Gibbs Sampling for LDA

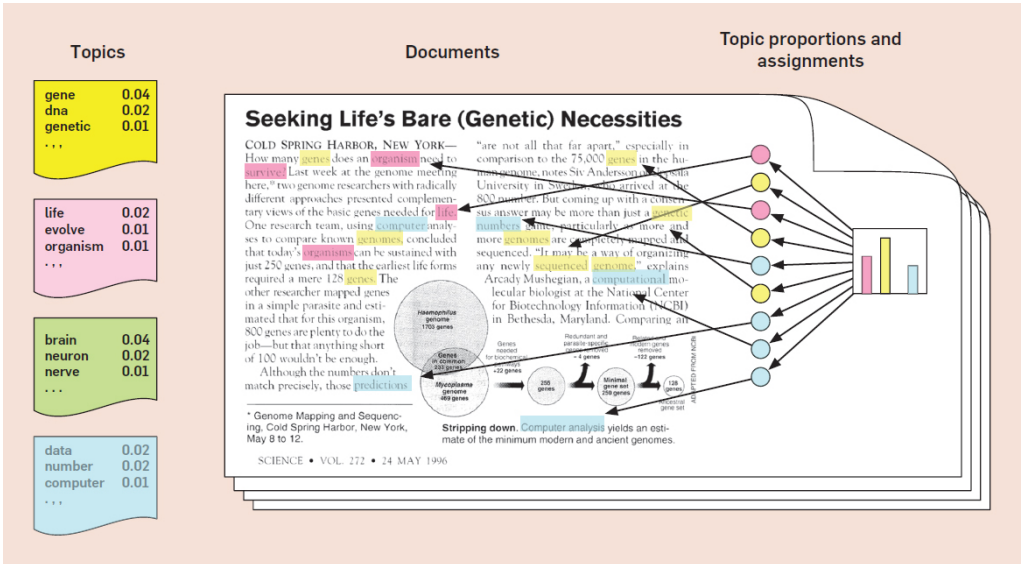
David Mareček

November 02, 2022

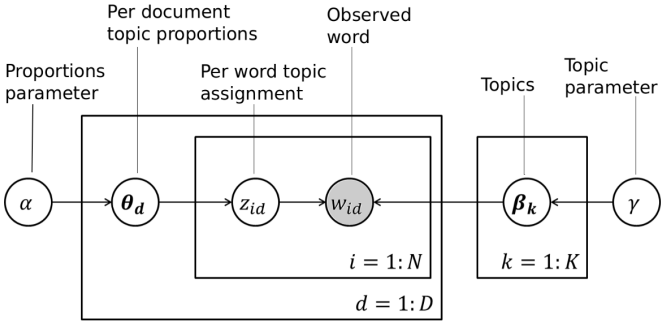

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Many of the slides in this presentation were taken from the presentations of Carl Edward Rasmussen (University of Cambridge)

Latent Dirichlet Allocation



Latent Dirichlet Allocation: generative model



1. For each document d draw a distribution $\vec{\theta}_d$ over topics from a $Dir(\vec{\alpha})$.
2. For each topic k draw a distribution $\vec{\beta}_k$ over words from a $Dir(\vec{\gamma})$.
3. Draw a topic z_{nd} for the n-th word in document d from a $Cat(\vec{\theta}_d)$.
4. Draw word $w_n d$ from a $Cat(\vec{\beta}_{z_{nd}})$.

In LDA, every word in a document can be drawn from a different topic and every document has its own distribution over topics $\vec{\theta}_d$.

The intractability of LDA

The probability of everything is:

$$p(\vec{\beta}, \vec{\theta}, z, w|\vec{\gamma}, \vec{\alpha}) = \prod_{k=1}^{K} p(\vec{\beta}_k|\vec{\gamma}) \prod_{d=1}^{D} [p(\vec{\theta}_d|\vec{\alpha}) \prod_{n=1}^{N_d} (p(z_{nd}|\vec{\theta}_d) p(w_{nd}|\vec{\beta}, z_{nd})]$$

Learning involves computing the posterior over the parameters, $\vec{\beta}$ and $\vec{\theta}$ given the words w, but this would require the marginalization of the latent veriables z_{nd}.
How many configurations are there?
The evidence (normalising constant of the posterior):

$$p(w) = \iint \sum_{z} \prod_{d=1}^{D} \prod_{k=1}^{K} \prod_{n=1}^{N_d} p(z_{nd}|\vec{\theta}_d) p(\vec{\theta}_d|\vec{\alpha}) p(w_{nd}|\vec{\beta}, z_{nd} p(\vec{\beta}_k|\vec{\gamma}) d\vec{\beta}_k d\vec{\theta}_d$$

We need to average over all possible set of values of all z_{nd}. If every document had N words, this means N^K configurations per document.

Expectation-Maximization vs. Gibbs sampling

The posterior is intractable because there are too many possible latent z_{nd}.

Expectation-Maximization: We would need to keep and update probabilities of all possible configurations in the memory.
Intractable.

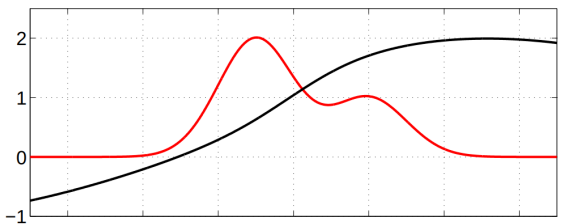
Gibbs Sampling:
We need to keep only current values of latent variables and parameters, and use a specific algorithm to update them and converge to good solutions.

Introduction to Gibbs Sampling

How do we do integrals withe respect to an intractable posterior?
Approximate expectations of a function $\Phi(x)$ with respect to probability $p(x)$:

$$\mathbb{E}_{p(x)}\Phi(x) = \int \Phi(x)p(x)dx,$$

when these are not analytically tractable, $x \in \mathbb{R}^D$ and $D \gg 1$
Assume that we can evaluate $\Phi(x)$ and $p(x)$.



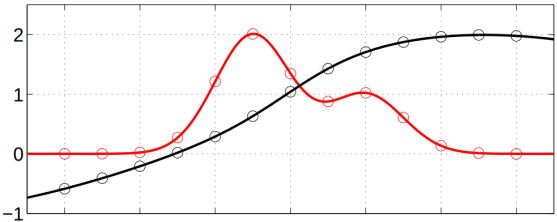
Introduction to Gibbs Sampling

Numerical integration on a grid

Approximate the integral by a sum of products

$$\int \Phi(x)p(x)dx \simeq \sum_{\tau=1}^{T} \Phi(x^{(\tau)})p(x^{(\tau)})\Delta x,$$

where $x^{(\tau)}$ lie on an equidistant grid (or fancier versions of this).



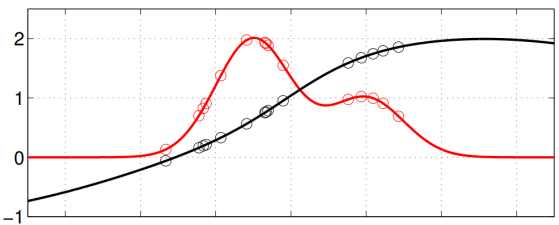
Problem: the number of grid points required is k^D. It grows exponentially with the dimension D. Practicable only to $D = 4$ or so.

Introduction to Gibbs Sampling

Monte Carlo: The fundamental basis for Monte Carlo approximations is

$$\mathbb{E}_{p(x)} \simeq \hat{\Phi} = \frac{1}{T} \sum_{\tau=1}^{T} \Phi(x^{(\tau)}),$$

where $x^{(\tau)}$ are samples from the distribution $p(x)$.



Under mild conditions, $\hat{\Phi} \to \mathbb{E}(\Phi(x))$ as $T \to \infty$. For moderate T, $\hat{\Phi}$ may still be a good approximation. The variance of the estimation depends only on the number of samples (T) and is independent if the dimension D of x.

Introduction to Gibbs Sampling

This is great, but how do we generate random samples from $p(x)$?
This is hard for a higher number of dimensions.

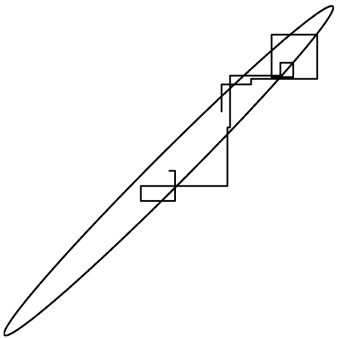
Gibbs Sampling
For each component i from the vector of variables x in turn we sample a new value from the conditional distribution of x_i given all other variables:

$$x_i' \sim p(x_i|x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_D).$$

It can be shown, that this will eventually generate dependent samples from the joint distribution $p(x)$.
Gibbs sampling reduces the task of sampling from a joint distribution, to sampling from a sequence of univariate conditional distributions.

Introduction to Gibbs Sampling

Example: 20 iterations of Gibbs sampling on a bivariate Gaussian distribution



See also `https://www.youtube.com/watch?v=AEwY6QXWoUg`

Notice that strong correlations can slow down Gibbs sampling.

Introduction to Gibbs Sampling

Gibbs sampling is a parameter free algorithm, applicable if we know how to sample from the conditional distributions.

Main disadvantage: depending on the target distribution, there may be very strong correlations between consecutive samples.

To get less dependence, Gibbs sampling is often run for a long time, and the samples are thinned by keeping only every 10th or 100th sample.

Burn-in: often, the initial sequence of samples is discarded, until the chain has converged to the desired distribution. What does convergence mean in this context?

It is often challenging to judge the effective correlation length of a Gibbs sampler.
Sometimes several Gibbs samplers are run from different starting points, to compare results.

Collapsed Gibbs sampling for Latent Dirichlet Allocation (LDA)

For LDA, we will sample only the latent variables z_{nd}.

The other variables (parameters $\vec{\theta}$ and $\vec{\beta}$) will be marginalized (integrated out).

We need to compute probability of a single latent variable z_{nd} (assignment of one particular word to a topic) given all other latent variables $z_{-nd} = \{z_{n'd'} : n' \neq n, d' \neq d\}$ (assignments of all other words) and hyperparameters $\vec{\alpha}$ and $\vec{\gamma}$.

We will use Bayes theorem:

$$p(z_{nd} = k|\{w\}, \{z_{-nd}\}, \vec{\gamma}, \vec{\alpha}) = \frac{p(z_{nd} = k|\{z_{-nd}\}, \vec{\gamma}, \vec{\alpha})\ p(\{w\}|z_{nd} = k, \{z_{-nd}\}, \vec{\gamma}, \vec{\alpha})}{p(\{w\}|\{z_{-nd}\}, \vec{\gamma}, \vec{\alpha})}$$

The denominator is constant with respect to z_{nd}; generation of topics does not depend on $\vec{\gamma}$; generation of words for given topic does not depend on $\vec{\gamma}$)

$$p(z_{nd} = k|\{w\}, \{z_{-nd}\}, \vec{\gamma}, \vec{\alpha}) \propto p(z_{nd} = k|\{z_{-nd}\}, \vec{\alpha})\ p(\{w\}|z_{nd} = k, \{z_{-nd}\}, \vec{\gamma})$$

Collapsed Gibbs sampling for Latent Dirichlet Allocation (LDA)

We have:

$$p(z_{nd} = k|\{w\}, \{z_{-nd}\}, \vec{\gamma}, \vec{\alpha}) \propto p(z_{nd} = k|\{z_{-nd}\}, \vec{\alpha})\ p(\{w\}|z_{nd} = k, \{z_{-nd}\}, \vec{\gamma})$$

Probability of data $p(w)$ can be rewritten as $p(w_{nd}|w_{-nd})p(w_{-nd})$ and $p(w_{-nd})$ is constant with respect to z_{nd})

For each predictive distribution, we integrate over all possible parameters $\vec{\beta}_k$ and $\vec{\theta}_d$.
These integrals can be easily computed; see predictive distribution for Dirichlet posteriors.

$$\begin{aligned} p(z_{nd} = k|\{w\}, \{z_{-nd}\}, \vec{\gamma}, \vec{\alpha}) \propto \\ &\propto p(z_{nd} = k|\{z_{-nd}\}, \vec{\alpha})\ p(w_{nd}|\{w_{-nd}\}, z_{nd} = k, \{z_{-nd}\}, \vec{\gamma}) \\ &\propto \int p(z_{nd} = k|\vec{\theta}_d)p(\vec{\theta}_d|z_{-nd}, \vec{\alpha})d\vec{\theta}_d \int p(w_{nd}|\vec{\beta}_k)p(\vec{\beta}_k|\{w_{-nd}\}, \{z_{-nd}\}, \vec{\gamma})d\vec{\beta}_k \end{aligned}$$

Collapsed Gibbs sampling for Latent Dirichlet Allocation (LDA)

We use the formula for the expected value for the Dirichlet distribution:

$$
\begin{aligned}
p(z_{nd} = k | \{w\}, \{z_{-nd}\}, \vec{\gamma}, \vec{\alpha}) &\propto \\
&\propto \int p(z_{nd} = k | \vec{\theta}_d) p(\vec{\theta}_d | z_{-nd}, \vec{\alpha}) d\vec{\theta}_d \int p(w_{nd} | \vec{\beta}_k) p(\vec{\beta}_k | \{w_{-nd}\}, \{z_{-nd}\}, \vec{\gamma}) d\vec{\beta}_k \\
&= \frac{\alpha + c_d[d][k]}{K\alpha + N_d - 1} \; \frac{\gamma + c_w[w_{nd}][k]}{M\gamma + \sum_{m=1}^{M} c_w[m][k]}
\end{aligned}
$$

Where:

- $c_d[d][k]$ = how many words in document d are assigned to topic k.
- $c_w[m][k]$ = how many times the word m is assigned to topic k (across all documents).

The current position z_{nd} is always excluded from the counts.

LDA Algorithm

initialize z_{nd} *randomly* $\forall d \in 1..D,\ \forall n \in 1..N_d$;
compute initial counts $c_d[d][k]$, $c_w[m][k]$, $c[k]$ $\forall d \in 1..D,\ \forall k \in 1..K,\ \forall m \in 1..M$;
for $i \leftarrow 1$ **to** I **do**
  for $d \leftarrow 1$ **to** D **do**
    for $n \leftarrow 1$ **to** N_d **do**
      $c_d[d][z_{nd}]{-}{-}$; $c_w[w_{nd}][z_{nd}]{-}{-}$; $c[z_{nd}]{-}{-}$;
      for $k \leftarrow 1$ **to** K **do**
        $p[k] = \frac{\alpha + c_d[d][k]}{K\alpha + N_d - 1} \ \frac{\gamma + c_w[w_{nd}][k]}{M\gamma + c[k]}$;
      end
      sample k *from probability distribution* $p[k]$;
      $z_{nd} \leftarrow k$;
      $c_d[d][k]{+}{+}$; $c_w[w_{nd}][k]{+}{+}$; $c[k]{+}{+}$;
    end
  end
end

LDA algorithm: example

Suppose we have only three documents using only 5 different words $\{a, b, c, d, e\}$ and we set the number of topics to $K = 2$ and hyperparemeters $\alpha = 0.2$, $\gamma = 0.1$.

Let's have the following documents and folowing topics assigned to their words:

$$\begin{aligned} D_1 &= a, a, b, a, c, & D_2 &= d, c, e, \mathbf{d}, c, & D_3 &= d, d, e, a, a. \\ z_1 &= 1, 2, 1, 2, 1, & z_2 &= 2, 2, 1, \mathbf{1}, 2, & z_3 &= 1, 1, 2, 2, 2. \end{aligned}$$

Let's pick e.g. the fourth word in the second document (w_{24}) and compute the probability distribution across topics given all other words in the collection.

$$p(z_{24} = 1) \propto \frac{\alpha + 1}{2\alpha + 4} \cdot \frac{\gamma + 2}{5\gamma + 7} = 0.072 \propto 0.45$$

$$p(z_{24} = 2) \propto \frac{\alpha + 3}{2\alpha + 4} \cdot \frac{\gamma + 1}{5\gamma + 8} = 0.088 \propto 0.55$$

So the new topic z_{24} sampled for the word w_{24} will be topic 2 with probability 55% and topic 1 with probability 45%.

LDA Algorithm - topics assignment on a new data

initialize z_{nd} *randomly* $\forall d \in 1..D,\ \forall n \in 1..N_d$;
fix the counts $c_w[m][k]$ *and* $c[k]$ *obtained during training*;
compute initial counts $c_d[d][k]\ \forall d \in 1..D,\ \forall k \in 1..K$;
for $i \leftarrow 1$ **to** I **do**
  for $d \leftarrow 1$ **to** D **do**
    for $n \leftarrow 1$ **to** N_d **do**
      $c_d[d][z_{nd}]--$;
      for $k \leftarrow 1$ **to** K **do**
        $p[k] = \frac{\alpha + c_d[d][k]}{K\alpha + N_d - 1} \ \frac{\gamma + c_w[w_{nd}][k]}{M\gamma + c[k]}$;
      end
      sample k *from probability distribution* $p[k]$;
      $z_{nd} \leftarrow k$;
      $c_d[d][k]++$;
    end
  end
end

Entropy of text

- joint probability of the text T: $p(T) = \prod_{i=1}^{N} p(w_i) = \prod_{m=1}^{M} p(m)^{c_m}$
- log probability of T: $\log\, p(T) = \sum_{i=1}^{N} \log p(w_i) = \sum_{m=1}^{M} c_m \log p(m)$
- entropy of T: $H(T) = -\frac{1}{N} \sum_{i=1}^{N} \log p(w_i) = -\sum_{m=1}^{M} \frac{c_m}{N} \log p(m) = \frac{-\log p(T)}{N}$
- perplexity of T: $PP(T) = 2^{H(T)}$

A perplexity of g corresponds to the uncertainity associated with a die with g sides, which generates each new word.

Note: All the logarithms used here are binary (with base 2)

Word entropy of a topic in LDA

Probability of word w given a topic k is

$$p(w|k) = \frac{\gamma + c_w[w][k]}{M\gamma + \sum_{m=1}^{M} c_w[m][k]},$$

where the counts c_w are taken from the training data and M is the size of the vocabulary. The entropy of a topic is computed as follows:

$$H(k) = -\sum_{w=1}^{M} p(w|k) \log_2 p(w|k)$$

Perplexity is $PP(k) = 2^{H(k)}$.

Perplexity of the LDA model on the test data

Probability of word w in document d is

$$p(w|d) = \sum_{k=1}^{K} p(w|k)p(k|d) = \sum_{k=1}^{K} \frac{\gamma + c_w[w][k]}{M\gamma + \sum c_w[m][k]} \frac{\alpha + c_d[d][k]}{K\alpha + N_d},$$

where the counts c_w are taken from the training data, and counts c_d and N_d are taken from the test data.

The entropy is computed as the average of the log probabilities over all words in the test data.

$$H = -\frac{1}{N_{test}} \sum_{d=1}^{D_{test}} \sum_{n=1}^{N_d} \log_2 p(w_{nd}),$$

where N_{test} is the total number of words in the test data. Perplexity is $PP = 2^H$.

Note: To make it more properly, we should take into account more training iterations, not only the final counts c_w. However, the results would be very similar, because of averaging over the whole data.

Perplexity of the simple model without using topics

Probability of word w in the test data given the training data is

$$p(w) = \frac{\gamma + c_w[w]}{M\gamma + \sum c_w[m]}$$

where the counts c_w are taken from the training data.
The entropy is computed as the average of the log probabilities over all words in the test data.

$$H = -\frac{1}{N_{test}} \sum_{d=1}^{D_{test}} \sum_{n=1}^{N_d} \log_2 p(w_{nd}),$$

where N_{test} is the total number of words in the test data. Perplexity is $PP = 2^H$.