Darth Fader

Documentation written by Adrienne Leonard[*]

October 17, 2013

1 Introduction

Darth Fader is a software package that computes the redshifts of a set of test galaxy spectra by cross-correlation with a set of template spectra. The methodology is described in detail in Machado et al., 2013, arXiv:1309.3579. The templates are generated from a training set of galaxy spectra at zero redshift (with or without noise). The continuum is subtracted from the training set spectra in an automated way, and eigentemplates computed from the continuum-free training spectra via Principal Component Analysis. The most significant eigentemplates are retained for cross-correlation in order to estimate the test galaxy redshifts. The test spectra undergo an identical continuum-subtraction procedure, and line features are then identified by denoising using filtering in wavelet space according to a set False Discovery Rate (FDR) threshold. Redshifts are then estimated by cross-correlation of the continuum-free and (optionally) denoised test spectra with the retained eigentemplates, with the redshift corresponding to the shift between the eigentemplate and test spectrum that minimises the χ^2 distance between the two. The redshift catalogue is then cleaned by removing galaxies whose spectra return fewer than a user-specified number N of line features (in Machado et al., 2013, this critical number was taken to be $N = 6$). Tests have demonstrated that galaxies in this category are less likely to obtain a correct redshift estimate after cross-correlation with the eigentemplates. Their removal significantly increases the catastrophic failure rate after cleaning, as compared with the full redshift catalogue.

Please note the following:

- Please follow the **full installation instructions on** `http://cosmostat.org/darth_fader.html` prior to running the software.
- Darth Fader requires **all** spectra to be pixellated **logarithmically** with respect to wavelength. This affects the **redshift estimation** portion of Darth Fader; the denoising and continuum estimation routines will still function regardless of whether the the spectra are binned linearly or logarithmically with respect to wavelength.
- Darth Fader assumes that all test spectra contain the same number of pixels, and have the same start and end wavelengths. Similarly, it assumes that the training set spectra contain the same number of pixels, and have the same start and end wavelengths.
- The training set and the test data do not need to cover the same wavelength range; in fact, it is **advantageous** to have training set spectra covering a **larger** range of wavelength than the test data. *It is very important, however, that the training set and test data have the same logarithmic pixel scaling.*
- Both templates and test data must be **continuum-free** before being passed to the redshift estimation routine.
- The eigentemplates should be unit normalised.
- If using denoised data for the redshift estimation step, *df_get_redshift.pro* will return a very large value for z_{est} for galaxies that return no features after denoising.

[*]adrienne.leonard@cea.fr

- The Darth Fader software is modular. This means that there is a separate routine for each step in the Darth Fader process. Each of the routines called by *darth_fader.pro* may be replaced by the user's own software designed for that purpose (e.g. one may replace our redshift estimation by cross-correlation with a different redshift estimation script, whilst retaining the remainder of the Darth Fader denoising/continuum-estimation/cleaning software).

2 The Software Package

Darth Fader consists of IDL routines, to be used in conjunction with the iSAP package[1]. The IDL routines are described in detail below.

2.0 Main program: *darth_fader.pro*

This is the main routine that runs the components of the Darth Fader software, described below.

Calling Sequence	`darth_fader, spectra_components, templates, redshift_estimates, clean_catalogue, verbose = verbose, S_TabPeaks = S_TabPeaks, Data = Data, Training = Training, TnoBaseline = TnoBaseline`
Inputs	All inputs are specified in *df_input_params.pro*
Calls	*df_input_params.pro, df_del_baseline.pro, df_get_eigenvectors.pro, df_get_spectra_components.pro, df_get_redshifts.pro, df_peaks.pro*
Options	
verbose	If set, Darth Fader will update you on its progress as it processes your data.
Returns	
spectra_components	A structure containing the continuum (*.baseline*), emission line (*.posline*), absorption line (*.negline*), and noise (*.noise*) spectra as well as continuum-subtracted, noisy data (*.data_nobaseline*), and continuum-subtracted, denoised data (*.data_clean*)
templates	Array containing the user-specified number of eigentemplates
redshift_estimates	Array containing the redshift estimates for each galaxy
clean_catalogue	Structure listing the locations of spectra meeting the N-peak criterion (*.indices*) and the redshifts of those galaxies (*.redshifts*)
Keyword Returns	
S_TabPeaks	Returns a structure containing the emission (*.empeaks*), absorption (*.abspeaks*), and total (*.allpeaks*) peak count for each galaxy spectrum
Data	Returns the original input data
Training	Returns the original training set
TnoBaseline	Returns the continuum-subtracted training set

2.1 Input parameters: *df_input_params.pro*

This is the most important routine for the user, as this is where the user is able to specify all the inputs to the Darth Fader main code. This code is a function, called via the calling sequence indicated below. It takes no command line inputs; all modifications must be done inside the body of the routine itself. This is the only Darth Fader routine that the user should need to modify, as it specifies all the options given tot he other Darth Fader routines.

[1]Software and documentation at `http://cosmostat.org/software.html`

Calling Sequence `inputs = df_input_params()`

User parameters

INDIR	Directory where your input data are located. Note: Darth Fader assumes that all training, test and template spectra are located in the same directory
incat	Name of the *.fits* or *.fits.gz* file containing the test data as a 2D array. Note, spectra must be log-binned with respect to wavelength
trainingcat	Name of the *.fits* or *.fits.gz* file containing the training data as a 2D array. If eigenvectors have already been computed, the user may specify trainingcat = ' '
rmscurve	Name of the *.fits* or *.fits.gz* file containing the RMS noise curve associated with the data. This may be either a 1D array, if the same noise curve is to be used for all spectra, or a 2D array with a noise curve for each spectrum. Note the dimensions must match those of *incat*. In the case of white Gaussian noise, specify rmscurve = ' '
templatecat	Name of the *.fits* or *.fits.gz* file containing the eigentemplates, if already computed. **One of trainingcat and templatecat must be specified**
lstep	Logarithmic pixel scale of the data δ such that $\lambda_i = \lambda_{min} 10^{i\delta}$, where λ is in angstroms
training_lmin; training_lmax	Minimum and maximum wavelength of the training set in angstroms*
data_lmin; data_lmax	Minimum and maximum wavelength of the data in angstroms* *These data are used to compute the pixel shift between the end of the test spectra and the end of the training set spectra. This can, alternatively, be hardwired by specifying *shiftpar*
shiftpar	Pixel shift between the end of the test spectra and the end of the training set spectra
EstimRedshiftFromNoisyData	Boolean operator specifying whether the cross-correlation should be carried out using the denoised, continuum-free spectrum (recommended for wavelength-dependent noise) or the noisy, continuum-free spectrum (recommended for white Gaussian noise)
ComputeEigentemplates	Boolean operator; set to 0 if you have already computed eigentemplates and specified *templatecat*
nscale	Number of scales to use in the multiscale transforms for denoising. Set to 0 to compute automatically from the data
EnergPercent	Darth Fader will retain all eigentemplates such that the total eigenvalue weight of the retained templates is $\geq EnergPercent$
Ntemplates	Forces the specified number of templates to be retained if set greater than 0, overriding *EnergPercent*
FDR	False discovery rate threshold to be used in the denoising steps, specified as n where the FDR parameter $\alpha = 1 - \mathrm{erf}(n/\sqrt{2})$ is the probability outside the $n\sigma$ interval of the standard normal distribution
NpeakCrit	Critical number of peaks required for cleaning
NsigmaPeak	Detection level for peaks during peak counting
CleanCatalogue	Boolean operator specifying whether to clean the catalogue using the N-peak criterion
OUTDIR	Directory in which to place the outputs of the Darth Fader run
OutputComponents, CompFileName	Boolean operator specifying whether to output the spectra components as a fits file *CompFileName*
OutputTemplates, TempFileName	Boolean operator specifying whether to output the retained eigentemplates as a fits file *TempFileName*
OutputAllEigen, AllEigenName	Boolean operator specifying whether to output all eigentemplates as a fits file *AllEigenName*
OutputRedshifts, RedshiftFileName	Boolean operator specifying whether to output redshift estimates as a fits file *RedshiftFileName*
OutputCleanCatalogue, CleanFileName	Boolean operator specifying whether to output the cleaned catalogue as a fits file *CleanFileName*

2.2 Continuum-subtraction: *df_del_baseline.pro*

This function takes a set of galaxy spectra and removes the continuum. This is called by *darth_fader.pro*, or may be used independently, and may take as an argument a single spectrum as a 1D array or many galaxy spectra as a 2D array.

Calling Sequence	`Spectra_no_baseline = df_del_baseline(spectra, nscale = nscale, rms = rms, baseline = baseline)`
Inputs	
spectra	1D or 2D array containing the galaxy spectrum information
Keyword Inputs	
nscale	Number of scales for the multiscale transforms
rms	RMS error curve for the data, either as a 1D array (if the same noise curve applies to all spectra) or a 2D array of the same dimensions as *spectra*.
Returns	Continuum-subtracted spectra are returned in the named variable *Spectra_no_baseline*
Keyword Returns	
baseline	Returns the continuum estimate to the named variable *baseline*

2.3 Computation of eigentemplates: *df_get_eigenvector.pro*

Calling sequence	`Templates = df_get_eigenvector(Training, AllEigen=AllEigen, Denoising=Denoising, OptFil = OptFil, EnergPercent = EnergPercent, NTemp = NTemp)`
Inputs	
Training	2D array containing the training spectra
Keyword Inputs	
Denoising	Set to 1 to denoise the training set before estimating the continuum
OptDen	If *Denoising* is set, specify options if different from default
EnergPercent	Eigenvector energy to be captured by the retained eigenvectors
NTemp	Number of templates to be retained (overrides EnergPercent)
Returns	Array of eigenvectors matching the *EnergPercent* or *NTemp* criteria specified returned in the named variable *Templates*
Keyword Returns	
AllEigen	Returns the full eigenvector set in the named variable *AllEigen*

2.4 Decomposition of spectra: *df_get_spectra_components.pro*

Calling sequence	`DecSpectra = df_get_spectra_components(TabSpectra, RMS=RMS, nscale=nscale, nsigma=nsigma)`
Inputs	
Spectra	1D or 2D array containing the test spectrum/spectra
Keyword Inputs	
RMS	Error curve(s) associated with the spectra
nscale	Number of scales for the multiresolution denoising steps
nsigma	FDR parameter expressed as n where α is the probability external to the $n\sigma$ interval of the standard normal distribution
Returns	Structure containing the continuum estimate (*.baseline*), emission line spectrum (*.posline*), absorption line spectrum (*.negline*), noise estimate (*.noise*), continuum-subtracted noisy data (*.data_nobaseline*) and continuum-subtracted noise-free data (*.data_clean*)

2.5 Redshift estimation: *df_get_redshift.pro*

Calling sequence	`Redshifts = df_get_redshift(Spectra, TabTemplate, lstep= lstep, shiftpar=shiftpar, InfoTemp=InfoTemp)`
Inputs	
Spectra	1D or 2D array containing the continuum-subtracted (either noisy or denoised) spectrum/spectra
TabTemplate	Template spectra for cross-correlation
Keyword Inputs	
lstep	Logarithmic pixel scale
shiftpar	Pixel shift between the end of the template spectra and the end of the test spectra
Returns	Array of redshift estimates for each spectrum passed to the routine in the named variable *Redshifts*. Note *df_get_redshift.pro* returns a very large value for z_{est} for spectra that return no features after denoising, when denoised data are used for redshift estimation
Keyword Returns	
InfoTemp	Retains key information about the size and contents of the test spectra. Useful if *df_get_redshift.pro* is called multiple times for spectra of the same dimensions and properties.

2.6 Peak counting: *df_peaks.pro*

Calling sequence	`TabPeak = df_peaks(DecSpectra, noise=noise, nsigma=nsigma)`
Inputs	
DecSpectra	Output of *df_get_spectra_components.pro*: a structure containing the spectra components.
Keyword Inputs	
noise	RMS error curve for the spectra
nsigma	Detection threshold for peaks
Returns	Structure containing the number of emission line (*.empeaks*), absorption line (*.abspeaks*) and total (*.allpeaks*) peaks detected in each spectrum

3 Example

On the Darth Fader webpage, we provde example data to benchmark your installation of the software. We provide an example code *example_darth_fader.pro* which may be run in IDL to test the software. Before running this example code, please ensure that the specified file paths in *df_input_params.pro* are correct, but make no other modifications to the other codes.

Calling sequence `example_darth_fader`

Returns

```
Reading data...
done!
Computing eigentemplates from training data...
Nbr of templates = 20, PercentEnergy = 99.927853
done!
Getting spectra components...
done!
Computing redshifts...
done!
Cleaning catalogue...
done!
% of catastrophic failures before cleaning = 22.027972
% of galaxies retained after cleaning = 75.804196
% of catastrophic failures after cleaning = 4.2896679
```

4 Contact

Please refer to the paper Machado et al., 2013, arXiv:1309.3579 for more detailed information about the method used in this software. If you have any queries regarding installation or utilisation of the Darth Fader software package, or of the methods described in Machado et al. (2013), please contact Adrienne Leonard.

5 Copyright

If you find this software useful for your application, we'd appreciate it if you would drop us a quick email to tell us a bit about what your application!

Users are free to modify and use the Darth Fader software in whatever way they see fit. However, any publications using this software should clearly cite Machado et al., 2013, arXiv:1309.3579 and the CosmoStat software webpage.