Project: Astronomy & Astrophysics (SPHY0804)

Cross-Matching Astronomical Data

Kunal Singh(UID-219114, Roll No.-9)
Tanishka Manjrekar (UID-219128, Roll No.-19)

Abstract**:** *Cross-matching is an important tool in astronomy data analysis. This project aims to design a program to cross match two data catalogues and improve the time complexity. We cross-matched AT20G with SuperCOSMOS, AT20G with SDSS and BSS with SuperCOSMOS.*

Keywords: Cross-Matching, kd tree, time complexity, SuperCOSMOS, AT20G,Bright Source Sample(BSS), SDSS

Theoretical Background and Review of Relevant Literature:

For the study of different astronomical objects, it is necessary to study the object in different wavelengths. We obtain the data from different wavelengths using different telescopes, detectors and imaging techniques. However, to find the emissions of the same object in different catalogues cross matching is necessary. Cross matching is the process of finding counterparts of objects in different catalogues and combining the information for further analysis.

Positional cross matching uses the positional coordinates in both the catalogues to find the objects. There is some positional error in all catalogues so we calculate the distance between them and if the distance between the objects is within a certain search radius, we consider the object a match.

However, we have to take into account the spherical nature of the celestial sphere on which these coordinates are positioned and we need to calculate the great circle distance or angular distance between the two objects.

$$d = 2arcsin\sqrt{sin^2\frac{|\delta_1 - \delta_2|}{2} + cos\delta_1 cos\delta_2 \sin^2\frac{|\alpha_1 - \alpha_2|}{2}}$$

Where α_1, α_2 are Right Ascension of object in catalogue 1 and catalogue 2 respectively

δ_1, δ_2 are Declination of object in catalogue 1 and catalogue 2 respectively

d is angular distance

However, this method is essentially designed for cross-matching between point sources. It is good for merging optical catalogues of stars and small extended sources but mostly useless for associating the largest galaxies or radio sources. For example, a single radio galaxy can have several distinct components corresponding to its core and lobes. Traditional methods can quickly link the radio core to the optical centre, but the jet or lobes would be ignored since they are far away from the centre. Currently such components are identified and associated by eye or machine learning algorithms.

There has been a lot of research in cross matching large catalogues to build all sky surveys which provide various properties of objects like the flux in different wavelengths, redshifts, magnitude etc.

The *'WISE × SuperCOSMOS PHOTOMETRIC REDSHIFT CATALOG: 20 MILLION GALAXIES OVER 3π STERADIANS'* provides information about the cross matching of the SuperCOSMOS and WISE catalogues.

In the paper *'Matching radio catalogues with realistic geometry: application to SWIRE and ATLAS (Dongwei Fan, Tamás Budavári, Ray P. Norris, Andrew M. Hopkins)'* cross matches SWIRE and ATLAS catalogues using probabilistic cross identification.

'SOURCE MATCHING IN THE SDSS AND RASS: WHICH GALAXIES ARE REAL John K. Parejko, Anca Constantin, Michael S. Vogeley' demonstrates cross matching of SDSS and RSS to identify galaxies emitting X rays.

Statement of the Problem:

Designing a program in Python to perform efficient cross-matching among different catalogues from astronomical surveys.

Details of Method of Study:

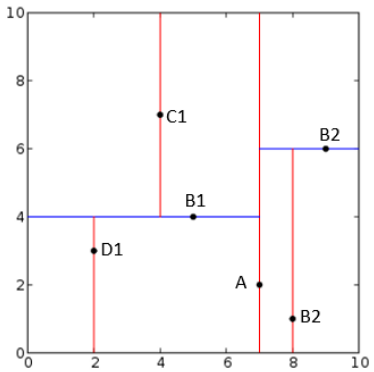
For crossmatching our two catalogues, we need to search the second catalogue to find a counterpart for each object in the first catalogue. To do this, we usually search within a given radius, based on the uncertainties in the position.

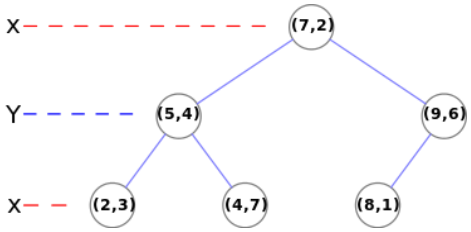
1. **Brute force algorithm:**
   i. For each source_A in the first catalogue, we look through each source_B in the second catalogue.
   ii. We calculate the angular distance between each pair of sources.
   iii. If the angular distance is less than our search radius, and if that offset, is the lowest we've seen so far, for source_A, we consider that pair a match.

When we run it on two catalogues, we end up with a list of matches between galaxies in the first catalogue, galaxies in the second catalogue and the great circles offset between them. The time complexity of this algorithm is $O(n^2)$ which is very poor.
To increase the time complexity, we need a better algorithm. We will use the KD tree algorithm as it has time complexity of O(nlogn).

2. **KD Tree Algorithm:**

Step 1: construct the KD tree



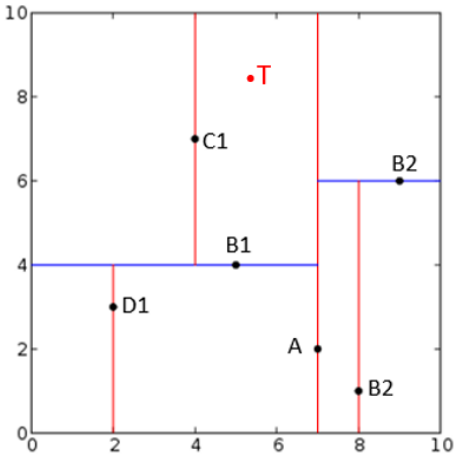


We consider all our points in a cartesian space. We first find a median in the x direction. That point becomes the root of the tree(say A). and we split the plane at that point(vertical line). We then consider points to the left of A and split the plane in the y dimension. And again, at the median point, which is B. We repeat this process, alternating between the x and y dimensions, until the left-hand side of the tree is complete. Finally, the process is repeated for the data to the other side of A until every data point in our original dataset is either a node or a leaf in the tree. Now we have our KD tree

Step 2:

Now that our KD tree is completed, we can do our fast neighbour search.

Searching for a nearest neighbour in a k-d tree proceeds as follows:



1. Starting with the root node , the algorithm moves down the tree recursively, in the same way that it would if the search point were being inserted (i.e. it goes left or right depending on whether the point is lesser than or greater than the current node in the split dimension).
2. Once the algorithm reaches a leaf node, it checks that node point and if the distance is better, that node point is saved as the "current best".
3. The algorithm unwinds the recursion of the tree, performing the following steps at each node:
    1. If the current node is closer than the current best, then it becomes the current best.
    2. The algorithm checks whether there could be any points on the other side of the splitting plane that are closer to the search point than the current best.
4. When the algorithm finishes this process for the root node, then the search is complete.

Step 3:

We set a maximum distance radius(search radius) and consider only those best distances as a count which is less than the maximum distance.

Data:

AT20G catalogue: The Australia Telescope 20 GHz (AT20G) Survey
SuperCOSMOS: SuperCOSMOS Science Archive
(SCOS_XSC_mCl1_B21.5_R20_noStepWedges.csv.gz)
SDSS: Sky Server
SQL Query:

```
-- This query does a table JOIN between the imaging (PhotoObj) and spectra
--(SpecObj) tables and includes the necessary columns in the SELECT to upload
--the results to the SAS(Science Archive Server) for FITS file retrieval.
SELECT TOP 500000
p.objid,p.ra,p.dec,p.u,p.g,p.r,p.i,p.z,
p.run, p.rerun, p.camcol, p.field,
s.specobjid, s.class, s.z as redshift,
s.plate, s.mjd, s.fiberid
FROM PhotoObj AS p
JOIN SpecObj AS s ON s.bestobjid = p.objid
WHERE
  class='GALAXY'
  AND s.z BETWEEN 0 AND 2
  AND p.u BETWEEN 0 AND 19.6
  AND g BETWEEN 0 AND 20
```

Results and Discussions:

1. Demonstrating Time Complexity:
   Cross-Matching a subset of AT20G catalogue(BSS) with subset of SUPERCOSMOS all sky galaxy catalogue(500):
   Search radius=1 arc second.
   Time taken-
   - Brute force algorithm: 1.3846648979997553s
   - KD tree Algorithm: 0.039312627999606775 s

   The KD tree algorithm is much faster than the brute force algorithm. However, for small datasets the difference is not appreciable.

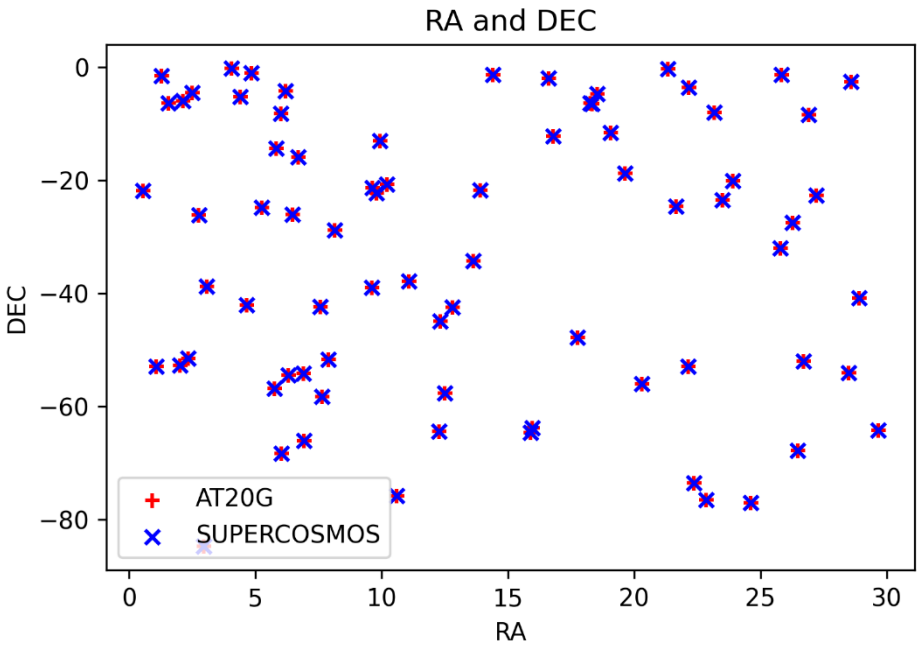
   Cross-Matching entire AT20G catalogue with 5000 objects from SUPERCOSMOS all sky galaxy catalogue:
   Search radius=1 arc second.
   Time taken-
   - Brute force algorithm: 504.07104657699983 s
   - KD tree algorithm: 2.287129633999939 s

   For large datasets the difference becomes very significant.

   For the brute force algorithm when we increase the data points by a factor of n, the time is scaled up by a factor of n^2(i.e., Time Complexity is $O(n^2)$)Hence for large datasets this algorithm is not suitable.

   For KD tree algorithm Time Complexity is $O(nlog(n))$ and hence it is much more efficient at handling large amount of data.

2. Cross-Matching AT20G catalogue with SUPERCOSMOS All-Sky galaxy survey:
   We cross matched the entire AT20G catalogue with 10000000 objects from SUPERCOSMOS catalogue.
   Search radius=1 arc second.
   Time taken for KD tree algorithm: 32.354639793997194 s



Sample:

Match: 101

First 10 values from the cross-matched catalogue

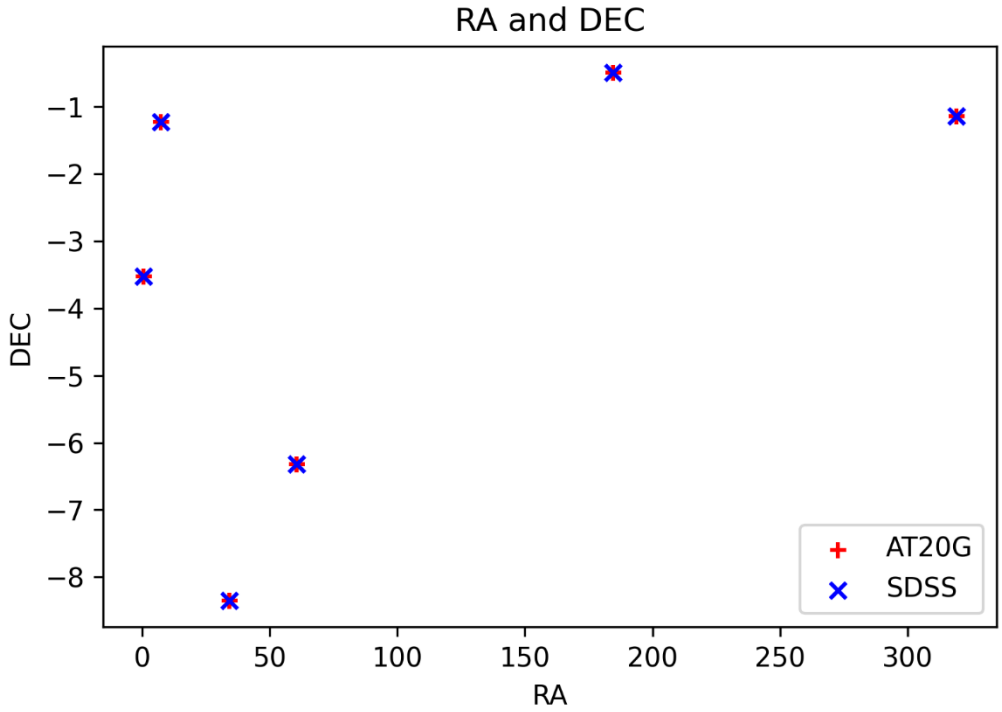
index	ra1	Dec-01	ra2	Dec-02	flux	Bmag
1	37.66433	-85.4719	37.664	-85.4722	19.213	113
2	2.941126	-84.7222	2.94283	-84.7221	19.154	274
3	56.70818	-77.2435	56.70854	-77.2437	19.847	48
4	24.61511	-77.0658	24.61554	-77.066	21.254	147
5	47.98024	-76.8641	47.98054	-76.8642	17.635	1238
6	22.83617	-76.5409	22.83675	-76.5409	19.605	126
7	10.59231	-75.8106	10.59208	-75.8108	19.633	51
8	50.21714	-75.0106	50.21717	-75.0104	16.757	52
9	58.07025	-73.9503	58.07008	-73.9505	19.604	75
10	22.37562	-73.553	22.37529	-73.5531	18.33	67
11	36.76836	-72.2998	36.769	-72.2998	19.453	44

The entire cross-matched catalogue can be accessed here: Click Here

3. Cross-Matching AT20G catalogue with SDSS (DR17):
   We cross matched entire AT20G catalogue with SDSS catalogue data release 17.

   Search Radius: 1 arc second

   Time Taken for KD tree algorithm: 8.418119009998918 s



Cross-Matched Points:

Match: 6

index	ra1	Dec-01	ra2	Dec-02	flux20Hz	u	g	r	i	z
1	34.26112	-8.34784	34.26125	-8.34778	415	19.52138	18.90502	18.34309	17.96175	17.66989
2	60.57704	-6.32026	60.57721	-6.32022	99	19.2386	17.36408	16.30307	15.70507	15.29092
3	0.627612	-3.52791	0.6275	-3.52781	53	18.61899	18.18854	18.09792	18.2257	18.07231
4	7.254108	-1.22827	7.25421	-1.22825	207	17.50779	15.65948	14.69832	14.23905	13.93871
5	319.0135	-1.14123	319.0135	-1.14133	93	18.87722	18.36756	17.88198	17.58896	17.31564
6	184.4947	-0.4962	184.4948	-0.49594	404	18.63554	18.1183	17.67046	17.37298	17.12646

Limitations:

1. With advancement in technology and science the amount of data collected keeps on increasing. Hence even the most efficient algorithms take a lot of time and memory to process. It becomes necessary to come up with faster algorithms.
2. Further criteria can be introduced to improve the confidence level of the matches like comparison of redshifts, flux etc. This project does not consider any other criteria and relies only on positional cross matching. Hence is it relatively less accurate.

Relevance of the Study:

1. Objects which radiate at energies multiple wavelengths will be detected in multiple surveys, for e.g.: information about the different parts of the AGN can be understood by looking at different parts of the wavelength. E.g. the radiation from the Accretion Disc is brightest at optical and ultraviolet wavelengths. While infrared radiation comes from the dusty regions that surround the black hole. The jets produce a lot of radio emissions and hot gas surrounding the central black hole region can produce x-ray emissions. To fully understand the physics of the object we need to piece all these information together.

2. Crossmatching can also be used to segregate required objects from a catalogue. E.g if we need objects from a catalogue which has high flux density in radio region we can cross match that catalogue with a catalogue like AT20G to get our desired results.

Bibliography:

1. Line, J., Webster, R., Pindor, B., Mitchell, D., & Trott, C. (2017). PUMA: The Positional Update and Matching Algorithm. Publications of the Astronomical Society of Australia, 34, E003. doi:10.1017/pasa.2016.58
2. https://www.rohithkrishna.in/post/2020-08-11-cross-matching/
3. https://www.aanda.org/articles/aa/full_html/2017/11/aa30965-17/aa30965-17.html
4. https://www.atnf.csiro.au/research/AT20G/
5. http://ssa.roe.ac.uk/allSky.html
6. Murphy, T., “The Australia Telescope 20 GHz Survey: the source catalogue”, Monthly Notices of the Royal Astronomical Society, vol. 402, no. 4, pp. 2403–2423, 2010. doi:10.1111/j.1365-2966.2009.15961.x.

CODE Link