

# Using ArborZ

Adam J. Sypniewski (ajsyp@umich.edu) and David W. Gerdes  
University of Michigan

December 17, 2012

## 1 Obtaining and Installing ArborZ

The current ArborZ release can be found as a tarball (`arborz.tar.gz`) in the "Files" section of the ArborZ project wiki:

```
https://cdcv.s.fnal.gov/redmine/projects/arborz
```

ArborZ has no dependencies, making compiling and installing so simple that we currently do not have a makefile. To build ArborZ, we first extract the tarball and move into the newly extracted directory:

```
> tar -xvzf arborz.tar.gz && cd arborz
```

We compile ArborZ directly:

```
> g++ -o arborz *.cpp
```

Upon successful completion of this command, the executable "arborz" is created in the `arborz` directory. The source files can be deleted.

## 2 Using ArborZ

ArborZ uses boosted decision trees (BDTs) to determine the empirical mapping from observables (such as magnitudes or colors) to redshift. Like most machine-learning algorithms, ArborZ determines this mapping during a "training" step. The resulting mapping can then be applied in subsequent "evaluation" steps (also called "target" steps) to produce photometric redshift estimates for a galaxy catalog. We discuss the training and evaluation steps separately.

### 2.1 Training

#### 2.1.1 Basic Usage

The simplest form for training ArborZ is:

```
./arborz --state STATE_FILE --train TRAIN_FILE.tsv COLUMN_SPECIFICATION
```

Here, `STATE_FILE` is the name of the "state file" which is created during training. The state file contains the empirical mapping from observable space to redshift space. It is needed during the evaluation phase to produce photometric redshift estimates. Note that ArborZ will silently overwrite any existing state files of the same name, so be careful.

The `TRAIN_FILE.tsv` argument is the name of file which contains the training data for ArborZ. It should be a tab-spaced value (TSV) file which can contain an arbitrary number of header lines. Although this file can contain many columns (and types) of data—even data not used for training—it must contain the training observables (numeric values only) and the true redshift for each galaxy entry. Any lines encountered in this file which cannot be properly parsed, or which contain invalid redshifts<sup>1</sup>, are silently dropped from the training process.

`COLUMN_SPECIFICATION` indicates which columns of the training file should be used during training. This specification consists of a sequence of zero-based column indices, which can be a range (specified using the dash), a single number, or a combination of both (separated by a comma). The last column specified is assumed to be the true redshift. Note that there should be no spaces in the column specification.

---

<sup>1</sup>An invalid redshift  $z$  is defined as a redshift where  $z \leq 0$  or  $z \geq 20$ .

For example, if the training file’s first five columns are training observables (such as magnitudes) and the sixth column is the redshift column, then `COLUMN_SPECIFICATION` could be: `0-5`. If the redshift column is the first column and you wish to train on the values of the second, fourth, fifth, and sixth columns, `COLUMN_SPECIFICATION` could be either `1,3-5,0` or `1,3,4,5,0` (since the column indices are zero-based).

### 2.1.2 Advanced Usage

There are also several more advanced options which can be specified during training. The options, their descriptions, and default values (if they aren’t specified) are listed here.

- **Header Lines.** If the training file contains header lines, you can tell ArborZ to skip these lines using the skip option: `--skip NUM_LINES`, where `NUM_LINES` is the number of non-data lines at the beginning of the file. Default value: 0.
- **Binning Options.** ArborZ partitions the training set into true redshift bins and it uses these same bins during the evaluation phase to construct its output PDF and photometric redshift estimates. The binning options can be specified with: `--bin BIN_METHOD NUM_BINS`. Here, `NUM_BINS` indicates the number of bins to use.<sup>2</sup> `BIN_METHOD` tells ArborZ how to choose the bin edges, and can be either `fixed`, to indicate that “`NUM_BINS`” bins of equal width should be used, or `variable`, to indicate that bin edges should be chosen so that every bin has the same number of galaxies in it. Default value: 64 fixed-width bins.
- **Binning File.** This option can be used to tell ArborZ to output the bin edges it chooses for training to a file. Generally, it is a good idea to specify this option, as knowing the bin edges explicitly is often useful for post-evaluation analysis. It can be specified with: `--write-bins BIN_FILE`, where `BIN_FILE` is the name of the file to write to. This file will have a single, float-point value per line, and there will be `NUM_BINS + 1` lines total (one for each bin edge, including the lowest and highest edges). Default: None (no binning file is written).
- **Maximum Redshift.** By default, ArborZ uses the full redshift range of the training file for construct its bins. Thus, the lowest bin edge is equal to the redshift of the lowest-redshift training galaxy, and the highest bin edge is equal to the redshift of the highest-redshift training galaxy. If you want to explicitly change the redshift of the highest bin edge, you can use: `--max MAX_REDSHIFT`, where `MAX_REDSHIFT` is the highest redshift bin edge. Note that this is *not* equivalent to applying a training cut, since galaxies with redshifts higher than `MAX_REDSHIFT` may still be used as “background” galaxies during training. Default: Highest redshift in training set.
- **Expected Resolution.** The expected photometric redshift resolution  $\sigma_{\text{res}}$  for the training set. This is used during training to determine—for a given redshift bin—which galaxies should be considered “signal” and which should be considered “background” when presented to that bin’s BDT classifier. ArborZ considers all training objects which fall into a given redshift bin “signal” galaxies, and all training objects whose redshifts fall  $3\sigma_{\text{res}}$  beyond the edges of the same bin “background” objects. As a rule of thumb, when using  $N_{\text{bins}}$  redshift bins from  $z = 0$  to  $z = z_{\text{max}}$ ,  $\sigma_{\text{res}}$  should be chosen such that  $N_{\text{bins}}\sigma_{\text{res}} \approx z_{\text{max}}$ . Default value: 0.02.

## 2.2 Evaluation

### 2.2.1 Basic Usage

The simplest form for evaluating photometric redshifts for a given galaxy catalog is:

```
./arborz --state STATE_FILE --eval TARGET_FILE.tsv OUTPUT_FILE.tsv COLUMN_SPECIFICATION
```

Here, `STATE_FILE` is an ArborZ state file produced by training. If you are content with the training parameters used to create the state file, you can use the same state file on many evaluations; that is, you do not need to re-train ArborZ multiple times if you intend to use the same training parameters multiple times.

`TARGET_FILE.tsv` is a TSV file containing the galaxy catalog to produce photometric redshifts for. Its format is similar to the input training file from the training stage, but it does not need to have any redshift

---

<sup>2</sup>We have had good results with as few as 32 bins and as many as 100.

information present. The observable quantities used in training must be present in this file. The target file can also contain non-data header lines at the top, and lines which cannot be parsed are silently dropped.

`COLUMN_SPECIFICATION` is the sequence of zero-based column indices in the target file to use in evaluation. The column specification follows the same form as from the training step, although the redshift column need not be specified; however, if you are simply performing validation tests and have true redshifts available, you can again have the last column be the redshift column from the target set, and ArborZ will copy this column into the output file for your convenience.

Note that if the column ordering/indices in the target file are different from those in the training file, you will need to change the column specification appropriately. In other words, the first column you specify in training must correspond to the first column specified in evaluation, although the actual indices in the training and evaluation files can be different. For example, if you training on  $g$ -band magnitudes (say, column index 4) and  $r$ -band magnitudes (say, column index 5), then a possible value for `COLUMN_SPECIFICATION` during training could be `4-5,A` (where `A` is the index of the redshift column). Now, if in evaluation  $g$ -band magnitudes are in column 3 and  $r$ -band magnitudes are in column 10, you would set `COLUMN_SPECIFICATION` to `3,10,B` (where `B` is the index of the redshift column, although it can be omitted). The important take-away is that  $g$ -band magnitudes were specified before  $r$ -band magnitudes for both training and evaluation, even through the column indices are different.

`OUTPUT_FILE.tsv` is the file into which the ArborZ photometric redshift estimates are written. The ordering of results corresponds exactly with the ordering in the input target file (although invalid lines from the target file get dropped). It has no header lines, and will always have at least four columns, which correspond to  $z_{\text{phot}}$ ,  $\sigma_z$ ,  $p_{\text{peak}}$ , and  $z_{\text{spec}}$ .  $z_{\text{phot}}$  is the best-estimate photometric redshift for that galaxy, and  $\sigma_z$  is the error on that estimate.  $p_{\text{peak}}$  is a measure of ArborZ’s confidence in the photo- $z$  estimate. It is always between 0 and 1 inclusive (with 1 indicating strong confidence) and is correlated with  $\sigma_z$  (see Gerdes et al, 2010 for more details). If a true redshift column was indicated in the `COLUMN_SPECIFICATION` during evaluation, then  $z_{\text{spec}}$  is set to the redshift value indicated in the target file. If a true redshift column was not specified, then  $z_{\text{spec}} = 0$  for all galaxies.

### 2.2.2 Advanced Usage

There are advanced options which can be specified during evaluation. The options, their descriptions, and default values (if they aren’t specified) are listed here.

- **Header Lines.** If the training file contains header lines, you can tell ArborZ to skip these lines using the skip option: `--skip NUM_LINES`, where `NUM_LINES` is the number of non-data lines at the beginning of the file. Default value: 0.
- **PDF Production.** ArborZ produces a full redshift probability distribution  $p(z)$  for each target galaxy during evaluation. This  $p(z)$  is a PDF histogram<sup>3</sup> whose bin edges are the same as those used in training. This  $p(z)$  production is one of ArborZ’s strong suites. It is not always desirable, however, to write this PDF to the output file, since there are  $N_{\text{bins}}$  additional ASCII floats needed for each target galaxy, which can make for large output files. To enable PDF histogram outputs, use `--hist`. This will result in an output file with  $4 + N_{\text{bins}}$  columns, where the first four are the four described in the previous section, and the last  $N_{\text{bins}}$  columns the PDF. Default: do not write histograms to file.

## 2.3 Additional Usage

Although it is often desirable to separate the training and evaluation steps (so that the state file can be generated once but used many times), ArborZ can also train and evaluate in a single command. To do this, simply specify both the `--train` and `--eval` options in the command line. The `--skip` option can now take two values: `--skip NUM_LINES_TRAIN NUM_LINES_TARGET`, where `NUM_LINES_TRAIN` is the number of non-data lines to skip in the training file, and `NUM_LINES_TARGET` is the number of non-data lines to skip in the target file. If only one argument is given—for example, `--skip NUM_LINES`—then `NUM_LINES` is considered the number of non-data lines to skip in both training and evaluation files.

If you choose to execute both training and evaluation in one step, then it is possible to *not* create a state file (i.e., you are allowed to not specify `--state` at the command line). However, we discourage this usage,

---

<sup>3</sup>By PDF histogram we mean one in which the *area* of a bin, rather than a height, indicates the probability in that bin.

since, in principle, it is no longer possible to recover the observable–redshift mapping if the state file is lost (or not created).

## 2.4 Quick Start

Here is a quick example of a typical training command:

```
./arborz --state TRAIN.state --train TRAIN.tsv 0-5 --bin fixed 32 --write-bin TRAIN.bin --skip 1 --max 1.5 --res 0.04
```

And here is a typical evaluation step:

```
./arborz --state TRAIN.state --eval TARGET.tsv RESULTS.tsv 0-5 --hist --skip 1
```

## 3 Additional Information

We typically use magnitudes with ArborZ. Of course, there is no reason other information cannot be used, and we sometimes use additional information. Feel free to explore the importance and effect of different observables on the output of ArborZ.

## 4 FITS Support

As of version 2.01, ArborZ now supports reading and writing FITS files via the CFITSIO library. Support is still experimental, so bear with us as we perfect this.

### 4.1 Building

ArborZ now support reading and writing FITS files via the CFITSIO library. In order to build ArborZ with FITS support, first download and install CFITSIO. The command to build ArborZ is now slightly longer:

```
> g++ -o arborz *.cpp -I/usr/local/include -L/usr/local/lib -lcfitsio -DFITS
```

The `-L` and `-I` options may be different (or not necessary at all), depending on the setup of your CFITSIO installation. For reference, the `-I` flag is the directory containing `fitsio.h` and `-L` is the directory containing `libcfitsio.a`.

### 4.2 Usage

To use FITS files, simply ignore the `COLUMN_SPECIFICATION` part of the `--train` and `--eval` options. Instead, a new option is introduced: `--fits`. Without any additional arguments to the `--fits` flag, all available columns of the input FITS are printed. Thus, to list all of the FITS columns, data types, and lengths (if the column is a vector), try this:

```
./arborz --train FILE.FITS --fits
```

Specify the actual columns used in training or evaluation by comma-spacing the column names (case-insensitive). So if your FITS file has `MAG_G`, `MAG_R`, and `TRUE_Z` columns, use:

```
--fits MAG_G,MAG_R,TRUE_Z
```

Be sure not to put any spaces between the column names. Just as with TSV files, the last column specified is interpreted as the true redshift column (which may be omitted during evaluation).

ArborZ will properly handle FITS columns which are vectors. If you wish to use the entire vector column, you don't need to do anything special; simply use the column name as you would any other column. If, however, you wish to use parts of the vector, you can do so by placing `COLUMN_SPECIFICATION`-like entries enclosed in square brackets at the end of the column name (except that FITS indices are **one-based**). So if you have a column named `DATA` which is a 5-vector, you can use all five columns like this:

```
--fits DATA
```

If you do this while training, the last component of each `DATA` entry is interpreted as the true redshift. If you wish to use only the first three components of `DATA`, you can do so in many ways:

```
--fits DATA[1,2,3]
```

```
--fits DATA[1],DATA[2],DATA[3]
```

```
--fits DATA[1-3]
```

```
--fits DATA[1-2,3]
```

```
--fits DATA[1-2],DATA[3]
```

The output FITS file produced during evaluation will have at least three columns: ZPHOT, ZPHOTERR, and PEAKPROB. If you specified a true redshift during evaluation, it will be listed in the ZTRUE column. If you used the `--hist` option to output the PDF histogram, it will be in the P\_OF\_Z column (a vector column).

### 4.3 Limitations

Limitations of ArborZ's current FITS support:

- If the `--fits` option is used, then you must split the training and evaluation steps into two discrete commands. You cannot specify both together, which would otherwise allow you to kill two birds with one command, as well as allow you to skip production of the state file.
- You cannot mix file types during evaluation. This means that the input and output files given to `--eval` must both be FITS or both be TSV.
- FITS columns must be of the TDOUBLE type, although they can be TDOUBLE vectors.

Keep in mind that ArborZ's state files are still universal and not bound to any file type. So you can produce a state file by training on TSV data and then use that state file to evaluate against FITS data.

### 4.4 Quick Start

Here is a quick example of a typical FITS-based training command:

```
./arborz --state TRAIN.state --train TRAIN.fits --fits MAG[1-5],ZSPEC --bin fixed 32 --write-bin  
TRAIN.bin --skip 1 --max 1.5 --res 0.04
```

And here is a typical evaluation step:

```
./arborz --state TRAIN.state --eval TARGET.fits RESULTS.fits --fits MAG[1-5],ZSPEC --hist  
--skip 1
```