scHPF

Release 0.4.0

Hanna Mendes Levitin

Aug 25, 2020

SETUP

Single-cell Hierarchical Poisson Factorization (scHPF) is a tool for *de novo* discovery of discrete and continuous expression patterns in single-cell RNA-sequencing (scRNA-seq).

We find that scHPF's sparse low-dimensional representations, non-negativity, and explicit modeling of variable sparsity across genes and cells produces highly interpretable factors. The algorithm takes genome-wide molecular counts as input, avoids prior normalization, and has fast, memory-efficient inference on sparse scRNA-seq datasets.

Algorithmic details, benchmarking against alternative methods, and scHPF's application to a spatially sampled high-grade glioma can be found in [**?**].

You can find the software on github.

CHAPTER ONE

INSTALLATION

1.1 Environment & Dependencies

scHPF requires Python >= 3.6 and the packages:

- numba (*version requirement depends on python version*, but will be safe with 0.45, and probably 0.45+)
- scikit-learn
- pandas
- (optional) loompy

The easiest way to setup a python environment for scHPF is with anaconda (or its stripped-down version miniconda):

```
conda create -n schpf_p37 python=3.7 scikit-learn numba=0.45 pandas

# for newer anaconda versions
conda activate schpf_p37
# XOR older anaconda verstions
source activate schpf_p37

# Optional, for using loom files as input to preprocessing
pip install -U loompy
```

1.1.1 numba compatibility

Certain micro-versions of Python and numba do not play well together, resulting in segmentation faults and/or horrible performance (at least for the ops scHPF uses). In our experience, micro-version combos that avoid these issues are listed below, as well as known-bad combination, but note this is not an exhaustive list:

Python 3.7.9 Compatible numba: 0.45-0.50 DO NOT USE: 0.44 or earlier

Python 3.7.5 - 3.7.8 Not tested

Python 3.7.4 Compatible numba: 0.44, 0.45 DO NOT USE: 0.43 or earlier

Python <=3.7.3 Compatible numba: 0.39, 0.40, 0.44, 0.45 DO NOT USE: 0.41-0.43

Please let us know about any weird errors/slowness your experience so we can document!

1.2 Installing scHPF

Once you have set up the environment, clone `simslab/scHPF` from github and install.

```
git clone git@github.com:simslab/scHPF.git
cd scHPF
pip install .
```

CHAPTER

TWO

GENE LISTS

2.1 About

We recommend restricting analysis to protein-coding genes, and bundle premade lists of coding genes for human and mouse with the scHPF code. The *prep CLI command* optionally uses these lists to filter input data. Although ENSEMBL ids are theoretically unambiguous and consistent across releases (ie stable identifiers), you may want to generate your own list from a different annotation (that matches your alignment GENCODE version) or with different parameters for gene inclusion (eg including lncRNA).

2.2 Premade lists

The scHPF code includes tab-delimited lists of ENSEMBL ids and names for genes with protein coding, T-cell receptor constant, or immunoglobulin constant biotypes for human and mouse.

Premade lists can be found in the code's resources folder:

- Human (GENCODE v24, v29, v31)
- Mouse (GENCODE vM10, vM19)

2.3 Format

Example tab-delimited gene list:

```
ENSG00000186092     OR4F5
ENSG00000284733     OR4F29
ENSG00000284662     OR4F16
ENSG00000187634     SAMD11
ENSG00000188976     NOC2L
ENSG00000187961     KLHL17
```

By default, the prep command assumes a two-column, tab-delimited text file of ENSEMBL gene ids and names, and uses the first column (assumed to be ENSEMBL id) to filter genes. See the *prep command documentation* for other options.

Note: ENSEMBL ids may end in a period followed by an unstable version number (eg ENSG00000186092.6). By default, the prep command ignores anything after the period. This means `[ENS-ID].[VERSION]` is equivalent to `[ENS-ID]`. See the *prep command* for other options.

2.4 Making custom gene lists

Although ENSEMBL ids aim to be unambiguous and consistent across releases (ie stable identifiers), you may want to generate your own list from a different annotation or with different parameters for gene inclusion.

2.4.1 Example creation script

Reference files of ids and names for genes with with `protein_coding`, `TR_C_gene`, or `IG_C_gene` biotypes in the GENCODE main annotation (in this case `gencode.v29.annotation.gtf`) were generated as follows:

```
# Select genes with feature gene and level 1 or 2
awk '{if($3=="gene" && $0~"level (1|2);"){print $0}}' gencode.v29.annotation.gtf >␣
↪gencode.v29.annotation.gene_l1l2.gtf

# Only include biotypes protein_coding, TR_C_g* and IG_C_g*
awk '{if($12~"TR_C_g" || $12~"IG_C_g" || $12~"protein_coding"){print $0}}' gencode.
↪v29.annotation.gene_l1l2.gtf > gencode.v29.annotation.gene_l1l2.pc_TRC_IGC.gtf

# Retrieve ENSEMBL gene id and name
awk '{{OFS="\t"}{gsub(/"/, "", $10); gsub(/;/, "", $10); gsub(/"/, "", $14); gsub(/;/,
↪ "", $14); print $10, $14}}' gencode.v29.annotation.gene_l1l2.pc_TRC_IGC.gtf >␣
↪gencode.v29.annotation.gene_l1l2.pc_TRC_IGC.stripped.txt
```

Note: For older GENCODE versions, you may need to adjust the field indices in the third line of code (for example changing all instances of $14 to $16).

CHAPTER

THREE

SCHPF PREP

3.1 Basic usage

To preprocess genome-wide UMI counts for a typical run, use the command:

```
scHPF prep -i UMICOUNT_MATRIX -o OUTDIR -m 10 -w WHITELIST
```

As written, the command prepares a *matrix of molecular counts* for training and only includes genes that are:

- on a *whitelist*, for example one of the lists of protein coding genes bundled in the scHPF code's reference folder (`-w`/`--whitelist`)
- that we observe in at at least 10 cells (`-m`/`--min-cells`).

After running this command, `OUTDIR` should contain a matrix market file, `filtered.mtx`, and an ordered list of genes, `genes.txt`. An optional prefix argument can be added, which is prepended to to the output file names.

Now we can train the model with the `scHPF train` utility.

3.2 Input matrix format

`scHPF prep` takes a molecular count matrix for an scRNA-seq experiment and formats it for training. The input matrix has two allowed formats:

1. A **whitespace-delimited matrix** formatted as follows, with no header:

   ```
   ENSEMBL_ID    GENE_NAME    UMICOUNT_CELL0    UMICOUNT_CELL1 ...
   ```

2. A **loom file** (see loompy docs). The loom file must have at least one of the row attributes `Accession` or `Gene`, where `Accession` is an ENSEMBL id and `Gene` is a gene name.

3.3 Whitelisting genes

3.3.1 About

We recommend restricting analysis to protein-coding genes. The `-w`/`--whitelist` option removes all genes in the input data that are *not in* a two column, tab-delimited text file of ENSEMBL gene ids and names. Symmetrically, the `-b`/`--blacklist` option removes all genes that are *in* a file.

Whitelists for human and mouse are provided in the resources folder, and details on formatting and custom lists are in the *gene list documentation*.

Attention: ENSEMBL ids may end in a period followed by an unstable version number (eg ENSG00000186092.6). By default, the prep command ignores anything after the period. This means `[ENS-ID].[VERSION]` is equivalent to `[ENS-ID]`. This behavior can be overwritten with the `--no-split-on-dot` flag.

3.3.2 Whitespace-delimited input matrix

For whitespace-delimited UMI-count files, filtering is performed using the input matrix's first column (assumed to be a unique identifier) by default, but can be done with the gene name (next column) using the `--filter-by-gene-name` flag. This is useful for data that does not include a gene id.

3.3.3 loom input matrix

For loom files, we filter the loom `Accession` row attribute against the whitelist's ENSEMBLE if `Accession` is present in the loom's row attributes, and filter the loom's `Gene` row attribute against the gene name in the whitelist otherwise.

3.4 Complete options

For complete options, see the *complete CLI reference* or use the `-h` option on the command line:

```
scHPF prep -h
```

CHAPTER

FOUR

SCHPF TRAIN

4.1 Basic usage

A typical command to train an scHPF model (using data prepared by the `scHPF prep` command):

```
scHPF train -i TRAIN_FILE -o OUTDIR -p PREFIX -k 7 -t 5
```

This command performs approximate Bayesian inference on scHPF with, in this instance, seven factors and five different random initializations. scHPF will automatically select the trial with the lowest negative log-likelihood, and save the model in the OUTDIR in a serialized joblib file.

4.2 Input file format

scHPF's train command accepts two formats:

1. **Matrix Market (.mtx) files**, where rows are cells, columns are genes, and values are nonzero molecular counts. Matrix market files are output by the current `scHPF prep` command.
2. **Tab-delimited COO matrix coordinates**, output by a previous version of the preprocessing command. These files are essentially the same as .mtx files, except they do not have a header and are zero indexed.

4.3 Debugging

Hint: If you get an error like "Inconsistency detected by ld.so: dl-version.c: 224: _dl_check_map_versions" and are running numba 0.40.0, try downgrading to 0.39.0.

Hint: If you get an error like "Segmentation fault (core dumped)" and are running Python 3.7.4, try upgrading numba to version 0.45 or downgrading Python to 3.7.3 python *[More details]*

4.4 Complete options

For complete options, see the *complete CLI reference* or use the `-h` option on the command line:

```
scHPF train -h
```

CHAPTER

FIVE

SCHPF SCORE

5.1 Basic usage

To get gene- and cell-scores in a tab-delimited file, ordered like the genes and cells in the train file and with a column for each factor:

```
scHPF score -m MODEL_JOBLIB -o OUTDIR -p PREFIX
```

To also generate a tab-delimited file of gene names, ranked by gene-score for each factor:

```
scHPF score -m MODEL_JOBLIB -o OUTDIR -p PREFIX -g GENE_FILE
```

`GENE_FILE` is intended to be the gene.txt file output by the `scHPF prep` command, but can in theory be any tab-delimited file where the number of rows is equal to the number of genes in the scHPF model. The score command automatically uses the 1st (zero-indexed) column of `GENE_FILE` (or the only column if there is only one); however, the column used can be specified with `--name-col`.

If `OUTDIR` is omitted, the command will make a new subdirectory of the directory containing the model. The new subdirectory will have the same name as the model file, but without the joblib extension.

The command also outputs files which can be used to *select the number of factors* using trained models.

5.2 Complete options

For complete options, see the *complete CLI reference* or use the `-h` option on the command line:

```
scHPF score -h
```

CHAPTER

SIX

SELECTING *K*

6.1 General comments

The number of factors, K, determines scHPF's granularity. An appropriate number of factors depends on both the data being fit and the intended application of the scHPF model. In our experience, subsequent analyses on cell scores (eg. UMAP) are stable across a reasonable range of K, while interpretability (gene scores) can be more K-dependent.

6.2 Example workflows

6.2.1 1. Exploratory analysis on a single sample

In some cases, if a user has a single sample, it may be appropriate to increase or decrease K manually according to the desired resolution. Granularity at the level of expression programs can be assessed qualitatively using the per-factor ranked gene lists in *ranked_genes.txt* (from `scHPF score` with the `-g` option). For example, if genes for two cell types appear in the same factor, one might increase K. Resolution can also be assessed quantitatively using *cell type representation*, or *other quantitative criteria*.

When using this approach, we encourage the user to always try at least two values of K in any direction, as scHPF is multimodal and behavior is not always monotonic. K in the neighborhood of the number of clusters is often a good starting point.

6.2.2 2. Consistent choices across multiple models

Applying scHPF separately to multiple partitions (as in [**?**]) necessitates a uniform procedure for choosing the number of factors. To maximize interpretability while being quantitative and consistent across models, we usually train scHPF across a range of K's for each partition and select the per-dataset number of factors using a heuristic suitable to our intended application (*example criteria*). An example workflow might be:

1. Choose an appropriate selection criteria for the problem at hand (*examples*).
2. Guess a minimum number of factors, K_{min}. Values slightly less than the number of clusters in the dataset are usually a good starting point (e.g. K_{min} = number of clusters - 2). Guess a maximum number of factors, K_{max}, not worrying too much if we are low since we'll refine later (e.g. $K_{\text{max}} = K_{\text{min}} + 8$).
3. *Train* scHPF models for K in range(K_{min}, K_{max} +1). *Advanced note*: I sometimes use a step size of 2 or 3 on the first pass to check that the range is reasonable, but recommend a final step of 1 (scHPF is multimodal, so results may not be monotonic).
4. Evaluate the models using the selection criteria from 1. Expand/refine the range accordingly. For example, if K_{max} passes our criteria, we should increase K_{max}.

5. Repeat 3-5 as needed.

6.3 Example selection criteria

6.3.1 1. Cell type representation

In [**?**], we chose *K* based on scHPF's representation of cell types in the data. Specifically, we selected the smallest *K* such that every well-defined cluster was most strongly associated with at least one unique factor [Levitin2019, Appendix Figure S8]. This method is intuitive, and can work well when many cell types are present, but depends on the quality and granularity of clustering. It is also difficult to standardize across multiple models trained on different data.

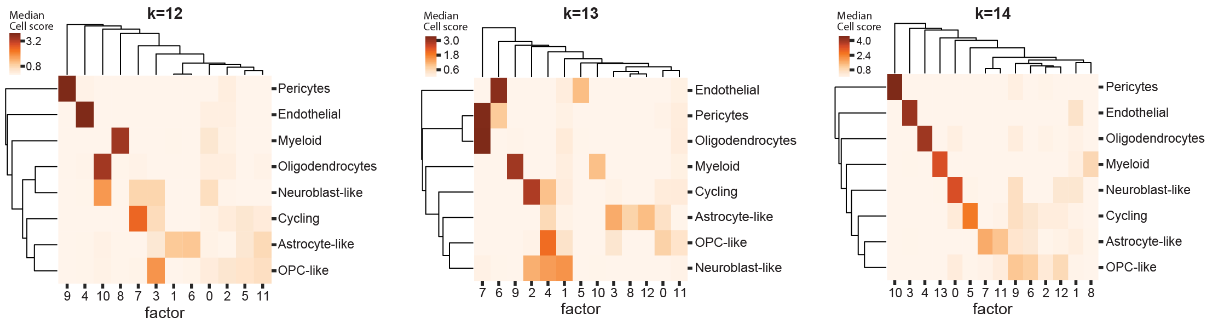
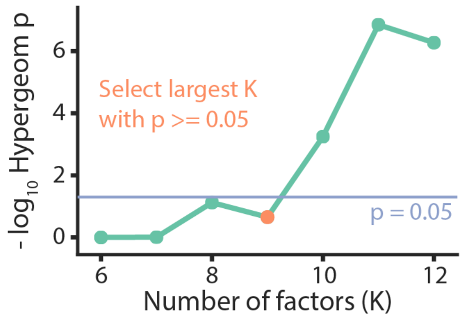


Fig. 1: Median cell score per factor and cluster in a high-grade glioma for 12, 13, and 14 factors in [**?**]. At 14 factors, all clusters are most closely associated with at least one unique factor.

Evaluating top gene overlap



Hypergeometric -log10 *p*-value of the maximum pairwise overlap of the highest scoring genes in each factor for Donor 2 Bone Marrow in [**?**] at different values of *K*.

6.3.2 2. Gene signature overlap

To find common patterns of gene expression across multiple models in [**?**], we selected K such that factors in the same model did not have significant overlap in their top genes (where top genes are defined as the n highest scoring genes per factor). This reflected our prior that programs should be distinctive with respect to gene scores, and the further requirement that models should have similar granularity across datasets with different levels of complexity.

The `scHPF score` command automatically produces the file *maximum_overlaps.txt*, which contains factors' maximum pairwise overlap and corresponding hypergeometric p values at different cutoffs.

For standard significance thresholds and reasonable n, this method can be quite strict, resulting in lower granularity factorizations for some datasets. Using *cellular resolution* or *cell type respresentation* may find higher resolution factorizations in these cases.

6.3.3 3. Cellular resolution

Cellular resolution directly evaluates a model's granularity by specifying how many factors, on average, should explain a given portion of a cell's total cell scores. We have found it especially useful for datasets where *gene signature overlap* is too strict.

We define cellular resolution as the maximum K such that, on average, cells' n highest scoring factors contain at least r*100 percent of their total score across all factors. So if we want to find a model where the 3 factors with the highest score in a cell contain at least 70% of its total score (on average), n would be 3 and r would be 0.7.

We can evaluate cellular resolution using one of `scHPF score`'s outputs, a file called *mean_cellscore_fraction.txt* (potentially with a prefix). The file's two columns, *nfactors* and *mean_cellscore_fraction*, represent the mean fraction of each cell's total cell score allocated to its top *nfactors* factors. If we want to find a model at n =3 and r =0.7 resolution, we might follow the *example workflow* above, and select the largest K such that *mean_cellscore_fraction* >= 0.7 when *nfactors* = 3.

6.3.4 4. Other metrics

Coming soon

CHAPTER SEVEN

PROJECTING DATA ONTO A TRAINED MODEL

Full writeup coming soon. Use the `prep-like` and `project` commandline programs.

7.1 Preparing data for projection

For complete options, see the *complete CLI reference* or use the `-h` option on the command line:

```
scHPF prep-like -h
```

7.2 Projecting new data

For complete options, see the *complete CLI reference* or use the `-h` option on the command line:

```
scHPF project -h
```

CHAPTER

EIGHT

COMPLETE CLI REFERENCE

8.1 scHPF prep

```
usage: scHPF prep [-h] -i INPUT [-o OUTDIR] [-p PREFIX] [-m MIN_CELLS]
                  [-w WHITELIST] [-b BLACKLIST] [-nvc N_VALIDATION_CELLS]
                  [-vgid VALIDATION_GROUP_IDS]
                  [--validation-max-group-frac VALIDATION_MAX_GROUP_FRAC]
                  [--filter-by-gene-name] [--no-split-on-dot]
```

8.1.1 Named Arguments

-i, --input	Input data. Currently accepts either: (1) a whitespace-delimited gene by cell UMI count matrix with 2 leading columns of gene attributes (ENSEMBL_ID and GENE_NAME respectively), or (2) a loom file with at least one of the row attributes *Accession* or *Gene*, where *Accession* is an ENSEMBL id and *Gene* is the name.
-o, --outdir	Output directory. Does not need to exist.
-p, --prefix	Prefix for output files. Optional. Default: “”
-m, --min-cells	Minimum number of cells in which we must observe at least one transcript of a gene for the gene to pass filtering. If 0 <*min_cells`< 1, sets threshold to be `min_cells* * ncells, rounded to the nearest integer. [Default 0.01] Default: 0.01
-w, --whitelist	Tab-delimited file where first column contains ENSEMBL gene ids to accept, and second column contains corresponding gene names. If given, genes not on the whitelist are filtered from the input matrix. Superseded by blacklist. Optional. Default: “”
-b, --blacklist	Tab-delimited file where first column contains ENSEMBL gene ids to exclude, and second column is the corresponding gene name. Only performed if file given. Genes on the blacklist are excluded even if they are also on the whitelist. Optional. Default: “”
-nvc, --n-validation-cells	Number of cells to randomly select for validation. Default: 0

-vgid, --validation-group-ids Single column file of cell group ids readable with np.readtxt. If *–n-validation-cells* is > 0, cells will be randomly selected approximately evenly across the groups in this file, under the constraint that at most *–validation-min-group-frac* * (ncells in group) are selected from every group.

--validation-max-group-frac If *-nvc*>0 and *validation-group-ids* is a valid file, at most `validation-min-group-frac`*(ncells in group) cells are selected from each group.

Default: 0.5

--filter-by-gene-name Use gene name rather than ENSEMBL id to filter (with whitelist or blacklist). Useful for datasets where only gene symbols are given. Applies to both whitelist and blacklist. Used by default when input is a loom file (unless there is an Accession attribute in the loom).

Default: False

--no-split-on-dot Don't split gene symbol or name on period before filtering whitelist and blacklist. We do this by default for ENSEMBL ids.

Default: False

8.2 scHPF train

```
usage: scHPF train [-h] -i INPUT [-o OUTDIR] [-p PREFIX] [-t NTRIALS]
                   [-v VALIDATION_CELLS] [-M MAX_ITER] [-m MIN_ITER]
                   [-e EPSILON] [-f CHECK_FREQ]
                   [--better-than-n-ago BETTER_THAN_N_AGO] [-a A] [-c C]
                   [--float32] [-bs BATCHSIZE] [-sl SMOOTH_LOSS] [-bts] [-sa]
                   [-rp] [--quiet]
                   nfactors
```

8.2.1 Named Arguments

-i, --input Training data. Expects either the mtx file output by the prep command or a tab-separated tsv file formatted like:*CELL_ID GENE_ID UMI_COUNT*. In the later case, ids are assumed to be 0 indexed and we assume no duplicates.

-o, --outdir Output directory for scHPF model. Will be created if does not exist.

-p, --prefix Prefix for output files. Optional.

Default: ""

nfactors Number of factors.

-t, --ntrials Number of times to run scHPF, selecting the trial with best loss (on training data unless validation is given). [Default 1]

Default: 1

-v, --validation-cells Cells to use to assess convergence and choose a model. Expects same format as `-i/--input`. Training data used by default.

-M, --max-iter Maximum iterations. [Default 1000].

Default: 1000

-m, --min-iter	Minimum iterations. [Default 30]
	Default: 30
-e, --epsilon	Minimum percent decrease in loss between checks to continue inference (convergence criteria). [Default 0.001].
	Default: 0.001
-f, --check-freq	Number of iterations to run between convergence checks. [Default 10].
	Default: 10
--better-than-n-ago	Stop condition if loss is getting worse. Stops training if loss is worse than *better_than_n_ago`*`check-freq* training steps ago and getting worse. Normally not necessary to change.
	Default: 5
-a	Value for hyperparameter a. [Default 0.3]
	Default: 0.3
-c	Value for hyperparameter c. [Default 0.3]
	Default: 0.3
--float32	Use 32-bit floats instead of default 64-bit floats in variational distrubtions
	Default: False
-bs, --batchsize	Number of cells to use per training round. All cells used if 0. Note that using batches changes the order of updates during inference.
	Default: 0
-sl, --smooth-loss	Average loss over the last *–smooth-loss* interations. Intended for when using minibatches, where int(ncells/batchsize) is a reasonable value
	Default: 1
-bts, --beta-theta-simultaneous	If False (default), compute beta update, then compute theta based on the updated beta. Note that if batching is used, this order is reverse. If True, update both beta and theta based on values from the last training round. The later slows the rate of convergence and sometimes results in better log-likelihoods, but may increase convergence time, especially for large numbers of cells.
	Default: False
-sa, --save-all	Save all trials
	Default: False
-rp, --reproject	Reproject data onto fixed global (gene) parameters after convergence, but before model selection. Recommended with batching
	Default: False
--quiet	Don’t print intermediate llh.
	Default: True

8.3 scHPF score

```
usage: scHPF score [-h] -m MODEL [-o OUTDIR] [-p PREFIX] [-g GENEFILE]
                   [--name-col NAME_COL]
```

8.3.1 Named Arguments

-m, --model	Saved scHPF model from train command. Should have extension`.joblib`
-o, --outdir	Output directory for score files. If not given, a new subdirectory of the dir containing the model will be made with the same name as the model file (without extension)
-p, --prefix	Prefix for output files. Optional. Default: “”
-g, --genefile	Create an additional file with gene names ranked by score for each factor. Expects the gene.txt file output by the scHPF prep command or a similarly formatted tab-delimited file without headers. Uses the zero-indexed `--name_col`’th column as gene names. Optional.
--name-col	The zero-indexed column of *genefile* to use as a gene name when (optionally) ranking genes. If `--name_col` is greater than the index of `--genefile`’s last column, it is automatically reset to the last column’s index. [Default 1] Default: 1

8.4 scHPF prep-like

```
usage: scHPF prep-like [-h] -i INPUT -r REFERENCE -o OUTDIR [-p PREFIX]
                       [--by-gene-name] [--no-split-on-dot]
```

8.4.1 Named Arguments

-i, --input	Input data to format. Currently accepts either: (1) a whitespace-delimited gene by cell UMI count matrix with 2 leading columns of gene attributes (ENSEMBL_ID and GENE_NAME respectively), or (2) a loom file with at least one of the row attributes *Accession* or *Gene*, where *Accession* is an ENSEMBL id and *Gene* is the name.
-r, --reference	Two-column tab-delimited file of ENSEMBL ids and gene names to select from *input* and order like. All genes in *reference* must be present in *input*.
-o, --outdir	Output directory. Does not need to exist.
-p, --prefix	Prefix for output files. Optional. Default: “”
--by-gene-name	Use gene name rather than ENSEMBL id to when matching against reference. Useful for datasets where only gene symbols are given. Used by default when input is a loom file (unless there is an Accession attr in the loom). Default: False

--no-split-on-dot	Don't split gene symbol or name on period before when matching to reference. We do this by default for ENSEMBL ids. Default: False

8.5 scHPF project

```
usage: scHPF project [-h] -m MODEL -i INPUT [-o OUTDIR] [-p PREFIX]
                     [--recalc-bp] [--max-iter MAX_ITER] [--min-iter MIN_ITER]
                     [--epsilon EPSILON] [--check-freq CHECK_FREQ]
```

8.5.1 Named Arguments

-m, --model	The model to project onto.
-i, --input	Data to project onto model. Expects either the mtx file output by the prep or prep-like commands or a tab-delimitted tsv file formated like: *CELL_ID GENE_ID UMI_COUNT*. In the later case, ids are assumed to be 0 indexed and we assume no duplicates.
-o, --outdir	Output directory for projected scHPF model. Will be created if does not exist.
-p, --prefix	Prefix for output files. Optional. Default: ""
--recalc-bp	Recalculate hyperparameter bp for the new data Default: False
--max-iter	Maximum iterations. [Default 500]. Default: 500
--min-iter	Minimum iterations. [Default 10] Default: 10
--epsilon	Minimum percent decrease in loss between checks to continue inference (convergence criteria). [Default 0.001]. Default: 0.001
--check-freq	Number of iterations to run between convergence checks. [Default 10]. Default: 10

8.6 scHPF train-pool

```
usage: scHPF train-pool [-h] -i INPUT [-o OUTDIR] [-p PREFIX] [-t NTRIALS]
                        [-v VALIDATION_CELLS] [-M MAX_ITER] [-m MIN_ITER]
                        [-e EPSILON] [-f CHECK_FREQ]
                        [--better-than-n-ago BETTER_THAN_N_AGO] [-a A] [-c C]
                        [--float32] [-bs BATCHSIZE] [-sl SMOOTH_LOSS] [-bts]
                        [-sa] [-rp] [--quiet] [--njobs NJOBS] -k NFACTORS
                        [NFACTORS ...]
```

8.6.1 Named Arguments

-i, --input	Training data. Expects either the mtx file output by the prep command or a tab-separated tsv file formatted like:*CELL_ID GENE_ID UMI_COUNT*. In the later case, ids are assumed to be 0 indexed and we assume no duplicates.
-o, --outdir	Output directory for scHPF model. Will be created if does not exist.
-p, --prefix	Prefix for output files. Optional. Default: “”
-t, --ntrials	Number of times to run scHPF, selecting the trial with best loss (on training data unless validation is given). [Default 1] Default: 1
-v, --validation-cells	Cells to use to assess convergence and choose a model. Expects same format as `-i/--input`. Training data used by default.
-M, --max-iter	Maximum iterations. [Default 1000]. Default: 1000
-m, --min-iter	Minimum iterations. [Default 30] Default: 30
-e, --epsilon	Minimum percent decrease in loss between checks to continue inference (convergence criteria). [Default 0.001]. Default: 0.001
-f, --check-freq	Number of iterations to run between convergence checks. [Default 10]. Default: 10
--better-than-n-ago	Stop condition if loss is getting worse. Stops training if loss is worse than *better_than_n_ago*`*`*check-freq* training steps ago and getting worse. Normally not necessary to change. Default: 5
-a	Value for hyperparameter a. [Default 0.3] Default: 0.3
-c	Value for hyperparameter c. [Default 0.3] Default: 0.3
--float32	Use 32-bit floats instead of default 64-bit floats in variational distrubtions Default: False
-bs, --batchsize	Number of cells to use per training round. All cells used if 0. Note that using batches changes the order of updates during inference. Default: 0
-sl, --smooth-loss	Average loss over the last *–smooth-loss* interations. Intended for when using minibatches, where int(ncells/batchsize) is a reasonable value Default: 1

-bts, --beta-theta-simultaneous If False (default), compute beta update, then compute theta based on the updated beta. Note that if batching is used, this order is reverse. If True, update both beta and theta based on values from the last training round. The later slows the rate of convergence and sometimes results in better log-likelihoods, but may increase convergence time, especially for large numbers of cells.

Default: False

-sa, --save-all Save all trials

Default: False

-rp, --reproject Reproject data onto fixed global (gene) parameters after convergence, but before model selection. Recommended with batching

Default: False

--quiet Don't print intermediate llh.

Default: True

--njobs Max number of processes to spawn. 0 will use the minimum of all available cores and ntrials.

Default: 0

-k, --nfactors Number of factors.

CHAPTER

NINE

CHANGELOG

9.1 0.4.0

- train-pool for training parallelized at the level of trials rather than computations
- reproject and save all options during training
- add separate joblib dependency (should be installed w/scikit-learn, scikit.externals.joblib is deprecated)

9.2 0.3.0

- Refactor so loss can be an arbitrary function
- Fix bugs in and expand options for projetion
- prep-like CLI to prepare data for projection onto a trained model
- cellscore fraction file for score CLI
- Verbose option for load_txt
- Update options for validation cells & selection
- Version as an object attribute
- Handle change in scipy API
- new GENCODE files
- (feature request) options to specify a and c from the train CLI
- Documentation with ReadTheDocs

9.3 0.2.4

- Emergency patch preprocessing error for loom files. Also fixed an errant test. Not really enough to justify a new release but fixed a pretty irritating/embarrassing error.

9.4 0.2.3

- fix no split on dot bug
- Max pairwise table + default max pairwise in score
- Note about ld.so error
- Fix max pairwise second greatest bug
- Some integration tests

9.5 0.2.2

- partial test suite
- max pairwise test for gene overlap
- faster preprocessing of larage text files
- refactor preprocessing and training control flow out of CLI
- move load and save methods outside of scHPF object

9.6 0.2.1

- Slight speedup during inference for Xphi
- Fix bug (occurred first in 0.2.0-alpha) that occurs when genes in whitespace-delim input to prep that have no counts

9.7 0.2.0

Numba implmentation with scikit-learn-like API

9.8 0.1.0

- Tensorflow implementation

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search