intro_project

March 30, 2022

[1]:
```
# Using ML to classify LAEs in the NEP Field
import tables as tb
import numpy as np

import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

from astropy import constants as const
from astropy.table import Table, column, join
from astropy.io import fits
import astropy.units as u
from astropy.coordinates import SkyCoord
from astropy.visualization import ZScaleInterval

from regions import CircleSkyRegion, CirclePixelRegion

from hetdex_api.survey import Survey, FiberIndex
from hetdex_api.config import HDRconfig
from hetdex_api.detections import Detections

from hetdex_tools.get_spec import get_spectra

import pandas as pd
import seaborn as sb
```

[2]:
```
det_object = Detections('hdr2.1', loadtable = False)
```

[13]:
```
hdr2 = Detections(curated_version = '2.1.3')
```

[14]:
```
print('hdr2.coords.size', hdr2.coords.size)
print('det_object.coords.size', det_object.coords.size)
```

```
hdr2.coords.size 578587
det_object.coords.size 1482880
```

[21]:
```
tb = hdr2.return_astropy_table()
```

[25]:
```
tb["field"] == "nep"
```

[25]:
```
array([False, False, False, …, False, False, False])
```

[28]:
```
np.unique(tb["field"])
```

[28]:
```
<Column name='field' dtype='str12' length=5>
    cosmos
  dex-fall
dex-spring
       egs
   goods-n
```

[19]:
```
# Once it has loaded you want to filter out the data by selecting those that␣
 ↪are in the NEP field
# to do this I will give you the verticies of a box that will encompass all the␣
 ↪NEP field - Oscar

# The center of the NEP field is given by:
# NEP Central Coordinates:
# R.A. = 18hours00minutes00seconds, decl. = 66 degree 33minute 38.552 arcmin
# Then make a radius of 3.5 degrees centered above and find all the RA and DEC␣
 ↪coordinates
# in the DF that are within this circle

############ USING HDR2 OBJECT ##########################
# creating the circle region in the sky (NEP field)
ra = '18h00m00s'
dec = '+66d33m38.552s'
center_sky_coords = SkyCoord(ra, dec, frame = 'icrs')

maskregion = hdr2.query_by_coords(center_sky_coords, 3.5 * u.deg)
detects_in_NEP = hdr2[maskregion]      # Sources within the NEP footprint
print('hdr2: ', end = "")
print(np.size(detects_in_NEP.detectid))

############ USING det_object OBJECT ##########################

ra = '18h00m00s'
dec = '+66d33m38.552s'
center_sky_coords = SkyCoord(ra, dec, frame = 'icrs')

maskregion = det_object.query_by_coords(center_sky_coords, 3.5 * u.deg)
detects_in_NEP = det_object[maskregion]      # Sources within the NEP footprint
print('det_object: ', end = "")
print(np.size(detects_in_NEP.detectid))
```

```
hdr2: 0
det_object: 69799
```

[17]:
```
np.sum(maskregion)
```

[17]: 0

[ ]:
```
# Once you have selected sources within the NEP footprint we can then go ahead␣
 ↪and find some
# spectra from these sources - Oscar
spectra = detects_in_NEP.hdfile.root.Spectra
```

center_ksy.separations(skycoords entire) return indeces return distance return 3d

separate by dist mask

main goal of algorithm want it to distinguish lae vs o2 emitter. wouldn't impose cuts unless training.

cut = filter cut = signal to noise could do plya

might need cuts for taining for confident lya and o2

increase confidence by visually inspecting

could visually inspect to increase confidence.

plotting histograms to look for outliers

hetdex isn't perfect and it catches emission lines that aren't real. visual inspections helps

no need for dataframes if i found another way

save as csv with astropy table

csv into get_spec()

`has nice documentation`

get_spectrum good for ids ****USE*** returns all fiber spec with corresponding weights.

`try to see if can get LAE samples. O2 samples. and ambigious samples. Clasify some of them. Si`

detection object filter by fields. turn to astropy table and then filter.

Want psf weighted.

[ ]:
```

```