CPSC 540: Machine Learning

Topic Models

Mark Schmidt

University of British Columbia

Winter 2019

Last Time: Empirical Bayes and Hierarchical Bayes

- 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,

$$\underset{\alpha,\beta}{\text{argmax}}\, p(x \mid \alpha, \beta) = \underset{\alpha,\beta}{\text{argmax}} \int_\theta p(x \mid \theta)p(\theta \mid \alpha, \beta)d\theta.$$

  - Can be used to optimize λ_j, polynomial degree, RBF σ_i, polynomial vs. RBF, etc.
- We also considered hierarchical Bayes, where you put a prior on the prior,

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

  - Further protection against overfitting, and can be used to model non-IID data.

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 occuring 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 occuring across documents.
     - E.g., 100% of documents contain "the" and 0.01% have "LeBron".
  3. TF-IDF: measures like (term frequency)*log 1/(document frequency).
     - Seeing "LeBron" tells you a lot about document, seeing 'the" tells you nothing.

- Many many many variations exist.

- TF-IDF features are very redundant.
  - Consider TF-IDF of "LeBron", "Durant", and "Kobe".
  - High values of these typically just indicate topic of "basketball".
  - Basically a weighted bag of words.

- 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.
    - Still ignores word order.
    - Tries to explain all words in terms of topics.

- The most cited ML paper in the 00s?

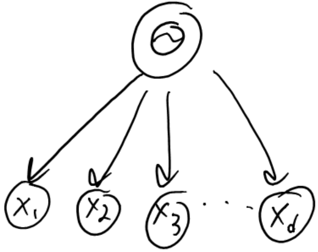
- LDA has several components, we'll build up to it by parts.
  - We'll assume all documents have d words and word order doesn't matter.

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.

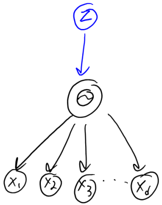


- Drawback: misses that dcouments are about different “topics”.
  - We want the word distribution to depend on the “topics”.

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".

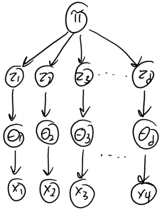
- So to generate a document with d words:
  - Sample a topic z from a categorical distribution.
  - Sample d word categorical distribution z.



- 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" π.
  - 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 π:
  - Sample d topics z_j from categorical distribution π.
  - Sample a word for each z_j from corresponding categorical distribution.

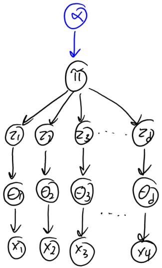


- Drawback: how do we compute π for a new document?
  - There is no generative model of π in this model.

Model 4: Latent Dirichlet Allocation

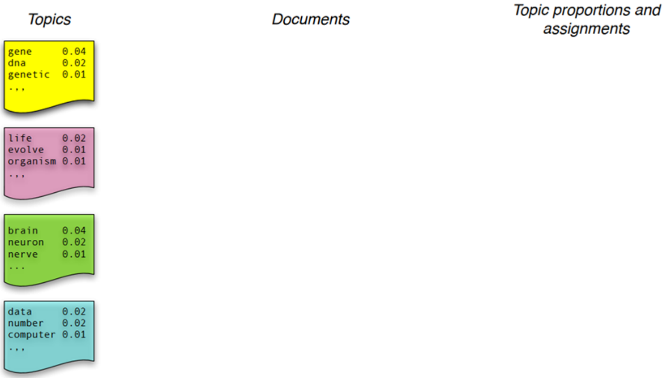
- Latent Dirichlet allocation (LDA) puts a prior on topic proportions.
  - Conjugate prior for categorical is Dirichlet distribution.

- So to generate a document with d words given Dirichlet prior:
  - Sample mixture proportions π from the Dirichlet prior.
  - Sample d topics z_j from categorical distribution π.
  - Sample a word for each z_j from corresponding categorical distribution.



- This is the generative model, typically fit with MCMC or variational methods.

Latent Dirichlet Allocation (LDA)



Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation (LDA)

1. Sample topic proportions θ from Dirichlet.

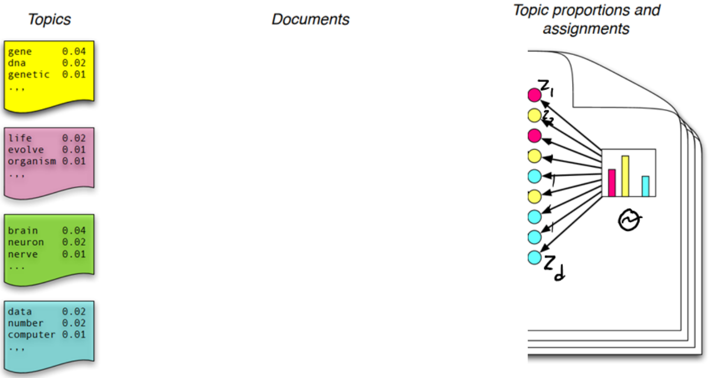
Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation (LDA)

1. Sample topic proportions Θ from Dirichlet.

2. Sample 'd' topics z_j from Θ.

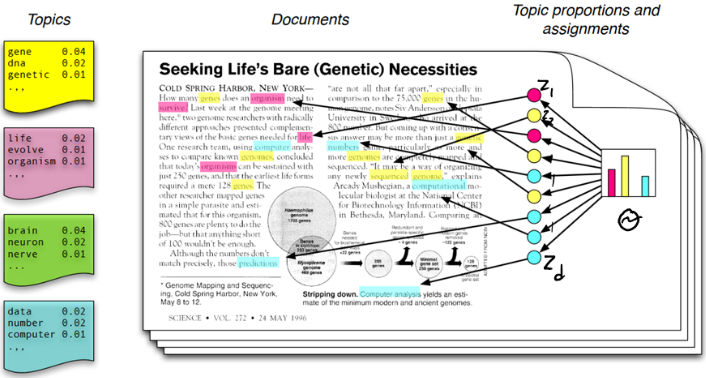


Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation (LDA)

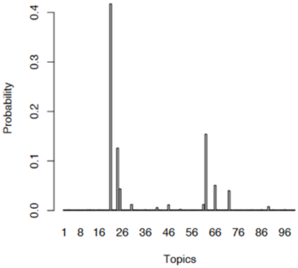
1. Sample topic proportions Θ from Dirichlet.
2. Sample 'd' topics z_j from Θ.
3. For each z_j sample a word based on frequencies for topic.



Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation Example



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

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.

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation Example

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."

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation Example

Health topics in social media:

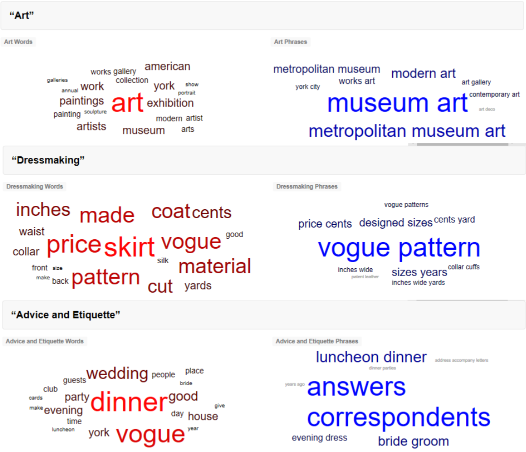
Non-Ailment Topics						
TV & Movies	**Games & Sports**	**School**	**Conversation**	**Family**	**Transportation**	**Music**
watch	killing	ugh	ill	mom	home	voice
watching	play	class	ok	shes	car	hear
tv	game	school	haha	dad	drive	feelin
killing	playing	read	ha	says	walk	lil
movie	win	test	fine	hes	bus	night
seen	boys	doing	yeah	sister	driving	bit
movies	games	finish	thanks	tell	trip	music
mr	fight	reading	hey	mum	ride	listening
watched	lost	teacher	thats	brother	leave	listen
hi	team	write	xd	thinks	house	sound

Ailments						
	Influenza-like Illness	**Insomnia & Sleep Issues**	**Diet & Exercise**	**Cancer & Serious Illness**	**Injuries & Pain**	**Dental Health**
General Words	better	night	body	cancer	hurts	dentist
	hope	bed	pounds	help	knee	appointment
	ill	body	gym	pray	ankle	doctors
	soon	ill	weight	awareness	hurt	tooth
	feel	tired	lost	diagnosed	neck	teeth
	feeling	work	workout	prayers	ouch	appt
	day	day	lose	died	leg	wisdom
	flu	hours	days	family	arm	eye
	thanks	asleep	legs	friend	fell	going
	xx	morning	week	shes	left	went
Symptoms	sick	sleep	sore	cancer	pain	infection
	sore	headache	throat	breast	sore	pain
	throat	fall	pain	lung	head	mouth
	fever	insomnia	aching	prostate	foot	ear
	cough	sleeping	stomach	sad	feet	sinus
Treatments	hospital	sleeping	exercise	surgery	massage	surgery
	surgery	pills	diet	hospital	brace	braces
	antibiotics	caffeine	dieting	treatment	physical	antibiotics
	fluids	pill	exercises	heart	therapy	eye
	paracetamol	tylenol	protein	transplant	crutches	hospital

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.

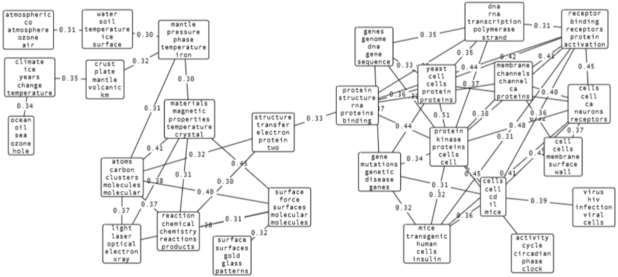
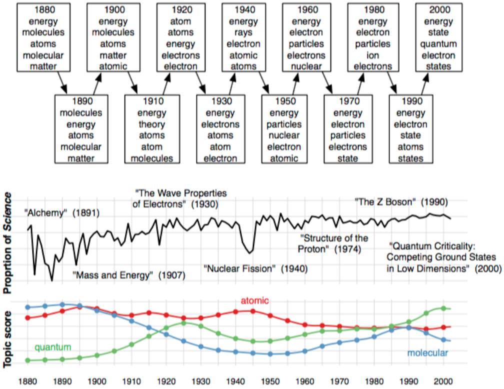


Figure 2: A portion of the topic graph learned from 15,744 OCR articles from *Science*. Each node represents a topic, and is labeled with the five most probable words from its distribution; edges are labeled with the correlation between topics.

http://people.ee.duke.edu/~lcarin/Blei2005CTM.pdf

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.



http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

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" (340, 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:

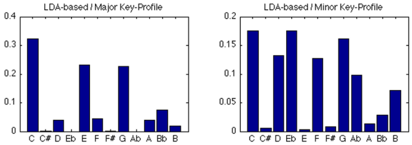


Figure 2: The C major and C minor key-profiles learned by our model, as encoded by the β matrix. Resulting key-profiles are obtained by transposition.

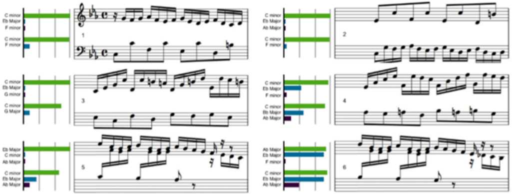


Figure 3: Key judgments for the first 6 measures of Bach's Prelude in C minor, WTC-II. Annotations for each measure show the top three keys (and relative strengths) chosen for each measure. The top set of three annotations are judgments from our LDA-based model; the bottom set of three are from human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

http://cseweb.ucsd.edu/~dhu/docs/nips09_abstract.pdf

Monte Carlo Methods for Topic Models

- Nasty integrals in topic models:

Inference [edit]

See also: Dirichlet-multinomial distribution

Learning the various distributions (the set of topics, their associated word probabilities, the topic of each word, and the particular topic mixture of each document) is a problem of Bayesian inference. The original paper used a variational Bayes approximation of the posterior distribution;[1] alternative inference techniques use Gibbs sampling[6] and expectation propagation.[7]

Following is the derivation of the equations for collapsed Gibbs sampling, which means φs and θs will be integrated out. For simplicity, in this derivation the documents are all assumed to have the same length N. The derivation is equally valid if the document lengths vary.

According to the model, the total probability of the model is:

$$P(\boldsymbol{W},\boldsymbol{Z},\boldsymbol{\theta},\boldsymbol{\varphi};\alpha,\beta)=\prod_{i=1}^{K}P(\varphi_i;\beta)\prod_{j=1}^{M}P(\theta_j;\alpha)\prod_{t=1}^{N}P(Z_{j,t}|\theta_j)P(W_{j,t}|\varphi_{Z_{j,t}}),$$

where the bold-font variables denote the vector version of the variables. First, $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ need to be integrated out.

$$\begin{aligned}P(\boldsymbol{Z},\boldsymbol{W};\alpha,\beta)&=\int_{\boldsymbol{\theta}}\int_{\boldsymbol{\varphi}}P(\boldsymbol{W},\boldsymbol{Z},\boldsymbol{\theta},\boldsymbol{\varphi};\alpha,\beta)\,d\boldsymbol{\varphi}\,d\boldsymbol{\theta}\\&=\int_{\boldsymbol{\varphi}}\prod_{i=1}^{K}P(\varphi_i;\beta)\prod_{j=1}^{M}\prod_{t=1}^{N}P(W_{j,t}\mid\varphi_{Z_{j,t}})\,d\boldsymbol{\varphi}\int_{\boldsymbol{\theta}}\prod_{j=1}^{M}P(\theta_j;\alpha)\prod_{t=1}^{N}P(Z_{j,t}\mid\theta_j)\,d\boldsymbol{\theta}.\end{aligned}$$

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

Monte Carlo Methods for Topic Models

- How do we actually *use* Monte Carlo for topic models?

- First we write out the posterior:

$$p(\underbrace{Z}_{\text{topics}}, \underbrace{\pi}_{\text{word prob.}}, \overset{\text{topic prop.}}{\theta} \mid \underbrace{X}_{\substack{\text{data}\\ \text{(words)}}}, \underbrace{\alpha}_{\substack{\text{prior}\\ \text{on topic}\\ \text{proportions}}}, \underbrace{\beta}_{\substack{\text{prior on}\\ \text{word}\\ \text{probabilities}}}) = \left[\prod_{i=1}^{n} \underbrace{p(\theta^i \mid \alpha)}_{\substack{\text{topic proportion}\\ \text{probability}\\ \text{(document 'i')}}} \prod_{j=1}^{d} \underbrace{p(z_j^i \mid \theta^i)}_{\substack{\text{topic probability}\\ \text{(topic at position}\\ \text{'j' in document 'i')}}} \underbrace{p(x_j^i \mid z_j^i, \pi_j)}_{\substack{\text{word probability}\\ \text{(word at position}\\ \text{'j' in document 'i')}}}\right] \left[\prod_{c=1}^{K} \underbrace{p(\pi_c \mid \beta)}_{\substack{\text{word probability}\\ \text{parameters}\\ \text{(topic 'c')}}}\right]$$

Monte Carlo Methods for Topic Models

- How do we actually *use* Monte Carlo for topic models?

- Next we generate samples from the posterior:
  - With Gibbs sampling we alternate between:
    - Sampling topics given word probabilities and topic proportions.
    - Sampling topic proportions given topics and prior parameters α.
    - Sampling word probabilities given topics, words, and prior parameters β.

  - Have a burn-in period, use thinning, try to monitor convergence, etc.

- Finally, we use posterior samples to do inference:
  - Distribution of topic proportions for sample i is frequency in samples.
  - To see if words come from same topic, check frequency in samples.

Outline

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 θ.
  - 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 leves (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,
   $$p(x) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}[x^i = x].$$
   - 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 π 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.

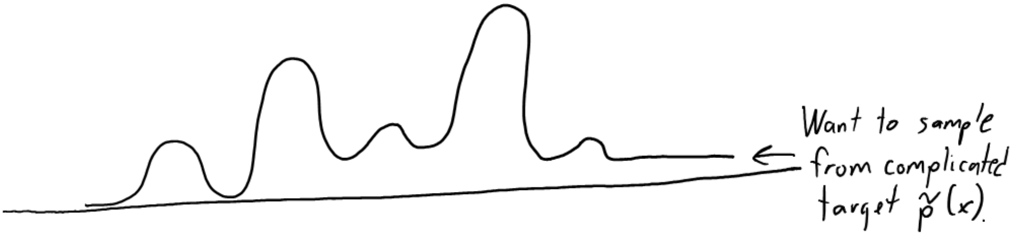
Beyond Inverse Transform and Conjugacy

- We want to use simple distributions to sample from complex distributions.
- Two common strategies are rejection sampling and importance sampling.

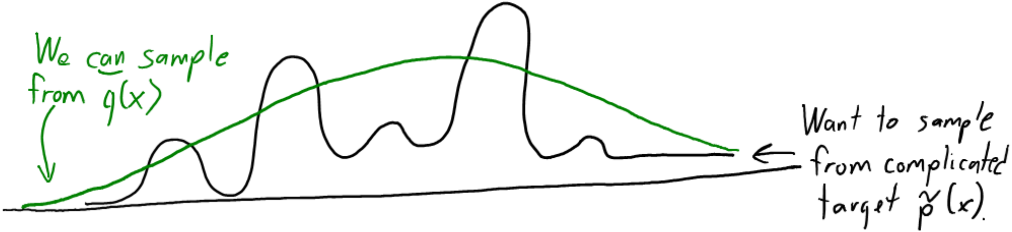
- We've previously seen rejection sampling to do conditional sampling:
  - Example: sampling from a Gaussian subject to $x \in [-1, 1]$.

  - Generate unconditional samples, throw out the ones that aren't in $[-1, 1]$.

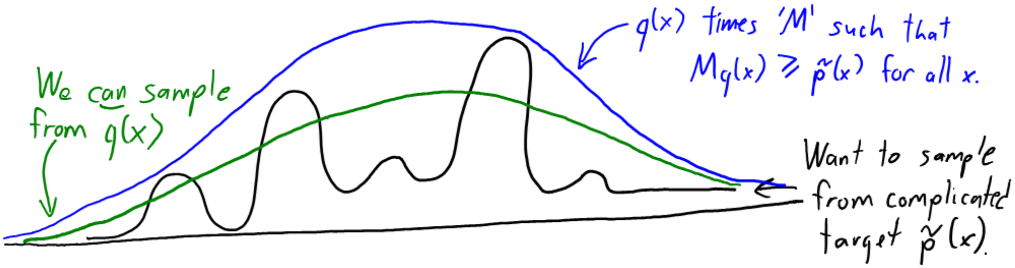
General Rejection Sampling Algorithm



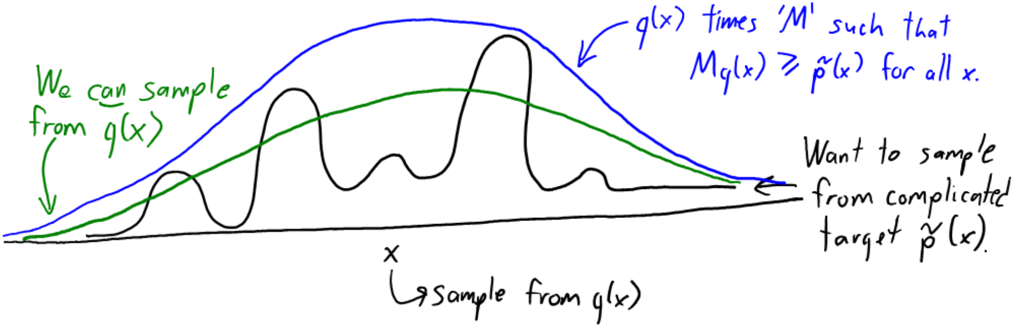
General Rejection Sampling Algorithm



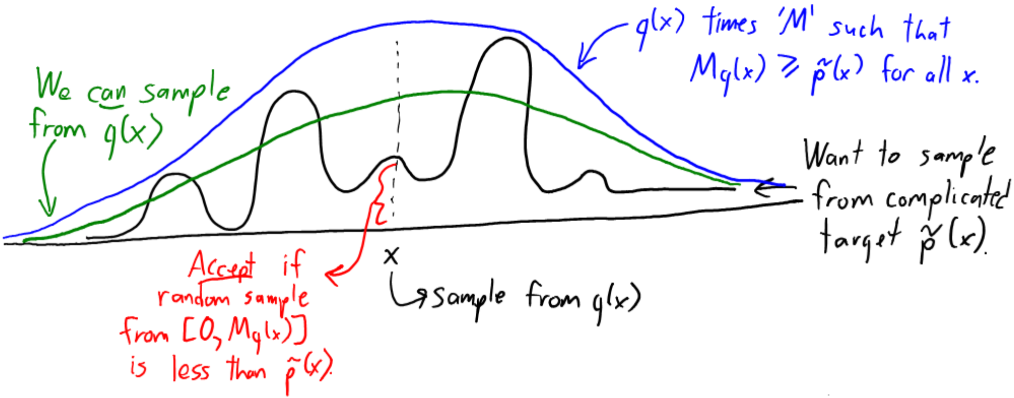
General Rejection Sampling Algorithm



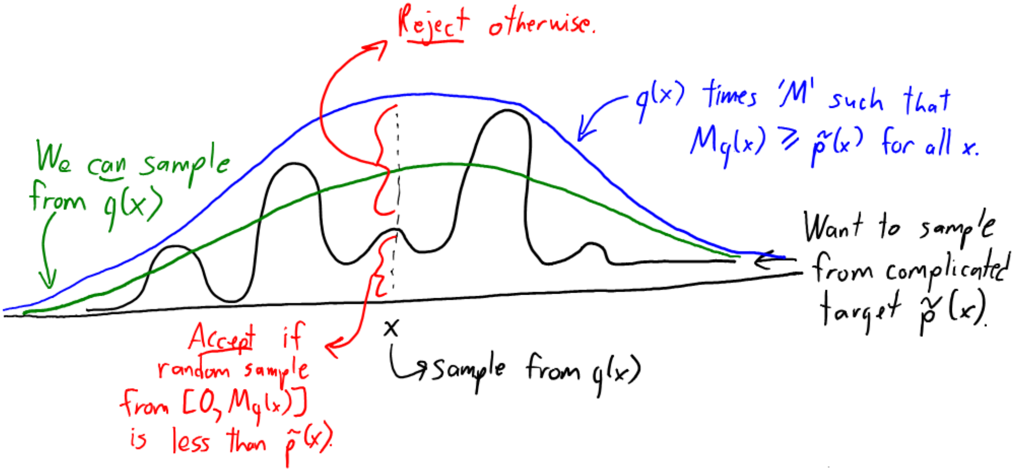
General Rejection Sampling Algorithm



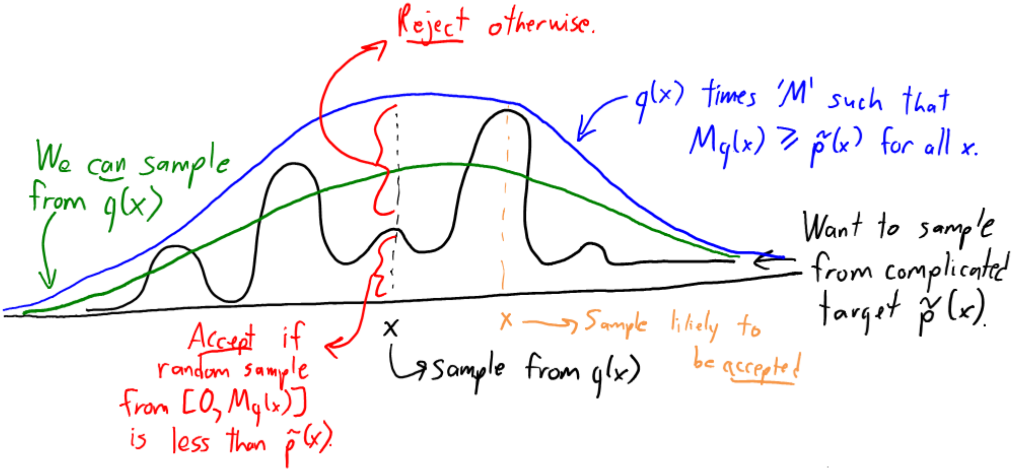
General Rejection Sampling Algorithm



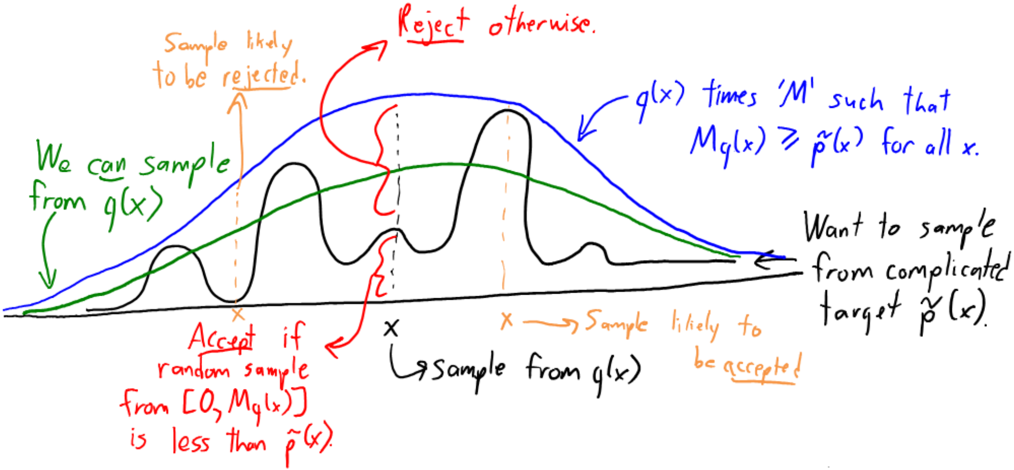
General Rejection Sampling Algorithm



General Rejection Sampling Algorithm



General Rejection Sampling Algorithm



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 $\mathcal{U}(0,1)$.
  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 $(M = 1)$,

$$\tilde{p}(\theta \mid x) = \underbrace{p(x \mid \theta)}_{\leq 1} p(\theta).$$

- 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,

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

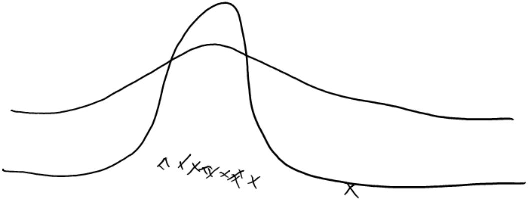
    and similarly for continuous distributions.

    - 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 can 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.
  - Though there is room to cleverly design q, like using mixtures.

Summary

- 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.
- Back to MCMC, and variational methods.

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 innefficient:
  - 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...

Using Context

- Consider these phrases:
  - “The *cat* purred”.
  - “The *kitten* purred”.

  - “black *cat* ran”.
  - “black *kitten* ran”
- Words that occur in the same context likely have similar meanings.
- Word2vec uses this insight to design an MDS distance function.

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").

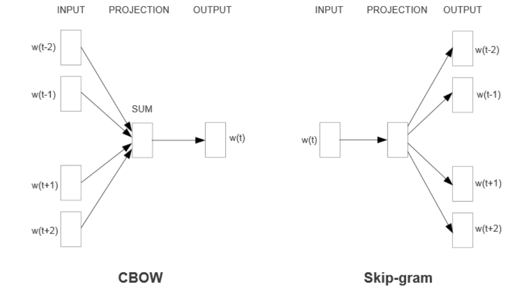


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

https://arxiv.org/pdf/1301.3781.pdf

- Train latent-factors to solve one of these supervised learning tasks.

Word2Vec

- In both cases, each word i is represented by a vector z^i.
- We optimize likelihood of word vectors z^i under the model

$$p(x_i \mid x_{\text{nei}}) = \prod_{j \in \text{nei}} p(x_i \mid x_j), \quad p(x_i \mid x_j) \propto \frac{\exp(\langle z^i, z^j \rangle)}{\sum_{c=1}^{k} \exp(\langle z^c, z^j \rangle)}.$$

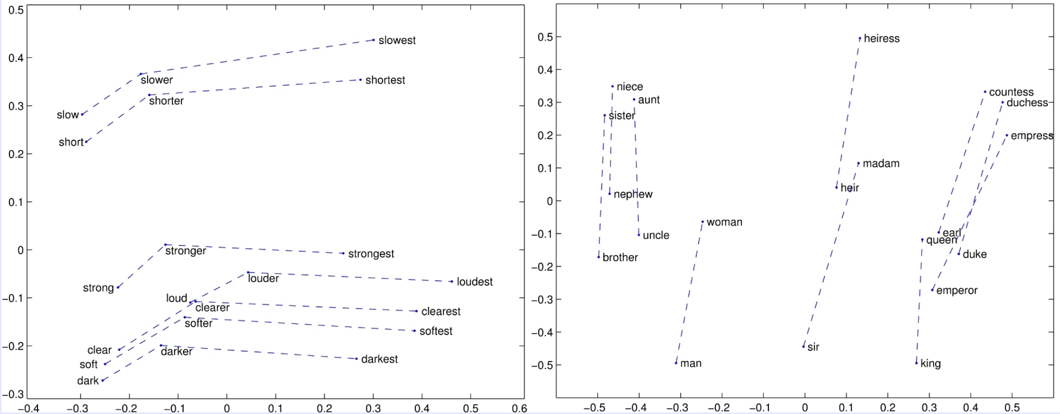
which is making a strong independence assumption.

- Apply gradient descent to NLL as usual:
  - Encourages $\langle z^i, z^j \rangle$ to be big for words in same context (making z^i close to z^j).
  - Encourages $\langle z^i, z^j \rangle$ to be small for words not appearing in same context.

- 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).

Bonus Slide: Word2Vec

MDS visualization of a set of related words.



http://sebastianruder.com/secret-word2vec

Distances between vectors might represent semantic relationships.

Bonus Slide: Word2Vec

- Subtracting word vectors to find related words:

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).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Table 8 shows words that follow various relationships. We follow the approach described above: the relationship is defined by subtracting two word vectors, and the result is added to another word. Thus for example, *Paris* - *France* + *Italy* = *Rome*. As it can be seen, accuracy is quite good, although

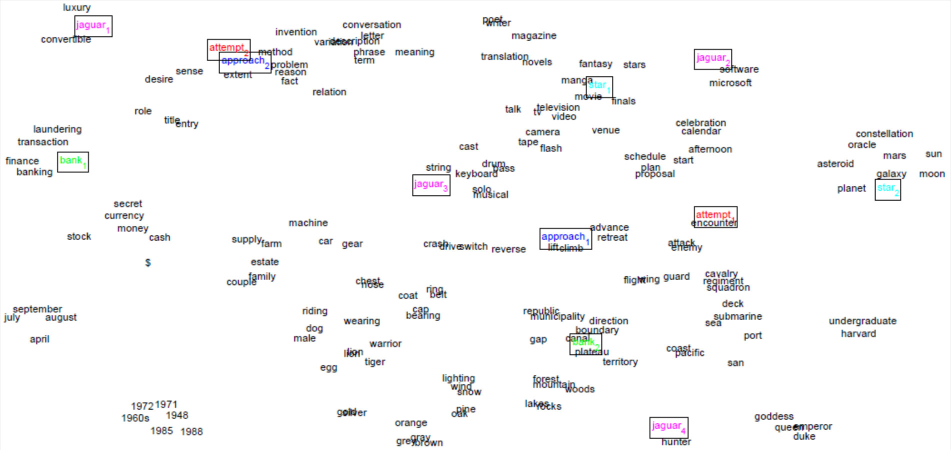
https://arxiv.org/pdf/1301.3781.pdf

- Word vectors for 157 languages:
  - `https://fasttext.cc/docs/en/crawl-vectors.html`

Multiple Word Prototypes

- 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