hca

Wray Buntine

June 2, 2014
Version 0.4

Abstract

hca is research software that does various versions of non-parametric topic models using Gibbs sampling including LDA, HDP-LDA, NP-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.

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 is research software, so not all options or combinations of options work correctly. Note that in this release, all the more experimental features have been stripped, so this release contains only moderately well tested components.

The programme runs a Gibbs sampler for a variety of non-parametric topic models. 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 coded into the `pctl.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 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.

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 of hyperparameters

-A*value* Use a symmetric Dirichlet prior for alpha with this `value` for each dimension. The value must be a positive float.

-A*type* A second version of the **-A** option with a string argument. It defines an alpha prior of `type` as follows:

- **hdp** alpha is modelled with a symmetric Dirichlet and theta is modelled with a Dirichlet Process.
- **hpdd** alpha is modelled with a (truncated) GEM and theta is modelled with a Pitman-Yor Process. This is the default.
- **pdp** alpha is uniform and theta is modelled with a Pitman-Yor Process.

-B*value* Use a symmetric Dirichlet prior with this `value` for each dimension. The value must be a positive float. Note the value stored internally and printed is the total of this over the number of words W.

-B*type* Second form of the **-B** option. Use a beta prior of `type` "hdp" "hpdd" or "pdp". Similar to the **-A** option.

- **hdp** beta is modelled with a Dirichlet, by default symmetric, or its mean can be set with the **-u** option, and phi is modelled with a Dirichlet Process.
- **hpdd** beta is modelled with a (truncated) GEM and phi is modelled with a Pitman-Yor Process. This is the default.
- **pdp** beta is by default uniform, or it can be set with the **-u** option, and phi is modelled with a Pitman-Yor Process. 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.

-u*file* the value for the beta vector is given by the text `file`. It should contain W floats, W being the number of words.

3.2 Controlling sampling of hyperparameters

-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,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.

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

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

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

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

-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.

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.*

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.

fract then the `fract` proportion at the tail of the document is held out. The initial proportion is used in estimating.

-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:

sp Estimate topic sparsity in the theta matrix for the words given in `DataStem.smap`. Results placed in `RepStem.smap`. The report gives "topic/weight" for topics including the word.

prog How often to do the standard diagnostic reports (default is every 5-th cycle).

phi Estimate the word probability vector for each topic. Stored in the `RepStem.phi` file.

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.

-o*score* Scoring rule to pick top words for printing. Methods are 'count', 'idf', 'cost' and 'phi'. Default is 'idf'.

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

-p Report topic coherency in the log file, and save results in the parameter 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.

-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 1.

`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.beta` If a constant beta vector is specified using the **-u** option, this saves the value, for possible use in a restart.

`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.top` 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.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.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 LDA statistics, and 0 if burstiness modelling says the word is a burst so does not contribute to LDA 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 mode is a symmetric 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.

7 Examples

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 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 line means:

"-v": use level one verbosity;

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

"-K20": use 20 topics (the truncation level);

"-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.

The parameters as they are sampled will be printed on a line beginning wth "Par:". At the end, the top 20 words will be printed and the final training perplexity printed. This is based on the posterior probability, not word probability estimates.

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!

7.2 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 command line means:

"-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.

This will have lines like:

```
Topic 15 (p=5.88%,ws=71.8%,ds=69.4%) words =,museum,salonika,byzantine
```

which means the topic is observed in 5.88% of word occurrences, word sparsity for the topic is 71.8% (71.8% of the words have zero data for this topic), and document sparsity for the topic is 69.4% (69.4% of the documents have zero data for this topic).

7.3 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 -lsp,2,1 -ltheta,2,1,0.001 -C20 data/ch c1
```

"-lsp2,1": 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 log-2 entropy of the topic distribution based on these counts is 1.149816.

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.

7.4 Run with testing

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 is:

```
PMI ::  0:2.051 1:3.172 2:5.170 3:0.353 ...
PMI = 1.257
```

The first line gives the PMI per topic. The second line gives the mean PMI.

7.5 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.

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. If the tag `XXX` is "SX, docXtopic PYP" then increase the bound `BM` using the option **-N***BM* with an increased `BM`. If the tag `XXX` is "SX, docXtopic PYP" then increase both bounds `BN,BM` using the option **-N***BM,BN* (note the order of the bounds).

For other errors, please report to the maintainer.

9 See Also

The command *linkBags*(1) is available from `http://mloss.org` in the software *text-bags* at `https://github.com/wbuntine/text-bags` . 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 0.4 of June 2, 2014. This incorporates parts of the library *libstb* version 1.8 also of June 2, 2014.

11 License and Copyright

12 Author

Prof. Wray Buntine
Email: `Wray.Buntine@monash.edu`
Some parts also done by Dr. Jinjin Li and Mr. Swapnil Mishra.