CPSC 540: Machine Learning

Topic Models

Mark Schmidt

University of British Columbia

Winter 2018
In Bayesian statistics we work with posterior over parameters,

\[ p(\theta \mid x, \alpha, \beta) = \frac{p(x \mid \theta)p(\theta \mid \alpha, \beta)}{p(x \mid \alpha, \beta)}. \]

We discussed empirical Bayes, where you optimize prior using marginal likelihood,

\[ \arg\max_{\alpha,\beta} p(x \mid \alpha, \beta) = \arg\max_{\alpha,\beta} \int p(x \mid \theta)p(\theta \mid \alpha, \beta)d\theta. \]

Can be used to optimize \( \lambda_j \), polynomial degree, RBF \( \sigma_i \), polynomial vs. RBF, etc.

We also considered hierarchical Bayes, where you put a prior on the prior,

\[ p(\alpha, \beta \mid x, \gamma) = \frac{p(x \mid \alpha, \beta)p(\alpha, \beta \mid \gamma)}{p(x \mid \gamma)}. \]

But is the hyper-prior really needed?
Hierarchical Bayes as a Graphical Model

- Let $x^i$ be a binary variable, representing if treatment works on patient $i$,

  $$x^i \sim \text{Ber}(	heta).$$

- As before, let’s assume that $\theta$ comes from a beta distribution,

  $$\theta \sim \mathcal{B}(\alpha, \beta).$$

- We can visualize this as a graphical model:
Hierarchical Bayes for Non-IID Data

- Now let $x^i$ represent if treatment works on patient $i$ in hospital $j$.
- Let’s assume that treatment depends on hospital,

$$x^i_j \sim \text{Ber}(\theta_j).$$

- So the $x^i_j$ are only IID given the hospital.

- Problem: we may not have a lot of data for each hospital.
  - Can we use data from one hospital to learn about others?
  - Can we say anything about a hospital with no data?
Hierarchical Bayes for Non-IID Data

- Common approach: assume $\theta_j$ drawn from common prior,

$$\theta_j \sim \mathcal{B}(\alpha, \beta).$$

- This introduces dependency between parameters at different hospitals:

- But, if you fix $\alpha$ and $\beta$ then you can’t learn across hospitals:
  - The $\theta_j$ and d-separated given $\alpha$ and $\beta$. 
Hierarchical Bayes for Non-IID Data

- Consider treating $\alpha$ and $\beta$ as random variables and using a hyperprior:

- Now there is a dependency between the different $\theta_j$.
  - Due to unknown $\alpha$ and $\beta$.

- Now you can combine the non-IID data across different hospitals.
  - Data-rich hospitals inform posterior for data-poor hospitals.
  - You even consider the posterior for new hospitals with no data.
Outline

1. Topic Models
2. Rejection and Importance Sampling
3. Metropolis-Hastings Algorithm
Motivation for Topic Models

We want a model of the “factors” making up a set of documents.

- In this context, latent-factor models are called topic models.

Suppose you have the following set of sentences:

- I like to eat broccoli and bananas.
- I ate a banana and spinach smoothie for breakfast.
- Chinchillas and kittens are cute.
- My sister adopted a kitten yesterday.
- Look at this cute hamster munching on a piece of broccoli.

What is latent Dirichlet allocation? It’s a way of automatically discovering topics that these sentences contain. For example, given these sentences and asked for 2 topics, LDA might produce something like:

- Sentences 1 and 2: 100% Topic A
- Sentences 3 and 4: 100% Topic B
- Sentence 5: 60% Topic A, 40% Topic B
- Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, ... (at which point, you could interpret topic A to be about food)
- Topic B: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, ... (at which point, you could interpret topic B to be about cute animals)

http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation

- “Topics” could be useful for things like searching for relevant documents.
Classic Approach: Latent Semantic Indexing

- Classic methods are based on scores like **TF-IDF**:
  1. **Term frequency**: probability of a word occurring within a document.
     - E.g., 7% of words in document $i$ are “the” and 2% of the words are “LeBron”.
  2. **Document frequency**: probability of a word occurring across documents.
     - E.g., 100% of documents contain “the” and 0.01% have “LeBron”.
  3. **TF-IDF**: measures like $(\text{term frequency}) \times \log \frac{1}{(\text{document frequency})}$.
     - Seeing “LeBron” tells you a lot about document, seeing ‘the” tells you nothing.

- TF-IDF features are **very redundant**.
  - Consider TF-IDF of “LeBron”, “Durant”, and “Kobe”.
  - High values of these typically just indicate topic of “basketball”.

- We want to find **latent factors** (“topics”) like “basketball”.

Modern Approach: Latent Dirichlet Allocation

- **Latent semantic indexing** (LSI) topic model:
  1. Summarize each document by its TF-IDF values.
  2. Run a latent-factor model like PCA or NMF on the matrix.
  3. Treat the latent factors as the “topics”.

- LSI has largely been replace by **latent Dirichlet allocation** (LDA).
  - Hierarchical Bayesian model of all words in a document.

- The most cited ML paper from the last 15 years?

- LDA has several components, we’ll build up to it by parts.
  - We’ll assume all documents have $d$ words and word order doesn't matter.
Model 1: Categorical Distribution of Words

- Base model: each word $x_j$ comes from a categorical distribution.

\[ p(x_j = \text{“the”}) = \theta_{\text{“the”}} \quad \text{where} \quad \theta_{\text{word}} \geq 0 \quad \text{and} \quad \sum_{\text{word}} \theta_{\text{word}} = 1. \]

- So to generate a document with $d$ words:
  - Sample $d$ words from the categorical distribution.

- Drawback: misses that documents are about different “topics”.
Model 2: Mixture of Categorical Distributions

- To represent “topics”, we’ll use a **mixture model**.
  - Each mixture has its own categorical distribution over words.
    - E.g., the “basketball” mixture will have higher probability of “LeBron”.

- So to generate a document with \( d \) words:
  - Sample a topic \( z \) from a categorical distribution.
  - Sample \( d \) word categorical distribution \( z \).

- Drawback: misses that documents may be about **more than one topics**.
Model 3: Multi-Topic Mixture of Categorical

- Our third model introduces a new vector of “topic proportions” $\pi$.
  - Gives percentage of each topic that makes up the document.
  - E.g., 80% basketball and 20% politics.
  - Called probabilistic latent semantic indexing (PLSI).

- So to generate a document with $d$ words given topic proportions $\pi$:
  - Sample $d$ topics $z_j$ from categorical distribution $\pi$.
  - Sample a word for each $z_j$ from corresponding categorical distribution.

- Drawback: how do we compute $\pi$ for a new document?
  - This is the same issue we had in our hospitals example.
Model 4: Latent Dirichlet Allocation

- **Latent Dirichlet allocation** (LDA) puts a prior on topic proportions.
  - Conjugate prior for categorical is **Dirichlet distribution**.

- So to generate a document with \(d\) words given Dirichlet prior:
  - Sample mixture proportions \(\pi\) from the Dirichlet prior.
  - Sample \(d\) topics \(z_j\) from categorical distribution \(\pi\).
  - Sample a word for each \(z_j\) from corresponding categorical distribution.
Latent Dirichlet Allocation (LDA)

Each topic is like a "principal component" or "latent factor"
Latent Dirichlet Allocation (LDA)

1. Sample topic proportions $\theta$ from Dirichlet.

Each topic is like a "principal component" or "latent factor"
1. Sample topic proportions $\Theta$ from Dirichlet.

2. Sample $d$ topics $z_j$ from $\Theta$.

Each topic is like a "principal component" or "latent factor"
Latent Dirichlet Allocation (LDA)

1. Sample topic proportions \( \Theta \) from Dirichlet.

2. Sample \( d \) topics \( z_j \) from \( \Theta \).

3. For each \( z_j \) sample a word based on frequencies for topic.

Each topic is like a "principal component" or "latent factor".
Figure 2: Real inference with LDA. We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left is the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.

Figure 3: A topic model fit to the *Yale Law Journal*. Here there are twenty topics (the top eight are plotted). Each topic is illustrated with its top most frequent words. Each word’s position along the x-axis denotes its specificity to the documents. For example “estate” in the first topic is more specific than “tax.”

Latent Dirichlet Allocation Example

Health topics in social media:

<table>
<thead>
<tr>
<th>Non-Alliment Topics</th>
<th>TV &amp; Movies</th>
<th>Games &amp; Sports</th>
<th>School</th>
<th>Conversation</th>
<th>Family</th>
<th>Transportation</th>
<th>Music</th>
</tr>
</thead>
<tbody>
<tr>
<td>watch</td>
<td>killing</td>
<td>ugh</td>
<td>ill</td>
<td>mom</td>
<td>home</td>
<td>voice</td>
<td>hear</td>
</tr>
<tr>
<td>watching</td>
<td>playing</td>
<td>school</td>
<td>ok</td>
<td>she</td>
<td>car</td>
<td>feelin</td>
<td>hear</td>
</tr>
<tr>
<td>tv</td>
<td>read</td>
<td>hahaha</td>
<td>ha</td>
<td>dad</td>
<td>drive</td>
<td>lil</td>
<td>feelin</td>
</tr>
<tr>
<td>killing</td>
<td>test</td>
<td>fine</td>
<td>says</td>
<td>hes</td>
<td>walk</td>
<td>night</td>
<td>feelin</td>
</tr>
<tr>
<td>movie</td>
<td>doing</td>
<td>yeah</td>
<td>sister</td>
<td>bit</td>
<td>bus</td>
<td>bit</td>
<td>night</td>
</tr>
<tr>
<td>seen</td>
<td>finish</td>
<td>thanks</td>
<td>tell</td>
<td>trip</td>
<td>driving</td>
<td>music</td>
<td>night</td>
</tr>
<tr>
<td>movies</td>
<td>flight</td>
<td>hey</td>
<td>mum</td>
<td>ride</td>
<td>house</td>
<td>listening</td>
<td>music</td>
</tr>
<tr>
<td>nr</td>
<td>reading</td>
<td>thats</td>
<td>brother</td>
<td>leave</td>
<td>listen</td>
<td>listening</td>
<td>listen</td>
</tr>
<tr>
<td>watched</td>
<td>teacher</td>
<td>xd</td>
<td>thinks</td>
<td>house</td>
<td>sound</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hi</td>
<td>write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ailments</th>
<th>Influenza-like Illness</th>
<th>Insomnia &amp; Sleep Issues</th>
<th>Diet &amp; Exercise</th>
<th>Cancer &amp; Serious Illness</th>
<th>Injuries &amp; Pain</th>
<th>Dental Health</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Words</td>
<td>better</td>
<td>night</td>
<td>body</td>
<td>cancer</td>
<td>hurts</td>
<td>dentist</td>
</tr>
<tr>
<td></td>
<td>hope</td>
<td>bed</td>
<td></td>
<td>pounds</td>
<td>knee</td>
<td>appointment</td>
</tr>
<tr>
<td></td>
<td>ill</td>
<td></td>
<td></td>
<td>gym</td>
<td>ankle</td>
<td>doctors</td>
</tr>
<tr>
<td></td>
<td>soon</td>
<td></td>
<td></td>
<td>weight</td>
<td>hurt</td>
<td>tooth</td>
</tr>
<tr>
<td></td>
<td>feel</td>
<td></td>
<td></td>
<td>lost</td>
<td>neck</td>
<td>teeth</td>
</tr>
<tr>
<td></td>
<td>feeling</td>
<td></td>
<td></td>
<td>workout</td>
<td>ouch</td>
<td></td>
</tr>
<tr>
<td></td>
<td>day</td>
<td></td>
<td></td>
<td>lose</td>
<td>leg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>flu</td>
<td></td>
<td></td>
<td>days</td>
<td>arm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>thanks</td>
<td></td>
<td></td>
<td>legs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td></td>
<td></td>
<td>week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symptoms</td>
<td>sick</td>
<td>sleep</td>
<td>sore</td>
<td>cancer</td>
<td>pain</td>
<td>infection</td>
</tr>
<tr>
<td></td>
<td>sore</td>
<td>headache</td>
<td>throat</td>
<td>breast</td>
<td>sore</td>
<td>pain</td>
</tr>
<tr>
<td></td>
<td>throat</td>
<td>fall</td>
<td>pain</td>
<td>lung</td>
<td>head</td>
<td>mouth</td>
</tr>
<tr>
<td></td>
<td>fever</td>
<td>insomnia</td>
<td>aching</td>
<td>prostate</td>
<td>foot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cough</td>
<td>sleeping</td>
<td>stomach</td>
<td>sad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Treatments</td>
<td>hospital</td>
<td>sleeping</td>
<td>exercise</td>
<td>surgery</td>
<td>massage</td>
<td>surgery</td>
</tr>
<tr>
<td></td>
<td>surgery</td>
<td>pills</td>
<td>diet</td>
<td>hospital</td>
<td>brace</td>
<td>braces</td>
</tr>
<tr>
<td></td>
<td>antibiotics</td>
<td>caffeine</td>
<td>dieting</td>
<td>treatment</td>
<td>physical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fluids</td>
<td>pill</td>
<td>exercises</td>
<td>heart</td>
<td>therapy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paracetamol</td>
<td>tylenol</td>
<td>protein</td>
<td>transplant</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0103408
Latent Dirichlet Allocation Example

Three topics in 100 years of “Vogue” fashion magazine:

http://dh.library.yale.edu/projects/vogue/topics/
Discussion of Topic Models

- There are many extensions of LDA:
  - We can put prior on the number of words (like Poisson).
  - Correlated and hierarchical topic models learn dependencies between topics.

Figure 2: A portion of the topic graph learned from 15,744 OCR articles from Science. Each node represents a topic, and is labeled with the five most probable words from its distribution; edges are labeled with the correlation between topics.

Discussion of Topic Models

- There are many extensions of LDA:
  - We can put prior on the number of words (like Poisson).
  - Correlated and hierarchical topic models learn dependencies between topics.
  - Can be combined with Markov models to capture dependencies over time.
Discussion of Topic Models

- There are *many* extensions of LDA:
  - We can put **prior on the number of words** (like Poisson).
  - **Correlated** and **hierarchical** topic models learn dependencies between topics.
  - Can be combined with **Markov models** to capture dependencies over time.
  - Recent work on better word representations like “word2vec” (bonus slides).
  - Now being applied **beyond text**, like “cancer mutation signatures”:

http://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1005657
Discussion of Topic Models

- Topic models for analyzing musical keys:

![Diagram of key profiles](http://cseweb.ucsd.edu/~dhu/docs/nips09_abstract.pdf)

Figure 2: The C major and C minor key-profiles learned by our model, as encoded by the $\beta$ matrix. Resulting key-profiles are obtained by transposition.

![Musical notations](http://cseweb.ucsd.edu/~dhu/docs/nips09_abstract.pdf)

Figure 3: Key judgments for the first 6 measures of Bach’s Prelude in C minor, WTC-II. Annotations for each measure show the top three keys (and relative strengths) chosen for each measure. The top set of three annotations are judgments from our LDA-based model; the bottom set of three are from human expert judgments [3].
Outline

1. Topic Models
2. Rejection and Importance Sampling
3. Metropolis-Hastings Algorithm
Overview of Bayesian Inference Tasks

- In **Bayesian** approach, we typically work with the posterior

\[ p(\theta \mid x) = \frac{1}{Z} p(x \mid \theta)p(\theta), \]

where \( Z \) makes the distribution sum/integrate to 1.

- Typically, we need to compute expectation of some \( f \) with respect to posterior,

\[ E[f(\theta)] = \int_{\theta} f(\theta)p(\theta \mid x)d\theta. \]

- Examples:
  - If \( f(\theta) = \theta \), we get **posterior mean** of \( \theta \).
  - If \( f(\theta) = p(\tilde{x} \mid \theta) \), we get **posterior predictive**.
  - If \( f(\theta) = \mathbb{I}(\theta \in S) \) we get probability of \( S \) (e.g., **marginals** or **conditionals**).
  - If \( f(\theta) = 1 \) and we use \( \tilde{p}(\theta \mid x) \), we get **marginal likelihood** \( Z \).
Need for Approximate Integration

- Bayesian models allow things that aren’t possible in other frameworks:
  - Optimize the regularizer (empirical Bayes).
  - Relax IID assumption (hierarchical Bayes).
  - Have clustering happen on multiple levels (topic models).

- But posterior often doesn’t have a closed-form expression.
  - We don’t just want to flip coins and multiply Gaussians.

- We once again need approximate inference:
  1. Variational methods.
  2. Monte Carlo methods.

- Classic ideas from statistical physics, that revolutionized Bayesian stats/ML.
Variational Inference vs. Monte Carlo

Two main strategies for approximate inference:

1. **Variational** methods:
   - Approximate $p$ with “closest” distribution $q$ from a tractable family,
     \[ p(x) \approx q(x). \]
   - Turns *inference into optimization* (need to find best $q$).
     - Called *variational Bayes*.

2. **Monte Carlo** methods:
   - Approximate $p$ with empirical distribution over samples,
     \[ p(x) \approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[x^i = x]. \]
   - Turns *inference into sampling*.
     - For Bayesian methods, we’ll typically need to *sample from posterior*. 
Conjugate Graphical Models: Ancestral and Gibbs Sampling

- For **conjugate DAGs**, we can use **ancestral sampling** for unconditional sampling.

- **Examples:**
  - For LDA, sample $\pi$ then sample the $z_j$ then sample the $x_j$.
  - For HMMs, sample the hidden $z_j$ then sample the $x_j$.

- We can also often use **Gibbs sampling** as an **approximate sampler**.
  - If **neighbours are conjugate** in UGMs.
  - To generate conditional samples in conjugate DAGs.

- However, **without conjugacy our inverse transform trick doesn’t work**.
  - We can’t even sample from the 1D conditionals with this method.
Beyond Inverse Transform and Conjugacy

- We want to use simple distributions to sample from complex distributions.
- Two common strategies are rejection sampling and importance sampling.
- We’ve previously seen rejection sampling to do conditional sampling:
  - Example: sampling from a Gaussian subject to $x \in [-1, 1]$.
  - Generate unconditional samples, throw out the ones that aren’t in $[-1, 1]$. 
General Rejection Sampling Algorithm

Want to sample from complicated target $\gamma(x)$. 
We can sample from \( g(x) \)

Want to sample from complicated target \( \gamma(x) \).
General Rejection Sampling Algorithm

We can sample from $g(x)$

$g(x)$ times $M$ such that $Mg(x) \geq \hat{p}(x)$ for all $x$. 

Want to sample from complicated target $\hat{p}(x)$. 
General Rejection Sampling Algorithm

We can sample from \( g(x) \)

\( g(x) \) times \( 'M' \) such that \( Mg(x) \geq \hat{p}(x) \) for all \( x \).

Want to sample from complicated target \( \hat{p}(x) \).

Sample from \( g(x) \)
General Rejection Sampling Algorithm

We can sample from \( g(x) \)

Accept if random sample from \([0, Mg(x)]\) is less than \( \hat{p}(x) \)

\( g(x) \) times \( M \) such that

\( Mg(x) \geq \hat{p}(x) \) for all \( x \)

Want to sample from complicated target \( \hat{p}(x) \)

Sample from \( g(x) \)
General Rejection Sampling Algorithm

We can sample from \( q(x) \).

Accept if random sample from \([0, M q(x)]\)

is less than \( \hat{\pi}(x) \).

\( x \) sample from \( q(x) \).

\( g(x) \) times \( 'M' \) such that \( M q(x) \geq \hat{\pi}(x) \) for all \( x \).

Want to sample from complicated target \( \hat{\pi}(x) \).
General Rejection Sampling Algorithm

We can sample from $q(x)$.

Accept if random sample from $[0, Mq(x)]$ is less than $\tilde{p}(x)$.

Sample likely to be accepted.

$\tilde{p}(x)$ times 'M' such that $Mq(x) \geq \tilde{p}(x)$ for all $x$.

Want to sample from complicated target $\tilde{p}(x)$. 
General Rejection Sampling Algorithm

We can sample from $g(x)$

Sample likely to be rejected.

Accept if random sample from $[0, M g(x)]$ is less than $\hat{p}(x)$.

$g(x)$ times $'M'$ such that $M g(x) \geq \hat{p}(x)$ for all $x$.

Want to sample from complicated target $\hat{p}(x)$.

Sample likely to be accepted.
General Rejection Sampling Algorithm

Ingredients of a more general rejection sampling algorithm:

1. Ability to evaluate unnormalized \( \tilde{p}(x) \),

\[
p(x) = \frac{\tilde{p}(x)}{Z}.
\]

2. A distribution \( q \) that is easy to sample from.

3. An upper bound \( M \) on \( \tilde{p}(x)/q(x) \).

Rejection sampling algorithm:

1. Sample \( x \) from \( q(x) \).
2. Sample \( u \) from \( U(0, 1) \).
3. Keep the sample if \( u \leq \frac{\tilde{p}(x)}{Mq(x)} \).

The accepted samples will be from \( p(x) \).
General Rejection Sampling Algorithm

- We can use general rejection sampling for:
  - Sample from Gaussian $q$ to sample from student t.
  - Sample from prior to sample from posterior ($M = 1$),
  
  $$p(\theta | x) = p(x | \theta) p(\theta).$$

- Drawbacks:
  - You may reject a large number of samples.
    - Most samples are rejected for high-dimensional complex distributions.
  - You need to know $M$.

- Extension in 1D for convex $- \log p(x)$:
  - Adaptive rejection sampling refines piecewise-linear $q$ after each rejection.
Importance Sampling

- **Importance sampling** is a variation that accepts all samples.
  - Key idea is similar to EM,

\[
\mathbb{E}_p[f(x)] = \sum_x p(x)f(x) \\
= \sum_x q(x) \frac{p(x)f(x)}{q(x)} \\
= \mathbb{E}_q \left[ \frac{p(x)}{q(x)} f(x) \right],
\]

and similarly for continuous distributions.

- We can sample from \( q \) but reweight by \( p(x)/q(x) \) to sample from \( p \).
- Only assumption is that \( q \) is non-zero when \( p \) is non-zero.
- If you only know unnormalized \( \tilde{p}(x) \), a variant gives approximation of \( Z \).
Importance Sampling

- As with rejection sampling, only efficient if \( q \) is close to \( p \).
- Otherwise, weights will be huge for a small number of samples.
  - Even though unbiased, variance will be huge.

- Can be problematic if \( q \) has lighter “tails” than \( p \):
  - You rarely sample the tails, so those samples get huge weights.

- As with rejection sampling, doesn’t tend to work well in high dimensions.
Outline

1. Topic Models
2. Rejection and Importance Sampling
3. Metropolis-Hastings Algorithm
Limitations of Simple Monte Carlo Methods

- The basic ingredients of our previous sampling methods:
  - Inverse CDF, rejection sampling, importance sampling.
  - Sampling in higher-dimensions: ancestral sampling, Gibbs sampling.

- These work well in low dimensions or for posteriors with analytic properties.

- But we want to solve high-dimensional integration problems in other settings:
  - Deep belief networks and Boltzmann machines.
  - Bayesian graphical models and Bayesian neural networks.
  - Hierarchical Bayesian models.

- Our previous methods tend not to work in complex situations:
  - Inverse CDF may not be available.
  - Conditionals needed for ancestral/Gibbs sampling may be hard to compute.
  - Rejection sampling tends to reject almost all samples.
  - Importance sampling tends to give almost zero weight to all samples.
Dependent-Sample Monte Carlo Methods

- We want an algorithm whose samples get better over time.

- Two main strategies for generating dependent samples:
  - **Sequential Monte Carlo:**
    - Importance sampling where proposal $q_t$ changes over time from simple to posterior.
    - “Particle Filter Explained without Equations”:
      https://www.youtube.com/watch?v=aUkBa1zMKv4
    - AKA sequential importance sampling, annealed importance sampling, particle filter.
  
  - **Markov chain Monte Carlo (MCMC).**
    - Design Markov chain whose stationary distribution is the posterior.

- These are the main tools to sample from high-dimensional distributions.
Markov Chain Monte Carlo

- We’ve previously discussed Markov chain Monte Carlo (MCMC).
  1. Based on generating samples from a Markov chain \( q \).
  2. Designed so stationary distribution \( \pi \) of \( q \) is target distribution \( p \).

- If we run the chain long enough, it gives us samples from \( p \).

- **Gibbs sampling** is an example of an MCMC method.
  - Sample \( x_j \) conditioned on all other variables \( x_{-j} \).

- Note that before we were sampling states according to a UGM, now we’re sampling parameters according to the posterior.
Limitations of Gibbs Sampling

- Gibbs sampling is nice because it has no parameters:
  - You just need to decide on the blocks and figure out the conditionals.

- But it isn’t always ideal:
  - Samples can be very correlated: slow progress.
  - Conditionals may not have a nice form:
    - If Markov blanket is not conjugate, need rejection/importance sampling.

- Generalization that can address these is Metropolis-Hastings:
  - Oldest algorithm among the “10 Best of the 20th Century”.
Warm-Up to Metropolis-Hastings: “Stupid MCMC”

Consider finding the expected value of a fair die:
- For a 6-sided die, the expected value is 3.5.

Consider the following “stupid MCMC” algorithm:
- Start with some initial value, like “4”.
- At each step, roll the die and generate a random number $u$:
  - If $u < 0.5$, “accept” the roll and take the roll as the next sample.
  - Otherwise, “reject” the roll and take the old value (“4”) as the next sample.
Warm-Up to Metropolis-Hastings: “Stupid MCMC”

- Example:
  - Start with “4”, so record “4”.
  - Roll a 6 and generate 0.234, so record 6.
  - Roll a 3 and generate 0.612, so record 6.
  - Roll a 2 and generate 0.523, so record 6.
  - Roll a 3 and generate 0.125, so record 3.

- So our samples are 4, 6, 6, 6, 3, …
  - If you run this long enough, you will spend $1/6$ of the time on each number.
  - So the dependent samples from this Markov chain could be used within Monte Carlo.

- It’s stupid since you should just accept every sample (they’re IID samples).
  - It works but it’s twice as slow.
The Metropolis algorithm for sampling from a continuous target \( p(x) \):

Start from some \( x^0 \) and on iteration \( t \):

1. Add zero-mean Gaussian noise to \( x^t \) to generate \( \tilde{x}^t \).
2. Generate \( u \) from a \( U(0, 1) \).
3. Accept the sample and set \( x^{t+1} = \tilde{x}^t \) if

\[
    u \leq \frac{\tilde{p}(\tilde{x}^t)}{\tilde{p}(x^t)},
\]

and otherwise reject the sample and set \( x^{t+1} = x^t \).

A random walk, but sometimes rejecting steps that decrease probability:

- A valid MCMC algorithm on continuous densities, but convergence may be slow.
- You can implement this even if you don’t know normalizing constant.
Metropolis Algorithm in Action

\[ M = 0.615, 0.398; \ N_{\text{pro}} = 1000, \ \frac{N_{\text{acc}}}{N_{\text{pro}}} = 0.39 \]
Metropolis Algorithm Analysis

- Markov chain with transitions $q_{ss'} = q(x^t = s' | x^{t-1} = s)$ is reversible if
  \[ \pi(s)q_{ss'} = \pi(s')q_{s's}, \]
  for some distribution $\pi$ (this condition is called detailed balance).

- Assuming we reach stationary, reversibility implies $\pi$ is stationary distribution.
  - By summing reversibility condition over all $s$ values we get
    \[ \sum_s \pi(s)q_{ss'} = \sum_s \pi(s')q_{s's} \]
    \[ \sum_s \pi(s)q_{ss'} = \pi(s') \sum_s q_{s's} = \pi(s') \]
    \[ \sum_s \pi(s)q_{ss'} = \pi(s') \]
    (stationary condition).

- Metropolis is reversible (bonus slide) so has correct stationary distribution.
Metropolis-Hastings

- **Metropolis-Hastings** algorithms allows general proposal distribution \( q \):
  - Value \( q(\tilde{x}^t \mid x^t) \) is probability of proposing \( \tilde{x}^t \).
  - Metropolis algorithm is special case where \( q \) is zero-mean Gaussian.

- It **accepts** a proposed \( \tilde{x}^t \) if
  \[
  u \leq \frac{\tilde{p}(\tilde{x}^t)q(x^t \mid \tilde{x}^t)}{\tilde{p}(x^t)q(\tilde{x}^t \mid x^t)},
  \]
  where extra terms ensure reversibility for asymmetric \( q \):
    - E.g., if you are more likely to propose to go from \( x^t \) to \( \tilde{x}^t \) than the reverse.

- This again works under very weak conditions, such as \( q(\tilde{x}^t \mid x^t) > 0 \).
- **Gibbs sampling** is a special case, but it’s often not the best choice:
  - You can make performance much better/worse with an appropriate \( q \).
Metropolis-Hastings

Metropolis-Hastings for sampling from mixture of Gaussians:

- With a random walk $q$ we may get stuck in one mode.
- We could have proposal be mixture between random walk and “mode jumping”.

[Diagram showing 1000 iterations]

Simple choices for proposal distribution $q$:
- Metropolis originally used random walks: $x^t = x^{t-1} + \epsilon$ for $\epsilon \sim \mathcal{N}(0, \Sigma)$.
- Hastings originally used independent proposal: $q(x^t \mid x^{t-1}) = q(x^t)$.
- Gibbs sampling updates single variable based on conditional:
  - In this case the acceptance rate is 1 so we never reject.
- Mixture model for $q$: e.g., between big and small moves.
- “Adaptive MCMC”: tries to update $q$ as we go: needs to be done carefully.
- “Particle MCMC”: use particle filter to make proposal.

Unlike rejection sampling, we don’t want acceptance rate as high as possible:
- High acceptance rate may mean we’re not moving very much.
- Low acceptance rate definitely means we’re not moving very much.
- Designing $q$ is an “art”.
Advanced Monte Carlo Methods

Some other more-powerful MCMC methods:

- **Block Gibbs sampling** improves over single-variable Gibb sampling.

- **Collapsed Gibbs sampling (Rao-Blackwellization)**: integrate out variables that are not of interest.
  - E.g., integrate out hidden states in Bayesian hidden Markov model.
  - E.g., integrate over different components in topic models.
  - Provably decreases variance of sampler (if you can do it, you should do it).

- **Auxiliary-variable sampling**: introduce variables to sample bigger blocks:
  - E.g., introduce $z$ variables in mixture models.
  - Also used in Bayesian logistic regression (beginning with Albert and Chib).
Advanced Monte Carlo Methods

- **Trans-dimensional MCMC:**
  - Needed when *dimensionality of problem can change* on different iterations.
  - Most important application is probably Bayesian feature selection.

- **Hamiltonian Monte Carlo:**
  - Faster-converging method based on Hamiltonian dynamics.
  - I think Alex will discuss this next time.

- **Population MCMC:**
  - Run multiple MCMC methods, each having different “move” size.
  - Large moves do exploration and small moves refine good estimates.

- **Combinations of variational inference and stochastic methods:**
  - **Variational MCMC:** Metropolis-Hastings where variational $q$ can make proposals.
  - **Stochastic variational inference (SVI):** variational methods using stochastic gradient.
Summary

- **Relaxing IID** assumption with hierarchical Bayes.
- **Latent Dirichlet allocation**: factor/topic model for discrete data like text.
- **Rejection sampling**: generate exact samples from complicated distributions.
- **Importance sampling**: reweights samples from the wrong distribution.
- **Markov chain Monte Carlo** generates a sequence of dependent samples:
  - But asymptotically these samples come from the posterior.
- **Metropolis-Hastings** allows arbitrary “proposals”.
  - With good proposals works much better than Gibbs sampling.

- Guest lecture by Alex Bouchard, then next week npBayes/variational/VAE/GANs.
Metropolis Algorithm Analysis

- Metropolis algorithm has \( q_{ss'} > 0 \) (sufficient to guarantee stationary distribution is unique and we reach it) and satisfies detailed balance with target distribution \( p \),

\[
p(s)q_{ss'} = p(s')q_{s's}.
\]

- We can show this by defining transition probabilities

\[
q_{ss'} = \min \left\{ 1, \frac{\tilde{p}(s')}{\tilde{p}(s)} \right\},
\]

and observing that

\[
p(s)q_{ss'} = p(s) \min \left\{ 1, \frac{\tilde{p}(s')}{\tilde{p}(s)} \right\} = p(s) \min \left\{ 1, \frac{1}{Z\tilde{p}(s')} \right\}
\]

\[
= p(s) \min \left\{ 1, \frac{p(s')}{p(s)} \right\} = \min \left\{ p(s), p(s') \right\}
\]

\[
= p(s') \min \left\{ 1, \frac{p(s)}{p(s')} \right\} = p(s')q_{s's}.
\]
Latent-Factor Representation of Words

- In natural language, we often represent words by an index.
  - E.g., “cat” is word 124056 among a “bag of words”.

- But this may be inefficient:
  - Should “cat” and “kitten” share parameters in some way?

- We want a latent-factor representation of words.
  - Closeness in latent space should indicate similarity, distances could represent meaning?

- We could use PCA, LDA, and so on.
- But recent “word2vec” approach is getting a lot of popularity...
Word2Vec

- Two variations of word2vec:
  1. Try to predict word from surrounding words ("continuous bag of words").
  2. Try to predict surrounding words from word ("skip-gram").

![Diagram of CBOW and Skip-gram](https://arxiv.org/pdf/1301.3781.pdf)

- Train latent-factors to solve one of these supervised learning tasks.
In both cases, each word $i$ is represented by a vector $z^i$.

We optimize likelihood of word vectors $z^i$ under the model

$$p(x^i \mid x^j) \propto \exp((z^i)^T z^j),$$

and we usually assume everything is independent while training.
- Apply gradient descent to minimize NLL as usual.

- In **CBOW**, denominator sums over all words.
- In **skip-grams**, denominator sums over all possible surround words.
  - Common trick to speed things up:
    - Hierarchical softmax.
    - Negative sampling (sample terms in denominator).
MDS visualization of a set of related words.

Distances between vectors might represent semantic relationships.

http://sebastianruder.com/secret-word2vec
**Bonus Slide: Word2Vec**

- Subtracting word vectors to find related words:

  ![Table of word pairs](https://arxiv.org/pdf/1301.3781.pdf)

  **Table 8:** Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).

- Word vectors for 157 languages:
What about homonyms and polysemy? 
  - The word vectors would need to account for all meanings.

More recent approaches:
  - Try to cluster the different context where words appear.
  - Use different vectors for different contexts.
Multiple Word Prototypes

http://www.socher.org/index.php/Main/ImprovingWordRepresentationsViaGlobalContextAndMultipleWordPrototypes