GibbsLDA++

A C/C++ Implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling for Parameter Estimation and Inference

http://gibbslda.sourceforge.net/

Xuan-Hieu Phan

hieuxuan at ecei dot tohoku dot ac dot jp or pxhieu at gmail dot com
Graduate School of Information Sciences
Tohoku University

---

1. Introduction
    1.1. Description
    1.2. News, Comments, and Bug Reports
    1.3. License
2. Compile GibbsLDA++
    2.1. Download
    2.2. Compiling
3. How to Use GibbsLDA++
    3.1. Command Line & Input Parameters
        3.1.1. Parameter Estimation from Scratch
        3.1.2. Parameter Estimation from a Previously Estimated Model
        3.1.3. Inference for Previously Unseen (New) Data
    3.2 Input Data Format

3.3. Outputs
    3.3.1. Outputs of Gibbs Sampling Estimation of GibbsLDA++
    3.3.2. Outputs of Gibbs Sampling Inference for Previously Unseen Data
3.4. Case Study
4. Links, Acknowledgements, and References

1. Introduction

1.1. Description

GibbsLDA++ is a C/C++ implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling technique for parameter estimation and inference. It is very fast and is designed to analyze hidden/latent topic structures of large-scale datasets including very large collections of text/Web documents. LDA was first introduced by David Blei et al [Blei03]. There have been several implementations of this model in C (using Variational Methods), Java, and Matlab. We decided to release this implementation of LDA in C/C++ using Gibbs Sampling to provide an alternative choice to the topic-model community.

GibbsLDA++ is useful for the following potential application areas:

- Information Retrieval (analyzing semantic/latent topic/concept structures of large text collection for a more intelligent information search.
- Document Classification/Clustering, Document Summarization, and Text/Web Data Mining community in general.
- Collaborative Filtering
- Content-based Image Clustering, Object Recognition, and other applications of Computer Vision in general.
- Other potential applications in biological data.

1.2. News, Comments, and Bug Reports.

We highly appreciate any suggestion, comment, and bug report.

1.3. License

2. Compile GibbsLDA++

2.1. Download

You can find and download document, source code, and case studies of GibbsLDA++ at http://sourceforge.net/projects/gibbslda

Here are some other tools developed by the same author:

- FlexCRFs: Flexible Conditional Random Fields
- CRFTagger: CRF English POS Chunker
- CRFChunker: CRF English Phrase Chunker
- JTextPro: A Java-based Text Processing Toolkit
- JWebPro: A Java-based Web Processing Toolkit
- JVnSegmenter: A Java-based Vietnamese Word Segmentation Tool

2.2. Compiling

On Unix/Linux/Cygwin/MinGW environments:

- System requirements:

  + A C/C++ compiler and the STL library. In the `Makefile`, we use `g++` as the default compiler command, if the C/C++ compiler on your system has another name (e.g., `cc, cpp, CC, CPP`, etc.), you can modify the `CC` variable in the `Makefile` in order to use `make` utility smoothly.

  + The computational time of GibbsLDA++ much depends on the size of input data, the CPU speed, and the memory size. If your dataset is quite large (e.g., larger than

100,000 documents or so), it is better to train GibbsLDA++ on a minimum of 2GHz CPU, 1Gb RAM system.

- Untar and unzip GibbsLDA++:

  ```
  $ gunzip GibbsLDA++.tar.gz

  $ tar -xf GibbsLDA++.tar
  ```

- Go to the home directory of GibbsLDA++ (i.e., GibbsLDA++ directory), type:

  ```
  $ make clean

  $ make all
  ```

3. How to Use GibbsLDA++

3.1. Command Line & Input Parameters

After compiling GibbsLDA++, we have an executable file `lda` in the `GibbsLDA++/src` directory. We use this for parameter estimation and inference for new data.

3.1.1. Parameter Estimation from Scratch

```
$ lda -est [-alpha <double>] [-beta <double>] [-ntopics
<int>] [-niters <int>] [-savestep <int>] [-twords <int>] -
dfile <string>
```

in which (parameters in [ ] are optional):

- `-est`: Estimate the LDA model from scratch
- `-alpha <double>`: The value of alpha, hyper-parameter of LDA. The default value of alpha is 50 / K (K is the the number of topics). See [Griffiths04] for a detailed discussion of choosing alpha and beta values.
- `-beta <double>`: The value of beta, also the hyper-parameter of LDA. Its default value is 0.1
- `-ntopics <int>`: The number of topics. Its default value is 100. This depends on the input dataset. See [Griffiths04] and [Blei03] for a more careful discussion of selecting the number of topics.

- `-niters <int>`: The number of Gibbs sampling iterations. The default value is 2000.
- `-savestep <int>`: The step (counted by the number of Gibbs sampling iterations) at which the LDA model is saved to hard disk. The default value is 200.
- `-twords <int>`: The number of most likely words for each topic. The default value is zero. If you set this parameter a value larger than zero, e.g., 20, GibbsLDA++ will print out the list of top 20 most likely words per each topic each time it save the model to hard disk according to the parameter `savestep` above.
- `-dfile <string>`: The input training data file. See Section 3.2 for a description of input data format.

3.1.2. Parameter Estimation from a Previously Estimated Model

```
$ lda -estc -dir <string> -model <string> [-niters <int>] -
savestep <int>] [-twords <int>]
```

in which (parameters in [ ] are optional):

- `-estc`: Continue to estimate the model from a previously estimated model.
- `-dir <string>`: The directory contain the previously estimated model
- `-model <string>`: The name of the previously estimated model. See Section 3.3 to know how GibbsLDA++ saves outputs on hard disk.
- `-niters <int>`: The number of Gibbs sampling iterations to continue estimating. The default value is 2000.
- `-savestep <int>`: The step (counted by the number of Gibbs sampling iterations) at which the LDA model is saved to hard disk. The default value is 200.
- `-twords <int>`: The number of most likely words for each topic. The default value is zero. If you set this parameter a value larger than zero, e.g., 20, GibbsLDA++ will print out the list of top 20 most likely words per each topic each time it save the model to hard disk according to the parameter `savestep` above.

3.1.3. Inference for Previously Unseen (New) Data

```
$ lda -inf -dir <string> -model <string> [-niters <int>] [-
twords <int>] -dfile <string>
```

in which (parameters in [ ] are optional):

- `-inf`: Do inference for previously unseen (new) data using a previously estimated LDA model.
- `-dir <string>`: The directory contain the previously estimated model
- `-model <string>`: The name of the previously estimated model. See Section 3.3 to know how GibbsLDA++ saves outputs on hard disk.
- `-niters <int>`: The number of Gibbs sampling iterations for inference. The default value is 20.
- `-twords <int>`: The number of most likely words for each topic of the new data. The default value is zero. If you set this parameter a value larger than zero, e.g., 20, GibbsLDA++ will print out the list of top 20 most likely words per each topic after inference.
- `-dfile <string>`:The file containing new data. See Section 3.2 for a description of input data format.

3.2 Input Data Format

Both data for training/estimating the model and new data (i.e., previously unseen data) have the same format as follows:

$$\begin{array}{l}\texttt{[M]}\\ \texttt{[document}_1\texttt{]}\\ \texttt{[document}_2\texttt{]}\\ \texttt{...}\\ \texttt{[document}_M\texttt{]}\end{array}$$

in which the first line is the total number for documents `[M]`. Each line after that is one document. $\texttt{[document}_i\texttt{]}$ is the $\texttt{i}^{\texttt{th}}$ document of the dataset that consists of a list of $\texttt{N}_\texttt{i}$ words/ terms.

$$\texttt{[document}_i\texttt{] = [word}_{i1}\texttt{] [word}_{i2}\texttt{] ... [word}_{iNi}\texttt{]}$$

in which all $\texttt{[word}_{ij}\texttt{] (i=1..M, j=1..N}_i\texttt{)}$ are text strings and they are separated by the blank character.

Note that the terms *document* and *word* here are abstract and should not only be understood as normal text documents. This is because LDA can be used to discover the underlying topic structures of any kind of discrete data. Therefore, GibbsLDA++ is not limited to text and natural language processing but can also be applied to other kinds of data like images and biological sequences. Also, keep in mind that for text/Web data collections, we should first preprocess the data (e.g., removing stop words and rare words, stemming, etc.) before estimating with GibbsLDA++.

3.3. Outputs

3.3.1. Outputs of Gibbs Sampling Estimation of GibbsLDA++

Outputs of Gibbs sampling estimation of GibbsLDA++ include the following files:

```
<model_name>.others
<model_name>.phi
<model_name>.theta
<model_name>.tassign
<model_name>.twords
```

in which:

- `<model_name>`: is the name of a LDA model corresponding to the time step it was saved on the hard disk. For example, the name of the model was saved at the Gibbs sampling iteration 400th will be `model-00400`. Similarly, the model was saved at the 1200th iteration is `model-01200`. The model name of the last Gibbs sampling iteration is `model-final`.
- `<model_name>.others`: This file contains some parameters of LDA model, such as:

  ```
  alpha=?
  beta=?
  ntopics=? # i.e., number of topics
  ndocs=? # i.e., number of documents
  nwords=? # i.e., the vocabulary size
  liter=? # i.e., the Gibbs sampling iteration at which the model was saved
  ```

- `<model_name>.phi`: This file contains the word-topic distributions, i.e., $p(word_w | topic_t)$. Each line is a topic, each column is a word in the vocabulary
- `<model_name>.theta`: This file contains the topic-document distributions, i.e., $p(topic_t | document_m)$. Each line is a document and each column is a topic.

- `<model_name>.tassign`: This file contains the topic assignments for words in training data. Each line is a document that consists of a list of `<word`$_{ij}$`>:<topic of word`$_{ij}$`>`

- `<model_file>.twords`: This file contains `twords` most likely words of each topic. `twords` is specified in the command line (see Sections 3.1.1 and 3.1.2).

GibbsLDA++ also saves a file called `wordmap.txt` that contains the maps between words and word's IDs (integer). This is because GibbsLDA++ works directly with integer IDs of words/ terms inside instead of text strings.

3.3.2. Outputs of Gibbs Sampling Inference for Previously Unseen Data

The outputs of GibbsLDA++ inference are almost the same as those of the estimation process except that the contents of those files are of the new data. The `<model_name>` is exactly the same as the filename of the input (new) data.

3.4. Case Study

For example, we want to estimate a LDA model for a collection of documents stored in file called `models/casestudy/trndocs.dat` and then use that model to do inference for new data stored in file `models/casestudy/newdocs.dat`.

We want to estimate for 100 topics with `alpha` = 0.5 and `beta` = 0.1. We want to perform 1000 Gibbs sampling iterations, save a model at every 100 iterations, and each time a model is saved, print out the list of 20 most likely words for each topic. Supposing that we are now at the home directory of GibbsLDA++, We will execute the following command to estimate LDA model from scratch:

```
$ src/lda -est -alpha 0.5 -beta 0.1 -ntopics 100 -niters
1000 -savestep 100 -twords 20 -dfile models/casestudy/
trndocs.dat
```

Now look into the `models/casestudy` directory, we can see the outputs as described in Section 3.3.1.

Now, we want to continue to perform another 800 Gibbs sampling iterations from the previously estimated model `model-01000` with `savestep` = 100, `twords` = 30, we perform the following command:

```
$ src/lda -estc -dir models/casestudy/ -model model-01000 -
niters 800 -savestep 100 -twords 30
```

Now, look into the casestudy directory to see the outputs.

Now, if we want to do inference (30 Gibbs sampling iterations) for the new data `newdocs.dat` (note that the new data file is stored in the same directory of the LDA models) using one of the previously estimated LDA models, for example `model-01800`, we perform the following command:

```
$ src/lda -inf -dir models/casestudy/ -model model-01800 -
niters 30 -twords 20 -dfile newdocs.dat
```

Now, look into the `casestudy` directory, we can see the outputs of the inferences:

```
newdocs.dat.others
newdocs.dat.phi
newdocs.dat.tassign
newdocs.dat.theta
newdocs.dat.twords
```

4. Links, Acknowledgements, and References

4.1. Links

Here are some pointers to other implementations of LDA:

- LDA-C (Variational Methods)
- Matlab Topic Modeling
- Java version of LDA-C and a short Java version of Gibbs Sampling for LDA
- LDA package (using Variational Methods, including C and Matlab code)

4.2. Acknowledgements

Our code is based on the Java code of Gregor Heinrich and the theoretical description of Gibbs Sampling for LDA in [Heinrich]. I would like to thank Heinrich for sharing the code and a comprehensive technical report.

We would like to thank Sourceforge.net for hosting this project.

4.3. References

- [Andrieu03] C. Andrieu, N.D. Freitas, A. Doucet, and M. Jordan: An introduction to MCMC for machine learning, Machine Learning (2003)
- [Blei03] D. Blei, A. Ng, and M. Jordan: Latent Dirichlet Allocation, Journal of Machine Learning Research (2003).
- [Blei07] D. Blei and J. Lafferty: A correlated topic model of Science, The Annals of Applied Statistics (2007).
- [Griffiths] T. Griffiths: Gibbs sampling in the generative model of Latent Dirichlet Allocation, Technical Report.
- [Griffiths04] T. Griffiths and M. Steyvers: Finding scientific topics, Proc. of the National Academy of Sciences (2004).
- [Heinrich] G. Heinrich: Parameter estimation for text analysis, Technical Report.
- [Hofmann99] T. Hofmann: Probabilistic latent semantic analysis, Proc. of UAI (1999).
- [Wei06] X. Wei and W.B. Croft: LDA-based document models for ad-hoc retrieval, Proc. of ACM SIGIR (2006).

Last updated July 15, 2007