TheCannon Documentation

Release 1.0

Anna Ho

January 13, 2015

CONTENTS

CHAPTER

ONE

INTRODUCTION

This is the software package used for *The Cannon*, a data-driven approach to determining stellar parameters and abundances (hereafter referred to as *labels*) corresponding to a vast set of stellar spectra.

A brief overview of *The Cannon* is below. For more details, see our publication! :D

For *The Cannon* to work, this vast set must have a subset of *reference objects*: spectra that already have high-fidelity labels. Labels for the remaining spectra are determined via a process called *label transfer.*

There are two fundamental steps in label transfer.

1. The *training step*: the reference objects are assembled into a *training set* that is used to solve for a flexible generative model (with ~80,000 parameters). The model describes the flux in every pixel of the continuum-normalized spectrum as a function of stellar labels.
2. The *test step*: we assume that this generative model holds for all of the other objects in the survey (dubbed *survey objects*). Then, the spectra of the survey objects and the generative model from the reference objects allow us to solve for - or infer - the labels of the survey objects.

A few notable features of this package are:

- Helpful (optional) methods for preparing spectra and labels to be fed into *The Cannon*
- Generate a training set and a test set by supplying spectra and training labels
- Merge multiple training sets into one
- Generate a model by running the training set through Step 1 of *The Cannon*
- Determine labels for the test set by running the model and training set through Step 2 of *The Cannon*.

CHAPTER

TWO

GETTING STARTED

The basic workflow for creating a training set and test set, solving for the model (Step 1) and finally determining labels (Step 2) is shown below.

2.1 Step 1: Prepare Data (`prepdata.py`)

The Cannon expects all spectra - for reference and survey objects - to be continuum-normalized in a consistent way, and sampled on a consistent rest-frame wavelength grid, with the same line-spread function. It also assumes that the flux variance, from photon noise and other sources, is known at each spectral pixel of each spectrum.

Preparing data thus involves: putting spectra into a 3D array (nstars, npixels, 3), putting label names into an array (nlabels), and putting training label values into a 2D array (nstars, nlabels). The user is left to do this him or herself, but we include some basic methods and functionalities in `prep_data.py` that might be helpful.

2.2 Step 2: Construct Training Set

The training set is a set of stars from the survey under consideration for which the user has spectra and also high-fidelity labels (that is, stellar parameters and element abundances that are deemed both accurate and precise.) The set of reference objects is critical, as the label transfer to the survey objects can only be as good as the quality of the training set.

Labels for a training set will not necessarily come from the same data, and therefore spectra will not necessarily be in the same data format. For flexibility, *The Cannon* software package allows the user to construct training subsets, one data type each, and then merge the subsets.

```
>>> dataset import Dataset
>>> fts_trainingset = Dataset(objectIDs = [], spectra = [], labelnames = [], labelvals = [])
>>> vesta_trainingset = Dataset(objectIDs = [], spectra = [], labelnames = [], labelvals = [])
>>> cluster_trainingset = Dataset(objectIDs = [], spectra = [], labelnames = [], labelvals = [])
>>> trainingset = mergesets(fts_trainingset, vesta_trainingset, cluster_trainingset)
```

There are a few ways to examine the dataset. You can retrieve the spectra as follows:

```
>>> pixels = trainingset.spectra[:,:,0]
>>> fluxes = trainingset.spectra[:,:,1]
>>> fluxerrs = trainingset.spectra[:,:,2]
```

2.3 Step 3: Construct Test Set

```
>>> testset = Dataset(objectIDs = [], spectra = [], labelnames = [], labelvals = None)
```

2.4 Step 4: *The Cannon* Step 1 - Generate Model

```
>>> from spectral_model import SpectralModel
>>> model = SpectralModel(label_names, modeltype)
>>> model.train(trainingset)
```

2.5 Step 5: *The Cannon* Step 2 - Infer Labels

```
>>> from cannon_labels import CannonLabels
>>> labels = CannonLabels(label_names)
>>> labels.solve(model, testset)
```

CHAPTER

THREE

USING THE CANNON

The details of using The Cannon package are provided in the following sections:

CHAPTER

FOUR

REFERENCE/API

prep_data.**continuum_normalize**(*spectra*)
Continuum-normalizes a spectra array.

Assumes that there are no gaps in the spectrum and fits a 2nd order Chebyshev polynomial. Divides spectrum by the polynomial.

Input: spectra array, 2D float shape nstars,npixels,3 Returns: 3D continuum-normalized spectra (nstars, npixels,3) 2D continuum array (nstars, npixels)

prep_data.**get_spectra**(*filenames*)
Extracts spectra from aspcap fits files

Input: a list of data file names Returns: 3D float array spectra[:,:,0] = pixel wavelengths spectra[:,:,1] = flux values spectra[:,:,2] = flux err values

prep_data.**get_training_labels**(*filename*)
Extracts training label names and values from file

Assumes: – that the file format is as follows, with label names in first row – first row has a # then label names... – first column is the IDs...so strings – and the rest are floats – that you want *all* of the labels

Input: filename Returns: 2D np.array (size=ntrainingstars, nlabels) consisting of all training labels

p

C

G

P