hca

Wray Buntine

March 25, 2016
Version 0.62

Abstract

hca is research software that does various versions of non-parametric topic models using Gibbs sampling including LDA, HDP-LDA, NP-LDA, NG-LDA all with/without burstiness modelling. Various diagnostics, "document completion" testing and coherence measurements with PMI are also supported. The code runs on multi-core getting about 50% efficiency with 8 cores. Be warned, however, that it is research code. Not all combinations of options behave as expected, not all errors behave gracefully.

1 Synopsis

hca [**-?**] [**-?***Arg*] *DataStem RepStem*

2 Description

hca reads the collection of files with stem *DataStem* that form the input set of data. When checkpointing, or at termination, the output is written to files with stem *RepStem*. On restart with the **-r***0* option, some of these are also read initially to restore the previous state. A log of the run is reported to `stderr` if the **-e** option is used. By default, the log goes to `RepStem.log`.

The programme runs a Gibbs sampler for a variety of non-parametric topic models including simple HDP-LDA. The model selected has three parts:

alpha: this is the prior on topic vector (theta) for each document. LDA has a simple symmetric Dirichlet with parameter alpha and the vector has dimension T (the number of topics).

beta: this is the prior on word vector (phi) for each topic. LDA has a simple symmetric Dirichlet with parameter beta and the vector has dimension W (the number of words).

burst: this is the burstiness component which has a document specific variant of the word vector for each topic. This is not used by default.

These parts are set using the **-S**, **-A** and **-B** options.

There are various model parameters, notably the discount and concentrations for the different Pitman-Yor processes in the model. These are usually sampled using Adaptive Rejection Sampling. They are also kept bounded using constraints hard-coded into the `util/dimdir.h` header file. So when a parameter fails to change, check the sampling by increasing verbosity and you may observe the value tries to change but doesn't.

The programme uses generalised second order Stirling numbers with the library extracted from *libstb* version 1.8 released at `https://github.com/wbuntine/libstb`. This is (annoyingly) initialised with predefined bounds on the tables, and these can be modified with the **-N** option. This should be used for collections with larger numbers of documents, but its best to run first and on error, increase the bounds. See the *Errors* for details.

2.1 Caveat Emptor

The programme is research software, so not all options or combinations of options work correctly. Note that in this release, all the poorly tested experimental features have been stripped, so this release contains only moderately well tested components. However, different combinations of options are not thoroughly tested. Documentation, itself, may also be out of date in some places (unfortunately).

Note also, this is not intended to be code that others could easily modify. In order to get performance, to provide all the features, and to run multi-core, the code is quite convoluted. Researchers seeking simple code they can experiment and modify themselves should request a cutdown version from the author. The safest thing is to work with the examples given at the end of the document.

3 Options

Options have a single letter followed by a possible single argument. Options are grouped under the following functions:

- *setting of hyperparameters*,
- *controlling sampling of hyperparameters*,
- *general control*, and
- *testing and reports*.

3.1 Setting up the model and hyperparameters

For these, `theta` is a vector for each document representing the topic proportions and `phi` is a vector for each topic representing the word proportions. The task of the system is to estimate these. The vector theta and its various priors and parameters is the Alpha side and the vector phi and its various priors and parameters is the Beta side. All the scalar parameters can be set using the **-S***var=value* option and thereafter fixed using the **-F***var* option or by default sampled using adaptive rejection sampling.

-Avalue[**,file**] Use a symmetric Dirichlet prior on theta using this `value` (a float/real) for each dimension. The value must be a positive float. With the optional `file` argument, the file specifies the probability vector to use as the mean vector of the Dirichlet. The file is in text format representing `T` (the number of topics) floats, which will be normalised. Then multiply the mean vector by `T*value` to get the Dirichlet parameter vector. *i.e.*, the mean of the `T` values in the Dirichlet parameter vector is `value`.

-Adir[**,file**] Same as **-A***value[,file]* but `value` is set to a default, 0.05*`avelen/T` where `avelen` is the average document length in the training set.

-Atype[**,file**] This other version of the **-A** option changes the Alpha side Dirichlet on theta to a Pitman-Yor process, thus allowing estimation of hierarchical prior. It defines a distribution on theta and its prior mean (a vector) `alpha` of `type` as follows:

hdp theta is modelled with a Dirichlet Process with mean `alpha` and concentration `b`, and alpha is modelled with a symmetric Dirichlet with concentration `b0`. If the `file` optional argument is used then it specifies an input file giving the mean of the Dirichlet over `alpha`. By default the mean is a uniform vector.

hpdd theta is modelled with a Pitman-Yor Process with mean `alpha`, discount `a` and concentration `b`, and alpha is modelled with a (truncated) GEM with discount `a0` and concentration `b0`. This is the default. The `file` optional argument is ignored.

- **pdp** theta is modelled with a Pitman-Yor Process with mean `alpha`, discount `a` and concentration `b`, and the alpha vector is uniform. This is not hierarchical because alpha is constant. If the `file` optional argument is used then it specifies `alpha`.

-Ang The instigates the *normalised Gamma* prior on theta. Each topic now has its own independent gamma distribution, `Gamma(NGalpha,NGbeta)`, and these are normalised to get the topic probabilities. There are 2*`T` parameters now, the vector of alpha parameters for each topic and the vector of beta parameters for each topic. This means each dimension has its own independent variance parameter, so both mean and variance are fit per dimension. By default these are sampled in batches during standard hyper-parameter sampling. Do not use with burstiness as this has not been evaluated.

-Bvalue[**,file**] Use a symmetric Dirichlet prior with this `value` for each dimension. The value must be a positive float. When this mode is running, an extra latent variable is sampled per document, which is saved as `RepStem.UN` when checkpointing. *Warning:* the value stored internally and printed is the total of this over the number of words `W`. With the optional `file` argument, the file specifies the probability vector to use as the mean vector of the Dirichlet. The file is in text format representing `W` (the number of words) floats. Then multiply the mean vector by `W*value` to get the Dirichlet parameter vector. i.e, the mean of the `W` values in the Dirichlet parameter vector is `value`.

-Bdir[**,file**] Same as **-B***value[,file]* but `value` is set to a default, currently 0.001 (10 times the current minimum allowed for a Dirichlet).

-Btype[**,file**] The other form of the **-B** option similar to the **-A** option. Use a prior beta of `type` "hdp" "hpdd" or "pdp". Similar to the **-A** option.

- **hdp** phi is modelled with a Dirichlet Process with mean `beta` and concentration `bw` and beta is modelled with a Dirichlet with concentration `bw0` by default symmetric (a uniform mean) or its mean can be set with the `file` optional argument above. Setting `file` to the reserved word "data" uses the observed word frequencies as the mean.
- **hpdd** phi is modelled with a Pitman-Yor Process with mean `beta`, discount `aw` and concentration `bw`, and `beta` is modelled with a (truncated) GEM and discount `aw0` and concentration `bw0`. This is the default.
- **pdp** phi is modelled with a Pitman-Yor Process with mean `beta`, discount `aw` and concentration `bw`, and beta is by default uniform, or its mean can be set with the `file` optional argument above. Setting `file` to the reserved word "data" uses the observed word frequencies as the mean. This is not hierarchical because beta is constant.

-S*var=value* Set variable `var` to float `value`, where `var` can be one of:

- **a** discount parameter for the non-parametric distribution on the theta, topic distribution per document.
- **b** concentration parameter for the non-parametric distribution on theta, the topic distribution per document.
- **a0** discount parameter for the non-parametric distribution on alpha, the prior for theta.
- **b0** concentration parameter for the non-parametric distribution on alpha, the prior for theta.
- **aw** discount parameter for the non-parametric distribution on phi, word distribution per topic.

bw concentration parameter for the non-parametric distribution on phi, word distribution per topic.

aw0 discount parameter for the non-parametric distribution on beta, prior for phi.

bw0 concentration parameter for the non-parametric distribution on beta, prior for phi.

ad discount parameter for burstiness.

bdk concentration parameter for burstiness, a constant initially but subsequent sampling will allow a different value per topic.

3.2 Controlling sampling of hyperparameters

Most hyperparameters are fit with adaptive rejection sampling (ARS) by default. The discount parameter of a Pitman-Yor process, when set to zero is not fit, as it is assumed you want a Dirichlet process instead. Options give which cycles to run ARS on which hyperparameters and which hyperparameters not to sample.

-D*cycles,start* Start sampling `alpha` of the symmetric Dirichlet for alpha after `start` cycles and then repeat every `cycles` cycles.

-E*cycles,start* Start sampling `beta` of the symmetric Dirichlet for beta after `start` cycles and then repeat every `cycles` cycles.

-F*var* Fix the variable `var` where it takes the value **alpha**, **beta** or one of the arguments to the **-S** option.

-g*var,batch* The vector hyperparameters `bdk`, `NGalpha` and `NGbeta` are sampled in batches using a heuristic batch size. Set the batch size with **-g***bdk,10* or similar, though note they all share the same batchsize.

-G*var,cycles,start* Sample the variable `var` where it takes the value **alpha**, **beta** or one of the arguments to the **-S** option. The `start` and `cycles` integers are used as for the **-D** option.

3.3 General control

-c*cycles* Do a checkpoint every this many `cycles`. This saves the output statistics and the parameter file adequate to do a restart with **-r***0* option.

-C*cycles* Stop after this many `cycles`. Default is 100. Note **-C***0* should be used when one just wants reports, as the various output files (other than reports) will be left unaltered.

-d*dots* For really big batches of data, print a "." every `dots` documents within a single cycle.

-e Reroute logging to the `stderr`.

-f*format* Read input data from data formatted according to the type `format`. Data is expected to come from an input file with name `DataStem.Suff` where `Suff` is an appropriate suffix. These are given with Input Files below. Allowed formats are: `ldac`, `witdit`, `docword`, `bag` and `lst`.

-K*topics* Set T the maximum number of topics. Default is 10.

-M*maxtime* Quit early when total training time exceeds this many seconds.

-N*maxN,maxT* Set maximum for the Stirling number tables to count `maxN` and table count `maxT`. Default is 10000,1000. On collections with more than 20k documents, can require more.

-q*threads* If compiled with threading, enables this many threads. Default is 1.

-r*0* Restart with all data. Currently must use the `offset` equal to "0" for a normal restart.

-r*phi* Another version of the **-r** option using the string “phi” as the argument. Restart but now fix the word by topic matrix to the previously estimated values saved at `RepStem.phi`, and the beta side is held constant and not sampled. Can significantly speed up testing or querying sometimes.

-r*theta* Second version of the **-r** option using the string “phi” as the argument. Restart but now fix the document by topic matrix to the previously estimated values saved at `RepStem.theta` and `RepStem.testprob`.

-s*seed* Initialise the random number seed.

-v Up verbosity by one increment. Starts at zero and currently understands 0-3.

-x Enable use of exclude topics with **-Q**.

3.4 Testing and reports

-h*Hold,arg* Do document completion testing on the test set. There are three styles of document completion implemented given by the `Hold` parameter.

- **dict** every `arg`-th word in the dictionary is held out in estimating and used for testing. So if a word has dictionary index `arg-1`, `2*arg-1`, *etc.*, it is held out.
- **doc** every `arg`-th word is held out in estimating the latent variables (like theta) for the document and used instead for testing of perplexity. That is, words at document positions `arg-1`, `2*arg-1`, *etc.*
- **fract** then the `fract` proportion at the tail of the document is held out. The initial proportion is used in estimating. Some documents vary in topic over the length, so this method is not advised.

-l*Diag,cycles,start* Do a run-time estimation of the diagnostic `Diag` starting after the `start` cycle and then taking the estimate every `cycles` cycle. Diagnostics are:

- **alpha** Estimate the prior topic probability vector. Stored in the `RepStem.alpha` file. Note useable with the **-A***pdp* option on restart as the `RepStem.alpha` will be read, though `a` and `b` will need to be set.
- **phi** Estimate the word probability vector for each topic. Stored in the `RepStem.phi` file. If the model is not a symmetric Dirichlet model, then the word prior vector will be estimated and saved in the `RepStem.beta` file as well. Note useable with the **-B***pdp* option on restart as the `RepStem.beta` will be read, though `aw` and `bw` will need to be set.
- **prog** How often to do the standard diagnostic reports (default is every 5-th cycle).
- **sparse** Estimate topic sparsity in the theta matrix for the words given in `DataStem.smap`. If `DataStem.smap` is not there then this defaults to all words. Note, the default can be quite wasteful for multicore, is it duplicates the theta matrix for each thread, so only do for small data sets. Results placed in `RepStem.smap`. The report gives “topic/weight” for topics including the word.
- **testprob** Estimate the topic probability vector for each test document. Stored in the `RepStem.testprob` file.
- **theta** Estimate the topic probability vector for each training document. Stored in the `RepStem.theta` file.

Note that for `Diag`=“testprob” or “theta”, an additional argument after `start` giving the lowerbound on probabilities. Lower ones are dropped.

-L*Diag,cycles,start* Do a diagnostic estimate `Diag` after all Gibbs sampling is complete. Sampling of the estimate starts after the `start` cycle and goes for a total of `cycles` cycles (including the starting ones). Diagnostics are:

- **class** Estimate class probabilities with "true" classes given in `DataStem.class` and then produce confusion matrix for the test data. Output to files `DataStem.cnfs` and `DataStem.pcnfs`.
- **like** Estimate likelihood/perplexity on the test set using the standard (biased) document likelihood, or document completion if the **-h** option is used. Can also be instigated during run-time with the **-P** option.

-oscore[,count] Scoring rule to pick top words for printing. Methods are 'count', 'idf', 'cost' and 'phi'. Default is 'idf'. Ranking done for top `count` words, default is 20. Methods are

- **cost:** rank by proportion of this word in topic minus estimated proportion assuming topic and word independent.
- **count:** rank by count in topic.
- **idf:** rank by fraction of the total occurrences of this word that are in this topic.
- **phi:** rank by computed phi value (if loaded).
- **rat:** rank by ratio with the beta prior ("background topic") produced by NP-LDA.

-O Report log likelihood, not log perplexity. Both are done in base 2.

-p Report topic coherency in the log file, and save the detail (per topic) in the `RepStem.toppmi` file. This requires a `DataStem.pmi` or `DataStem.pmi.gz` file exist in the right format. This can be created with the *mkmat.pl* and *cooc2pmi.pl* scripts in the scripts directory of the release. The format is a simple sparse matrix form with lines of the form "N M PMI" for word indices (offset by 0) N and M and PMI value. *WARNING:* the file `DataStem.pmi` needs to be specifically built for the dataset as the word indices must align. By default, PMI computed for top 10 words. Give option twice, and PMI will be done for all top words ranked (as per the **-o** option).

-P*secs* Calculate test perplexity (using document completion) every interval in `secs` seconds. If Gibbs cycles are long, will report only after the cycle finishes.

-Q*nres,file* submit list queries given in the file, and return `nres` results for each. Must use the **-r***phi* option with a pre-estimated phi matrix (for efficiency).

-t*size* Specify size of training set. It takes the first `size` entries in the data set. Default is all the set minus the test data.

-T*filestem* Specify a separate test set. Assumes the same suffix as for `DataStem`. When using this, be sure to fix the training set size with **-t***size* if you do not want to train on the full data set.

-T*size* Specify size of test set. It takes the `size` entries immediately following the training set. Default is zero. This option can be confused with the above, so do not use filestems that are just integers.

-V load the dictionary from the `DataStem.tokens` file for use in reporting. It has one token per line. Must have at least level two verbosity or this is ignored.

-X Instigate report on naive Bayes classification using the topic model and classes given in `DataStem.class` file. The report is a confusion matrix to file `RepStem.tbyc` built on the training data.

4 Input Files

The following files provide details about the dataset. The filenames are constructed by adding a suffix to the data stem. The data (document+word) format itself can be one of four different formats and is specified with the **-f** option.

`DataStem.class` Class index for each document, one per line. Optional file used with some reports instigated by **-X** or **-L***class* options.

`DataStem.dit`+`DataStem.wit` Simple document index and word index files, both indices offset by 1, one index per line. Words in the collection are listed by document. The `DataStem.dit` file gives the document index, and the corresponding line in `DataStem.wit` gives the dictionary index.

`DataStem.docword` This format appears in some UCI data sets at `http://archive.ics.uci.edu/ml/datasets/Bag+of+Words`. Word indices offset by 1.

`DataStem.ldac` Standard LdaC format. Word indices to the dictionary are offset by 0.

`DataStem.smap` A list of word indices (offset by 0) about which one wants a sparsity report generated. The report is instigated by the **-l***sp* option.

`DataStem.tokens` tokens/words in the dictionary, one per line. Optional file used with **-V** option.

`DataStem.txtbag` default bag or list format for *linkBags*(1) command of `text-bags`. Word indices offset by 0.

The various output files such as `RepStem.par` (Parameter and dimension output file) are also read on restart with the **-r***0* option.

5 Output Files

The following files are output when the system checkpoints or at the end of the run. These are built by adding a suffix to the report stem, `RepStem`. The first set of files are:

`RepStem.alpha` If the alpha vector is being estimated with the **-lalpha** option, then this will contain the estimated value.

`RepStem.beta` If a constant beta vector is specified using the **-u** option, this saves the value, for possible use in a restart. Otherwise, if the phi matrix is being estimated with the **-lphi** option and the beta vector is not fixed, then this will contain the estimated value.

`RepStem.cnfs`+`RepStem.pcnfs` Best prediction and probability vector confusion matrices built on the test data with the **-L***class* command.

`RepStem.log` Log file created if **-e** option not used.

`RepStem.par` Parameter and dimensions file in simple "var = value" format. These are detailed in the next section.

`RepStem.phi` The Phi matrix written as a binary file: first W (dictionary size), T (topics), C (sample size) are written as 32 bit integers and then the full Phi matrix as native floats with W as the minor index. Only generated with appropriate use of the **-l***phi* option.

`RepStem.smap` Optional sparsity report on the word indices listed in `DataStem.smap`. The report is instigated by the **-l***sp* option.

`RepStem.tbyc` Optional confusion matrix printed when the **-X** option is used.

`RepStem.topcor` File of correlations between topic. Created with the `RepStem.topset` file.

`RepStem.toplst` A simple text report giving the top word indices for each topic. If a hierarchical model in use, then the "-1" topic is for the base distribution of words. Word indices are offset from 0.

`RepStem.toppmi` A simple text report giving the top word indices and the associated mean PMI for the word.

`RepStem.topset` Full diagnostic output for topics and their words instigated with a command sequence like

```
hca -r0 -C0 -v -V -V -oidf,100 DATA STEM
```

The first few lines of the file are comment lines giving header information.

`RepStem.theta` Estimated topic probabilities for each training document written in a simple sparse form. The class index ("-1" or "+1" for binary classes, otherwise just the index) is also added if it exists. Topic indices are offset by 0. Only generated with appropriate use of the **-l***theta* option.

`RepStem.testprob` Like the `-ltheta` option but for the test documents. Only generated with appropriate use of the **-l***testprob* option.

The second set of files gives the actual runtime statistics. Output matrices are in a simple readable sparse vector format the same as the `DataStem.docword` format.

`RepStem.ndt` Document by topic counts.

`RepStem.nwt` Word by topic counts.

`RepStem.tdt` Document by topic table counts if the Alpha side of the model is non-parametric.

`RepStem.twt` Word by topic table counts if the Beta side of the model is non-parametric.

`RepStem.UN` latent "mass" variable kept for each document when the **-A***ng* option is used.

`RepStem.zt` With no burstiness, gives topic index (offset by 0), one per line. With burstiness, gives one "z,r" per line where "z" is the topic index (offset by 0) and "r" is the burst table indicator, which is 1 if the word contributes to standard topic model statistics, and 0 if burstiness modelling says the word is a burst so does not contribute to topic model statistics.

These files along with `RepStem.par` are input on a restart using **-r***0*.

6 The Parameter File

The parameter file has the following *dimensions*:

N – number of words in the full collection, summed over all documents.

NT – number of words in the training set, summed over all training documents.

W – number of words in the dictionary.

D – number of documents in total.

TRAIN – number of documents to train on, is always the the first ones in the file.

TEST – number of documents to test on, is always the the last ones in the file.

T – maximum number of topics.

ITER – number of major cycles made last.

In addition, the float parameters allowed to be specified with the **-F** and **-G** options are also given. Finally, the type of model for alpha as specified by the **-A** option is coded in the `PYalpha` variable. It is 0 if the model is a Dirichlet, the LDA default. It is 1 for hdp, 2 for hpdd and 3 for pdp. Likewise for the `PYbeta` variable and the **-B** option.

If the **-A***ng* option is used then vectores `NGalpha` and `NGbeta` are saved as well.

7 Examples

Examples are given for

- *basic running*,
- *different models*,
- *diagnostic reports*,
- *restarts and printing words*,
- *sparsity mappings and topic probabilities*
- *testing*
- *estimating model parameters*
- *burstiness*

7.1 Basic running

These examples work as is on late model Linux, Macs and Windows. However, you need to replace the executable, `hca`, by the system dependent one, from the install directory where the `data/` directory is. For instance, on Windows that might be `hca/hca.exe`.

Run basic LDA with default parameters and full parameter fitting on the full dataset and no testing, sending logging to *stderr*.

```
hca -v -e -K20 -Adir -Bdir -C100 data/ch c1
```

Alternatively, run basic HDP-LDA with parameter fitting on the full dataset and no testing, sending logging to *stderr*.

```
hca -v -e -K20 -B0.001 -C100 data/ch c1
```

The command lines mean:

"-v": use level one verbosity;

"-e": send the log file to `stderr`, not to "c1.log";

"-K20": use 20 topics (the truncation level if using **-A***hpdd*));

"-Adir": use a symmetric Dirichlet prior on topic probability vectors for documents with default value;

"-Bdir": use a symmetric Dirichlet prior on word probability vectors (i.e., topics) with default value;

"-B0.001": use a symmetric Dirichlet prior on word probability vectors (i.e., topics) with this value;

"-C100": run for 100 cycles;

"data/ch": stem for data file;

"c1": stem for results file.

Consider the HDP-LDA version. Before the runtime logging starts, initial details are printed:

```
Version 0.5, H.Pitman-Yor sampler for topics, Dirichlet sampler for words
Sampling pars: b(3), b0(3), betatot(4),
Setting seed = 1403582987
Read from ldac file: D=395, W=4258, N=84010
S-table 'a, ad,  all zero PYP': a=0.000000, N=812/1000, M=100/1000, +S+U/V float mem=626k
mem   = 1.3 (MByte)
seed  = 1403582987
N     = 84010
W     = 4258
D     = 395
TRAIN   = 395
TEST    = 0
T     = 20
ITER  = 100
PYbeta  = 0
betatot  = 4.258000 # total over W=4258 words
PYalpha  = 2
a     = 0.000000
b     = 10.000000
a0     = 0.000000
b0     = 10.000000
Initialised with 20 classes
```

Note the following:

- the `betatot` value is the total of the input `beta` (0.001) over the $W = 4258$ words; internally the `betatot` is maintained and subsequently sampled;
- the "Sampling pars:" line indicates hyperparameters being sampled, which are `b`, `b0`, `betatot`, with `b` and `b0` being sampled every 3 major cycles and `betatot` every 4 major cycles;
- in this case `a` and `a0` are not sampled because they are fixed at 0, meaning the alpha side is modelled with a Dirichlet process;
- the memory allocated is approximately 1.3Mb, actual usage will vary with stack memory and some items not recorded;
- the seed for the random number generator is 1403582987 so use "-s1403582987" to repeat the same sampling;
- there are 395 documents, 4258 different words/tokens in the dictionary and a total of 84010 words/tokens in the documents;
- `PYbeta=0` means the beta side is a Dirichlet;
- `PYalpha=2` means the alpha side is a truncated GEM prior at the top level and Pitman-Yor process or Dirichlet process at the document level;
- and `TEST=0` means there is no test data.

7.2 Different models

The list below gives different models that one might run. Note all hyperparameters will subsequently be fit during sampling, unless you use the **-F** option to switch individual fitting off.

-Adir -Bdir : this is standard LDA using the default settings for symmetric Dirichlet priors. Replace the word "dir" with a float to get specific values initialised.

-B0.001 : this is HDP-LDA using a Dirichlet prior for phi (word probability vector), and a default non-parametric prior (HPDD) for theta (topic probability vector).

`default`**:** default is full non-parametric topic modelling *without* burstiness, we call NP-LDA, both priors for theta and phi use the default non-parametric prior (HPDD).

-Ang : this uses a normalised Gamma for the prior for theta, which means different dimensions have both mean and variance fit, we call NG-LDA. Phi has the default non-parametric prior (HPDD).

-Sbdk=100 : full non-parametric topic modelling *with burstiness*, bursty NP-LDA, both priors for theta and phi use the default non-parametric prior (HPDD).

7.3 Diagnostic reports

By default, every 5 cycles, a short report is printed:

```
[26/05/2014:10:01:38] cycles:  81 82 83 84 85
log_2(perp)=11.5182,9.9503
Pars:  b=2.041296, b0=3.007822, betatot=301.019289
```

The report frequency is modified with the **-l***prob,...* option, and the report can be extended by adding verbosity with **-v**. The entry in square brackets is the system clock time at the start of cycle 81. Here cycles 81-85 are run. The two perplexities reported are normalised per token and then given in log to base 2. The first is from the posterior probability with all real-valued probability vectors marginalised out using Pitman-Yor process theory but with the latent counts (counts of tables, not full table configurations) included. The second is the running total of word probabilities encountered during sampling. This does not include the probability cost of latent variables (for instance, the topics) so always less. After `Pars:` appears the list of hyperparameters being sampled and their current values.

Adding an extra level of verbosity using an additional **-v**, one gets a brief one line report for every hyperparameter being sampled, such as

```
  myarmsMH(b) = 3.272891<-3.432078, w 37 calls
```

This means the adaptive rejection sampler took 37 calls to sample `b`. The initial value was 3.432078 and the final value was 3.272891. This line will be printed every time a sampling is done, sometimes multiple ones per major Gibbs cycle. Moreover, topic probabilities are printed. These are estimated (with standard smoothing) from training data. For instance,

```
probs =  0.041541 0.062400 0.083437 0.060447 0.025652 0.069235 ....
conc. = 10.225621, empty = 0, exp.ent = 19.049888
```

The three diagnostics give additional details about the probabilities. The concentration (inverse of variance) applies to these, and it is computed differently depending on the model. If some topics have no data in them, `empty` will tell how much. The effective number of topics is 19.049888, which is the exponential of the entropy of the probability vector (ignoring empty topics). It should always be less than the truncation level.

At the end, a final report is printed.

```
[29/05/2014:21:07:27] Finished after 100 cycles on average of 0.193804+0.013074(s) per cycle

Topic 6/0 p=12.54% ws=76.1% ds=14.2% ew=584 ed=24 da=10 t1=4 ud=0.9344 pd=0.6448 co=-1.4%
Topic 3/1 p=6.82% ws=76.8% ds=39.0% ew=790 ed=56 da=6 t1=3 ud=0.8126 pd=0.7304 co=-0.8%
Topic 14/2 p=5.73% ws=83.2% ds=82.0% ew=442 ed=93 da=12 t1=5 ud=0.9223 pd=0.7350 co=-0.3%
...
```

```
Average topicXword sparsity = 82.93%
Average docXtopic sparsity = 66.14%
Underused topics = 0.0%

probs =  0.037662 0.031478 0.034289 0.020517 0.043002 0.097527 0.022766 0.068859 0.114952 ...
conc. = 1.784346, empty = 0, exp.ent = 15.296125
log_2(train perp) = 11.456566
```

The figures give 0.19380 seconds per cycle for the Gibbs sampler and 0.01307 seconds per cycle for the adative rejection sampling of hyperparameters. Note these figures are not collected correctly for the multi-core version.

Some basic details for the topics are given too. With verbosity level of 1 only diagnostics are given for topics. With higher verbosity word indices or words are reported as well, as ranked using the **-o** option. The topics are listed in terms of decreasing proportion. So "Topic 6/0" means "topic number 6, which is the most frequent" and "Topic 14/2" means "topic number 14, which is the 3rd most frequent."
Details of the diagnostics are as follows:

co: coherence as per Mimno, Wallach, Talley, Leenders and McCallum, EMNLP 2011.

da: documents with proportion for topic greater than 1/sqrt(T).

ds: document sparsity, proportion of documents having zero occurrences of this topic;

ed: effective number of documents, expenential of the entropy of the document distribution (the document by topic matrix normally normalised over topics; renormalise by documents for a given topic);

ew: effective number of words, exponential of the entropy of the word distribution for topic;

ewp: effective number of words, inverse of the expected word probability, Mallet's alternative to `ew`;

ng: with the **A**ng option, gives the expected topic probability computed by normalising the means of the topic gammas, and a measure of overdispersion given by the standard-deviation divided by the mean.

p: proportion of tokens tagged with this topic;

pd: Hellinger distance to the (training) population word distribution;

t1: documents with this topic as most common.

ud: Hellinger distance to the uniform distribution.

ws: word sparsity, proportion of words occurring zero times with this topic;

So the first topic has 6/0 given. This means it was index 6 in the run but is rank 0 in terms of proportion. In the saved data file it will be topic 6. With more verbosity, top topic words will be given as well ranked according to the *-o* option. Totals for some of the topics are also given: "Average topicXword sparsity" is the mean of the word sparsities (`ws`), "Average docXtopic sparsity" gives the mean of the document sparsities (`ds`), and the number of underused topics is the percentage of topics whose observed proportion is less than 1/T/100 or with less than 5 occurrences.

The `log_2(train perp)` figure is equivalent to the `log_2(perp)` figure above because there is no test data. At this point, a number of data files will have been written, the same as done with any checkpoint. The main one is the parameter file `c1.par` which gives all the dimensions as well as the final values of the hyper-parameters. Note the `probs` are also included, but these are for information only. The others can be used to restart the run.

If you have the multicore version compiled, and you have an 8-core CPU, then run with 8 threads:

```
hca -v -e -K20 -B0.001 -C100 -q8 data/ch c1
```

"-q8": use 8 threads for Gibbs sampling.

This just repeats the above but should be faster!

7.4 Restart and print words for the topics

Restart from checkpoint after the previous run but run no cycles. Input the tokens from `data/ch.tokens`, and print top 10 words for each topic.

```
hca -v -v -r0 -e -V -C0 data/ch c1
```

The new command line options mean:

"-v -v": use level two verbosity;

"-r0": restart from document 0, i.e., on all documents;

"-V": input the tokens from "data/ch.tokens," and print top 10 words for each topic. Note must have at least level two verbosity;

"-C0": do not run any cycles, just do reporting.

After printing initial details, this will print two sets of details. The first is a list of top topic words (if verbosity is greater than 1) and topic diagnostics. The topic diagnostics were explained in the precious subsection. Topics are printed in decreasing order of occurrence. The extra verbosity level and the **-V** means that topic words will be printed out too.
Here are some sample topic lists with just "**-v -v**", which uses word ranking "**-o***idf*" by default:

```
Topic 5/0 p=13.68% ws=37.0% ds=43.3% ew=732 ewp=424.9 ed=132.3 ...
topic 5/0 words=1679,1412,780,1234,1612,1096,1758,1552,1066,584
Topic 9/1 p=12.78% ws=37.0% ds=42.5% ew=715 ewp=396.8 ed=137.1 ...
topic 9/1 words=452,623,1241,1701,1275,1434,1448,1489,1062,1079
```

Here are some sample topic lists with words, "**-v -v -V**":

```
Topic 5/0 p=13.68% ws=37.0% ds=43.3% ew=732 ewp=424.9 ed=132.3 ...
topic 5/0 words=bernardin,miami,chicago,concert,beach,pop,designer,murders,killing,music
Topic 9/1 p=12.78% ws=37.0% ds=42.5% ew=715 ewp=396.8 ed=137.1 ...
topic 9/1 words=germany,nazi,papers,territory,hitler,crimes,chancellor,sentence,victims,troops
```

Here are some sample topic lists with words using ratio ranking, "**-o***rat*", which does not work with plain Dirichlet priors on phi:

```
Topic 5/0 p=13.68% ws=37.0% ds=43.3% ew=732 ewp=424.9 ed=132.3 ...
topic 5/0 words=bernardin,miami,chicago,concert,fans,pop,designer,music,killing,video
Topic 9/1 p=12.78% ws=37.0% ds=42.5% ew=715 ewp=396.8 ed=137.1 ...
topic 9/1 words=germany,nazi,papers,territory,hitler,nobel,crimes,german,prize,chancellor
```

For more detail to the `RepStem.topset` file and the `RepStem.topcor` file, use:

```
hca -v -v -r0 -e -V -V -oidf,100 -C0 data/ch c1
```

The command line means:

"-V -V": extra **-V** means create the `RepStem.topset` file of details.

"-oidf,100": means report on up to 100 words for each topic, and words ranked by the `idf` score.

The first two lines give brief column heads for the topic and word lines. The scores match those printed with diagnostics.

7.5 Produce sparsity mappings and document topic probabilities

Restart again and build a topic probability vector for each document, as well as sparsity mappings for the words in `data/ch.smap` file. This you need to create/edit ahead of time. This must run a number of cycles because the estimates are done during the Gibbs sampling.

```
hca -v -r0 -e -lsparse,2,1 -ltheta,2,1,0.001 -C20 data/ch c1
```

"-lsparse,2,1": sample for sparsity every 2nd cycle starting at the 1st.

"-ltheta,2,1,0.001": sample probabilities per document (theta) every 2nd cycle starting at the 1st. Only report probabilities above 0.001.

"-C20": sampling done for 20 cycles.

Now view the sparsity report at `c1.smap` and the topic probabilities at `c1.theta`, and the values saved in the parameter file `c1.par`. Again, add the **-q***8* option to run this faster, with 8 threads (if you have 8 cores).

Read lines in the sparsity report, `c1.smap`, as follows:

```
--(12): 5/2.6 14/1.3 19/219.0 perp=1.149816
```

Token with index 12 occurs in topics 5, 14 and 19. It has 2.6 counts (its a sample average so counts can be a fraction) in topic 5 and 219.0 in topic 19. The effective number of topics using this token is 1.149816. This is measured as the exponential of the entropy of the topic distribution (i.e., probability of topic given the single word and assuming topics are equally likely).

Read lines in the topic probabilities report, `c1.theta`, as follows:

```
15: 16:0.006699 17:0.088948 19:0.902410
```

Document 15 has 0.006699 for topic 15 and 0.902410 for topic 17. The three topics only add to 0.998057 because some smaller topics must have been dropped.

7.6 Run with testing

Testing discussed here only tests on the latest sample done with Gibbs. More sophisticated testing, described later first estimates the model parameters over a number of Gibbs iterations, and then perform testing using the estimates. This is described in later subsections.

First run basic LDA with training and parameter fitting on a subset and testing on the final 100 documents. The training subset is the full dataset minus the test data. Logging now to `c1.log`. Checkpoint every 20 cycles (note, we usually only do this for cycles taking over 10 minutes each).

```
hca -v -K20 -C100 -c20 -T100 data/ch c1
```

Again run multi-core with **-q***8* if needed.

"-c20": do a checkpoint with any reporting every 20 cycles.

"-T100": use the last 100 documents for testing, so the first (datasetsize-100) are used for training. The documents must be ordered so the test data is at the end. Alternatively, a file stem can be given if test data is in a separate file, so loaded from there.

View the end of the log file to get the test perplexity, which is printed after "log_2(test perpML)".

Now restart but use document completion (every 4th word) to get perplexity, with no more Gibbs cycles. Without **-h** the default is to use a standard likelihood calculation so will be biased.

```
hca -v -e -r0 -C0 -hdoc,4 -T100 data/ch c1
```

"-hdoc,4": hold out every 4-th word in the document.

"-T100": the test set size must be repeated, since it is not reloaded with the restart.

View the end of the log file to get the test perplexity, which is printed after "log_2(test perpHold)". Note it is also recorded in the parameter file.

Restart and record the PMI and the classification details on test data.

```
hca -v -v -V -r0 -C0 -Llike,0,0 -X -p -T100 data/ch c1
```

"-Llike,0,0": prevent it doing test likelihood calculations, which are potentially slow on larger data sets.

"-X": load up class data from `data/ch.clas` file to enable classification on test data.

"-p": initiate PMI calculation.

The PMI data has a value printed for each topic as well as a final average. It bases its calculations on the matrix `data/ch.pmi.gz` created explicitly for this test set. For other datasets, you will need to download prepared PMI matrices from the project homepage. The PMI output in the log file adds a PMI figure at the end of the second set of diagnostics:

```
Topic 0 stats: p=3.16%, ws=86.3%, ds=71.4%, pmi=2.565,
Topic 1 stats: p=6.73%, ws=81.7%, ds=76.2%, pmi=0.825,
Topic 2 stats: p=3.59%, ws=85.2%, ds=72.9%, pmi=1.392,
```

Moreover, the general diagnistics get an extra line:

```
Average PMI = 0.602
```

7.7 Estimating model parameters

The assumes a run has already been done. Now we restart and initiate estimation.

```
hca -v -e -r0 -C100 -lphi,3,1 -ltheta,3,1 -lalpha,3,1 data/ch c1
```

"-lalpha,3,1": estimate the `alpha` vector if the Alpha side is non-parametric, and save in the `c1.alpha` file. Estimation starts after the 1st cycle and a sample is added to the average every 3 cycles, that is, 1,4,7,...,94,97.

"-lphi,3,1": estimate the `phi` matrix, and if the Beta side is non-parametric, then also estimate the `beta` vector. Saved as the `c1.phi` and `c1.beta` files respectively. Estimation as before.

"-ltheta,3,1": estimate the `theta` matrix and save as the `c1.theta` file. Estimation as before.

The files `c1.alpha` and `c1.beta` are text but the file `c1.phi` is binary. The file `c1.theta` is written in a readable sparse form.

7.8 Burstiness

The burstiness version significantly improves everything. Our best bet, currently, is to run with optimisation of the hyperparameters:

```
hca -v -v -e -K20 -C100 -Sbdk=100 -Sad=0.5 data/ch c1
```

"-Sbdk=100": burstiness document concentration is different for every topic. This initialises all of them to 100. Default has no burstiness.

"-Sad=0.5": burstiness document discount set to 0.5, same for all topics. Default is zero.

The initial discount for the bursty topics is 0.5. The concentration we set quite high initially, and these will be sampled separately with each topic in batches, so `bdk` is a vector in the parameter file. The hyperparameter sampling slows it down quite a bit but seems to make a significant difference. Unused topics sometimes get a very low concentration. Alternatively, fix the burstiness discount with **-F***ad* and continue sampling burstiness concentration only, which is quite a lot faster. Note burstiness works well with multi-core as does sampling of hyperparameters.

Diagnostics reported for burstiness, printed at the end, are as follows:

```
Burst report:  multis=55.45%, tables=79.57%, tbls-in-multis=63.15%
```

These are:

multis: percentage of tokens in documents that occur more than once. Only these are affected by burstiness processing. So (100-`multis`) is proportion of tokens unique in their document.

tables: percentage of data being passed up by the burstiness sub-module to the topic model. Note 100% of the (100-`multis`)% unique tokens will be passed up as unique tokens always go to the topic model. Of the remaining `multis`% tokens, only `tbls-in-multis`% get passed up.

tbls-in-multis: the percentage of non-unique words in documents that are passed up by the burstiness sub-module to the topic model.

7.9 Sample Scripts

This section lists some useful scripts for doing combined runs. Scripts below have common shell parameters:

K: number of topics

T: number of documents at end of file to use for testing

DATA: stem for the data set

STEM: stem for the result set

This first example runs standard HDP-LDA for 1000 cycles on 4 cores, fitting all hyper-parameters. Check points are done every 100 cycles, and at that stage a test is done using document completion where every 3/4 words are done to train a theta for the test document and the remaining 1/4 are used to compute perplexity. The testing runs 40 Gibbs cycles with a burnin of 10 cycles. Basic diagnostics are reported. After the 1000 cycles, a restart then runs Gibbs for 200 cycles and theta, phi and alpha are estimated at that stage. Finally, a full diagnostic report is done (without holdout testing, though) to report on the words in the topics using the estimated phi.

```
hca -Bdir -K$K -C1000 -v -q4 -c100 -T$T -Llike,40,10 -hdoc,4 $DATA $STEM
hca -r0 -v -C200 -ltheta,5,1 -lphi,5,1 -lalpha,5,1 -q4 -T$T $DATA $STEM
hca -r0 -v -v -V -C0 -ophi -rphi -T$T $DATA $STEM
```

Using the holdout testing during checkpointing, we get both a training set and a test set perplexity computed for every 100 cycles of training.

This second example tests out burstiness. It runs three versions with testing: with burstiness using PYP discount (parameter `ad`) fixed at 0, burstiness using PYP discount (parameter `ad`) fixed at 0.5, and no burstiness. Files are saved in a common directory. Note the Stirling number tables are also initialised with larger values (60000,3000).

```
hca -N60000,3000 -K$K -C2000 -q4 -v -hdoc,4 -T$T -Sbdk=100 $DATA $STEM/npb
hca -N60000,3000 -K$K -C2000 -q4 -v -hdoc,4 -T$T -Sbdk=100 -Sad=0.5 -Fad $DATA $STEM/npba
hca -N60000,3000 -K$K -C2000 -q4 -v -hdoc,4 -T$T $DATA $STEM/np
```

8 Errors

There is some error reporting on failure.

If the software quits during a run on larger data with an error message like:

```
S_V(N,M,A) tagged 'XXX' hit bounds (BN,BM)
```

for integers `N,M` and label `XXX` then you need to increase the bounds `BN,BM`. If only the `BM` bound is violated, then set `BN` to its default (10000) and increase `BM` to, say 5000 (your choice) with the option **-N***10000,5000*. The `BN` bound should only be violated when the Beta side table is affected, in which case the label will be XXX="SB, topicXword PYP". Now increase `BN` to, say 30000 (your choice) with the option **-N***30000,1000*, leaving `BM` as it was.

For other errors, please report to the maintainer. Best bet is to recompile with "MYDEBUG=-g" set in the Makefile and possibly run under a memory checker to get details of the reason for the crash.

9 See Also

The command *linkBags*(1) is available from *text-bags* at `https://github.com/wbuntine/text-bags` and was previously released at `http://mloss.org`. The extended library *libstb*, parts of which are included, is available individually from `http://mloss.org` also at `https://github.com/wbuntine/libstb` .

10 Version

This programme is version 0.62 of March 25, 2016. This incorporates parts of the library *libstb* version 1.8 also of March 25, 2016.

11 License and Copyright

12 Author

Prof. Wray Buntine
Email: `Wray.Buntine@monash.edu`

Some parts also done by Dr. Jinjing Li and Mr. Swapnil Mishra.