astrorapid Documentation

Daniel Muthukrishna

Jan 22, 2021

Contents:

RAPID (Real-time Automated Photometric IDentification) can classify multiband photometric light curves into several different transient classes. It uses a deep recurrent neural network to produce time-varying classifications.

CHAPTER 1

Installation

1.1 Using pip

The easiest and preferred way to install DASH (to ensure the latest stable version) is using pip:

```
pip install astrorapid --upgrade
```

1.2 From Source

Alternatively, the source code can be downloaded from GitHub by running the following command:

```
git clone https://github.com/daniel-muthukrishna/astrorapid.git
```

1.3 Dependencies

Using pip to install astrorapid will automatically install the mandatory dependencies: `numpy`, `tensorflow`, `keras`, `astropy`, `pandas`, `extinction`, and `scikit-learn`.

If you wish to plot your classifications or train your own classifier with your data you will need matplotlib and scipy as well. These can be installed with

```
pip install matplotlib
pip install scipy
```

1.4 Platforms

RAPID can be run on both Python 2 and Python 3 distributions. It is also cross-platform, working on Mac, Windows, and Linux distributions. If you have any issues, please submit an issue at https://github.com/daniel-muthukrishna/

astrorapid/issues.

CHAPTER 2

Usage

2.1 Classify Light curves

Use the following example code:

```
from astrorapid.classify import Classify

# Each light curve should be a tuple in this form. Look at the example code for an␣
↪example of the input format.
light_curve_info1 = (mjd, flux, fluxerr, passband, photflag, ra, dec, objid, redshift,
↪ mwebv)
light_curve_list = [light_curve_info1,]
contextual_info_list = [{'hosttype': value},]  # Only use this parameter if you have␣
↪trained your own classifer with specific meta data. Otherwise set to None.

# Classify Light curves
classification = Classify(known_redshift=True)
predictions = classification.get_predictions(light_curve_list, contextual_info_list)
print(predictions)

# Plot light curve and classification vs time of the light curves at the specified␣
↪indexes
classification.plot_light_curves_and_classifications(indexes_to_plot=(0,1,4,6))
classification.plot_classification_animation(indexes_to_plot=(0,1,4,6))
```

2.2 Train your own classifier with your own data

You'll simply need to run the function `astrorapid.custom_classifier.create_custom_classifier()` to get started with training your own classifier. An example is shown below.

```
from astrorapid.custom_classifier import create_custom_classifier

script_dir = os.path.dirname(os.path.abspath(__file__))

create_custom_classifier(get_data_func=astrorapid.get_training_data.get_data_from_
↪snana_fits,
                         data_dir=os.path.join(script_dir, '..', 'data/ZTF_20190512'),
                         class_nums=(1, 2, 12, 14, 3, 13, 41, 43, 51, 60, 61, 62, 63,␣
↪64, 70),
                         class_name_map={1: 'SNIa-norm', 2: 'SNII', 12: 'SNII', 14:
↪'SNII', 3: 'SNIbc', 13: 'SNIbc', 41: 'SNIa-91bg', 43: 'SNIa-x', 51: 'Kilonova', 60:
↪'SLSN-I', 61: 'PISN', 62: 'ILOT', 63: 'CART', 64: 'TDE', 70: 'AGN'},
                         reread_data=False,
                         contextual_info=('redshift', 'some_contextual_info1'),
                         passbands=('g', 'r'),
                         retrain_network=False,
                         train_epochs=100,
                         zcut=0.5,
                         bcut=True,
                         ignore_classes=(61, 62, 64, 70),
                         nprocesses=None,
                         nchunks=10000,
                         otherchange='',
                         training_set_dir='data/training_set_files',
                         save_dir='data/saved_light_curves',
                         fig_dir='data/training_set_files/Figures'),
                         plot=True
                         )
```

You'll need to write your own function *get_data_func* to read your data and use the *astrorapid* preprocessing tools. Use the skeleton function here `astrorapid.get_custom_data.get_custom_data()`, or as rewritten below.

```
def get_custom_data(class_num, data_dir, save_dir, passbands, known_redshift,␣
↪nprocesses, redo):
    """
    Get data from custom data files.
    You will need to write this function with the following skeleton function:

    Parameters
    ----------
    class_num : int
        Class number. E.g. SNIa is 1. See helpers.py for lookup table.
        E.g. class_num = 1
    data_dir : str
        Directory where data is stored
        E.g. data_dir='data/ZTF_20190512/'
    save_dir : str
        Directory to save processed data
        E.g. save_dir='data/saved_light_curves/'
    passbands : tuple
        Passbands to use.
        E.g. passbands=('g', 'r')
    known_redshift : bool
        Whether to correct the light curves for cosmological time dilation using␣
↪redshift.
    nprocesses : int or None
        Number of processes to use
```

(continues on next page)

(continued from previous page)

```
    redo : bool
        Whether to redo reading the data and saving the processed data.


    Returns
    -------
    light_curves : dict of astropy.table.Table objects
        e.g light_curves['objid1'] =
            passband   time        flux      fluxErr    photflag
              str1    float32     float32    float32     int32
            -------- --------- ------------ ---------- --------
                   g -46.8942   -48.926975   42.277767        0
                   g -43.9352   -105.35379    72.97575        0
                   g -35.9161   -46.264206     99.9172        0
                   g -28.9377   -28.978344   42.417065        0
                   g -25.9787   109.886566    46.03949        0
                   g -15.0399     -80.2485    80.38155        0
                   g -12.0218     93.51743   113.21529        0
                   g  -6.9585    248.88364  108.606865        0
                   g  -4.0411    341.41498   47.765404        0
                   g       0.0    501.7441    45.37485     6144
                 ...       ...          ...         ...      ...
                   r  40.9147    194.32494   57.836903     4096
                   r  59.9162     67.59185    45.66463     4096
                   r  62.8976     80.85155   44.356197     4096
                   r  65.8974    28.174305    44.75049     4096
                   r  71.8966   -18.790287  108.049774     4096
                   r  74.9297   -3.1707647   125.15057     4096
                   r  77.9341  -11.0205965   125.784676     4096
                   r  80.8576    129.65466    69.99305     4096
                   r  88.8922   -14.259436   52.917866     4096
                   r 103.8734    27.178356  115.537704     4096

    """

    # If the data has already been run and processed load it. Otherwise read it and␣
↪save it
    save_lc_filepath = os.path.join(save_dir, f"lc_classnum_{class_num}.pickle")
    if os.path.exists(save_lc_filepath) and not redo:
        with open(save_lc_filepath, "rb") as fp:  # Unpickling
            light_curves = pickle.load(fp)
    else:
        light_curves = {}
        # Read in data from data_dir and get the mjd, flux, fluxerr, passband,␣
↪photflag as 1D numpy arrays for
        # each light curve. Get the ra, dec, objid, redshift, mwebv, model_num, peak_
↪mjd as floats or strings.
        # Set whether you'd like to train a model with a known redshift or not. Set␣
↪known_redshift as a boolean.

        # Enter your own data-reading code here that gets the mjds, fluxes, fluxerrs,␣
↪passbands, photflags,
        # ras, decs, objids, redshifts, mwebvs, model_nums, peak_mjds for all the␣
↪light curves from the data_dir

        # Once you have the required data information for each light curve, pass it␣
↪into InputLightCurve with
```

(continues on next page)

(continued from previous page)

```
        # something like the following code:
        for i, objid in enumerate(objids):
            inputlightcurve = InputLightCurve(mjds[i], fluxes[i], fluxerrs[i],␣
↪passbands[i], photflags[i],
                                              ras[i], decs[i], objids[i],␣
↪redshifts[i], mwebvs[i],
                                              known_redshift=known_redshift,
                                              training_set_parameters={'class_number
↪': int(class_num),
                                                                       'peakmjd':␣
↪peakmjds[i]},
                                              other_meta_data={'some_contextual_info1
↪': value})
            light_curves[objid] = inputlightcurve.preprocess_light_curve()

        # If you think that reading the data is too slow, you may want to replace the␣
↪for loop above with
        # multiprocessing. See the example function in get_training_data.py if you␣
↪need help doing this.

        # Next, we save it:
        with open(save_lc_filepath, "wb") as fp:  # Pickling
            pickle.dump(light_curves, fp)

    return light_curves
```

CHAPTER 3

Example

Example script classifying an example supernova light curve.

```
from astrorapid.classify import Classify

# Light curve information. It may be easier if this is read from a file and later␣
↪manipulated into this format.
mjd = [57433.4816, 57436.4815, 57439.4817, 57451.4604, 57454.4397, 57459.3963, 57462.
↪418 , 57465.4385, 57468.3768, 57473.3606, 57487.3364, 57490.3341, 57493.3154, 57496.
↪3352, 57505.3144, 57513.2542, 57532.2717, 57536.2531, 57543.2545, 57546.2703, 57551.
↪2115, 57555.2669, 57558.2769, 57561.1899, 57573.2133,57433.5019, 57436.4609, 57439.
↪4587, 57444.4357, 57459.4189, 57468.3142, 57476.355 , 57479.3568, 57487.3586, 57490.
↪3562, 57493.3352, 57496.2949, 57505.3557, 57509.2932, 57513.2934, 57518.2735, 57521.
↪2739, 57536.2321, 57539.2115, 57543.2301, 57551.1701, 57555.2107, 57558.191 , 57573.
↪1923, 57576.1749, 57586.1854]
flux = [2.0357230e+00, -2.0382695e+00,  1.0084588e+02,  5.5482742e+01,  1.4867026e+01,
↪ -6.5136810e+01,  1.6740545e+01, -5.7269131e+01,  1.0649184e+02,  1.5505235e+02,  3.
↪2445984e+02,  2.8735449e+02,  2.0898877e+02,  2.8958893e+02,  1.9793906e+02, -1.
↪3370536e+01, -3.9001358e+01,  7.4040916e+01, -1.7343750e+00,  2.7844931e+01,  6.
↪0861992e+01,  4.2057487e+01,  7.1565346e+01, -2.6085690e-01, -6.8435440e+01, 17.
↪573107  ,  41.445435  , -110.72664   ,  111.328964  ,  -63.48336   ,  352.44907   ,
↪  199.59058   ,  429.83075   ,  338.5255    ,  409.94604   ,  389.71262   ,  195.
↪63905   ,  267.13318   ,  123.92461   ,  200.3431    ,  106.994514  ,  142.96387   ,
↪   56.491238  ,   55.17521   ,   97.556946  ,  -29.263103  ,  142.57687   ,  -20.
↪85057   ,   -0.67210346,   63.353024  ,  -40.02601]
fluxerr = [42.784702,  43.83665 ,  99.98704 ,  45.26248 ,  43.040398,  44.00679 ,  41.
↪856007,  49.354336, 105.86439 , 114.0044  ,  45.697918,  44.15781 ,  60.574158,  93.
↪08788 ,  66.04482 ,  44.26264 ,  91.525085,  42.768955,  43.228336,  44.178196,  62.
↪15593 , 109.270035, 174.49638 ,  72.6023  ,  48.021034, 44.86118 ,  48.659588, 100.
↪97703 , 148.94061 ,  44.98218 , 139.11194 ,  71.4585  ,  47.766987,  45.77923 ,  45.
↪610615,  60.50458 , 105.11658 ,  71.41217 ,  43.945534,  45.154167,  43.84058 ,  52.
↪93122 ,  44.722775,  44.250145,  43.95989 ,  68.101326, 127.122025, 124.1893  ,  49.
↪952255,  54.50728 , 114.91599]
passband = ['g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g',
↪ 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g', 'r', 'r', 'r', 'r', 'r', 'r', 'r
↪', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r',
↪'r', 'r', 'r']
```

(continues on next page)

(continued from previous page)

```
photflag = [0,    0,    0,    0,    0,    0,    0,    0,    0,    0, 4096, 4096,    0,
↪    0,    0,    0,    0,    0,    0,    0,    0,    0, 0,    0,    0,    0,    0,
↪ 0,    0,    0,    0, 4096, 6144, 4096, 4096, 4096, 0,    0,    0,    0,    0,    0,
↪    0,    0,    0,    0,    0, 0,    0,    0,    0]
objid = 'transient_1'
ra = 3.75464531293933
dec = 0.205076187109334
redshift = 0.233557
mwebv = 0.0228761

light_curve_info1 = (mjd, flux, fluxerr, passband, photflag, ra, dec, objid, redshift,
↪ mwebv)

# Classification
light_curve_list = [light_curve_info1,]  # Add more light curves to be classified to
↪this list.

classification = Classify(known_redshift=True)
predictions = classification.get_predictions(light_curve_list)
print(predictions)

# Plot classifications vs time
classification.plot_light_curves_and_classifications()
classification.plot_classification_animation()
```

CHAPTER 4

API

RAPID has a few important classes and methods.

4.1 Classify light curves

- `astrorapid.classify.Classify` - Setup light curve classification
- `astrorapid.classify.Classify.get_predictions()` - Classify light curves!

4.2 Plot classifications

- `astrorapid.classify.Classify.plot_light_curves_and_classifications()` - Plot light curves and classifications
- `astrorapid.classify.Classify.plot_classification_animation()` - Plot animation of light curves and classifications

4.3 Train your own classifier

- `astrorapid.get_custom_data.get_custom_data()` - Read custom data files and save preprocessed light curves.
- `astrorapid.custom_classifier.create_custom_classifier()` - Run preprocessing and train neural network classification model.

The full documentation of useful classes and methods can be found below.

4.4 Full Documentation

CHAPTER 5

Contribute

- Issue Tracker: https://github.com/daniel-muthukrishna/astrorapid/issues
- Source Code: https://github.com/daniel-muthukrishna/astrorapid

CHAPTER 6

Support

If you are having issues, please let us know by submitting a GitHub issue at https://github.com/daniel-muthukrishna/astrorapid/issues

CHAPTER 7

License

The project is licensed under the MIT license.

CHAPTER 8

Citation

You can cite the following paper for this work: https://ui.adsabs.harvard.edu/abs/2019arXiv190400014M/abstract

CHAPTER 9

Author

Daniel Muthukrishna http://www.danielmuthukrishna.com