ANNz User Guide

Adrian Collister

May 14, 2005

1 Overview

ANNz is a tool for estimating photometric redshifts using Artificial Neural Networks (ANNs, see figure 1 and, for an introduction, Bishop 1995, Firth, Lahav & Somerville 2003 and Collister & Lahav 2004). In the usual broad categorisation of photometric redshift methods *ANNz* falls into the empirical camp. ANNs learn the relationship between photometry and redshift from photometry of galaxies whose redshifts have already been determined spectroscopically. Trained networks can then be used on test data, for which there is usually no prior redshift information, to estimate the photometric redshifts.

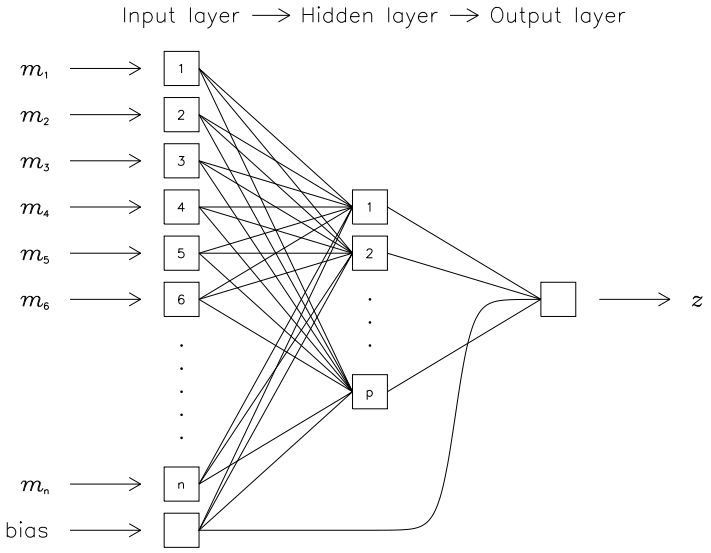

Figure 1: A schematic diagram of an ANN with input nodes taking, for example, magnitudes $m_i = -2.5 \log_{10} f_i$ in various filters, a single hidden layer, and a single output node giving redshift z. The architecture is n:p:1 in the notation used in *ANNz*. Each connecting line carries a weight w_j, the free parameters of the model. The bias node allows for an additive constant in the network function defined at each node. More complex nets can have additional hidden layers.

1.1 Data

We refer to the set of galaxies for which we wish to determine photometric redshifts as the *testing set*. In order to estimate photometric redshifts for these galaxies using *ANNz* we require a suitable *training set*. The training set consists of galaxies for which a reliable spectroscopic redshift is already available. The training and testing sets must have identically calibrated photometry for the same set of filters. The calibration procedure should include corrections for random effects

such as Milky Way reddening, but *ANNz* can be relied on to account for systematic effects such as internal reddening. It is very important that the training set is representative of the testing set, in terms of noise characteristics and sample distribution. For example, a network trained on the brightest galaxies in the sample will not be very successful at determining photometric redshifts for a testing set of very faint galaxies. Thus, as in all empirical photometric redshift techniques, the quality of the training data is of paramount importance.

For reasons explained in section 2.2, a portion of the training data is separated out for use as a *validation set*. This validation set should be chosen at random from the available training data. In summary therefore, three sets of data are required: the *training*, *validation* and *testing* sets. The training and validation sets are used to train the network and must include an estimate for the actual redshift of the galaxies (i.e. the spectroscopic redshift), while the testing set would usually contain galaxies for which no other estimate of redshift is available, and on which the trained network will be used to estimate photometric redshifts.

Data file format The data consists of the magnitudes of each galaxy in each of the filters used (there is no limit on the number of filters) together with the error in each photometric measurement (if these are not available enter `0.0` in place of the error). The final column in the data should be the spectroscopic redshift of each galaxy; this is not usually available for the testing set, but can optionally be included for the purpose of evaluating the performance of *ANNz* on unseen data (see section 2.3).

A section from an example data file, consisting of five-band photometry for six galaxies, is shown below: the first five columns are the magnitudes in the five filters, the next five are the respective photometric errors and the final column is the spectroscopic redshift. No file headers or similar are required. Columns should be delimited by one or more spaces.

```
20.1002 18.0422 17.0871 16.6813 16.4147  0.1130 0.0098 0.0060 0.0064 0.0165  0.0932
17.9550 16.1042 15.1750 14.7851 14.4281  0.0321 0.0036 0.0025 0.0025 0.0052  0.0853
18.9893 17.2083 16.3756 15.9731 15.6981  0.0567 0.0062 0.0045 0.0049 0.0120  0.0808
19.1883 18.3094 17.1918 16.7310 16.4222  0.1384 0.0249 0.0129 0.0136 0.0264  0.2520
19.2836 17.5224 16.6941 16.3132 16.0568  0.0622 0.0072 0.0054 0.0062 0.0148  0.0666
18.6364 17.6581 17.1622 16.7780 16.6394  0.0273 0.0062 0.0057 0.0060 0.0149  0.1458
```

1.2 Programs

The *ANNz* package comprises the following programs.

`annz_net` Used to specify the network architecture.

`annz_train` This is the network training program. It uses the training and validation data to determine the network weights required to predict the redshift from the photometry.

`annz_test` This program applies one or more trained networks to a given testing set, outputting the estimated photometric redshift for each object.

You are recommended to add the directory containing these programs to your path for ease of use from the command line.

2 Procedure

2.1 Specifying the network architecture: `annz_net`

The first step is to choose a network architecture. [1] The number of input nodes is fixed by the number of filters available, and there will usually be just one output node, the photometric redshift (but see §3); however, there is complete freedom over the number and size of the intervening (*hidden*) layers. The program `annz_net` is used to create architecture description files (which are suffixed `.net`).

[1] Firth et al. (2003) demonstrate the influence of the architecture on the accuracy of the network predictions.

Usage `annz_net`

Example

```
> annz_net
==============================
 ANNz: Network architecture
==============================
Number of filters (minimum 1): 5

Number of hidden layers (minimum 1): 2

Nodes in hidden layer 1: 10

Nodes in hidden layer 2: 10

Number of outputs (minimum 1): 1

====================================
Created "arch.5.10.10.1.net"
====================================
```

The `.net` file is named according to the architecture it describes, e.g. `arch.5.10.10.1.net`. A listing of the `.net` file produced in the example above follows. The structure of the `.net` file is simple and it may just as well be created manually using a text editor.

```
# Generated by annz_net
# arch.5.10.10.1.net
N_INPUTS 5
N_OUTPUTS 1
N_LAYERS 2 10 10
```

2.2 Training a network: `annz_train`

Usage `annz_train <net_file> <train_datafile> <valid_datafile> <save_file> <seed>`

Example

```
> annz_train arch.5:10:10:1.net sdss.ugriz.train sdss.ugriz.valid sdss.wts -7252

==================================
 ANNz: Network training
==================================

Training set: sdss.ugriz.train contains 15,000 patterns.
Validation set: sdss.ugriz.valid contains 5,000 patterns.

Maximum iterations: 1000

Initial cost function: 56,359.668  RMS (valid):   0.12813
Iteration:     1     Train cost function:  17,689.272     RMS (valid):   0.07237  (best RMS)
Iteration:     2     Train cost function:  10,336.376     RMS (valid):   0.05554  (best RMS)
                                          .
                                          .

Iteration:   998     Train cost function:   1,571.887     RMS (valid):   0.03198
Iteration:   999     Train cost function:   1,571.855     RMS (valid):   0.03195
Iteration: 1,000     Train cost function:   1,571.834     RMS (valid):   0.03196
Maximum iterations: 0
```

The program reads in the network architecture and the training and validation data as specified in the command line arguments. The `seed` argument is the seed for the random number generator used to initialise the weights; give any integer value.

The program prompts for the `Maximum iterations`: enter an integer number of iterations. The program gives you the option of extending the run once this number of iterations has been completed.

After each iteration of training the program reports two statistics:

`Train cost function` is the value of the network cost function evaluated on the *training* set.

`RMS (valid)` is the root mean square deviation between the network output and the spectroscopic redshift for the *validation* set.

The program works to minimise the cost function, but at each step it also calculates `RMS(valid)`. At the end of the training run the network weights saved are those for which `RMS(valid)` was minimal, *not* those for which the `Total` was minimal. This is to avoid over-fitting to the training set. Schematically, we might expect the errors on the training and validation set to vary with iteration epoch as shown in figure 2. By using the network weights for which the validation error is minimum we hope to optimise the performance of the network on data sets which it has not seen before (in neural network terms, we want to optimise its *generalisation* performance).

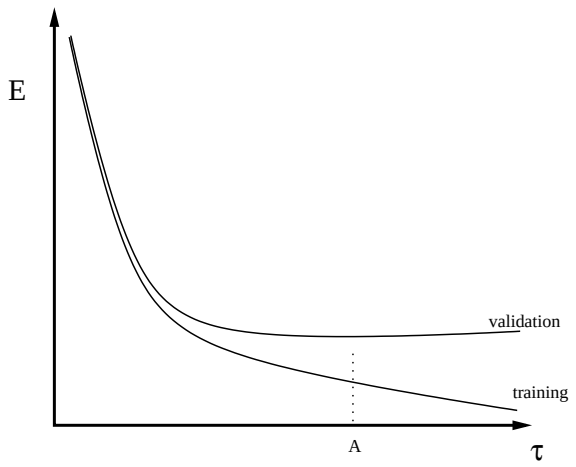

Figure 2: A schematic of the behaviour of training and validation set errors during training, as a function of training iteration τ. To achieve the best generalisation performance training should be halted at the minimum of the validation set error, that is, at A.

When the requested number of iterations are complete the program agains prompts for `Maximum iterations`: to stop training type `0`, or type e.g. `100` to carry on training. *Note:* For large data sets/low processing speeds, training can take a lot of processing time; this option to extend runs may be used to avoid having to enter an over-large number of iterations at the outset, thus avoiding unnecessarily long training runs.

The trained network state (i.e. the weights from the iteration at which `RMS(valid)` was minimal) are saved to the file name given as the argument `save_file`. The weights file produced in this example is shown below. The normalisation table describes the internal renormalisation used by the network to ensure all inputs and outputs receive equal weight in the cost function. It is not recommended to edit these files manually.

```
# Generated by annz_train
# Network architecture
N_INPUTS 5
N_OUTPUTS 1
N_LAYERS 2 10 10

# Normalisation table
INORM 0 19.1576 1.20349
INORM 1 17.4727 0.992614
INORM 2 16.5869 0.858345
```

```
INORM 3 16.1932 0.833334
INORM 4 15.8939 0.834703
ONORM 0 0.112356 0.0660118

WEIGHTS 181
0 6 -1.3335
1 6 -0.0592288
2 6 3.16965
3 6 -6.45469
4 6 2.14021
5 6 1.23082
0 7 1.09852
 .
 .
```

2.3 Using trained networks on test data: `annz_test`

Usage `annz_test <test_datafile> <results_file> <wts_file1> [<wts_file2> ...]`

Example `> annz_test sdss.ugriz.test sdss.set sdss.wts`

There are two distinct modes of running for `annz_test`, depending on whether the testing set includes spectroscopic redshifts.

No spectroscopic redshifts. This would be the case for most real applications of *ANNz*. The output from `annz_test` on running the command line above follows.

```
==================================
 ANNz: Photo-z determination
==================================
Using a committee of 1 networks.
WARNING: Error estimates not sensible with fewer than 3 networks in committee.
 # of inputs  : 5
 # of outputs : 1
Spectroscopic redshifts not present.
Calculating photometric redshifts...
Sample size: 10000
================================================
Photometric redshifts saved to "sdss.set"
================================================
```

The output file, `sdss.set`, contains two columns of numbers (per output): the photometric redshift estimated by the network for each galaxy in the testing set and an estimate for the error in this value (see below).

Spectroscopic redshifts provided. This case is useful for assessing the accuracy of the photometric redshifts estimated by *ANNz*, by running it on a testing set for which the spectroscopic redshifts are already known. On running the command line above, the following output is obtained.

```
==================================
 ANNz: Photo-z determination
==================================
Using a committee of 1 networks.
WARNING: Error estimates not sensible with fewer than 3 networks in committee.
 # of inputs  : 5
 # of outputs : 1
Found spectroscopic redshifts.
Calculating photometric redshifts...
Sample size: 10000
================================================
Output 1: Mean: 0.000397 rms: 0.022716
Photometric redshifts saved to "sdss.set"
================================================
```

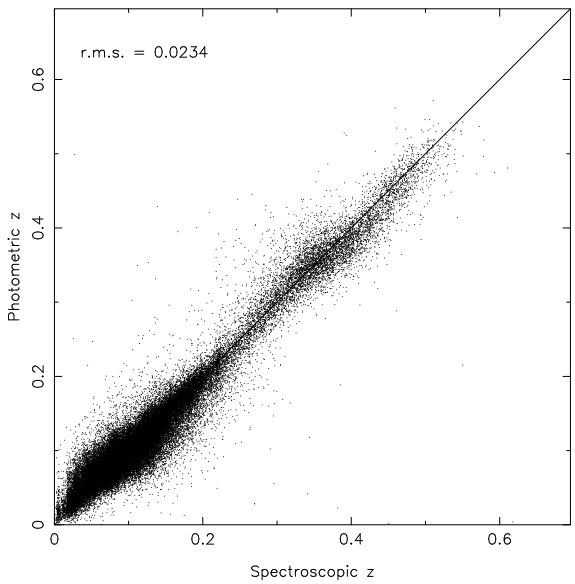

Figure 3: The plot of photometric redshift versus spectroscopic redshift produced in this example use of *ANNz*. The data used in this example is taken from the Sloan Digital Sky Survey Data Release 1 (Stoughton et al., 2002).

The program reports the mean deviation and rms deviation between the results from the network and the spectroscopic redshifts in the data file (a *positive* mean implies that the network has, on average, *overestimated* the redshift).

The output file consists of three columns (per output): the spectroscopic redshift, the photometric redshift estimated by *ANNz*, and the estimated error in the photometric redshift (see below).

2.4 Errors on the photometric redshifts

ANNz considers two independent sources of error in its photometric redshifts. The first is simply the photometric noise and the second is a product of the numerical neural network method. Each is briefly described here.

Photometric noise Since the output of a trained network is a precisely known function of the inputs, the variance in the outputs due to noisy inputs can be estimated using the standard chain-rule approach. Thus, if $y = F(\mathbf{x}; \mathbf{w})$ represents the function of the network (assuming a single output) the variance in the output is given by

$$\sigma_y^2 = \sum_i \left(\frac{\partial F}{\partial x_i} \right)^2 \sigma_{x_i}^2 \tag{1}$$

where the sum is over the inputs.

Network variance Depending on the particular initialisation used for the weights, the training process will usually converge to different local minima of the cost function. A common practice is to train a number of networks and select one based on the best performance on the validation set. However, this is wasteful of training effort and, in fact, the sub-optimal networks can be used to improve overall accuracy: the mean of the individual outputs from a group (known as a *committee*) of networks will usually be a better estimate for the true value than the outputs of any one committee member in isolation.

Using a committee also allows the uncertainty in the output due to the variance in the network weights vector to be estimated. Ideally, test cases should be robust to different intialisations of the weight vector, but some may show a large variance in the output as a result of the training process finding different local minima, particularly if they fall in regions of the input parameter space which are poorly sampled by the training data.

It is very straightforward to use a committee with *ANNz*. Trained networks can be added to the committee by simply appending their `.wts` file to the command line for `annz_test`. For example,

```
> annz_test sdss.ugriz.test sdss.set sdss.1.wts sdss.2.wts sdss.3.wts
```

applies a committee of three networks to the testing set (`sdss.1.wts`, `sdss.2.wts` and `sdss.3.wts` are the weights files for networks previously trained with `annz_train` using different values for the random number seed). The output from `annz_test` in this example is:

```
=================================
 ANNz: Photo-z determination
=================================
Using a committee of 3 networks.
 # of inputs  : 5
 # of outputs : 1
Found spectroscopic redshifts.
Calculating photometric redshifts...
Sample size: 10000
================================================
Output 1: Mean: 0.0004 rms: 0.0223152
Photometric redshifts saved to "sdss.set"
================================================
```

The ANNs which are included in the committee may have different architectures or even, in principle, be trained on different training/validation sets. `annz_test` applies each network individually to the testing set, then calculates the mean and standard deviation of the outputs from each network for each galaxy in the testing set. The mean is used as the estimate for the photometric redshift, and the standard deviation is combined in quadrature with the error due to the photometric noise to give the overall error estimate in the photometric redshift (Fig. 4). The minimum number of networks required to sensibly estimate the network variance term is three; `annz_test` warns the user if any fewer are used.

Important The error estimates made by *ANNz* are approximate at best. They should not be used as a substitute for proper evaluation of the performance of *ANNz* using a spectroscopic evaluation set.

The results are output in the same format as was described in §2.3.

3 Additional inputs & outputs

A particular strength of empirical photometric redshift methods is the ease with which additional inputs (i.e. other than magnitudes) and outputs can be added to the parameterisation. For example, measurements of distance dependent properties such as angular size of galaxies, or structural parameters such as concentration indices (which may help break degeneracies in colour-space) may be useful in improving the estimates of photometric redshifts. It is also possible to attempt to extract extra information from the inputs, for example the spectral type of a galaxy. Provided training data exists, it is equally straightforward to predict spectral type from the photometry as it is to predict photometric redshifts – the procedure is exactly the same. Indeed it is possible to train a network to predict both properties simultaneously by using a two output architecture (e.g. Collister & Lahav 2004).

References

Bishop C. M., 1995, Neural Networks for Pattern Recognition. Oxford University Press

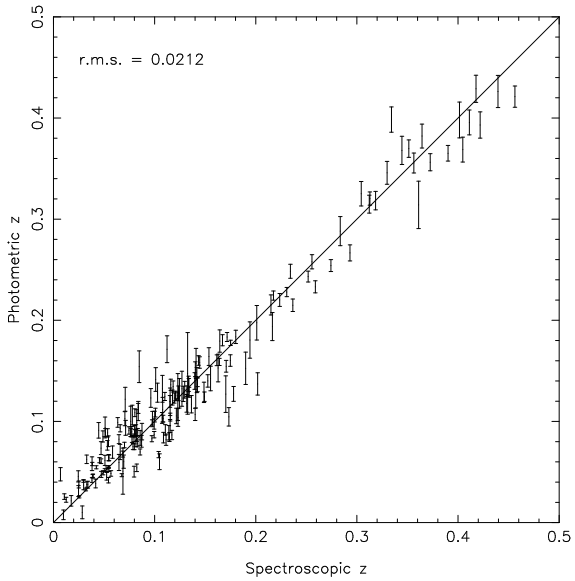

Figure 4: A subset of 200 galaxies from the testing set used in this example, having been submitted to a committee of three networks. The error bars estimated by `annz_test` are shown.

Collister A.A., Lahav O., 2004, PASP, 116, 345

Firth A. E., Lahav O., Somerville R. S., 2003, MNRAS, 339, 1195

Lahav O., Naim A., Sodré L., Storrie-Lombardi M. C., 1996, MNRAS, 283, 207

Stoughton et al. 2002, AJ, 123, 485