Filtering problem

- **Given:** Model and observations, i.e. :
  1. Transition matrix **q** and emission distribution $P(\,.\,|z)$.
  2. Observed sequence $x_{1:N} = (x_1, \ldots, x_N)$.
- **Estimate:** Probability of each hidden variable, i.e. $Q(Z_n = k|x_{1:n})$

Variant: **Smoothing problem**, in which we estimate $Q(Z_n = k|x_{1:N})$ instead.

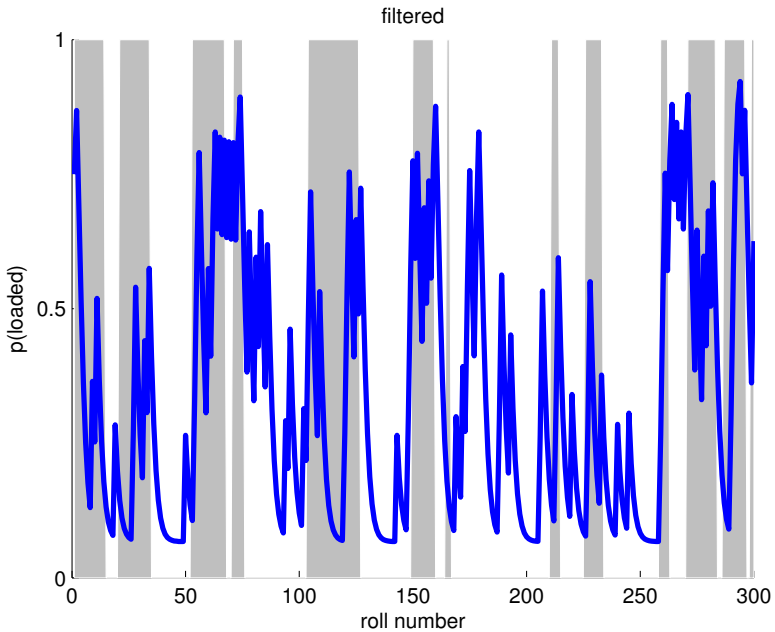
Decoding problem

- **Given:** Model (**q** and $P(\,.\,|z)$) and observed sequence $x_{1:N}$.
- **Estimate:** Maximum likelihood estimates $\hat{z}_{1:N} = (\hat{z}_1, \ldots, \hat{z}_N)$ of hidden states.

Learning problem

- **Given:** Observed sequence $x_{1:N}$.
- **Estimate:** Model (i.e. **q** and $P(\,.\,|z)$).

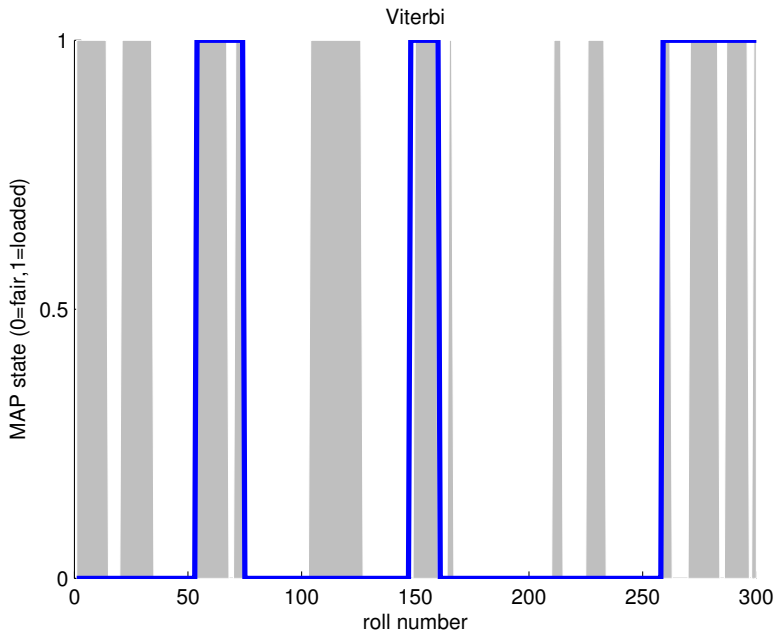
Before we look at the details, here are examples for the dishonest casino.



Filtering result.

Gray bars: Loaded dice used.

Blue: Probability $P(Z_n = \text{loaded} | x_{1:n})$



Decoding result.

Gray bars: Loaded dice used.

Blue: Most probable state Z_n.

The first estimation problem we consider is to estimate the probabilities $Q(z_n|x_{1:n})$.

Idea

We could use Bayes' equation (recall: $P(a|b) = \frac{P(b|a)P(a)}{P(b)}$) to write:

$$Q(k|x_n) = \frac{P(x_n|k)Q(Z_n = k)}{\sum_{k=1}^{K} P(x_n|k)Q(Z_n = k)} \ .$$

Since we know the Markov chain $(Q_{\text{init}}, \mathbf{q})$, we can compute Q, and the emission probabilities $P(x_n|k)$ are given.

Filtering

The drawback of the solution above is that it throws away all information about the past. We get a better estimate of Z_n by taking $x_1, \ldots, x_{n-1}$ into account. Reducing the uncertainty in Z_n using $x_1, \ldots, x_{n-1}$ is called **filtering**.

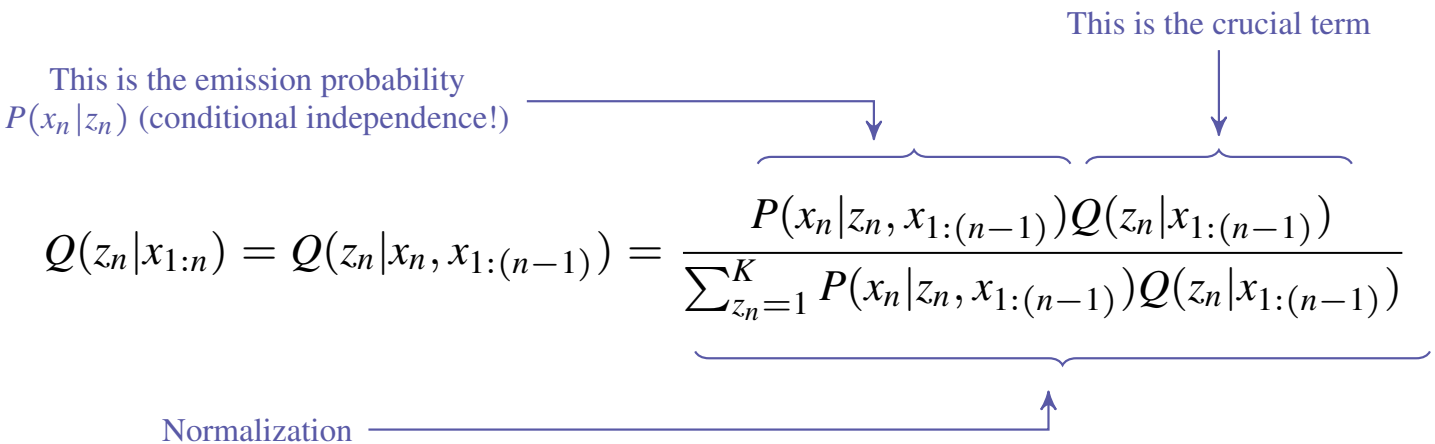
Filtering problem

Our task is to estimate the probabilities $Q(z_n|x_{1:n})$. Since the sequence has length n and each Z_i can take K possible values, this is a $N \times K$-matrix $\hat{Q}$, with entries

$$\hat{Q}_{nk} := Q(Z_n = k|x_{1:n}) \ .$$

Decomposition using Bayes' equation

We can use Bayes' equation (recall: $P(a|b) = \frac{P(b|a)P(a)}{P(b)}$) to write:



Reduction to previous step

The crucial idea is that we can use the results computed for step $n-1$ to compute those for step n:

$$Q(Z_n = k|x_{1:(n-1)}) = \sum_{l=1}^{K} \underbrace{Q(Z_n = k|Z_{n-1} = l)}_{= q_{lk} \text{ (transition matrix)}} \underbrace{Q(Z_{n-1} = l|x_{1:(n-1)})}_{= \hat{Q}_{(n-1)l}}$$

Summary

In short, we can compute the numerator in the Bayes equation as

$$a_{nk} := P(x_n|z_n) \sum_{l=1}^{K} q_{lk} \hat{Q}_{(n-1)l} \ .$$

The normalization term is

$$\sum_{z_n=1}^{K} \left(P(x_n|z_n) \sum_{l=1}^{K} q_{lk} \hat{Q}_{(n-1)l} \right) = \sum_{j=1}^{K} a_{nj} \ .$$

Solution to the filtering problem: The forward algorithm

Given is a sequence $(x_1, \ldots, x_N)$.

For $n = 1, \ldots, N$, compute

$$a_{nk} := P(x_n|z_n) \sum_{l=1}^{K} q_{lk} \hat{Q}_{(n-1)l} \ ,$$

and

$$\hat{Q}_{nk} = \frac{a_{nk}}{\sum_{j=1}^{K} a_{nj}} \ .$$

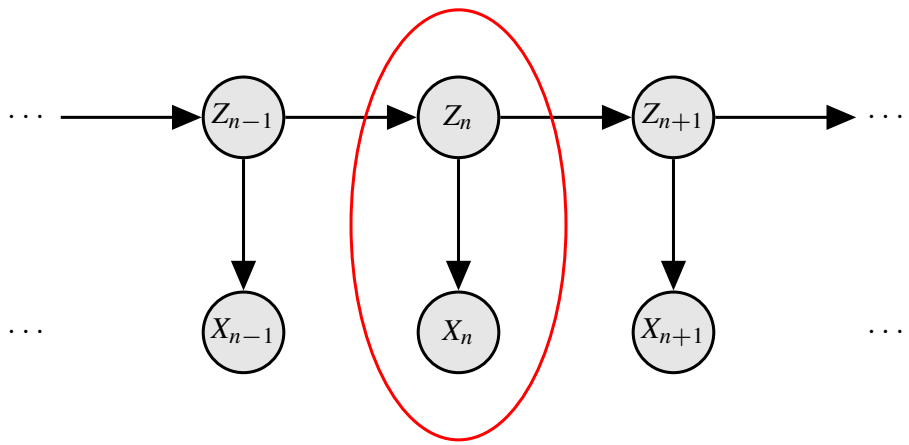
This method is called the **forward algorithm**.

Parametric emission model

We usually define the emission probabilities $P(x_n|z_n)$ using a parametric model $P(x|\theta)$ (e.g. a multinomial or Gaussian model). Then

$$P(x_n|Z_n = k) := P(x_n|\theta_k) \ ,$$

i.e. the emission distribution of each state k is defined by a parameter value θ_k.



Relation to mixture models

If we just consider a *single* pair (Z_n, X_n), this defines a finite mixture with K clusters:

$$\pi(x_n) = \sum_{k=1}^{K} c_k P(x_n|\theta_k) = \sum_{k=1}^{K} Q(Z_n = k) P(x_n|\theta_k)$$

Recall: EM for mixtures

E-step	M-step
Soft assignments $\mathbb{E}[M_{ik}] = \Pr(m_i = k)$	cluster weights c_k component parameters θ_k

HMM case

- For mixtures, $\Pr\{m_i = k\} = c_k$. In HMMs, the analogous probability $\Pr\{Z_n = k\}$ is determined by the transition probabilities.
- The analogue of the soft assignments a_{ik} computed for mixtures are state probabilities

$$b_{nk} = Q(Z_n = k|\theta, x_{1:n}) \ .$$

- Additionally, we have to estimate the transition matrix $\mathbf{q}$ of the Markov chain.

EM for HMMs

E-step	M-step
Transition probabilities q_{kj} State probabilities b_{nk}	component parameters θ_k

M-step

The M-step works exactly as for mixture models. E.g. for Gaussian emission distributions with parameters μ_k and σ_k^2,

State probabilities substituted for assignment probabilities

$$\mu_k = \frac{\sum_{n=1}^N b_{nk} x_n}{\sum_{n=1}^N b_{nk}} \qquad \text{and} \qquad \sigma_k^2 = \frac{\sum_{n=1}^N b_{nk} (x_n - \mu_k)^2}{\sum_{n=1}^N b_{nk}}$$

E-step

- Computing the state probabilities is a filtering problem:

  $$b_{nk}^{\text{new}} = Q(Z_n = k | \theta^{\text{old}}, x_{1:n}) \ .$$

  The forward algorithm assumes the emission probabilities are known, so we use the emission parameters θ^{old} computed during the previous M-step.

- Estimating the transition probabilities is essentially a filtering-type problem for *pairs* of states and can also be solved recursively, but we will skip the details since the equations are quite lengthy.

Problem

Given speech in form of a sound signal, determine the words that have been spoken.

Method

- Words are broken down into small sound units (called *phonemes*). The states in the HMM represent phonemes.
- The incoming sound signal is transformed into a sequence of vectors (feature extraction). Each vector x_n is indexed by a time step n.
- The sequence $x_{1:N}$ of feature vectors is the observed data in the HMM.

Phoneme

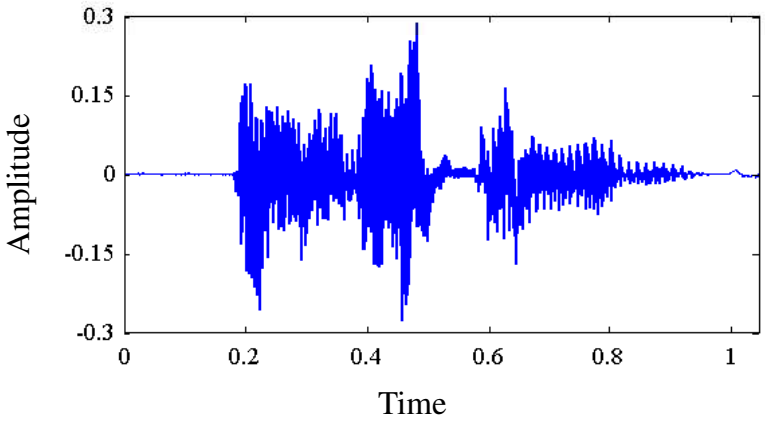
A **phoneme** is defined as the smallest unit of sound in a language that distinguishes between distinct meanings. English uses about 50 phonemes.

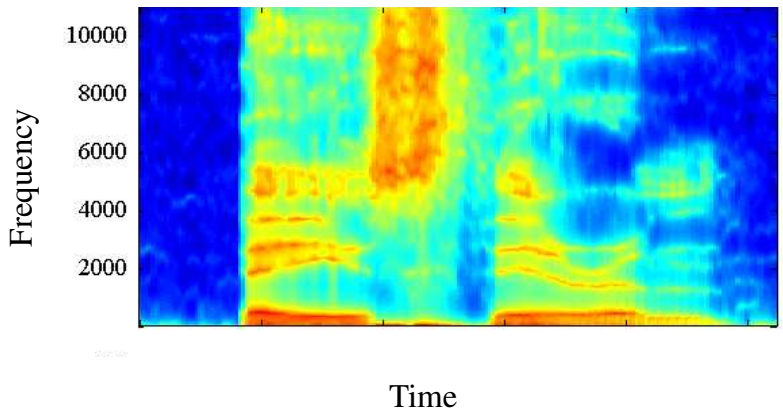
Example

Zero	Z IH R OW	Six	S IH K S
One	W AH N	Seven	S EH V AX N
Two	T UW	Eight	EY T
Three	TH R IY	Nine	N AY N
Four	F OW R	Oh	OW
Five	F AY V		

Subphonemes

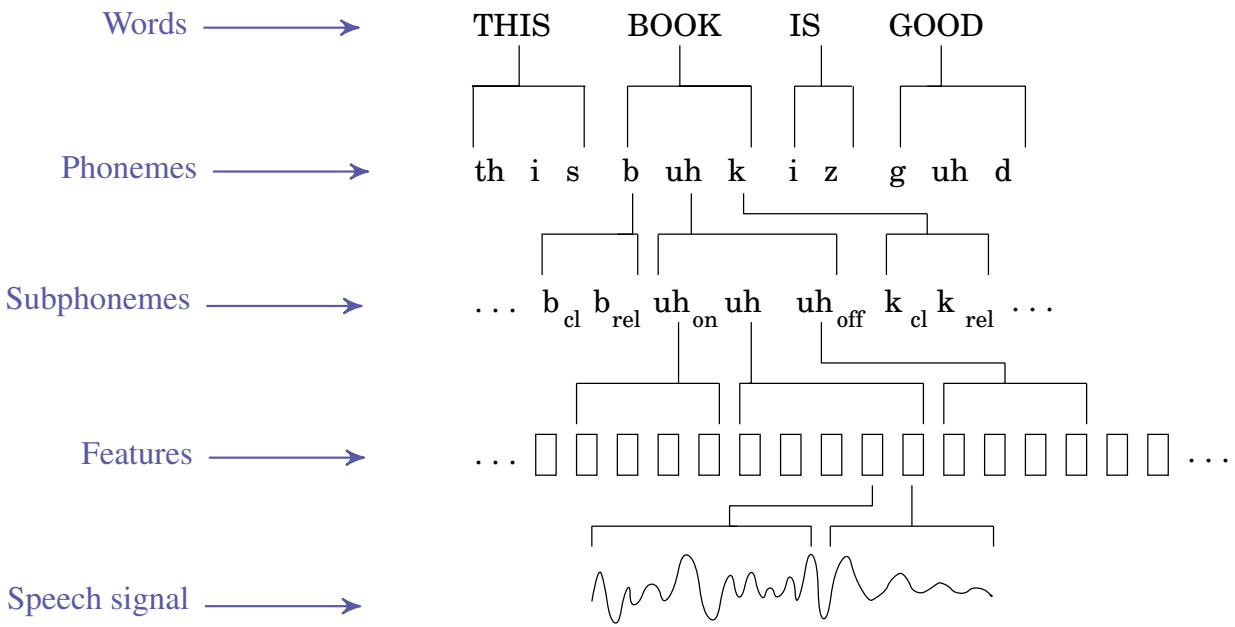
Phonemes can be further broken down into subphonemes. The standard in speech processing is to represent a phoneme by three subphonemes ("triphons").





Feature extraction

- A speech signal is measured as amplitude over time.
- The signal is typically transformed into various types of features, including (windowed) Fourier- or cosine-transforms and so-called "cepstral features".
- Each of these transforms is a scalar function of time. All function values for the different transforms at time t are collected in a vector, which is the feature vector (at time t).



HMM speech recognition

- Training: The HMM parameters (emission parameters and transition probabilities) are estimated from data, often using both supervised and unsupervised techniques.
- Recognition: Given a language signal (= observation sequence $x_{1:N}$, estimate the corresponding sequence of subphonemes (= states $z_{1:N}$). This is a decoding problem.

Factory model

Training requires a lot of data; software is typically shipped with a model trained on a large corpus (i.e. the HMM parameters are set to "factory settings").

The adaptation problem

- The factory model represents an average speaker. Recognition rates can be improved drastically by adapting to the specific speaker using the software.
- Before using the software, the user is presented with a few sentences and asked to read them out, which provides labelled training data.

Speaker adaptation

- Transition probabilities are properties of the language. Differences between speakers (pronounciation) are reflected by the emission parameters θ_k.
- Emission probabilities in speech are typically multi-dimensional Gaussians, so we have to adapt means and covariance matrices.
- The arguably most widely used method is **maximum likelihood linear regression** (**MLLR**), which uses a regression technique to make small changes to the covariance matrices.

More details on HMMs

If you feel enthusiastic, the following books provide more background:

- David Barber's "Bayesian reasoning and machine learning" (available online).
- Chris Bishop's "Pattern recognition and machine learning".
- Many books on speech, e.g. Rabiner's classic "Fundamentals of speech recognition".

HTK

If you would like to try out speech recognition software, have a look at the HTK (**H**MM **T**ool**k**it) package, which is the de-facto standard in speech research. HTK implements both HMMs for recognition and routines for feature extraction.

BAYESIAN MIXTURE MODELS

The concept of a Bayesian mixture will come up a few times in this class, so we will briefly review it on the next few slides.

Idea

- Recall that the defining idea of a Bayesian model is to treat the model parameters as random variables.
- That requires specifying a distribution for those parameters, known as the *prior distribution*.
- Instead of asking for an value of the parameters estimated from data (a *point estimate*), we determine their conditional distribution given the data. This distribution is called the *posterior distribution*.
- A Bayesian mixture model is a finite mixture model whose model parameters are treated as random.

Inference: Sampling

These models are examples of models in which the exact posterior is intractable. Inference uses Markov chain Monte Carlo sampling, which will be our main topic for the last two lectures.

Recall: Finite mixture models

$$\pi(x) = \sum_{k=1}^{K} c_k p(x|\theta_k)$$

The model parameters are the component parameters θ_k and the weights c_k. For a Bayesian version of the model, we hence have to generate random component parameters $\Theta_1, \ldots, \Theta_K$ and random weights $C_1, \ldots, C_K$.

More precisely

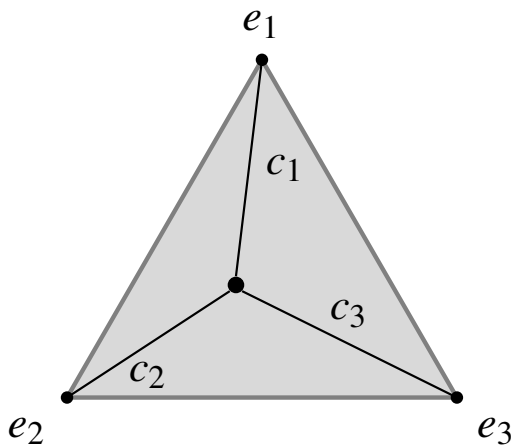
- The mixture components $p(x|\theta)$ are an exponential family model (as discussed in SML).
- Which prior we choose for Θ depends on the choice of p.
- The prior of the vector $(C_1, \ldots, C_K)$ is always chose as a *Dirichlet distribution*.

The Dirichlet distribution

The Dirichlet distribution, for some specified number $K \in \mathbb{N}$, generates a random vector $(C_1, \ldots, C_K)$ with the property that $C_k > 0$ for all $k = 1, \ldots, K$ and $\sum_{k \leq K} C_k = 1$. In other words, $(C_1, \ldots, C_K)$ is a (randomly generated) probability distribution on K categories.

Recall: Probability simplex

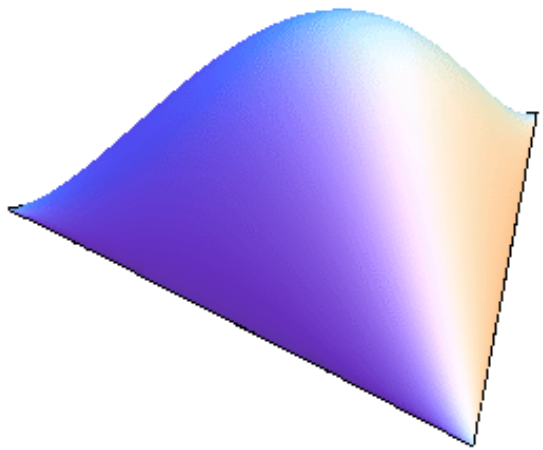
The set of all probability distributions on K events is the *simplex* $\Delta_K := \{(c_1, \ldots, c_k) \in \mathbb{R}^K \mid c_k \geq 0 \text{ and } \sum_k c_K = 1\}$.



Dirichlet distribution

The **Dirichlet distribution** is the distribution on Δ_K with density

$$q_{\text{Dirichlet}}(c_{1:K}|\alpha, g_{1:K}) := \frac{1}{K(\alpha, g_{1:K})} \exp\Big(\sum_{k=1}^{K}(\alpha g_k - 1)\log(c_k)\Big)$$



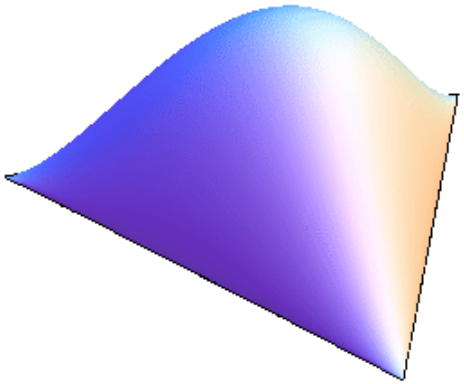
Parameters:

- $g_{1:K} \in \Delta_K$: Mean parameter, i.e. $\mathbb{E}[c_{1:K}] = g_{1:K}$.
- $\alpha \in \mathbb{R}_+$: Concentration.
  Larger $\alpha \to$ sharper concentration around $g_{1:K}$.

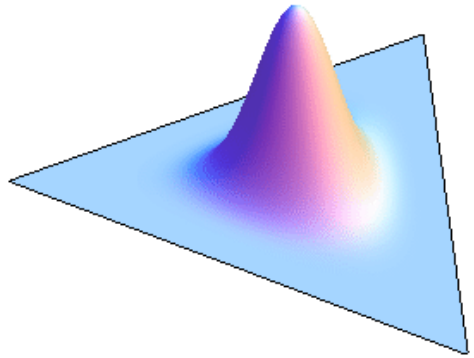
The Dirichlet Distribution

In all plots, $g_{1:K} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. Light colors = large density values.

Density plots

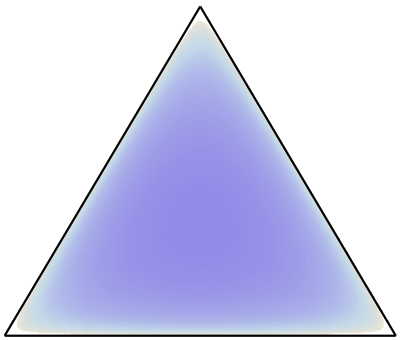


$\alpha = 1.8$

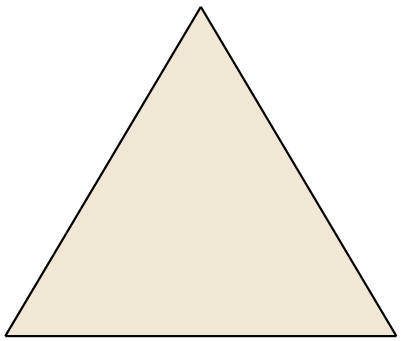


$\alpha = 10$

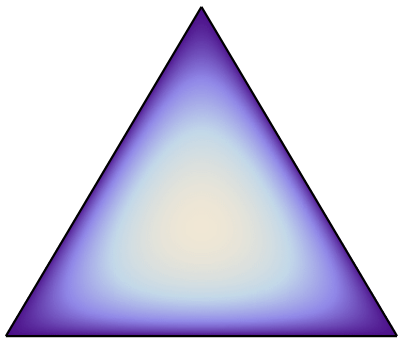
As heat maps



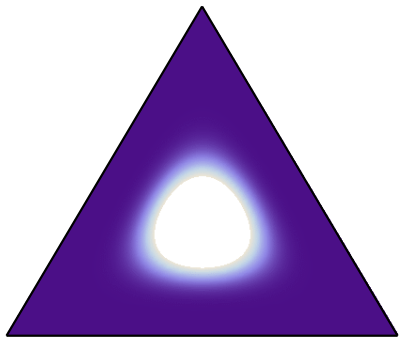
$\alpha = 0.8$
Large density values
at extreme points



$\alpha = 1$
Uniform distribution
on Δ_K



$\alpha = 1.8$
Density peaks
around its mean



$\alpha = 10$
Peak sharpens
with increasing α

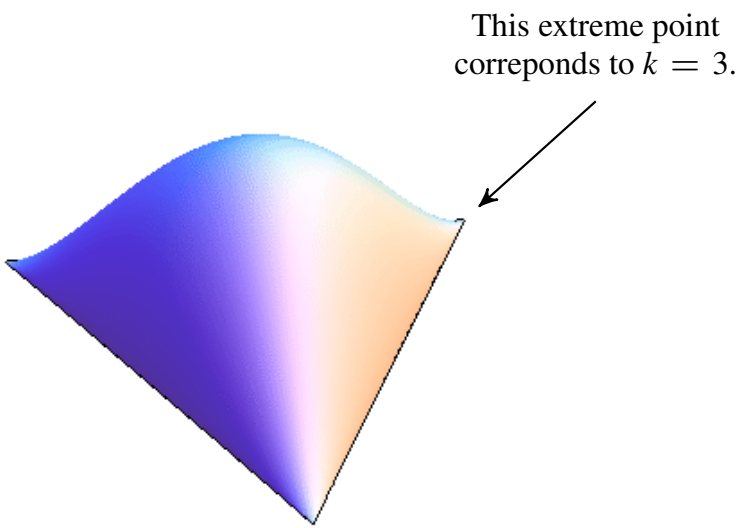
- Suppose we observe data from a multinomial distribution on K categories. The multinomial is parametrized by a finite probability distribution on K events. We generate this distribution using a Dirichlet (with hyperparameters α and $g_{1:K}$). What is the posterior?
- If we observe h_k counts in category k, the posterior is

$$\Pi(c_{1:K}|h_1,\ldots,h_k) = q_{\text{Dirichlet}}(c_{1:K}|\alpha+n,(\alpha g_1+h_1,\ldots,\alpha g_K+h_K))$$

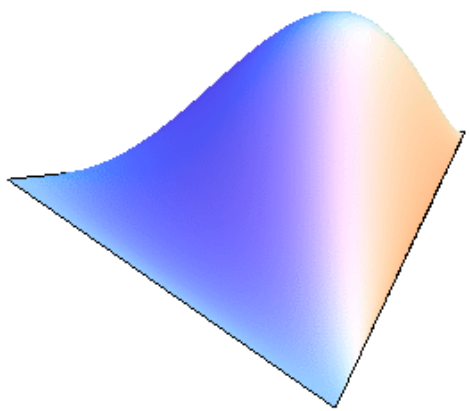
  where $n = \sum_k h_k$ is the total number of observations.
- The posterior distribution of a Dirichlet prior combined with a multinomial observation model is again a Dirichlet distribution.

Illustration: One observation

Suppose $K = 3$ and we obtain a single observation in category 3.



Prior: Mean at the center.



Posterior: Shifted mean, increased concentration.

Definition

A model of the form

$$\pi(x) = \sum_{k=1}^{K} C_k p(x|\Theta_k)$$

is called a **Bayesian mixture model** if $p(x|\theta)$ is an exponential family model and

- $\Theta_1, \ldots, \Theta_K \sim_{\text{iid}} q$, where q is a prior we have chosen for Θ.
- $(C_1, \ldots, C_K)$ is sampled from a K-dimensional Dirichlet distribution.

Posterior distribution

The posterior of a BMM under observations $x_1, \ldots, x_n$ is (up to normalization):

$$\Pi(c_{1:K}, \theta_{1:K} | x_{1:n}) \propto \prod_{i=1}^{n} \Big(\sum_{k=1}^{K} c_k p(x_i | \theta_k) \Big) \Big(\prod_{k=1}^{K} q(\theta_k) \Big) q_{\text{Dirichlet}}(c_{1:K})$$

The posterior is analytically intractable

- Thanks to conjugacy, we *can* evaluate each term of the posterior.
- However: Due to the $\prod_{i=1}^{n} \Big(\sum_{k=1}^{K} \ldots \Big)$ bit, the posterior has K^n terms!
- Even for 10 clusters and 100 observations, that is impossible to compute.

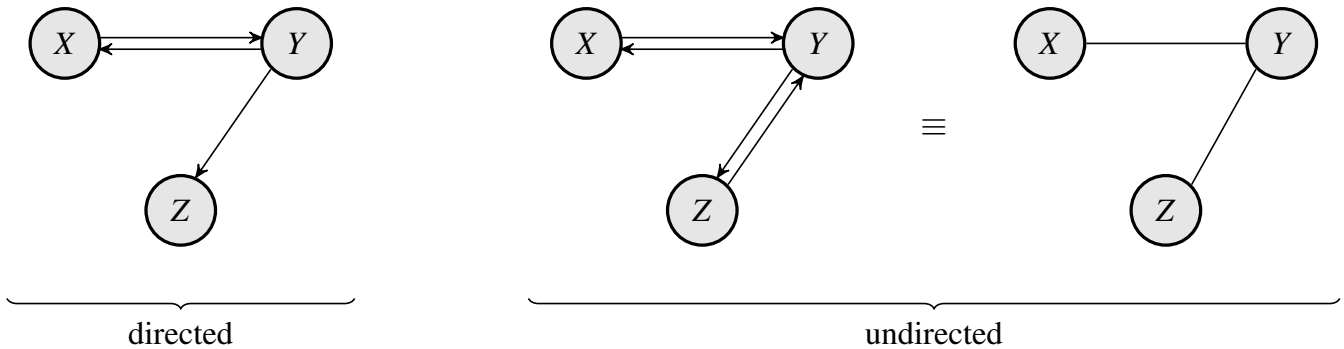
Solution

The posterior can be sampled with a very simple MCMC sampler (which looks strikingly similar to an EM algorithm, and will be discussed later).

MARKOV RANDOM FIELDS

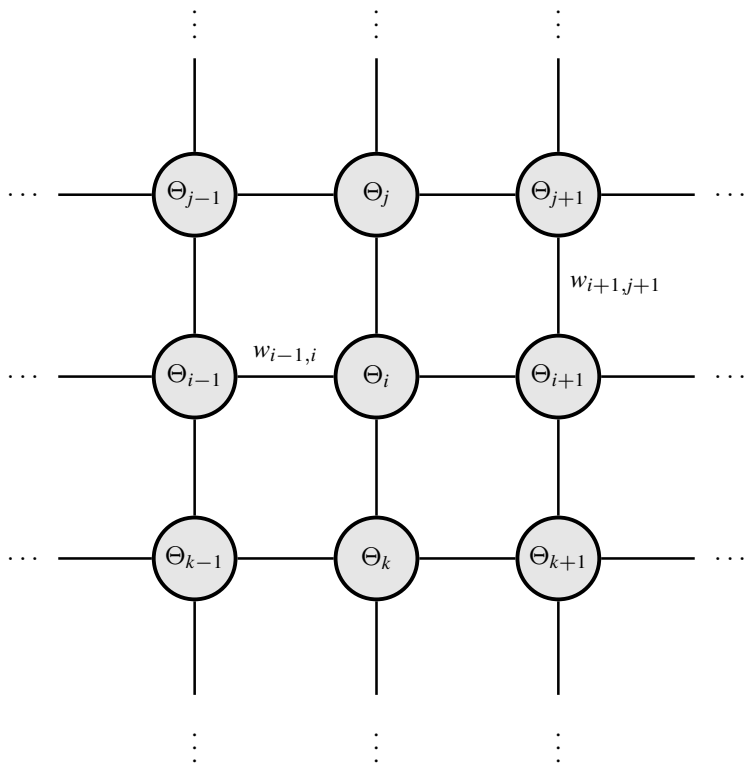
Undirected Graphical Model

- A graphical model is **undirected** when its dependency graph is undirected; equivalently, if each edge in the graph is either absent, or present in both directions.



- An undirected graphical model is more commonly known as a **Markov random field**.
- Markov random fields are special cases of (directed) graphical models, but have distinct properties. We treat them separately.
- We will consider the undirected case first.

We start with an undirected graph:



A random variable Θ_i is associated with each vertex. Two random variables interact if they are neighbors in the graph.

- We define a **neighborhood graph**, which is a weighted, undirected graph:

$$\mathcal{N} = (V_{\mathcal{N}}, W_{\mathcal{N}})$$

(vertex set: $V_{\mathcal{N}}$; set of edge weights: $W_{\mathcal{N}}$)

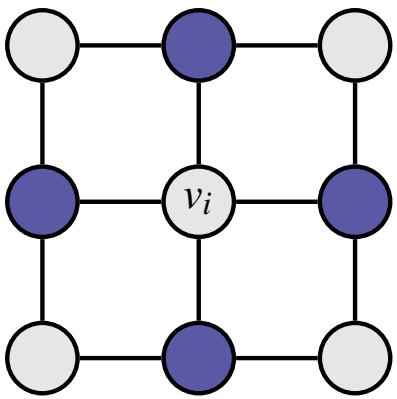
  The vertices $v_i \in V_{\mathcal{N}}$ are often referred to as **sites**.
- The edge weights are scalars $w_{ij} \in \mathbb{R}$. Since the graph is undirected, the weights are symmetric ($w_{ij} = w_{ji}$).
- An edge weight $w_{ij} = 0$ means "no edge between v_i and v_j".

Neighborhoods

The set of all neighbors of v_j in the graph,

$$\partial(i) := \{j \mid w_{ij} \neq 0\}$$

is called the **neighborhood** of v_i.



purple = $\partial(i)$

Given a neighborhood graph $\mathcal{N}$, associate with each site $v_i \in V_{\mathcal{N}}$ a RV Θ_i.

The Markov property

We say that the joint distribution P of $(\Theta_1, \ldots, \Theta_n)$ satisfies the **Markov property** with respect to $\mathcal{N}$ if

$$\mathcal{L}(\Theta_i | \Theta_j,\ j \neq i) = \mathcal{L}(\Theta_i | \Theta_j,\ j \in \partial(i)) \ .$$

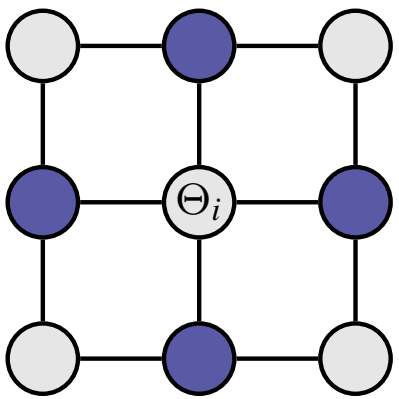
The set $\{\Theta_j, j \in \partial(i)\}$ of random variables indexed by neighbors of v_i is called the **Markov blanket** of Θ_i.

In words

The Markov property says that each Θ_i is conditionally independent of the remaining variables given its Markov blanket.

Definition

A distribution $\mathcal{L}(\Theta_1, \ldots, \Theta_n)$ which satisfies the Markov property for a given graph $\mathcal{N}$ is called a **Markov random field**.



Markov blanket of Θ_i

Probabilities and energies

A (strictly positive) density $p(x)$ can always be written in the form

$$p(x) = \frac{1}{Z} \exp(-H(x)) \qquad \text{where} \qquad H : \mathbf{X} \to \mathbb{R}_+$$

and Z is a normalization constant.

The function H is called an **energy function**, or **cost function**, or a **potential**.

MRF energy

In particular, we can write a MRF density for RVs $\Theta_{1:n}$ as

$$p(\theta_1, \ldots, \theta_n) = \frac{1}{Z} \exp(-H(\theta_1, \ldots, \theta_n))$$

Definition

Suppose $\mathcal{N} = (V_{\mathcal{N}}, W_{\mathcal{N}})$ a neighborhood graph with n vertices and $\beta > 0$ a constant. Then

$$p(\theta_{1:n}) := \frac{1}{Z(\beta, W_{\mathcal{N}})} \exp\Big(\beta \sum_{i,j} w_{ij} \mathbb{I}\{\theta_i = \theta_j\}\Big)$$

defines a joint distribution of n random variables $\Theta_1, \ldots, \Theta_n$. This distribution is called the **Potts model**.

Interpretation

- If $w_{ij} > 0$: The overall probability *increases* if $\Theta_i = \Theta_j$.
- If $w_{ij} < 0$: The overall probability *decreases* if $\Theta_i = \Theta_j$.
- If $w_{ij} = 0$: No interaction between Θ_i and Θ_j.

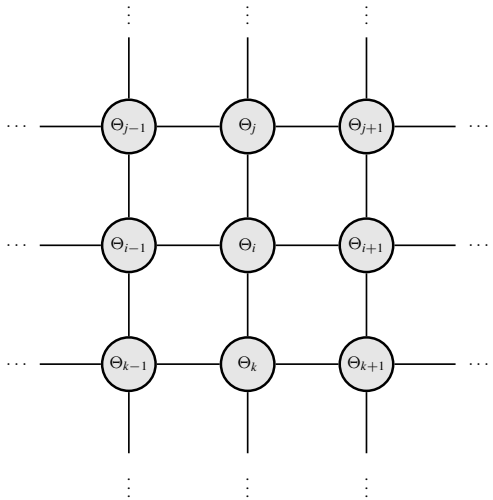
Positive weights encourage *smoothness*.

Ising model

The simplest choice is $w_{ij} = 1$ if (i, j) is an edge.

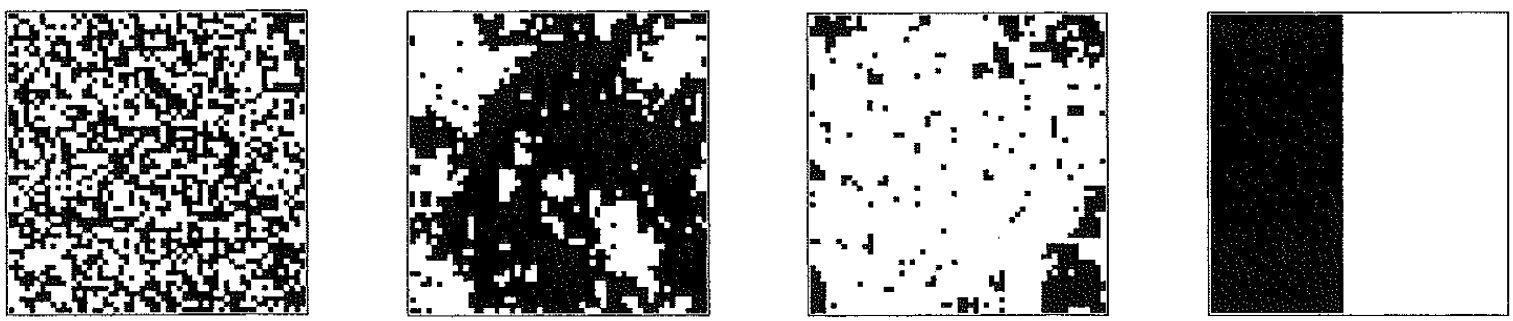
$$p(\theta_{1:n}) = \frac{1}{Z(\beta)} \exp\Big(\sum_{(i,j) \text{ is an edge}} \beta \mathbb{I}\{\theta_i = \theta_j\} \Big)$$

If $\mathcal{N}$ is a d-dim. grid, this model is called the **Ising model**.



Example

Samples from an Ising model on a 56×56 grid graph.

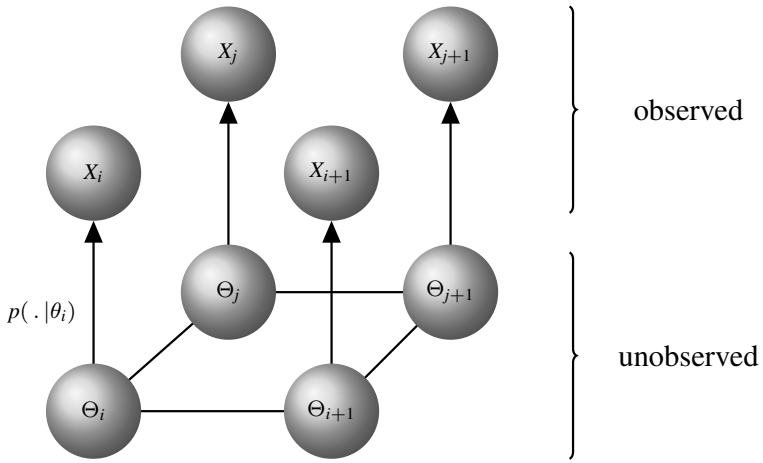


Increasing $\beta \longrightarrow$

We consider a spatial problem with observations X_i. Each i is a location on a grid.

Spatial model

Suppose we model each X_i by a distribution $\mathcal{L}(X|\Theta_i)$, i.e. each location i has its own parameter variable Θ_i. This model is Bayesian (the parameter is a random variable). We use an MRF as prior distribution.



We can think of $\mathcal{L}(X|\Theta_i)$ as an emission probability, similar to an HMM.

Spatial smoothing

- We can define the joint distribution $(\Theta_1, \ldots, \Theta_n)$ as a MRF on the grid graph.
- For positive weights, the MRF will encourage the model to explain neighbors X_i and X_j by the same parameter value. $\rightarrow$ Spatial smoothing.

Example: Segmentation of Noisy Images

Mixture model

- A BMM can be used for image segmentation.
- The BMM prior on the component parameters is a natural conjugate prior $q(\theta)$.
- In the spatial setting, we index the parameter of each X_i separately as θ_i. For K mixture components, $\theta_{1:n}$ contains only K different values.
- The joint BMM prior on $\theta_{1:n}$ is

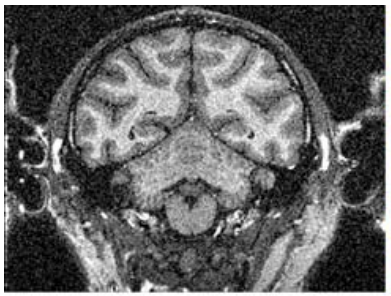
$$q_{\text{BMM}}(\theta_{1:n}) = \prod_{i=1}^{n} q(\theta_i) \ .$$

Smoothing term

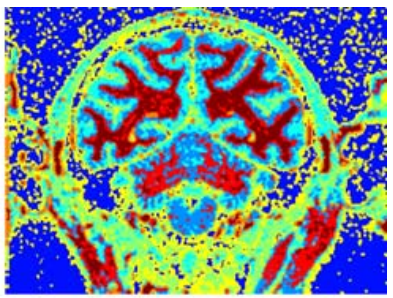
We multiply the BMM prior $q_{\text{BMM}}(\theta)$ with an MRF prior

$$q_{\text{MRF}}(\theta_{1:n}) = \frac{1}{Z(\beta)} \exp\Big(\beta \sum_{w_{ij} \neq 0} \mathbb{I}\{\theta_i = \theta_j\}\Big)$$

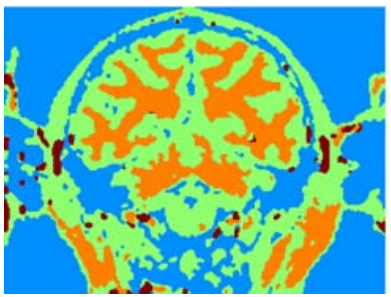
This encourages spatial smoothnes of the segmentation.



Input image.



Segmentation w/o smoothing.



Segmentation with MRF smoothing.

MRFs pose two main computational problems.

Problem 1: Sampling

Generate samples from the joint distribution of $(\Theta_1, \ldots, \Theta_n)$.

Problem 2: Inference

If the MRF is used as a prior, we have to compute or approximate the posterior distribution.

Solution

- MRF distributions on grids are *not* analytically tractable. The only known exception is the Ising model in 1 dimension.
- Both sampling and inference are based on Markov chain sampling algorithms.

SAMPLING ALGORITHMS

In general

- A **sampling algorithm** is an algorithm that outputs samples $X_1, X_2, \ldots$ from a given distribution P or density p.
- Sampling algorithms can for example be used to approximate expectations:

$$\mathbb{E}_p[f(X)] \approx \frac{1}{n} \sum_{i=1}^{n} f(X_i)$$

Inference in Bayesian models

Suppose we work with a Bayesian model whose posterior $\hat{Q}_n := \mathcal{L}(\Theta | X_{1:n})$ cannot be computed analytically.

- We will see that it can still be possible to *sample* from $\hat{Q}_n$.
- Doing so, we obtain samples $\Theta_1, \Theta_2, \ldots$ distributed according to $\hat{Q}_n$.
- This reduces posterior estimation to a density estimation problem (i.e. estimate $\hat{Q}_n$ from $\Theta_1, \Theta_2, \ldots$).

Posterior expectations

If we are only interested in some statistic of the posterior of the form $\mathbb{E}_{\hat{Q}_n}[f(\Theta)]$ (e.g. the posterior mean), we can again approximate by

$$\mathbb{E}_{\hat{Q}_n}[f(\Theta)] \approx \frac{1}{m}\sum_{i=1}^{m} f(\Theta_i) \ .$$

Example: Predictive distribution

The **posterior predictive distribution** is our best guess of what the next data point x_{n+1} looks like, given the posterior under previous observations. In terms of densities:

$$p(x_{n+1}|x_{1:n}) := \int_{\mathbf{T}} p(x_{n+1}|\theta)\hat{Q}_n(d\theta|X_{1:n} = x_{1:n}) \ .$$

This is one of the key quantities of interest in Bayesian statistics.

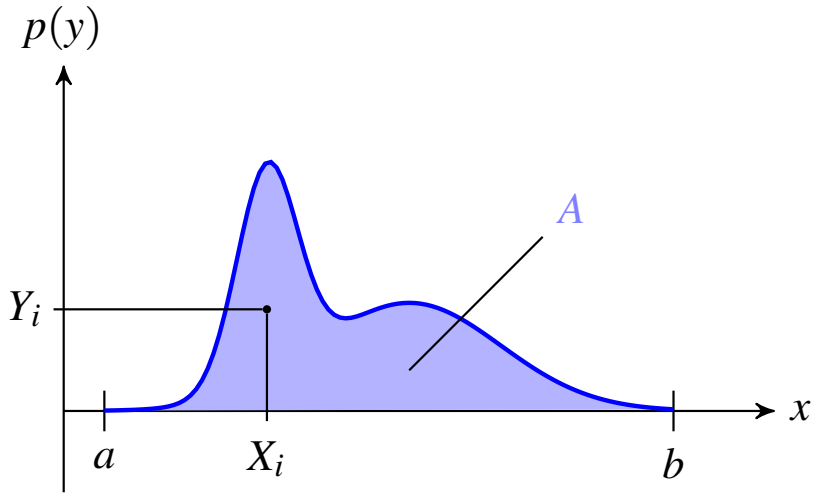
Computation from samples

The predictive is a posterior expectation, and can be approximated as a sample average:

$$p(x_{n+1}|x_{1:n}) = \mathbb{E}_{\hat{Q}_n}[p(x_{n+1}|\Theta)] \approx \frac{1}{m}\sum_{i=1}^{m} p(x_{n+1}|\Theta_i)$$

Basic Sampling: Area Under Curve

Say we are interested in a probability density p on the interval $[a, b]$.



Key observation

Suppose we can define a uniform distribution U_A on the blue area A under the curve. If we sample

$$(X_1, Y_1), (X_2, Y_2), \ldots \sim_{\text{iid}} U_A$$

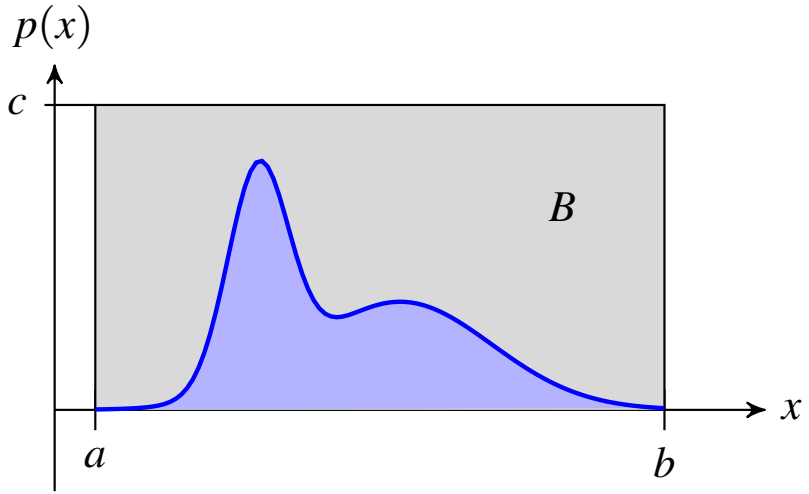
and discard the vertical coordinates Y_i, **the X_i are distributed according to p**,

$$X_1, X_2, \ldots \sim_{\text{iid}} p \ .$$

Problem: Defining a uniform distribution is easy on a rectangular area, but difficult on an arbitrarily shaped one.

Solution: Rejection sampling

We can enclose p in box, and sample uniformly from the box B.



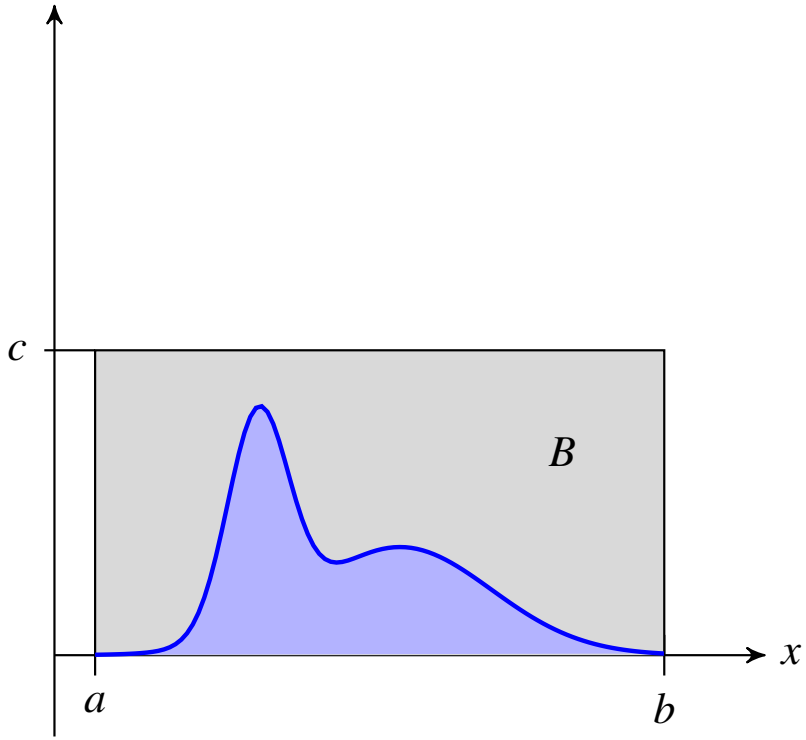
- We can sample (X_i, Y_i) uniformly on B by sampling

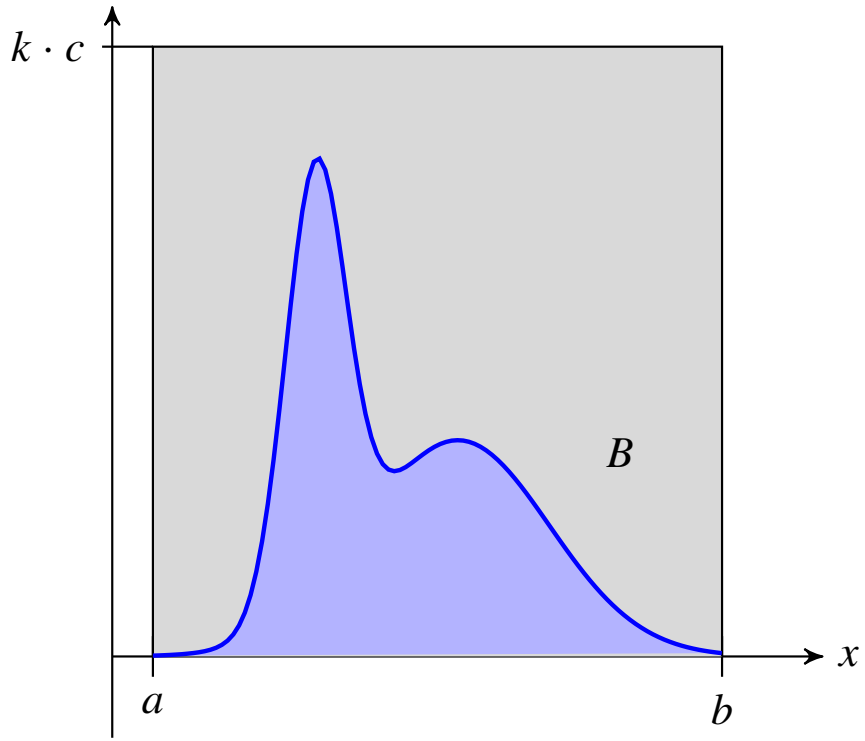
$$X_i \sim \text{Uniform}[a, b] \qquad \text{and} \qquad Y_i \sim \text{Uniform}[0, c] \ .$$

- If $(X_i, Y_i) \in A$, keep the sample.
  That is: If $Y_i \leq p(X_i)$.
- Otherwise: Discard it ("reject" it).

Result: The remaining (non-rejected) samples are uniformly distributed on A.

This strategy still works if we scale the vertically by some constant $k > 0$.





We simply draw $Y_i \sim \text{Uniform}[0, kc]$ instead of $Y_i \sim \text{Uniform}[0, c]$.

Consequence

For sampling, it is sufficient if p is known only up to normalization
(only the shape of p is known).

Sampling methods usually assume that we can evaluate the target distribution p up to a constant. That is:

$$p(x) \quad = \quad \frac{1}{\tilde{Z}} \tilde{p}(x) \ ,$$

and we can compute $\tilde{p}(x)$ for any given x, but we do not know $\tilde{Z}$.

We have to pause for a moment and convince ourselves that there are useful examples where this assumption holds.

Example 1: Simple posterior

For an arbitrary posterior computed with Bayes' theorem, we could write

$$\Pi(\theta|x_{1:n}) = \frac{\prod_{i=1}^{n} p(x_i|\theta)q(\theta)}{\tilde{Z}} \qquad \text{with} \qquad \tilde{Z} = \int_{\mathbf{T}} \prod_{i=1}^{n} p(x_i|\theta)q(\theta)d\theta \ .$$

Provided that we can compute the numerator, we can sample without computing the normalization integral $\tilde{Z}$.

Example 2: Bayesian Mixture Model

Recall that the posterior of the BMM is (up to normalization):

$$\hat{q}_n(c_{1:K}, \theta_{1:K} | x_{1:n}) \propto \prod_{i=1}^{n} \Big(\sum_{k=1}^{K} c_k p(x_i|\theta_k) \Big) \Big(\prod_{k=1}^{K} q(\theta_k|\lambda, y) \Big) q_{\text{Dirichlet}}(c_{1:K})$$

We already know that we can discard the normalization constant, but can we evaluate the non-normalized posterior $\tilde{q}_n$?

- The problem with computing $\tilde{q}_n$ (as a function of unknowns) is that the term $\prod_{i=1}^{n} \left(\sum_{k=1}^{K} \dots \right)$ blows up into K^n individual terms.
- If we *evaluate* $\tilde{q}_n$ for specific values of c, x and θ, $\sum_{k=1}^{K} c_k p(x_i|\theta_k)$ collapses to a single number for each x_i, and we just have to multiply those n numbers.

So: Computing $\tilde{q}_n$ as a formula in terms of unknowns is difficult; evaluating it for specific values of the arguments is easy.

Example 3: Markov random field

In a MRF, the normalization function is the real problem.

For example, recall the Ising model:

$$p(\theta_{1:n}) = \frac{1}{Z(\beta)} \exp\Big(\sum_{(i,j) \text{ is an edge}} \beta \mathbb{I}\{\theta_i = \theta_j\} \Big)$$

The normalization function is

$$Z(\beta) = \sum_{\theta_{1:n} \in \{0,1\}^n} \exp\Big(\sum_{(i,j) \text{ is an edge}} \beta \mathbb{I}\{\theta_i = \theta_j\} \Big)$$

and hence a sum over 2^n terms. The general Potts model is even more difficult.

On the other hand, evaluating

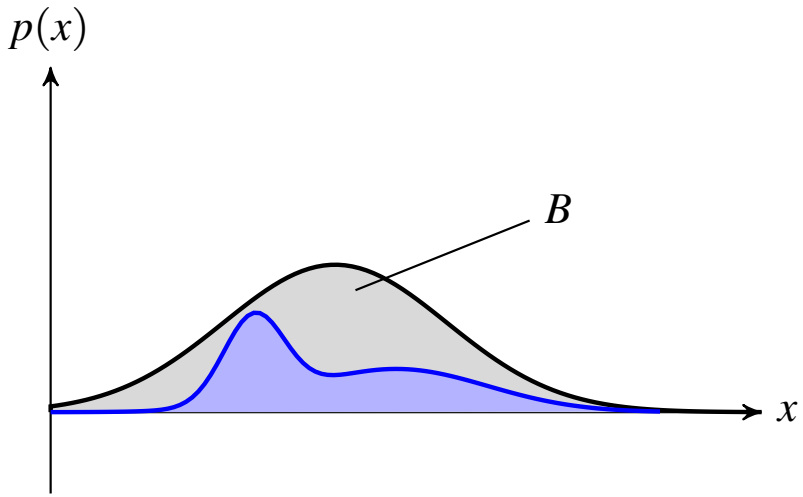
$$\tilde{p}(\theta_{1:n}) = \exp\Big(\sum_{(i,j) \text{ is an edge}} \beta \mathbb{I}\{\theta_i = \theta_j\} \Big)$$

for a given configuration $\theta_{1:n}$ is straightforward.

If we are not on the interval, sampling uniformly from an enclosing box is not possible (since there is no uniform distribution on all of $\mathbb{R}$ or $\mathbb{R}^d$).

Solution: Proposal density

Instead of a box, we use *another distribution* r to enclose p:

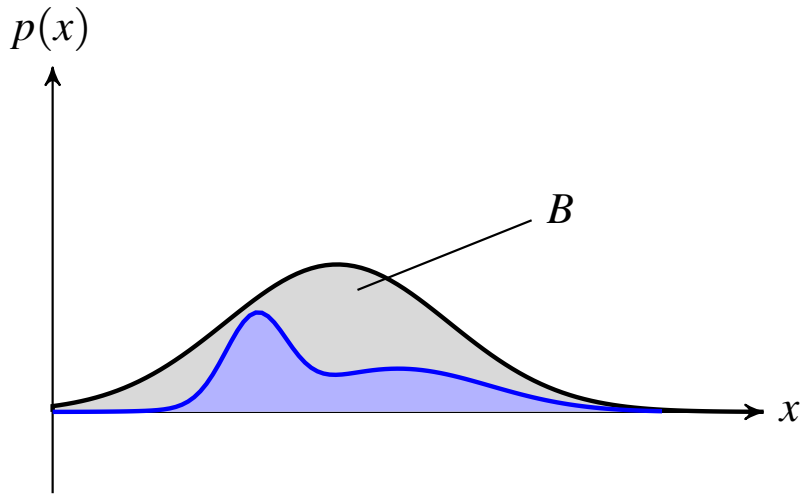


To generate B under r, we apply similar logic as before backwards:

- Sample $X_i \sim r$.
- Sample $Y_i | X_i \sim \text{Uniform}[0, r(X_i)]$.

r is always a simple distribution which we can sample and evaluate.

Rejection Sampling on $\mathbb{R}^d$



- Choose a simple distribution r from which we know how to sample.
- Scale $\tilde{p}$ such that $\tilde{p}(x) < r(x)$ everywhere.
- Sampling: For $i = 1, 2, \ldots,$:
  1. Sample $X_i \sim r$.
  2. Sample $Y_i | X_i \sim \text{Uniform}[0, r(X_i)]$.
  3. If $Y_i < \tilde{p}(X_i)$, keep X_i.
  4. Else, discard X_i and start again at (1).
- The surviving samples $X_1, X_2, \ldots$ are distributed according to p.

The rejection step can be interpreted in terms of probabilities and densities.

Factorization

We factorize the target distribution or density p as

$$p(x) \quad = \quad r(x) \cdot A(x)$$

r: distribution from which we know how to sample

A: probability function we can evaluate *once a specific value of x is given*

Sampling from the factorization

$$X \quad = \quad X' \cdot Z$$

where $X' \sim r$ and $Z|X' \sim \text{Bernoulli}(A(X'))$

Sampling Bernoulli variables with uniform variables

$$Z|X' \sim \text{Bernoulli}(A(X')) \quad \Leftrightarrow \quad Z = \mathbb{I}\{U < A(X')\} \quad \text{where} \quad U \sim \text{Uniform}[0,1] \ .$$

If we draw proposal samples X_i i.i.d. from r, the resulting sequence of accepted samples produced by rejection sampling is again i.i.d. with distribution p. Hence:

Rejection samplers produce i.i.d. sequences of samples.

Important consequence

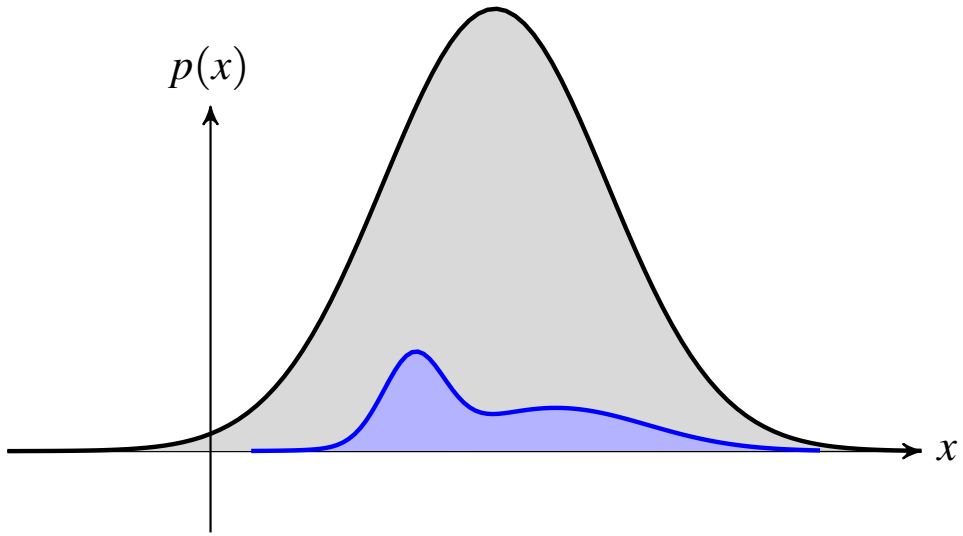
If samples $X_1, X_2, \ldots$ are drawn by a rejection sampler, the sample average

$$\frac{1}{m}\sum_{i=1}^{m} f(X_i)$$

(for some function f) is an unbiased estimate of the expectation $\mathbb{E}_p[f(X)]$.

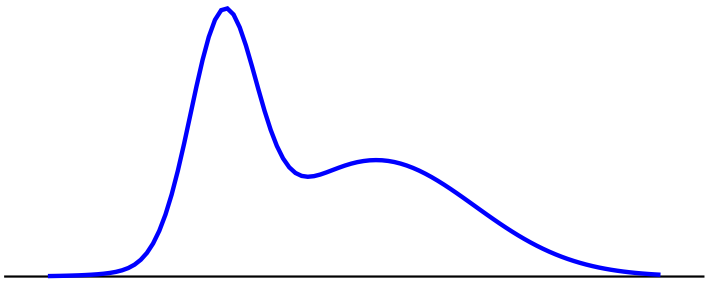
EFFICIENCY

The fraction of accepted samples is the ratio $\frac{|A|}{|B|}$ of the areas under the curves $\tilde{p}$ and r.

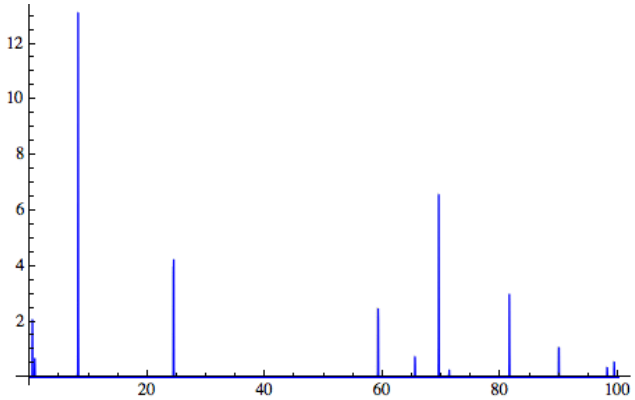


If r is not a reasonably close approximation of p, we will end up rejecting a lot of proposal samples.

AN IMPORTANT BIT OF IMPRECISE INTUITION



Example figures for sampling methods tend to look like this.



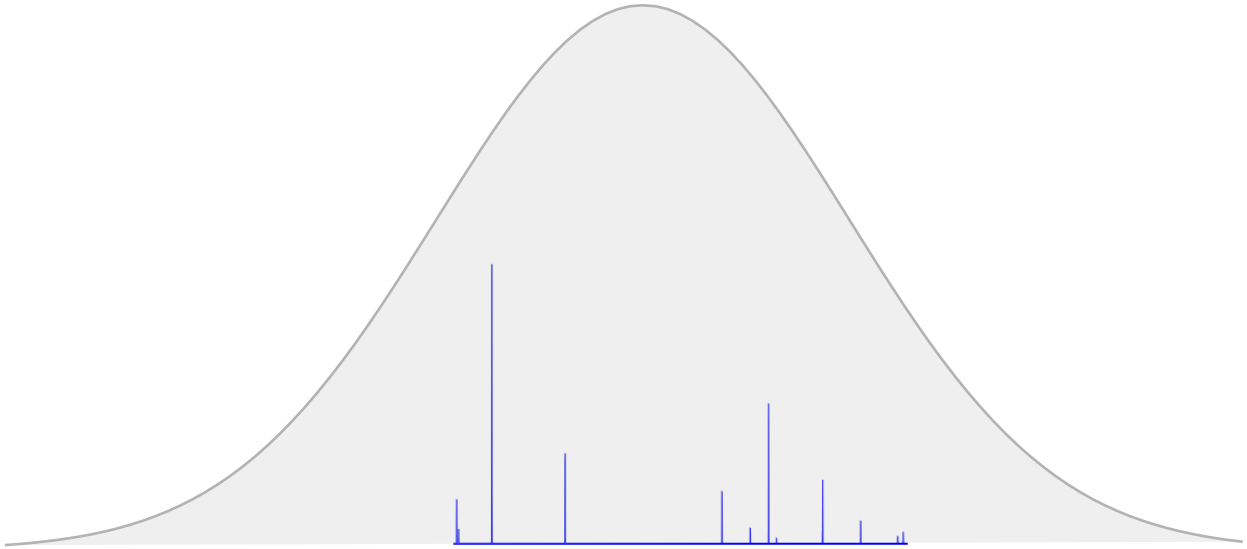
A high-dimensional distribution of correlated RVs will look rather more like this.

Sampling is usually used in multiple dimensions. Reason, roughly speaking:

- Intractable posterior distributions arise when there are several *interacting* random variables. The interactions make the joint distribution complicated.
- In one-dimensional problems (1 RV), we can usually compute the posterior analytically.
- Independent multi-dimensional distributions factorize and reduce to one-dimensional case.

Warning: Avoid sampling if you can solve analytically.

WHY IS NOT EVERY SAMPLER A REJECTION SAMPLER?



We can easily end up in situations where we accept only one in 10^6 (or 10^{10}, or 10^{20},...) proposal samples. Especially in higher dimensions, we have to expect this to be not the exception but the rule.

The rejection problem can be fixed easily if we are only interested in approximating an expectation $\mathbb{E}_p[f(X)]$.

Simple case: We can evaluate p

Suppose p is the target density and q a proposal density. An expectation under p can be rewritten as

$$\mathbb{E}_p[f(X)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_q\left[\frac{f(X)p(X)}{q(X)}\right]$$

Importance sampling

We can sample $X_1, X_2, \ldots$ from q and approximate $\mathbb{E}_p[f(X)]$ as

$$\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^{m} f(X_i)\frac{p(X_i)}{q(X_i)}$$

There is no rejection step; all samples are used.

This method is called **importance sampling**. The coefficients $\frac{p(X_i)}{q(X_i)}$ are called **importance weights**.

General case: We can only evaluate $\tilde{p}$

In the general case,

$$p = \frac{1}{Z_p}\tilde{p} \qquad \text{and} \qquad q = \frac{1}{Z_q}\tilde{q}\ ,$$

and Z_p (and possibly Z_q) are unknown. We can write $\frac{Z_p}{Z_q}$ as

$$\frac{Z_p}{Z_q} = \frac{\int \tilde{p}(x)dx}{Z_q} = \frac{\int \tilde{p}(x)\frac{q(x)}{q(x)}dx}{Z_q} = \int \tilde{p}(x)\frac{q(x)}{Z_q \cdot q(x)}dx = \mathbb{E}_q\left[\frac{\tilde{p}(X)}{\tilde{q}(X)}\right]$$

Approximating the constants

The fraction $\frac{Z_p}{Z_q}$ can be approximated using samples $x_{1:m}$ from q:

$$\frac{Z_p}{Z_q} \quad = \quad \mathbb{E}_q\left[\frac{\tilde{p}(X)}{\tilde{q}(X)}\right] \quad \approx \quad \frac{1}{m}\sum_{i=1}^{m}\frac{\tilde{p}(X_i)}{\tilde{q}(X_i)}$$

Approximating $\mathbb{E}_p[f(X)]$

$$\mathbb{E}_p[f(X)] \quad \approx \quad \frac{1}{m}\sum_{i=1}^{m} f(X_i)\frac{p(X_i)}{q(X_i)} \quad = \quad \frac{1}{m}\sum_{i=1}^{m} f(X_i)\frac{Z_q}{Z_p}\frac{\tilde{p}(X_i)}{\tilde{q}(X_i)} \quad = \quad \sum_{i=1}^{m}\frac{f(X_i)\frac{\tilde{p}(X_i)}{\tilde{q}(X_i)}}{\sum_{j=1}^{m}\frac{\tilde{p}(X_j)}{\tilde{q}(X_j)}}$$

Conditions

- Given are a target distribution p and a proposal distribution q.
- $p = \frac{1}{Z_p}\tilde{p}$ and $q = \frac{1}{Z_q}\tilde{q}$.
- We can evaluate $\tilde{p}$ and $\tilde{q}$, and we can sample q.
- The objective is to compute $\mathbb{E}_p[f(X)]$ for a given function f.

Algorithm

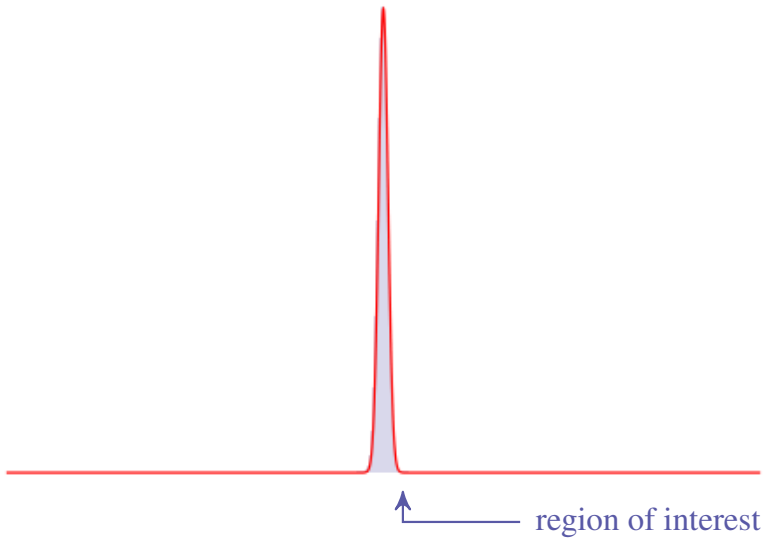
1. Sample $X_1, \ldots, X_m$ from q.
2. Approximate $\mathbb{E}_p[f(X)]$ as

$$\mathbb{E}_p[f(X)] \approx \frac{\sum_{i=1}^{m} f(X_i)\frac{\tilde{p}(X_i)}{\tilde{q}(X_i)}}{\sum_{j=1}^{m}\frac{\tilde{p}(X_j)}{\tilde{q}(X_j)}}$$

MARKOV CHAIN MONTE CARLO

MOTIVATION

Suppose we rejection-sample a distribution like this:



Once we have drawn a sample in the narrow region of interest, we would like to continue drawing samples within the same region. That is only possible if each sample *depends on the location of the previous sample*.

Proposals in rejection sampling are i.i.d. Hence, once we have found the region where p concentrates, we forget about it for the next sample.

Recall: Markov chain

- A sufficiently nice Markov chain (MC) has an invariant distribution P_{inv}.
- Once the MC has converged to P_{inv}, each sample X_i from the chain has marginal distribution P_{inv}.

Markov chain Monte Carlo

We want to sample from a distribution with density p. Suppose we can define a MC with invariant distribution $P_{\text{inv}} \equiv p$. If we sample $X_1, X_2, \ldots$ from the chain, then once it has converged, we obtain samples

$$X_i \sim p \ .$$

This sampling technique is called **Markov chain Monte Carlo** (**MCMC**).

Note: For a Markov chain, X_{i+1} can depend on X_i, so at least in principle, it is possible for an MCMC sampler to "remember" the previous step and remain in a high-probability location.

The Markov chains we discussed so far had a finite state space $\mathbf{X}$. For MCMC, state space now has to be the domain of p, so we often need to work with continuous state spaces.

Continuous Markov chain

A continuous Markov chain is defined by an initial distribution P_{init} and conditional probability $t(y|x)$, the **transition probability** or **transition kernel**.

In the discrete case, $t(y = i|x = j)$ is the entry $\mathbf{p}_{ij}$ of the transition matrix $\mathbf{p}$.

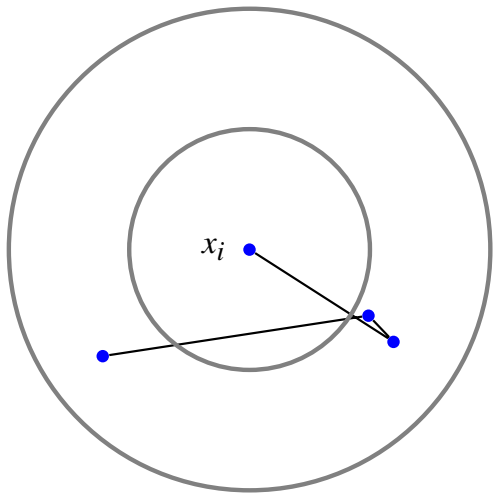
Example: A Markov chain on $\mathbb{R}^2$

We can define a very simple Markov chain by sampling

$$X_{i+1}|X_i = x_i \quad \sim \quad g(\,.\,|x_i, \sigma^2)$$

where $g(x|\mu, \sigma^2)$ is a spherical Gaussian with fixed variance. In other words, the transition distribution is

$$t(x_{i+1}|x_i) := g(x_{i+1}|x_i, \sigma^2)\ .$$



A Gaussian (gray contours) is placed around the current point x_i to sample X_{i+1}.

Recall: Finite case

- The invariant distribution P_{inv} is a distribution on the finite state space $\mathbf{X}$ of the MC (i.e. a vector of length $|\mathbf{X}|$).
- "Invariant" means that, if X_i is distributed according to P_{inv}, and we execute a step $X_{i+1} \sim t(\,.\,|x_i)$ of the chain, then X_{i+1} again has distribution P_{inv}.
- In terms of the transition matrix $\mathbf{p}$:

$$\mathbf{p} \cdot P_{\text{inv}} = P_{\text{inv}}$$

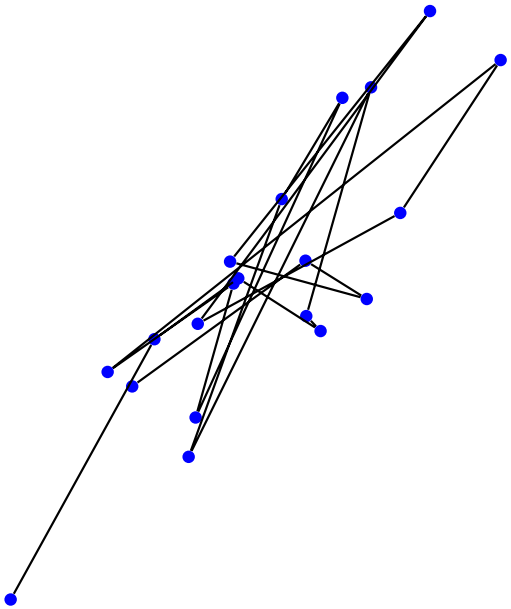
Continuous case

- $\mathbf{X}$ is now uncountable (e.g. $\mathbf{X} = \mathbb{R}^d$).
- The transition matrix $\mathbf{p}$ is substituted by the conditional probability t.
- A distribution P_{inv} with density p_{inv} is invariant if

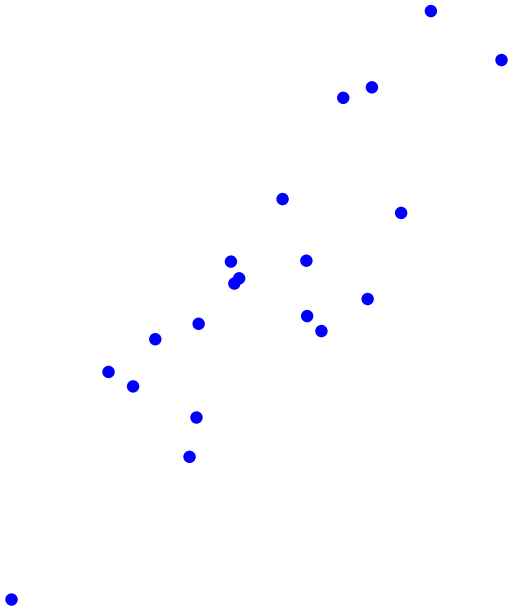
$$\int_{\mathbf{X}} t(y|x) p_{\text{inv}}(x) dx = p_{\text{inv}}(y)$$

This is simply the continuous analogue of the equation $\sum_i \mathbf{p}_{ij} (P_{\text{inv}})_i = (P_{\text{inv}})_j$.

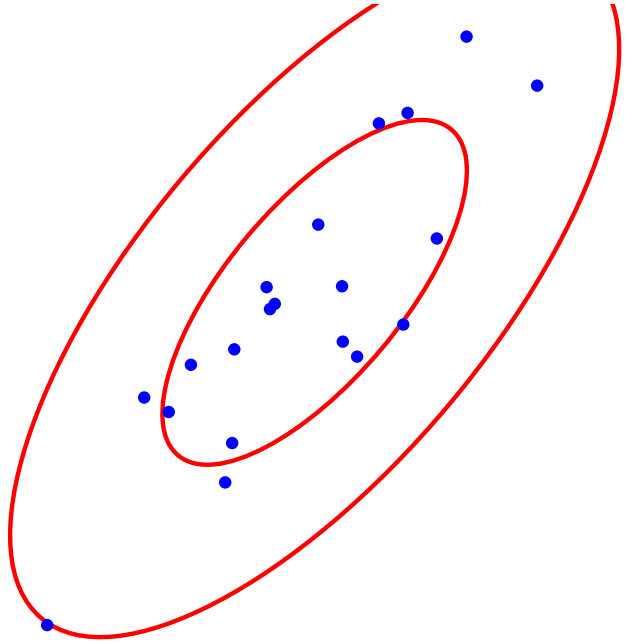
MARKOV CHAIN SAMPLING



We run the Markov chain n for steps. Each step moves from the current location x_i to a new x_{i+1}.



We "forget" the order and regard the locations $x_{1:n}$ as a random set of points.



If p (red contours) is both the invariant and initial distribution, each X_i is distributed as $X_i \sim p$.

Problems we need to solve

1. We have to construct a MC with invariant distribution p.
2. We cannot actually start sampling with $X_1 \sim p$; if we knew how to sample from p, all of this would be pointless.
3. Each point X_i is *marginally* distributed as $X_i \sim p$, but the points are *not* i.i.d.

Given is a continuous target distribution with density p.

Metropolis-Hastings (MH) kernel

1. We start by defining a conditional probability $q(y|x)$ on $\mathbf{X}$.
   q has nothing to do with p. We could e.g. choose $q(y|x) = g(y|x, \sigma^2)$, as in the previous example.
2. We define a **rejection kernel** A as

$$A(x_{n+1}|x_n) := \min\left\{1, \frac{q(x_i|x_{i+1})p(x_{i+1})}{q(x_{i+1}|x_i)p(x_i)}\right\}$$

   The normalization of p cancels in the quotient, so knowing $\tilde{p}$ is again enough.

3. We define the transition probability of the chain as

$$t(x_{i+1}|x_i) := q(x_{i+1}|x_i)A(x_{i+1}|x_i) + \delta_{x_i}(x_{i+1})c(x_i) \quad \text{where} \quad c(x_i) := \overbrace{\int q(y|x_i)(1-A(y|x_i))dy}^{\text{total probability that a proposal is sampled and then rejected}}$$

Sampling from the MH chain

At each step $i+1$, generate a proposal $X^* \sim q(\,.\,|x_i)$ and $U_i \sim \text{Uniform}[0,1]$.

- If $U_i \leq A(x^*|x_i)$, accept proposal: Set $x_{i+1} := x^*$.
- If $U_i > A(x^*|x_i)$, reject proposal: Set $x_{i+1} := x_i$.

Recall: Fundamental theorem on Markov chains

Suppose we sample $X_1 \sim P_{\text{init}}$ and $X_{i+1} \sim t(\,.\,|x_i)$. This defines a distribution P_i of X_i, which can change from step to step. If the MC is nice (recall: recurrent and aperiodic), then

$$P_i \to P_{\text{inv}} \qquad \text{for} \qquad i \to \infty \ .$$

Note: Making precise what aperiodic means in a continuous state space is a bit more technical than in the finite case, but the theorem still holds. We will not worry about the details here.

Implication

- If we can show that $P_{\text{inv}} \equiv p$, we do not have to know how to sample from p.
- Instead, we can start with *any* P_{init}, and will get arbitrarily close to p for sufficiently large i.

The number m of steps required until $P_m \approx P_{\text{inv}} \equiv p$ is called the **mixing time** of the Markov chain. (In probability theory, there is a range of definitions for what exactly $P_m \approx P_{\text{inv}}$ means.)

In MC samplers, the first m samples are also called the **burn-in** phase. The first m samples of each run of the sampler are discarded:

$$\underbrace{X_1, \ldots, X_{m-1}}_{\text{Burn-in; discard.}}, \underbrace{X_m, X_{m+1}, \ldots}_{\text{Samples from (approximately) } p\text{; keep.}}$$

Convergence diagnostics

In practice, we do not know how large j is. There are a number of methods for assessing whether the sampler has mixed. Such heuristics are often referred to as **convergence diagnostics**.

Even after burn-in, the samples from a MC are not i.i.d.

Strategy

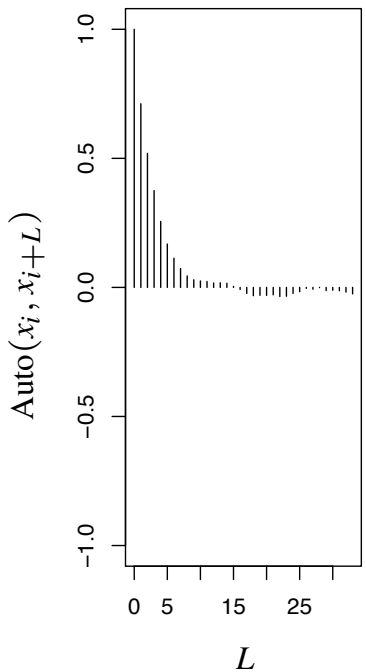
- Estimate empirically how many steps L are needed for x_i and x_{i+L} to be approximately independent. The number L is called the **lag**.
- After burn-in, keep only every Lth sample; discard samples in between.

Estimating the lag

The most commen method uses the **autocorrelation function**:

$$\text{Auto}(x_i, x_j) := \frac{\mathbb{E}[x_i - \mu_i] \cdot \mathbb{E}[x_j - \mu_j]}{\sigma_i \sigma_j}$$

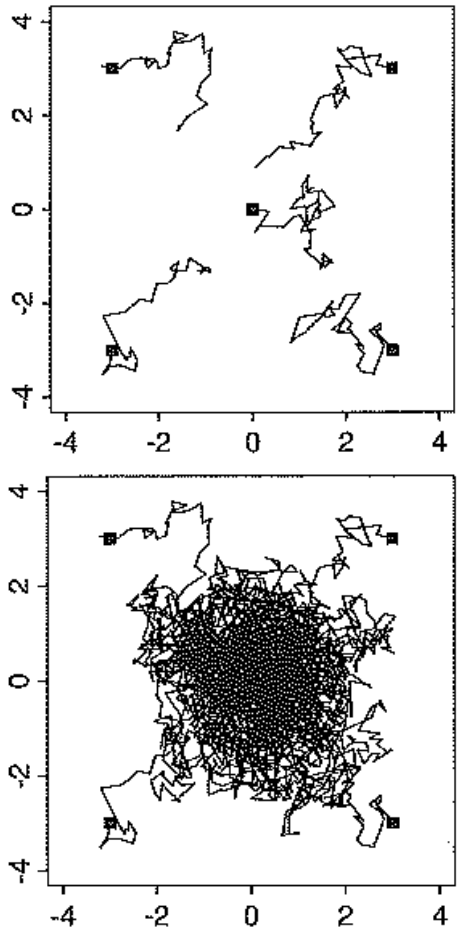
We compute $\text{Auto}(x_i, x_{i+L})$ empirically from the sample for different values of L, and find the smallest L for which the autocorrelation is close to zero.



There are about half a dozen popular convergence crieteria; the one below is an example.

Gelman-Rubin criterion

- Start several chains at random. For each chain k, sample X_i^k has a marginal distribution P_i^k.
- The distributions of P_i^k will differ between chains in early stages.
- Once the chains have converged, all $P_i = P_{\text{inv}}$ are identical.
- Criterion: Use a hypothesis test to compare P_i^k for different k (e.g. compare P_i^2 against null hypothesis P_i^1). Once the test does not reject anymore, assume that the chains are past burn-in.



Reference: A. Gelman and D. B. Rubin: "Inference from Iterative Simulation Using Multiple Sequences", *Statistical Science*, Vol. 7 (1992) 457-511.

The Metropolis-Hastings rejection kernel was defined as:

$$A(x_{n+1}|x_n) = \min\left\{1, \frac{q(x_i|x_{i+1})p(x_{i+1})}{q(x_{i+1}|x_i)p(x_i)}\right\} .$$

Hence, we certainly accept if the second term is larger than 1, i.e. if

$$q(x_i|x_{i+1})p(x_{i+1}) > q(x_{i+1}|x_i)p(x_i) .$$

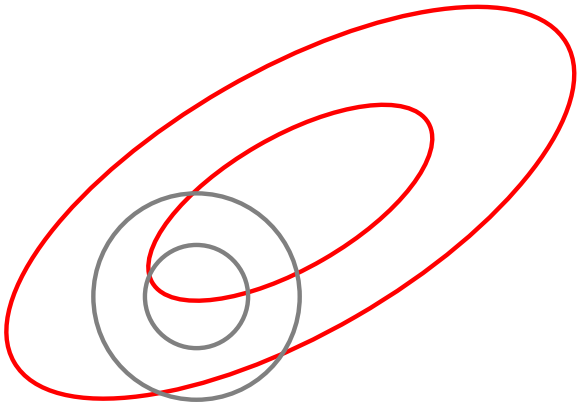
That means:

- We always accept the proposal value x_{i+1} if it *increases* the probability under p.
- If it *decreases* the probability, we still accept with a probability which depends on the difference to the current probability.

Hill-climbing interpretation

- The MH sampler somewhat resembles a gradient ascent algorithm on p, which *tends* to move in the direction of increasing probability p.
- However:
  - The actual steps are chosen at random.
  - The sampler can move "downhill" with a certain probability.
  - When it reaches a local maximum, it does not get stuck there.

Everyone's favorite example: Two Gaussians



red = target distribution p
gray = proposal distribution q

- $\mathrm{Var}[q]$ too large:
  Will overstep p; many rejections.
- $\mathrm{Var}[q]$ too small:
  Many steps needed to achieve good coverage of domain.

If p is unimodal and can be roughly approximated by a Gaussian, $\mathrm{Var}[q]$ should be chosen as smallest covariance component of p.

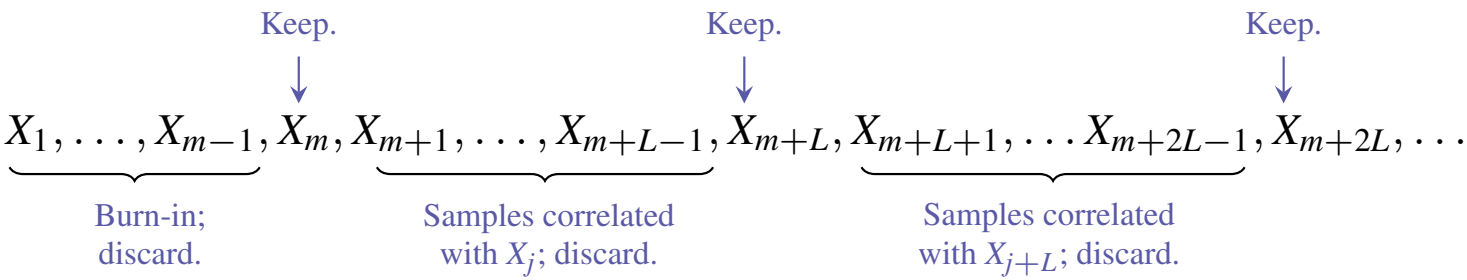
More generally

For complicated posteriors (recall: small regions of concentration, large low-probability regions in between) choosing q is much more difficult. To choose q with good performance, we already need to know something about the posterior.

There are many strategies, e.g. mixture proposals (with one component for large steps and one for small steps).

SUMMARY: MH SAMPLER

- MCMC samplers construct a MC with invariant distribution p.
- The MH kernel is one generic way to construct such a chain from p and a proposal distribution q.
- Formally, q does not depend on p (but arbitrary choice of q usually means bad performance).
- We have to discard an initial number m of samples as burn-in to obtain samples (approximately) distributed according to p.
- After burn-in, we keep only every Lth sample (where $L = \text{lag}$) to make sure the x_i are (approximately) independent.



THE GIBBS SAMPLER

By far the most widely used MCMC algorithm is the Gibbs sampler.

Full conditionals

Suppose $\mathcal{L}(X)$ is a distribution on $\mathbb{R}^{\mathrm{D}}$, so $X = (X_1, \ldots, X_{\mathrm{D}})$. The conditional probability of the entry X_d given all other entries,

$$\mathcal{L}(X_d | X_1, \ldots, X_{d-1}, X_{d+1}, \ldots, X_{\mathrm{D}})$$

is called the **full conditional** distribution of X_d.

On $\mathbb{R}^{\mathrm{D}}$, that means we are interested in a density

$$p(x_d | x_1, \ldots, x_{d-1}, x_{d+1}, \ldots, x_{\mathrm{D}})$$

Gibbs sampling

The Gibbs sampler is the special case of the Metropolis-Hastings algorithm defined by

$$\text{propsoal distribution for } X_d \quad = \quad \text{full conditional of } X_d \ .$$

- Gibbs sampling is only applicable if we can compute the full conditionals for each dimension d.
- If so, it provides us with a *generic* way to derive a proposal distribution.

Proposal distribution

Suppose p is a distribution on $\mathbb{R}^{\mathrm{D}}$, so each sample is of the form $X_i = (X_{i,1}, \ldots, X_{i,\mathrm{D}})$. We generate a proposal X_{i+1} coordinate-by-coordinate as follows:

$$
\begin{aligned}
X_{i+1,1} &\sim p(\,.\,|x_{i,2}, \ldots, x_{i,\mathrm{D}}) \\
&\vdots \\
X_{i+1,d} &\sim p(\,.\,|x_{i+1,1}, \ldots, x_{i+1,d-1}, x_{i,d+1}, \ldots, x_{i,\mathrm{D}}) \\
&\vdots \\
X_{i+1,\mathrm{D}} &\sim p(\,.\,|x_{i+1,1}, \ldots, x_{i+1,\mathrm{D}-1})
\end{aligned}
$$

Note: Each new $X_{i+1,d}$ is *immediately* used in the update of the next dimension $d+1$.

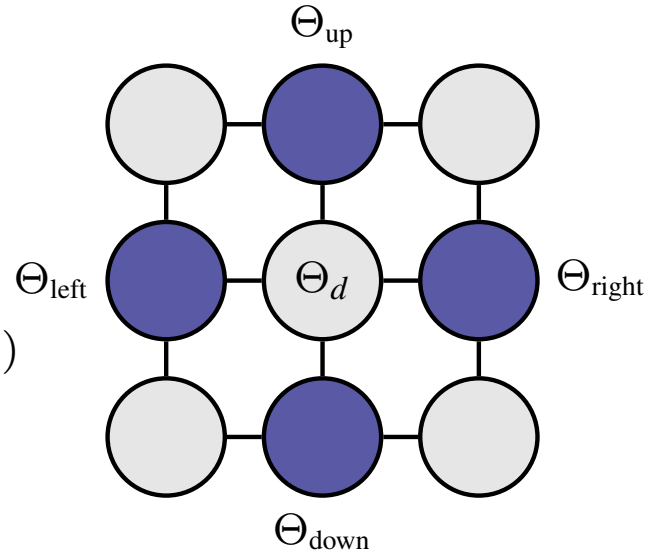
A Metropolis-Hastings algorithms with proposals generated as above is called a **Gibbs sampler**.

No rejections

It is straightforward to show that the Metropolis-Hastings acceptance probability for each $x_{i+1,d+1}$ is 1, so *proposals in Gibbs sampling are always accepted.*

EXAMPLE: MRF

In a MRF with D nodes, each dimension d corresponds to one vertex.



Full conditionals

In a grid with 4-neighborhoods for instance, the Markov property implies that

$$p(\theta_d|\theta_1,\ldots,\theta_{d-1},\theta_{d+1},\ldots,\theta_{\text{D}}) = p(\theta_d|\theta_{\text{left}},\theta_{\text{right}},\theta_{\text{up}},\theta_{\text{down}})$$

Specifically: Potts model with binary weights

Recall that, for sampling, knowing only $\tilde{p}$ (unnormalized) is sufficient:

$$\tilde{p}(\theta_d|\theta_1,\ldots,\theta_{d-1},\theta_{d+1},\ldots,\theta_{\text{D}}) =$$
$$\exp\Big(\beta(\mathbb{I}\{\theta_d=\theta_{\text{left}}\}+\mathbb{I}\{\theta_d=\theta_{\text{right}}\}+\mathbb{I}\{\theta_d=\theta_{\text{up}}\}+\mathbb{I}\{\theta_d=\theta_{\text{down}}\}\Big)$$

This is clearly very efficiently computable.

Sampling the Potts model

Each step of the sampler generates a sample

$$\theta_i = (\theta_{i,1}, \ldots, \theta_{i,\mathrm{D}}) \ ,$$

where D is the number of vertices in the grid.

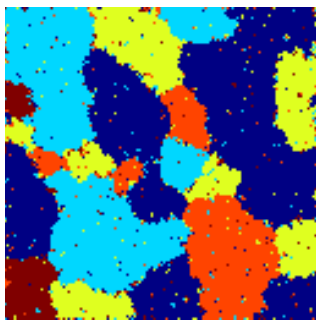
Gibbs sampler

Each step of the Gibbs sampler generates n updates according to

$$\begin{aligned}\theta_{i+1,d} &\sim p(\,.\,|\theta_{i+1,1}, \ldots, \theta_{i+1,d-1}, \theta_{i,d+1}, \ldots, \theta_{i,\mathrm{D}}) \\ &\propto \exp\Big(\beta(\mathbb{I}\{\theta_{i+1,d} = \theta_{\text{left}}\} + \mathbb{I}\{\theta_{i+1,d} = \theta_{\text{right}}\} + \mathbb{I}\{\theta_{i+1,d} = \theta_{\text{up}}\} + \mathbb{I}\{\theta_{i+1,d} = \theta_{\text{down}}\})\Big)\end{aligned}$$

This example is due to Erik Sudderth (UC Irvine).

MRFs as "segmentation" priors

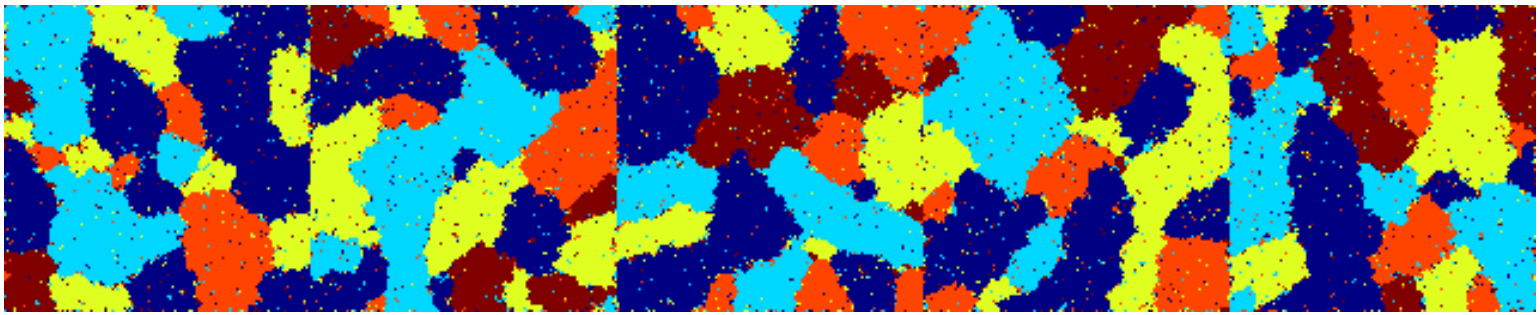


- MRFs where introduced as tools for image smoothing and segmentation by D. and S. Geman in 1984.
- They sampled from a Potts model with a Gibbs sampler, discarding 200 iterations as burn-in.
- Such a sample (after 200 steps) is shown above, for a Potts model in which each variable can take one out of 5 possible values.
- These patterns led computer vision researchers to conclude that MRFs are "natural" priors for image segmentation, since samples from the MRF resemble a segmented image.

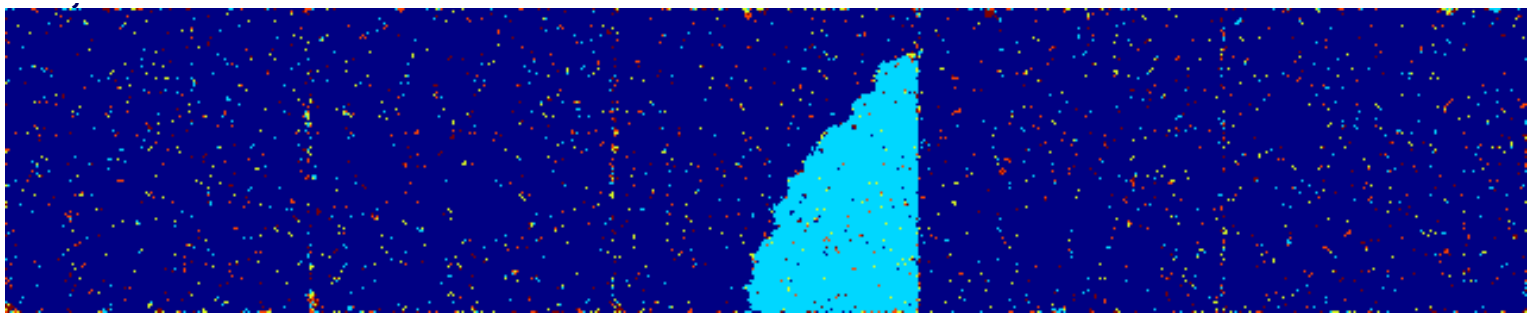
EXAMPLE: BURN-IN MATTERS

E. Sudderth ran a Gibbs sampler on the same model and sampled after 200 iterations (as the Geman brothers did), and again after 10000 iterations:

200 iterations



10000 iterations



Chain 1 Chain 5

- The "segmentation" patterns are not sampled from the MRF distribution $p \equiv P_{\text{inv}}$, but rather from $P_{200} \neq P_{\text{inv}}$.
- The patterns occur not because MRFs are "natural" priors for segmentations, but because *the sampler's Markov chain has not mixed.*
- MRFs are smoothness priors, not segmentation priors.

VARIATIONAL INFERENCE

Problem

We have to solve an inference problem where the correct solution is an "intractable" distribution with density p^* (e.g. a complicated posterior in a Bayesian inference problem).

Variational approach

Approximate p^* as

$$q^* := \arg\min_{q \in \mathcal{Q}} \phi(q, p^*)$$

where $\mathcal{Q}$ is a class of simple distributions and ϕ is a cost function (small ϕ means good fit). That turns the inference problem into a constrained optimization problem

$$\min \phi(q, p^*)$$
$$\text{s.t. } q \in \mathcal{Q}$$

Variational inference approximates a complicated distribution by minimizing the distance (or discrepancy) to a class of tractable distributions.

Recall: Optimization approach to problems

Formulate your problem such that the solution $x^* \in \mathbb{R}^d$ is the minimum of some function f, and solve

$$x^* := \arg\min_{x \in \mathbb{R}^d} f(x)$$

possibly under constraints.

Examples: Support vector machines, linear regression, logistic regression, ...

Inference problem as above

$$q^* := \arg\min_{q \in \mathcal{Q}} \phi(q, p^*)$$

- q is now a function (a density), not a point in $\mathbb{R}^d$.
- We have to optimize over a space of functions. Such spaces are in general infinite-dimensional.
- Often: $\mathcal{Q}$ is a parametric model, with parameter space $\mathbb{T} \subset \mathbb{R}^d$
  $\rightarrow$ reduces to optimization over $\mathbb{R}^d$.
- However: Optimization over infinite-dimensional spaces is in principle possible.

- Let $\mathcal{F}$ be a space of functions (e.g. all continuous functions on $\mathbb{R}$).
- A function $\phi : \mathcal{F} \to \mathbb{R}$ (a function whose arguments are functions) is called a **functional**.
- Examples: (1) The integral of a function. (2) The differential entropy of a density.

Recall: Derivatives (on $\mathbb{R}$)

The differential of $f : \mathbb{R}^d \to \mathbb{R}$ at point x is

$$\delta f(x) = \lim_{\varepsilon \searrow 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon} \quad \text{if } d = 1 \text{ or} \quad \delta f(x) = \lim_{\|\tilde{x}\| \searrow 0} \frac{f(x+\tilde{x}) - f(x)}{\|\tilde{x}\|} \quad \text{in general.}$$

The d-dimensional case works by reducing to the 1-dimensional case using a norm.

Derivatives of functionals

If $\mathcal{F}$ is a function space and $\| \bullet \|$ a norm on $\mathcal{F}$, we can apply the same idea to $\phi : \mathcal{F} \to \mathbb{R}$:

$$\delta\phi(f) := \lim_{\|\tilde{f}\| \searrow 0} \frac{\phi(f+\tilde{f}) - \phi(f)}{\|\tilde{f}\|}$$

$\delta\phi(f)$ is called the **Fréchet derivative** of ϕ at f.

f is a minimum of a Fréchet-differentiable functional ϕ only if $\delta\phi(f) = 0$.

Optimization

We can *in principle* find a minimum of ϕ by gradient descent: Add increment functions Δf_k "in the direction of" $\delta\phi(f_k)$ to the current solution candidate f_k.

The maximum entropy problem is often cited as an example.

Horseshoes

- We have to represent the infinite-dimensional quantities f_k and Δf_k in some way.
- Many interesting functionals ϕ are not Fréchet-differentiable as functionals on $\mathcal{F}$. They only become differentiable when constrained to a much smaller subspace.

One solution is "variational calculus", an analytic technique that addresses both problems. (We will not need the details.)

Recall: Maximum entropy principle

- The maximum entropy principle chooses a distribution within some set $\mathcal{P}$ of candidates by selecting the one with the largest entropy.
- That is: It solves the optimization problem

$$\begin{aligned} &\max \mathbb{H}(p) \\ &\text{s.t. } p \in \mathcal{P} \end{aligned}$$

- For example, if $\mathcal{P}$ are all those distributions under which some given statistic S takes a given expected value, we obtain exponential family distributions with sufficient statistic S.

Maximum entropy as functional optimization

- The entropy $\mathbb{H}$ assigns a scalar to a distribution $\rightarrow$ functional!
- Problem: The entropy as a functional e.g. on all distributions on $\mathbb{R}$ is concave, but it is *not* differentiable; it is not even continuous.
- The solution for exponential families can be determined using variational calculus.

Functional optimization in machine learning

- We will be interested in problems of the form

$$\min_q \phi(q)$$
$$\text{s.t. } q \in \mathcal{Q}$$

  where $\mathcal{Q}$ is a parametric family.
- That means each element of $\mathcal{Q}$ is of the form $q(\bullet|\theta)$, for $\theta \in \mathbb{T} \subset \mathbb{R}^d$.
- The problem then reduces back to optimization in $\mathbb{R}^d$:

$$\min_\theta \phi(q(\bullet|\theta))$$
$$\text{s.t. } \theta \in \mathbb{T}$$

- We can apply gradient descent, Newton, etc.

Recall

The **information** in observing $X = x$ under a probability mass function P is

$$J_P(x) := \log \frac{1}{P(x)} = -\log P(x) \ .$$

Its expectation $\mathbb{H}(P) := \mathbb{E}_P[J_P(X)]$ is the **entropy** of P.

The **Kullback-Leibler divergence** of P and Q is

$$D_{\mathrm{KL}}(P\|Q) := \mathbb{E}_P[J_Q(X)] - \mathbb{H}(P) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Entropy and KL divergence for densities

If p and q are probability densities, then

$$\mathbb{H}(p) := -\int p(x) \log p(x) dx \qquad \text{and} \qquad D_{\mathrm{KL}}(p\|q) := \int p(x) \log \frac{p(x)}{q(x)} dx$$

are the **differential entropy** of p and the **Kullback-Leibler divergence** of p and q.

Be careful

- The differential entropy does not behave like the entropy (e.g. it can be negative).
- The KL divergence for densities has properties analogous to the mass function case.

Recall VI optimization problem

$$q^* := \arg\min_{q \in \mathcal{Q}} \phi(q, p^*)$$

We have to choose a cost function.

The term "variational inference" in machine learning typically implies ϕ is a KL divergence,

$$q^* := \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}(q, p^*)$$

Order of the arguments

Recall that D_{KL} is not symmetric, so

$$D_{\mathrm{KL}}(q, p^*) \neq D_{\mathrm{KL}}(p^*, q)$$

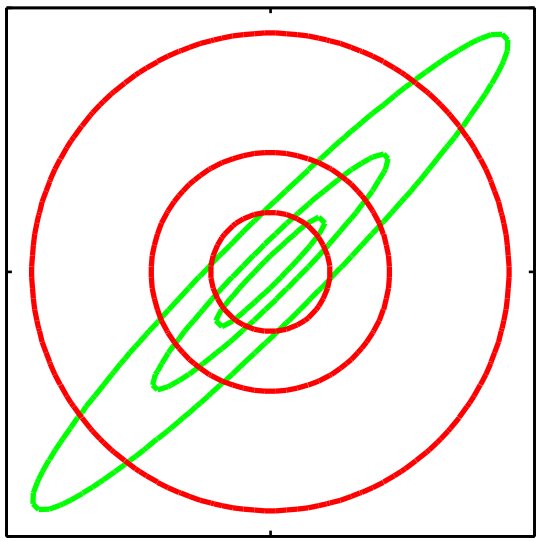
Which order should we use?

- Recall $D_{\mathrm{KL}}(p\|q)$ is an expectation with respect to p.
- $D_{\mathrm{KL}}(p^*\|q)$ emphasizes regions where the "true" model p^* has high probability. That is what we should use if possible.
- We use VI because p^* is intractable, so we can usually not compute expectations under it.
- We use the expectation $D_{\mathrm{KL}}(q, p^*)$ under the approximating simpler model instead.

We have to understand the implications of this choice.

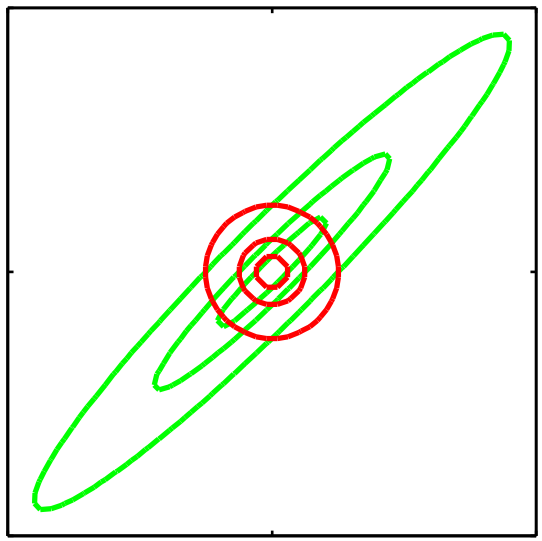
Approximating a Gaussian by a spherical Gaussian

What VI would do if possible



$D_{\mathrm{KL}}(p^* \| q) = D_{\mathrm{KL}}(green \| red)$

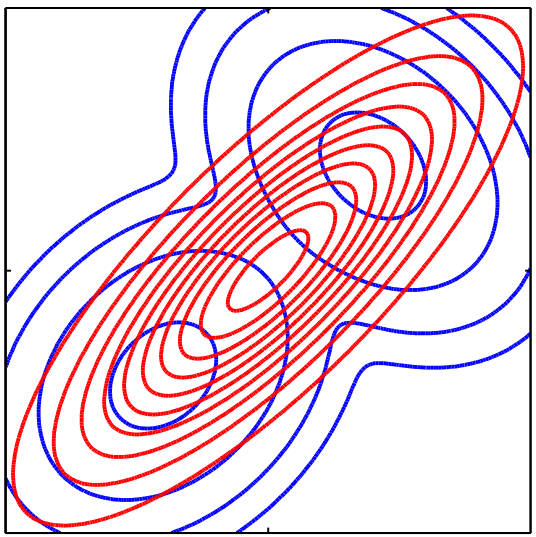
What VI does



$D_{\mathrm{KL}}(q \| p^*) = D_{\mathrm{KL}}(red \| green)$

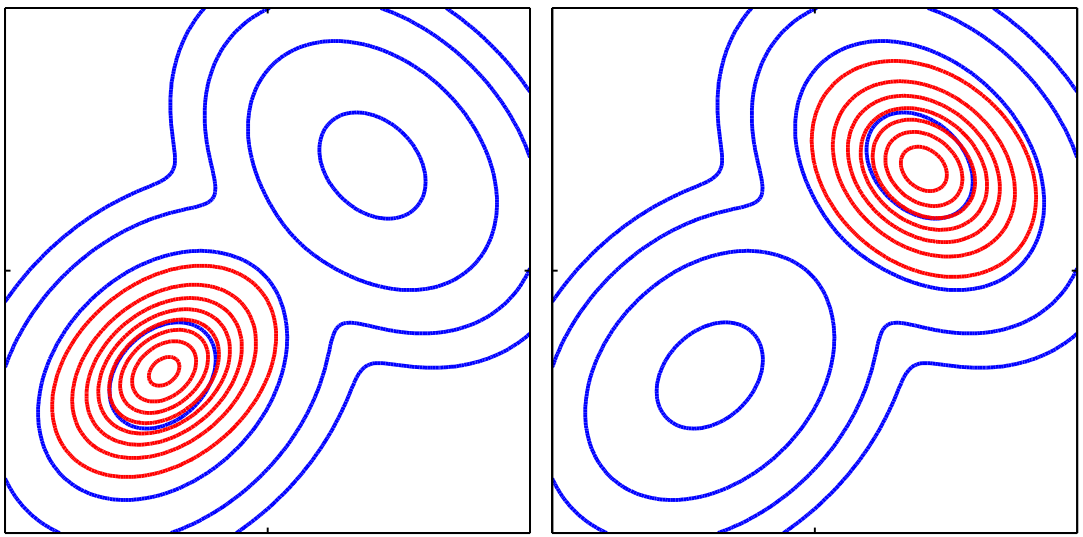
Approximating a Gaussian mixture by a single Gaussian

What VI would do if possible



$D_{\mathrm{KL}}(p^* \| q) = D_{\mathrm{KL}}(\textit{blue} \| \textit{red})$

What VI does



$D_{\mathrm{KL}}(q \| p^*) = D_{\mathrm{KL}}(\textit{red} \| \textit{blue})$

Often: Target distribution is a posterior of a parameter or latent variable $\mathbf{Z}$, given data $\mathbf{x}$.

Basic approximation problem

If the posterior density is $p^*(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$, then

$$q^*(\bullet) = \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}(q(\bullet)|p(\bullet|\mathbf{x})) \ .$$

Transforming the objective function

$$\begin{aligned} D_{\mathrm{KL}}(q(\bullet)|p(\bullet|\mathbf{x})) &= \mathbb{E}\Big[\log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{x})}\Big] \\ &= \mathbb{E}[\log q(\mathbf{Z})] - \mathbb{E}[\log p(\mathbf{Z}|\mathbf{x})] \\ &= \mathbb{E}[\log q(\mathbf{Z})] - \mathbb{E}[\log p(\mathbf{Z}, \mathbf{x})] + \log p(\mathbf{x}) \end{aligned}$$

- The evidence $p(\mathbf{x})$ is hard to compute (it is an integral over $p^*(\mathbf{x}|\mathbf{z})$).
- It depends only on $\mathbf{x}$, so it is an additive constant w.r.t. the optimization problem.
- Dropping it from the objective function does not change the location of the minimum.

$$F(q) := \mathbb{E}[\log q(\mathbf{Z})] - \mathbb{E}[\log p(\mathbf{Z}, \mathbf{x})]$$

Summary: VI approximation

$$\begin{array}{ll} \min & F(q) \\ \text{s.t.} & q \in \mathcal{Q} \end{array} \qquad \text{where} \qquad F(q) = \mathbb{E}[\log q(\mathbf{Z})] - \mathbb{E}[\log p(\mathbf{Z}, \mathbf{x})]$$

Terminology

- The function F is called a **free energy** in statistical physics.
- Since there are different forms of free energies, various authors attach different adjectives (variational free energy, Helmholtz free energy, etc).
- Parts of the machine learning literature have renamed F, by maximizing the objective function $-F$ and calling it an *evidence lower bound*, since

  $$-F(q) + D_{\mathrm{KL}}(q \,||\, p(\,\bullet\,|\mathbf{x})) = \log p(\mathbf{x}) \qquad \text{hence} \qquad e^{-F(q)} \leq p(\mathbf{x}) \;,$$

  and $p(\mathbf{x})$ is the "evidence" in the Bayes equation.

Definition

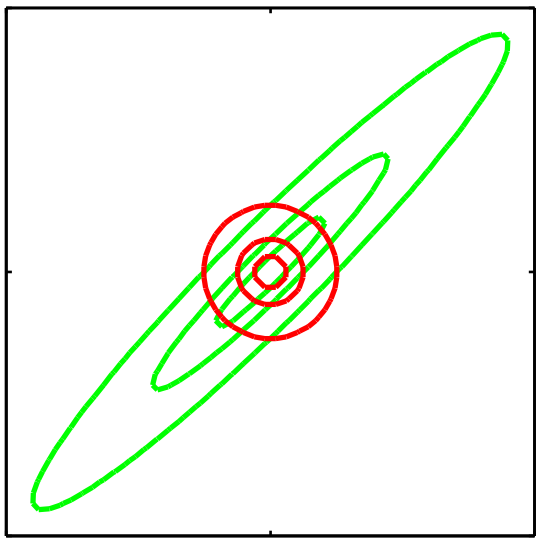
A variational approximation of a probability distribution p on a d-dimensional space

$$q^* := \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}(q, p^*)$$

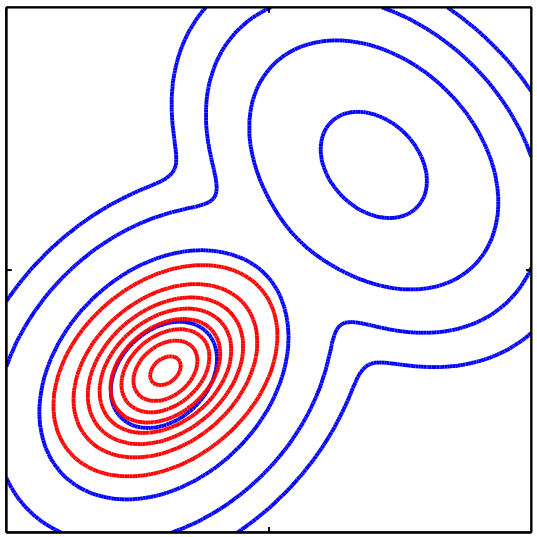
is called **mean field** approximation if each element of $\mathcal{Q}$ is factorial,

$$q(\mathbf{z}) = q_1(z_1) \dots q_d(z_d) \ .$$

In previous example



Mean field (Gaussian spherical)



Not a mean field

Model

We consider a MRF distribution for $X_1, \dots, X_n$ with values in $\{-1, +1\}$, given by

$$P(X_1, \dots, X_n) = \frac{1}{Z(\beta)} \exp(-\beta H(X_1, \dots, X_n)) \qquad \text{where } H = -\sum_{i,j=1}^{n} w_{ij} X_i X_j - \underbrace{\sum_{i=1}^{n} h_i X_i}_{\text{“external field”}}$$

w_{ij} and h_i are constant weights.

Physicists call this a “Potts model with an external magnetic field”.

Variational approximation

We choose $\mathcal{Q}$ as the family

$$\mathcal{Q} := \left\{ \prod_{i=1}^{n} Q_{m_i} \,\middle|\, m_i \in [-1, 1] \right\} \qquad \text{where} \qquad Q_m(X) := \begin{cases} \frac{1+m}{2} & X = 1 \\ \frac{1-m}{2} & X = -1 \end{cases}$$

Each factor is a Bernoulli$\left(\frac{1+m}{2}\right)$, except that the range is $\{-1, 1\}$ not $\{0, 1\}$.

Optimization Problem

$$\min D_{\mathrm{KL}}\Big(\prod_{i=1}^{n} Q_{m_i} \Big\| P\Big)$$
$$\text{s.t. } m_i \in [-1,1] \text{ for } i = 1,\ldots,n$$

Mean field solution

The mean field approximation is given by the parameter values m_i satisfying the equations

$$m_i = \tanh\Big(\beta\Big(\sum_{j=1}^{n} w_{ij} m_j + h_i\Big)\Big) \ .$$

That is: For given values of w_{ij} and h_i, we have to solve for the values m_i.

EXAMPLE: MEAN FIELD FOR THE POTTS MODEL

We have approximated the MRF

$$P(X_1, \ldots, X_n) \qquad \text{by} \qquad \prod_{i=1}^{n} \text{Bernoulli}(m_i) \quad \text{satisfying} \quad m_i = \tanh\left(\beta\left(\sum_{j=1}^{n} w_{ij} m_j + h_i\right)\right)$$

(where we interpret a 0 generated by the Bernoulli as a -1).

Interpretation

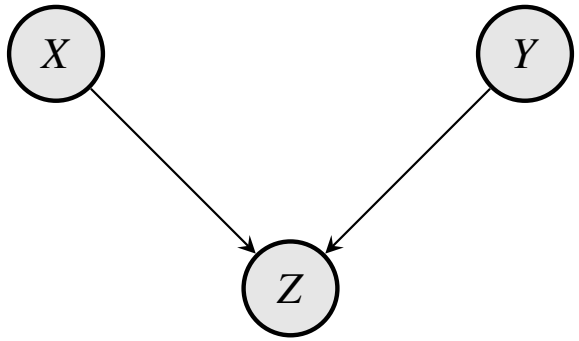
- In the MRF P, the random variables X_i interact.
- There is no interaction in the approximation.
- Instead, the effect of interactions is approximated by encoding them in the parameters.
- This is somewhat like a single effect ("field") acting on all variables simultanuously ("mean field").

How accurate is the approximation?

- In physics, P is used to model a ferromagnet with an external magnetic field.
- In this case, β is the inverse temperature.
- These systems exhibit a phenomenon called *spontanuous magnetization* at certain temperatures. The mean field solution predicts *spontanuous magnetization*, but at the wrong temperature values.

DIRECTED GRAPHICAL MODELS

Explaining Away



Conditioning on Z makes X and Y dependent.

Example

- Suppose X and Y are normal variables.
- $Z = X + Y$.

If we know Z, and someone reveals the value of Y to us, we know everything about X.

Explaining away

This effect is known as "explaining away"; this terminology makes sense if Z is an effect that either X or Y can account for, for example:

- X and Y are binary.
- $Z = \max\{X, Y\}$.

Suppose Z is binary. That may be explained by $X = 1$, but if we learn that $Y = 1$, it will make us less convinced that $X = 1$ (so Y "explains away" the effect attributed to X).

Directed Graphical Models: Mixtures and Admixtures

Next: Bayesian Mixtures and Admixtures

We will consider two variations on finite mixtures:

- *Bayesian mixture models* (mixtures with priors).
- *Admixtures*, in which the generation of each observation (e.g. document) can be influenced by several components (e.g. topics).
- One particular admixture model, called *latent Dirichlet allocation*, is one of the most succesful machine learning models of the past ten years.

$$\pi(x) = \sum_{k \leq K} c_k p(x|\theta_k)$$

Sampling from this model

1. Fix c_k and θ_k for $k = 1, \ldots, K$.
2. Generate $Z_i \sim \text{Multinomial}(c_1, \ldots, c_K)$.
3. Generate $X_i|Z_i \sim p(\bullet|\theta_{Z_i})$.

As a graphical model

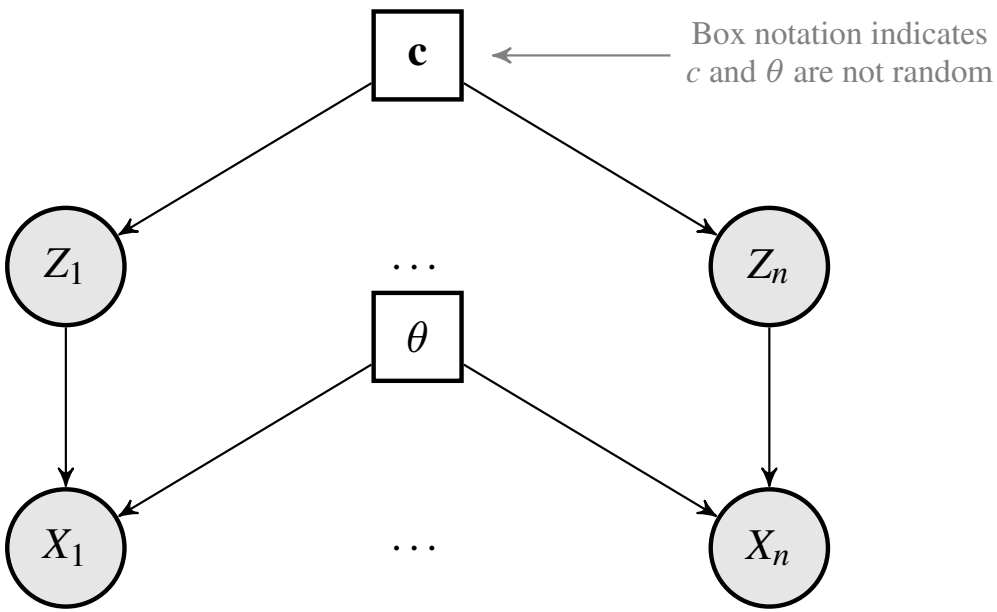
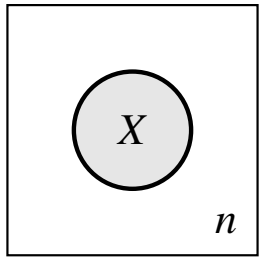


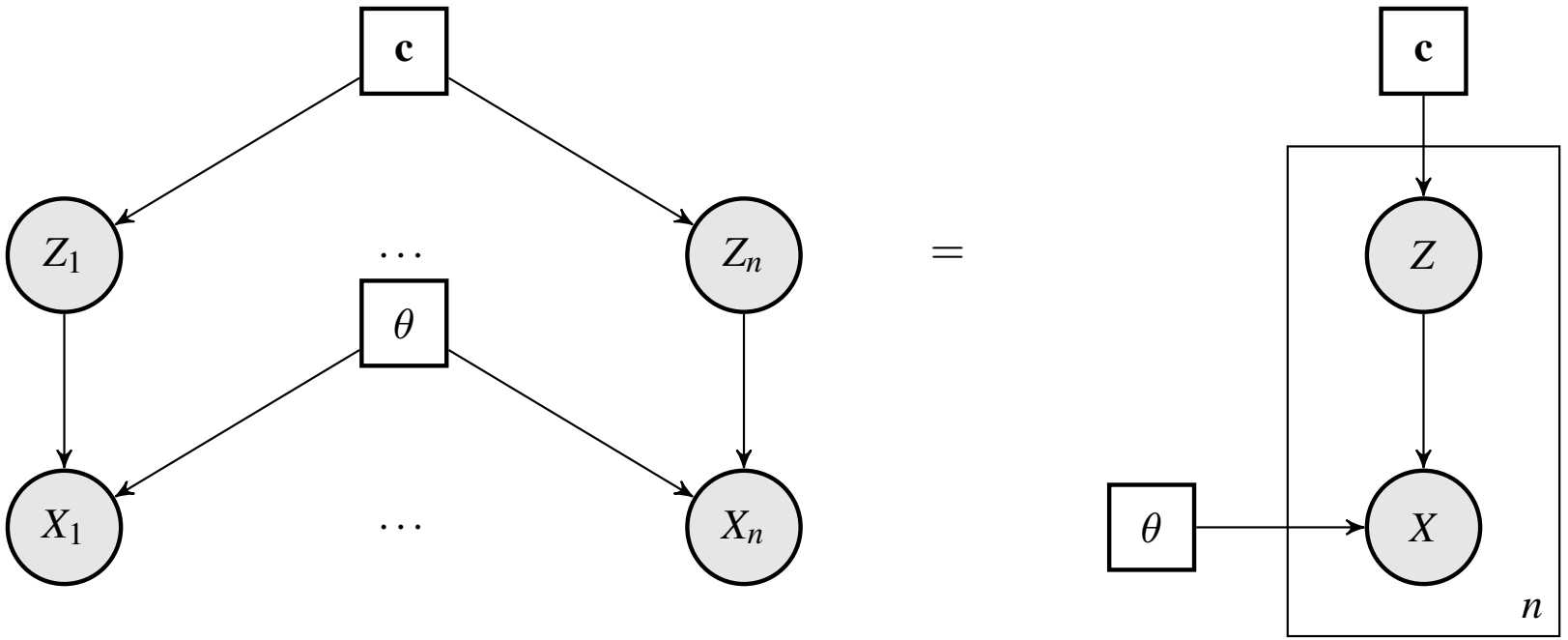
PLATE NOTATION

If variables are sampled repeatedly in a graphical model, we enclose these variables in "plate".



This means: Draw n (conditionally) independent samples from X.

Finite mixture with plate notation



Recall: Mixing distribution of a FMM

$$\pi(x) = \sum_{k=1}^{K} c_k p(x|\theta_k) = \int_{\mathbf{T}} p(x|\theta) m(\theta) d\theta \qquad \text{with} \qquad m := \sum_{k=1}^{K} c_k \delta_{\theta_k}$$

All parameters are summarized in the *mixing distribution m*.

Bayesian mixture model: Idea

In a Bayesian model, parameters are random variables. Here, that means a *random* mixing distribution:

$$M(\,.\,) = \sum_{k=1}^{K} C_k \delta_{\Theta_k}(\,.\,)$$

How can we define a random distribution?

Since M is discrete with finitely many terms, we only have to generate the random variables C_k and Θ_k:

$$M(\,.\,) = \sum_{k=1}^{K} C_k \delta_{\Theta_k}(\,.\,)$$

More precisely

Specifically, the term BMM implies that all priors are natural conjugate priors. That is:

- The mixture components $p(x|\theta)$ are an exponential family model.
- The prior on each Θ_k is a natural conjugate prior of p.
- The prior of the vector $(C_1, \ldots, C_K)$ is a Dirichlet distribution.

Explanation: Dirichlet distribution

- When we sample from a finite mixture, we choose a component k from a multinomial distribution with parameter vector $(c_1, \ldots, c_k)$.
- The conjugate prior of the multinomial is the Dirichlet distribution.

Definition

A model of the form

$$\pi(x) = \sum_{k=1}^{K} C_k p(x|\Theta_k) = \int_{\mathbf{T}} p(x|\theta) M(\theta) d\theta$$

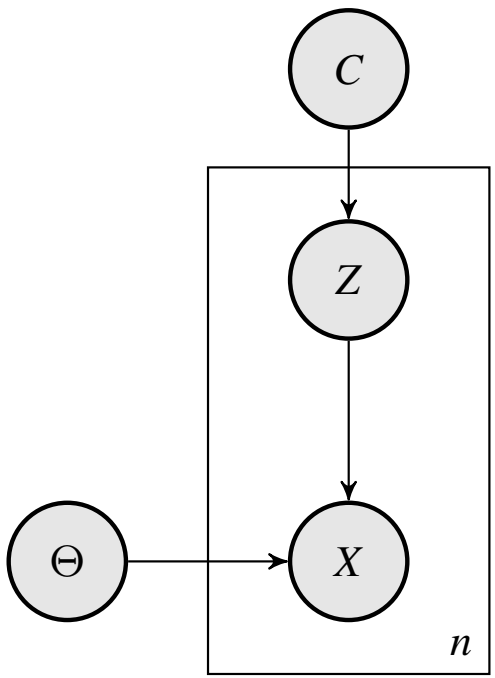
is called a **Bayesian mixture model** if $p(x|\theta)$ is an exponential family model and M a random mixing distribution, where:

- $\Theta_1, \ldots, \Theta_K \sim_{\text{iid}} q(\,.\,|\lambda, y)$, where q is a natural conjugate prior for p.
- $(C_1, \ldots, C_K)$ is sampled from a K-dimensional Dirichlet distribution.

Sampling from a Bayesian Mixture

1. Draw $C = (C_1, \ldots, C_k)$ from a Dirichlet prior.
2. Draw $\Theta_1, \ldots, \Theta_K \sim_{\text{iid}} q$, where q is the conjugate prior of p.
3. Draw $Z_i | C \sim \text{Multinomial}(C)$.
4. Draw $X_i | Z_i, \Theta \sim p(\bullet | \Theta_{Z_i})$.

As a graphical model



Posterior distribution

The posterior density of a BMM under observations $x_1, \ldots, x_n$ is (up to normalization):

$$\Pi(c_{1:K}, \theta_{1:K} | x_{1:n}) \propto \prod_{i=1}^{n} \Big(\sum_{k=1}^{K} c_k p(x_i | \theta_k) \Big) \Big(\prod_{k=1}^{K} q(\theta_k | \lambda, y) \Big) q_{\text{Dirichlet}}(c_{1:K})$$

The posterior is analytically intractable

- Thanks to conjugacy, we *can* evaluate each term of the posterior.
- However: Due to the $\prod_{k=1}^{K} \Big(\sum_{i=1}^{n} \ldots \Big)$ bit, the posterior has K^n terms!
- Even for 10 clusters and 100 observations, that is impossible to compute.

This Gibbs sampler is a bit harder to derive, so we skip the derivation and only look at the algorithm.

Recall: Bayesian mixture model

- Exponential family likelihood $p(x|\theta_k)$ for each cluster $k = 1, \ldots, K$.
- Natural conjugate prior q for all θ_k.
- Dirichlet prior Dirichlet(α, g) for the mixture weights $c_{1:K}$.

Assignment probabilities

Each step of the Gibbs sampler computes an assignment matrix:

$$\mathbf{a} = \begin{pmatrix} a_{11} & \ldots & a_{1K} \\ \vdots & & \vdots \\ a_{n1} & \ldots & a_{nK} \end{pmatrix} = \Big(\Pr\{x_i \text{ in cluster } k\}\Big)_{ik}$$

Entries are computed as they are in the EM algorithm:

$$a_{ik} = \frac{C_k p(x_i|\Theta_k)}{\sum_{l=1}^{K} C_l p(x_i|\Theta_l)}$$

In contrast to EM, the values C_k and Θ_k are random.

GIBBS FOR BMM: ALGORITHM

In each iteration j, the algorithm cycles through these steps:

1. For each x_i, sample an assignment

$$Z_i^j \sim \text{Multinomial}(a_{i1}^j, \ldots, a_{iK}^j) \qquad \text{where} \qquad \overbrace{a_{ik}^j = \frac{C_k^{j-1} p(x_i|\Theta_k^{j-1})}{\sum_{l=1}^K C_l^{j-1} p(x_i|\Theta_l^{j-1})}}^{\text{exactly as in EM}}$$

2. For each cluster k, sample a new value for Θ_k^j from the conjugate posterior $\Pi(\Theta_k)$ under the observations *currently* assigned to k:

$$\Theta_k^j \sim \Pi\Big(\Theta \,\Big|\, \lambda + \underbrace{\sum_{i=1}^n \mathbb{I}\{Z_i^j = k\}}_{\substack{\#\text{ points currently} \\ \text{assigned to } k}}, y + \underbrace{\sum_{i=1}^n \mathbb{I}\{Z_i^j = k\} S(x_i)}_{\substack{\text{aggregate } S(x_i) \text{ over} \\ \text{cluster } k}}\Big)$$

3. Sample new cluster proportions $C_{1:K}^j$ from the Dirichlet posterior (under all x_i):

$$C_{1:K}^j \sim \text{Dirichlet}(\alpha + n, g_{1:K}^j) \qquad \text{where} \qquad g_k^j = \frac{\alpha \cdot g_k + \overbrace{\sum_{i=1}^n \mathbb{I}\{Z_i^j = k\}}^{\substack{\#\text{ points currently} \\ \text{assigned to } k}}}{\underbrace{\alpha + n}_{\text{normalization}}}$$

prior expectation of C_k — prior concentration

COMPARISON: EM AND K-MEANS

The BMM Gibbs sampler looks very similar to the EM algorithm, with maximization steps (in EM) substituted by posterior sampling steps:

	Representation of assignments	Parameters
EM	Assignment probabilities $a_{i,1:K}$	a_{ik}-weighted MLE
K-means	$m_i = \arg\max_k(a_{i,1:K})$	MLE for each cluster
Gibbs for BMM	$m_i \sim \text{Multinomial}(a_{i,1:K})$	Sample posterior for each cluster

Recall: Multinomial text clustering

We assume the corpus is generated by a multinomial mixture model of the form

$$\pi(\mathbf{H}) = \sum_{k=1}^{K} c_k P(\mathbf{H}|\theta_k) \ ,$$

where $P(\mathbf{H}|\theta_k)$ is multionmial.

- A document is represented by a histogram $\mathbf{H}$.
- Topics $\theta_1, \ldots, \theta_K$.
- $\theta_{kj} = \Pr\{ \text{word } j \text{ in topic } k\}$.

Problem

Each document is generated by a single topic; that is a very restrictive assumption.

Parameters

Suppose we consider a corpus with K topics and a vocubulary of d words.

- $\phi \in \Delta_K$ topic proportions ($\phi_k = \Pr\{\text{ topic } k\}$).
- $\theta_1, \ldots, \theta_K \in \Delta_d$ topic parameter vectors ($\theta_{kj} = \Pr\{\text{ word } j \text{ in topic } k\}$).

Note: For random generation of documents, we assume that ϕ and the topic parameters θ_k are given (they properties of the corpus). To train the model, they have to be learned from data.

Model 1: Multinomial mixture

To sample a document containing M words:

1. Sample topic $Z \sim \text{Multinomial}(\phi)$.
2. For $i = 1, \ldots, M$: Sample $\text{word}_i | Z \sim \text{Multinomial}(\theta_Z)$.

The entire document is sample from topic Z.

Mixtures of topics

Whether we sample words or entire documents makes a big difference.

- When we sample from the multinomial mixture, we choose a topic at random, then sample the *entire* document from that topic.
- For several topics to be represented in the document, we have to sample each word individually (i.e. choose a new topic for each word).
- Problem: If we do that in the mixture above, every document has the same topic proportions.

Model 2: Admixture model

Each document explained as a *mixture* of topics, with mixture weights $C_{1:K}$.

Fix a matrix θ of size #topics $\times$ #words, where

$$\theta_{kj} := \text{ probability that word } j \text{ occurs under topic } k$$

1. Sample topic proportions $c_{1:K} \sim \text{Dirichlet}(\phi)$.
2. For $i = 1, \ldots, M$:
   - 2.1 Sample topic for word i as $Z_i | C_{1:K} \sim \text{Multinomial}(C_{1:K})$.
   - 2.2 Sample word$_i | Z_i \sim \text{Multinomial}(\theta_{Z_i})$.

This model is known as **Latent Dirichlet Allocation** (LDA).

Observation

LDA is *almost* a Bayesian mixture model: Both use multinomial components and a Dirichlet prior on the mixture weights. However, they are not identical.

Comparison

Bayesian MM	Admixture (LDA)
Sample $c_{1:K} \sim \text{Dirichlet}(\phi)$.	Sample $c_{1:K} \sim \text{Dirichlet}(\phi)$.
Sample topic $k \sim \text{Multinomial}(c_{1:K})$.	For $i = 1, \ldots, M$:
For $i = 1, \ldots, M$:	Sample topic $k_i \sim \text{Multinomial}(c_{1:K})$.
Sample word$_i \sim \text{Multinomial}(\theta_k)$.	Sample word$_i \sim \text{Multinomial}(\theta_{k_i})$.

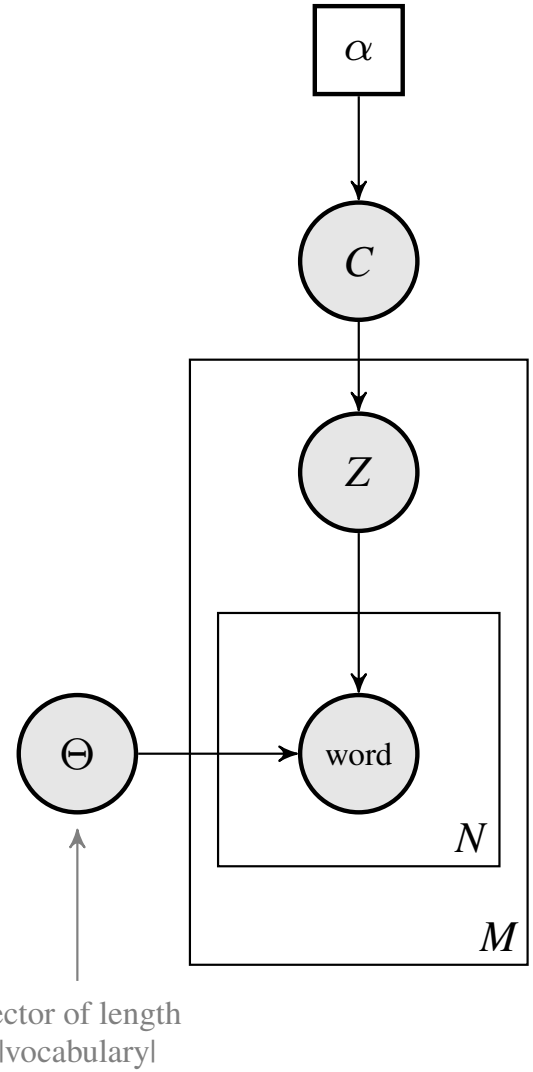
In admixtures:

- $c_{1:K}$ is generated at random, *once for each document*.
- Each word is sampled from its own topic.

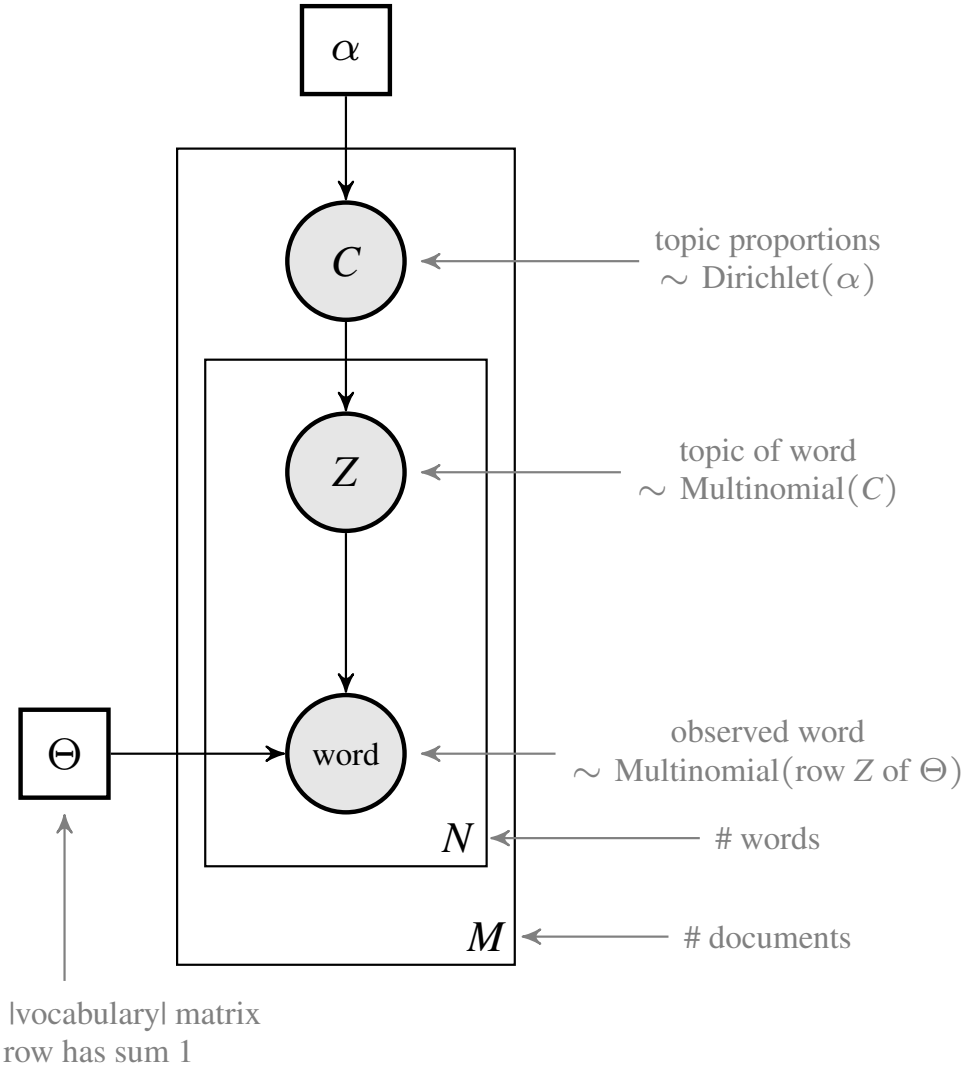
What do we learn in LDA?

LDA explains each document by a separate parameter $c_{1:K} \in \Delta_K$. That is, LDA models documents as *topic proportions*.

LDA as a Graphical Model



Bayesian mixture
(for M documents of N words each)



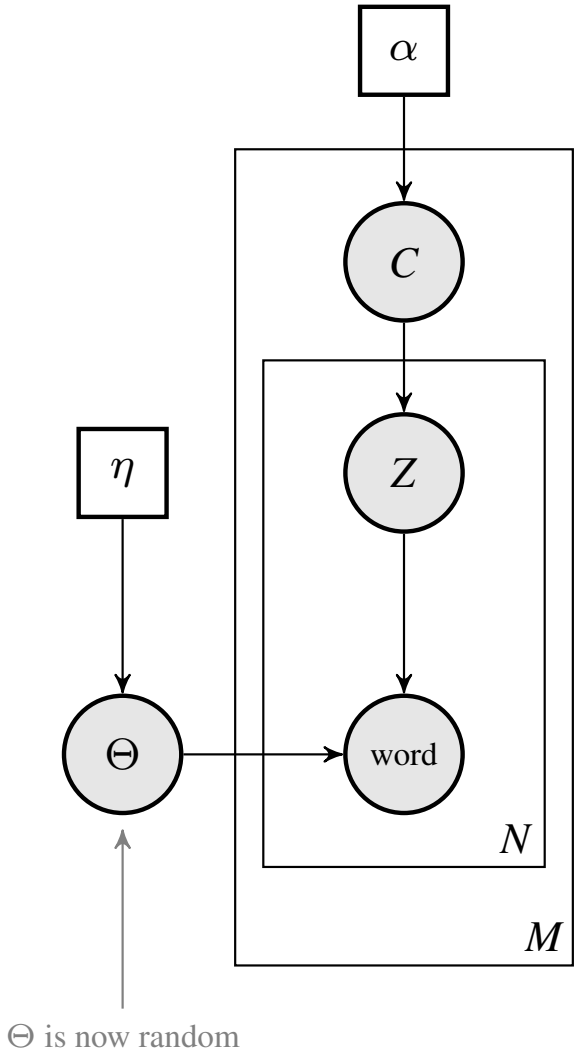
LDA

Prior on Topic Probabilities

- The parameter matrix θ is of size

$$\#\text{topics} \times |\text{vocabulary}|.$$

- Meaning: θ_{ki} = probability that term i is observed under topic k.
- Note entries of θ are non-negative and each row sums to 1.
- To learn the parameters θ along with the other parameters, we add a Dirichlet prior with parameters η. The rows are drawn i.i.d. from this prior.



Target distribution

$$\text{Posterior } \mathcal{L}(Z, C, \Theta | \text{words}, \alpha, \eta)$$

Recall: $Z|C \sim \text{Multinomial}(C)$ and $C, \Theta_1, \ldots, \Theta_K \sim \text{Dirichlet}$.

Variational approximation

$$\mathcal{Q} = \{q(z, c, \theta | \lambda, \phi, \gamma) \mid \lambda, \phi, \gamma\}$$

where

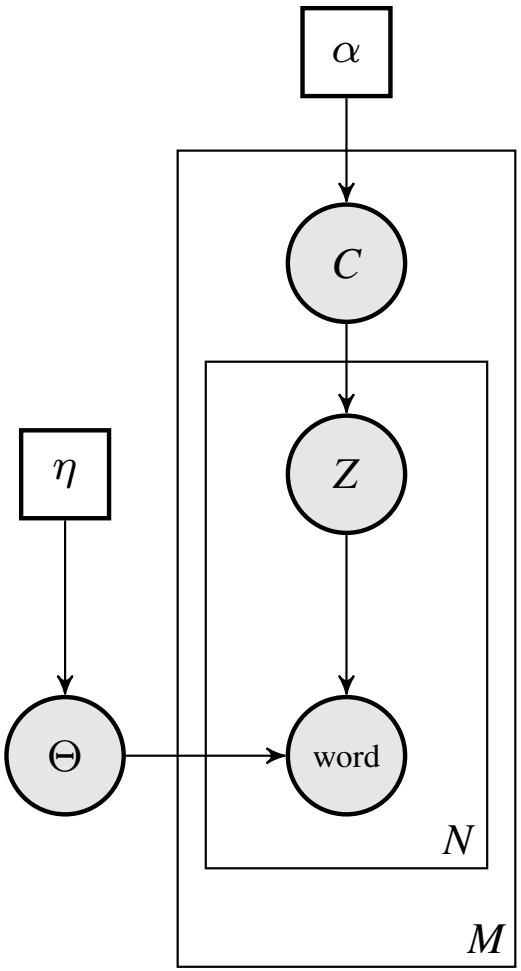
$$q(z, c, \theta | \lambda, \phi) := \prod_{k=1}^{K} p(\theta_k | \lambda) \prod_{m=1}^{M} \Big(q(c_m | \gamma_m) \prod_{n=1}^{N} p(z_{mn} | \phi_{mn}) \Big)$$

Dirichlet — Multinomial

We have introduced a new set of "variational" parameters λ, γ, ϕ.

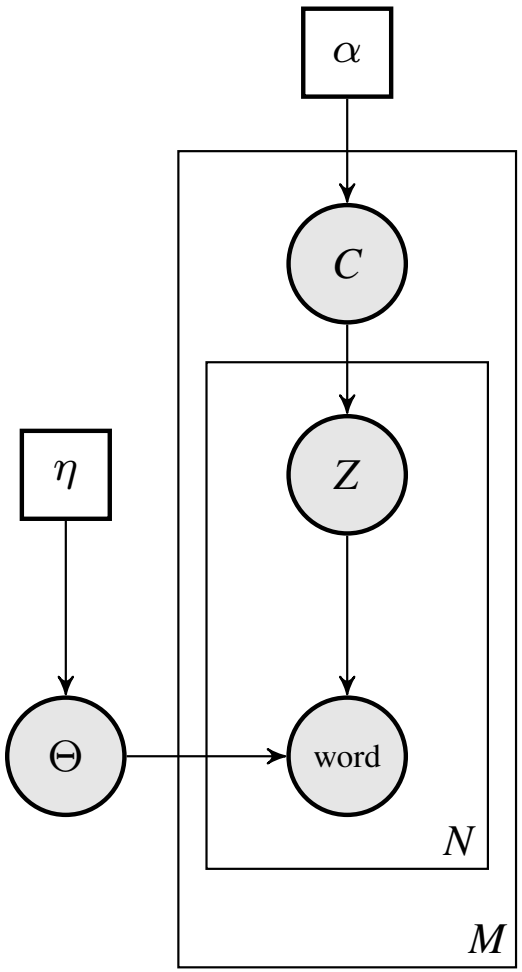
Variational inference problem

$$\begin{aligned} \min \quad & D_{\mathrm{KL}}(q(z, c, \theta | \lambda, \phi) || p(z, c, \theta | \alpha, \eta)) \\ \text{s.t.} \quad & q \in \mathcal{Q} \end{aligned}$$

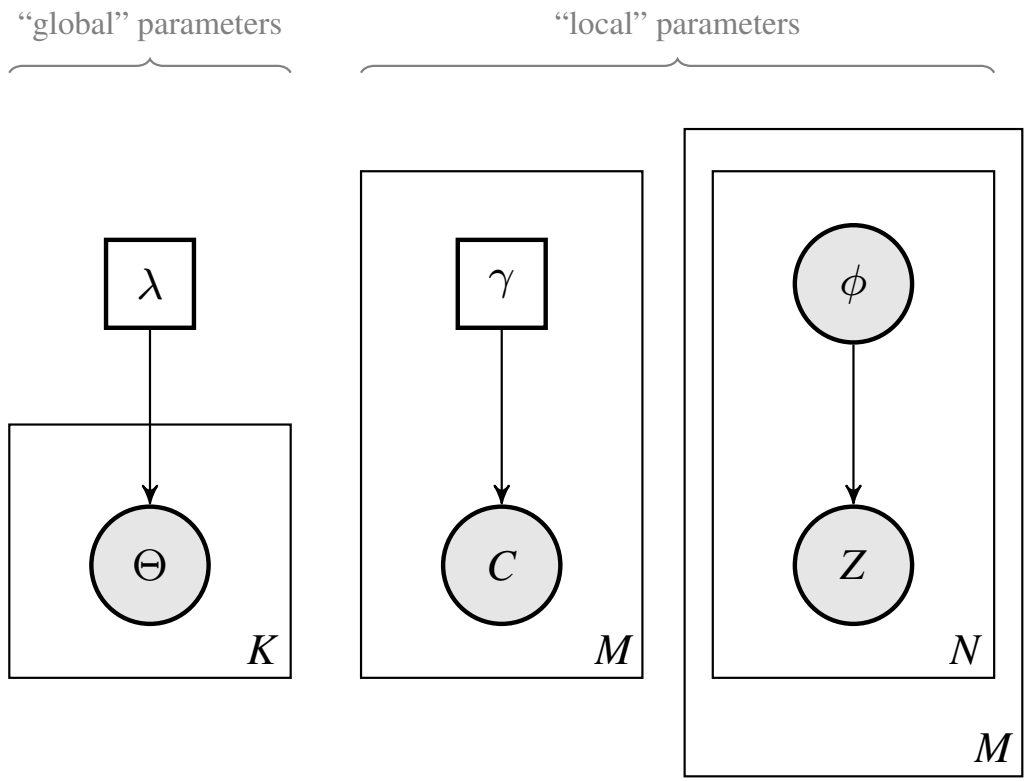


VARIATIONAL INFERENCE FOR LDA

$$q(z, c, \theta|\lambda, \phi) := \prod_{k=1}^{K} p(\theta_k|\lambda) \prod_{m=1}^{M} \left(q(c_m|\gamma_m) \prod_{n=1}^{N} p(z_{mn}|\phi_{mn}) \right)$$



LDA



Variational approximation

Algorithmic solution

We solve the minimization problem

$$\begin{aligned} \min \quad & D_{\mathrm{KL}}(q(z, c, \theta|\lambda, \phi)||p(z, c, \theta|\alpha, \eta)) \\ \text{s.t.} \quad & q \in \mathcal{Q} \end{aligned}$$

by applying gradient descent to the resulting free energy.

VI algorithm

It can be shown that gradient descent amounts to the following algorithm:

repeat (using update equations on next page)

- update local parameters
- update global parameters

until free energy has converged

In iteration t of the algorithm:

Local updates

$$\phi_{mn}^{(t+1)} := \frac{\tilde{\phi}_{mn}}{\sum_n \tilde{\phi}_{mn}}$$

where

$$\tilde{\phi}_{mn} = \exp\Big(\mathbb{E}_q[\log(C_{m1}), \ldots, \log(C_{md})|\gamma^{(t)}] + \mathbb{E}_q[\log(\Theta_{1,w_n}), \ldots, \log(\Theta_{d,w_n})|\lambda^{(t)}]\Big)$$

$$\gamma^{(t+1)} := \alpha + \sum_{n=1}^{N} \phi_n^{(t+1)}$$

Global updates

$$\lambda_k^{(t+1)} = \eta + \sum_m \sum_n \text{word}_{mn} \phi_{mn}^{(t+1)}$$

EXAMPLE: MIXTURE OF TOPICS

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.