**A CENTRALITY INDEX FOR SAMPLE POINTS**

In the analysis of samples, the location of modes is a difficult problem. Similar difficulties are found in density estimation and in disentangling Gaussian mixtures. The method I’m here presenting is drawn from a mechanical interpretation of sample points as material ‘points in a gravitational field. From the Newton’s gravitational theory, you learn that, if you disregard the attraction of points external to a points’ cloud, at the center of the cloud the gravitational field is null. Therefore, you are at the bottom of a potential well, like it happens for the Sun that attracts the other celestial bodies but has zero field at its center when one disregards the attractions from celestial bodies out of the Solar System. Thus, if you are able to create a potential field that wraps all sample points, you can try to isolate the potential wells and use them to:

1. Locate the modes in the sample
2. Find the initial seeds for a Cluster Analysis.
3. Isolate the normal components in a Gaussian mixture

A more detailed presentation of this approach is in the attachment. In this post I will use Python to apply the method to the Fisher Iris problem. I assume that Iris data are known to the reader. In case, Fisher Iris data are available from many public sources.

The work flow is the following

1. Read and normalize data
2. Create a potential field that wraps the data
3. Isolate the field’s potential wells
4. Read and normalize data

*import numpy as np*

*import pandas as pd*

*dataraw = pd.read\_excel("C:\Pyth\iris.xlsx") #specify your data*

*#standardize data --- dataraw is a DataFrame*

*#locate data in the DataFrame*

*datar = dataraw.iloc[:,1:5] #the field 0 is the point’s ID*

*means = datar.mean(axis = 0)*

*stdev = datar.std(axis = 0)*

*data = (datar-means)/stdev*

1. Create a potential field. You start with the pairwise distance between points.

*scalar = pd.merge(data, data, how = 'cross')*

*point1 = scalar.loc[:, 'sepal length \_x':'petal width \_x']*

*point2 = scalar.loc[:, 'sepal length \_y':'petal width \_y']*

*apoint1 = point1.to\_numpy(dtype = float)*

*apoint2 = point2.to\_numpy(dtype = float)*

*delta = (apoint1 - apoint2)*

The scalar product of *data* with itself is done with the *pandas merge* method using the *how = ‘cross’* specification. Unlike SQL, that always creates the scalar product unless you specify a *where* clause, *pandas merge* needs the *how* = parameter to differentiate a scalar product from *inner* or *outer* joins. Python automatically adds a trailer *\_x* and *\_y* to the variable names found in *data* to distinguish the two points in the record generated by the *merge* method. You can therefore extract *point1* and *point2* from *scalar.* The two DataFrames keep the data of the first point and of the second one in each couple found in *scalar*. Separating the two points makes it easier to find the differences of the 4 variables in each couple of points because you can run computations by block on the 2 ndarrays *apoint1* and *apoint2* you get with *point1.to\_numpy()* and *point2.to\_numpy()*.

Now you compute the pairwise attraction force. Remember that the force is a vector that always points to the point you are considering. You state the pointing direction with the function *sig()*, the signum of the distance *delta*. My suggestion is to use an exponential force, like *exp(-abs(delta))*. I strongly suggest not to use any *delta inverse* due to the risk of too strong forces when by chance two points are too near.

*force = 0*

*if delta.any() != 0:*

*force = np.exp(-abs(delta))*

*sig = np.sign(delta)*

*sforce = sig\*force*

*dsforce = pd.DataFrame(sforce)*

Now, for each point you need to sum all forces converging to it. The following code does the trick by variable. Variables are the force components. The array *sforce* is taken in chunks of 150 rows that bear the force’s components that point to a given point from all the others. Force’s components are thus summed up by chunk.

*arr = np.ones((150, 1),)*

*sforcet = sforce.T*

*sum\_force = np.zeros((1, 4),) #do not use empty arrays*

*start = 0*

*end = 150*

*for i in range(150):*

*s\_forcet = sforcet[:, start:end]*

*work = np.matmul(s\_forcet, arr)*

*sum\_force = np.concatenate((sum\_force, work.reshape(1, 4)), axis = 0)*

*start = end*

*end +=150*

*sumforce = sum\_force[1:, :] #omit the starting block*

*dsumforce = pd.DataFrame(sumforce)*

*dsumforce.to\_excel('C:\Pyth\sumforce\_sqc.xlsx')*

The array *arr* is a service array you need to sum through the chunk of 150 rows. As you can see, I don’t use any empty arrays to initialize the concatenation. I found that it doesn’t always work properly. So I decided to start with a true array I’ll later remove with *sumforce = sum\_force[1:, :]* that starts from the second row of the array *sum\_force*. The array keeps 4 numbers: the sums of the 4 force components along the 4 axes given by the 4 original variables. The norm of the vector having the 4 components is easily computed as follows

*sum\_force\_square = sumforce\*\*2*

*ssT = np.ones((4, 1),)*

*T\_w\_ = np.sqrt(np.matmul(sum\_force\_square, ssT))*

1. Isolate the field’s potential wells

Potential wells are where |*T\_w\_*| is at a local minimum. Because in the use of *matplotlib* I’m still a beginner, after exporting |*T\_w\_*| to EXCEL I got the following line plot. Data are ordered by the row number of the original data. So that the first 50 points correspond to *Setosa*, then 50 *Versicolor* follow, and last 50 *Virginica* close the sequence.

The first minimum is at point 16 with coordinates (5.7, 4.4, 1.5, 0.4). It is a *Setosa*. The second minimum is at the point 67 with coordinates (5.6, 3.0, 4.5, 1.5). It is a *Versicolor*. The third minimum is at point 92 with coordinates (6.1, 3.0, 4.6, 1.4). It is still a *Versicolor*. Now we find the point 128 with coordinates (6.1, 3.0, 4.9, 1.8). It is a *Virginica*. The point 139 follows, a *Virginica* too, with coordinates (6.4, 3.1, 5.5, 1.8). We have 5 points as candidates for seeds in a Clustering procedure. However, we have to choose those that are not too near to some other ones. They would belong to the same potential well. We can use different strategies here. A hierarchical clustering with the Ward distance would agglomerate the points 128 with 139 and the point 67 with the point 92. You could also choose one in the two couples to be agglomerated. Last you could start with 5 clusters and then agglomerate the final clusters.

Any cluster method would easily find the three clusters with a good correspondence with the true species.

Good seeds are a strong winning factor. In fact, iterative clustering methods suffer too much from the initial seeds due to the sequential process implicit in the many of them.

To compare the points in the potential well with the data averages by species, first I took the average of the two couples of too near points. They stay in the same potential well, though this doesn’t appear in the graph due to the points being ordered as in the raw data table. I got:

Couple (67, 92) average = (5.85, 3.00, 4.555, 1.45)

Couple (128, 139) average = (6.25, 3.05, 5.20, 1,80)

The point 16 (*Setosa*) is well separated from the other ones. So, I take it as it is. In the table, *PW()* means points from the potential wells and *Avg()* means sample average by species.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sepal length | Sepal width | Petal length | Petal width |
| Avg(Setosa) | 5.006 | 3.428 | 1.462 | 0.246 |
| PW(Setosa) | 5.700 | 4.400 | 1.500 | 0.400 |
| Avg(Versicolor) | 5.936 | 2.770 | 4.260 | 1.326 |
| PW(versicolor) | 5.850 | 3.000 | 4.555 | 1.450 |
| Avg(Virginica) | 6.588 | 2.974 | 5.552 | 2.026 |
| PW(Virginica) | 6.250 | 3.050 | 5.200 | 1.800 |

**APPENDIX: partial reprint of a previous post**

**Locating Modes in Samples by a Weighted Laplace Convolution**

**Abstract**

In the case of continuous random variables, a straight application of the theorem of Fermat is enough to locate modes when they are located inside of the variables’ range. The zeros of their Densities’ derivatives are in fact the points where they attain their maximum, minimum or inflection points. In order to decide between the three cases, a check on the second derivative is needed. Unfortunately, derivatives are of no help in the search of modes in a sample. Discrete by its very nature, a sample makes derivatives meaningless. However, keeping in mind that by definition modes are local Density’s maximums, one can completely overhaul the derivative-based approach and exploit the expectation that sample points would cluster around the sample’s modes. This paper shows how this approach can be implemented through a Gravitational model that endows each sample point with a “centrality” score in its surrounding topological neighbourhood. Thus, instead of a direct search of modes, a two-steps method is suggested: (1) searching points that bear a good centrality score and then (2) checking the local density in their neighbourhoods.

Keywords: mode; clustering; mixture decomposition; gravitation; deep learning.

1. **How the paper is organized**

After an introduction, the theory underpinning the method is presented. Then, the case of the Logistic distribution is addressed. It enjoys a Cumulative in closed form that under the gravitational law between the couple of points allows a closed form representation of the corresponding gravitational field . The zeros of locate the mode when it is inside of the variable’s range. The value of the method lies in the fact that in a sample the gravitational field is easy to compute. Details needed to demonstrate the statements can be found in the Appendix.

1. **Introduction**

Due to the randomness of data patterns in a sample, even after a preliminary careful smoothing of sample’s data, the smartest attempt to locate modes may end in a failure. In Author’s opinion the simple approach here sketched could attract some attention from readers. The core of the method stays in the study of the gravitational field generated by a negative exponential gravitational law of points’ inter-distance that corresponds to a weighted convolution of the density with the Laplace distribution. Because the convolution acts on the Density of continuous random variables and because Densities are absolutely summable over the whole real line, any convergence concern disappears. The method can work well when the data body is a sample from a continuous distribution and the mode is an internal point in the variable’s range. In this paper, therefore, mode means “a central point in a dense neighbourhood”. This definition implies that the method can be viewed as a new variant of Clustering algorithms.

1. **The gravitational model**

The gravitational field represents the accumulation on each sample point of the information about where the other points are. The value of the method is in its ability to provide each sample point with a “centrality” score in a suitably narrow surrounding neighbourhood. Borrowing from other fields of science, it is possible to endow each sample point with a smooth function of the pairwise point inter-distances from all the other points. We know that, by the notion of mutual attraction between couples of points, the gravitational Newton’s model does exactly this. Here the Newton model is taken only as a mental paradigm, without pretending the applicability to Statistics of the gravitational hypothesis as such. Therefore, we won’t use the Newton law that, among others, would lead to singular points, the infamous black holes. Narrowly circulated attempts by the Author with a plain application of the Newton’s law show that the gravitational force between some couples of sample points may be too big thus remarkably hindering the search. Mechanical paradigms are not new in Statistics (Capra et al. 1970). The simplest case is the interpretation of the mean as the barycentre of the data. All sample points, bearing the information of their position in the variable’s range, concur to the estimation of the sample mean. In addition, it is known that some multivariate models plunge their roots in mechanics. Principal Components Analysis is the straight replication of the theory of the axis of permanent rotation of a rigid body. In fact, the variance is the statistical companion of the moment of inertia that measures the resistance of a rigid body to rotation. Coming to samples, the sample variance is the resistance to rotation of a cloud of points keeping their inter-distances invariant during rotation. Viewing a data body as a collection of asteroids is thus certainly not new.

In this paper, keeping the basic characteristics of the Newton approach, we use a suitable law of attraction, both mathematically tractable and free from singularities. Namely:

1. for each couple of points in the sample, an attraction force that declines with the distance , is defined,
2. each point is endowed with the sum  of all the attraction forces emanating from all the other points in the sample

Often, we will refer to the ’s intensity as . A central point in a neighbourhood enjoys a local minimum of the total attraction’s intensity because the nearest surrounding points, pivotal in the sum of forces acting on that point, balance their contribution making to decrease to a local minimum. Central points in a neighbourhood lay therefore at the bottom of a potential well. The points where the intensity of the sum of the attraction forces has a local minimum are thus candidates as local modes in the sample. They can be taken as modes only after checking that the local density attains a local maximum there. In fact, the attraction forces can reach a balance because of the mutual disposition of points in the sample without being central in any dense neighbourhood. They wouldn’t be candidate as modes, however until directly checked. The centrality effect here described can be theoretically verified when the cumulative Distribution can be written in closed form.

From what has been said, the attraction force between the two points is

where

is the sign of the difference between the two points and . Taking as the point under consideration, all forces are always directed from to . So, they are positive when and negative when . This entails that the sum of forces is the integral in of (1)

is the mass of the point , actually the local Density at , actually its likelihood

is the mass of the point , actually the local density at , actually its likelihood

is the law of the force intensity’s decline with

Given the point , the total force acting on due to all the other points is given by

Twointegrals are needed because the forces change their direction at the point Please note that, being the integral in , the density is a constant, so that it can be moved out from the integral symbol.

All symmetric distributions obviously have = mean. In fact, taking without loss of generality , the force’s intensity that links the couple ((, ) is equal to the force intensity that links the two points  because is fixed, and also is the same. Thus, forces’ intensities are also the same with opposite signs, however. When the density is symmetric, the function is thus antisymmetric with . Summing over all points symmetrically positioned with respect to the mean , one gets the sum of 0. You conclude that, in the case of a unimodal symmetric distribution, the root of is also the single mode of the distribution. In fact, happens when is central to a dense neighbourhood. starts negative and ends positive so that admits at least 1 root in the range. The root is actually only one. In fact, both and are positive. For a generic and similarly . Therefore, and Buy the Ulysses Dini theorem, crosses the x-axis. The point of crossing is due to the symmetry between and

1. **The Logistic distribution**

Unfortunately, can’t be put in closed form for the normal Distribution. Integrating by parts, the Cumulative will show up at some point. Though admits approximations, they are of no help in the search of the roots of . A suitable approach is to leverage Distributions having a Cumulative in closed form. The Logistic distribution can work well with the distances .

The Logistic Density and the Logistic Cumulative distributions are written as

For each and the distance you have

For the total force acting on the point you have

A detailed computation for the Logistic distribution follows. Three basic integrals underpin the key steps. The first one is a definite integral spanning from to and linking the Logistic density to its companion Cumulative. The second and the third ones are indefinite integrals you can retrace in the mathematical literature (ex. Weast et al., 1964). Omitting the arbitrary constant you have

Also, the substitution is used when needed. The total force is written as

Separately dealing with A and B, you get

Putting all together

is antisymmetric in. The factor outside the major parenthesis is the density at , symmetric for a logistic distribution. Easy to prove, anyway. Substituting with you come back to the same form with inverted sign. You easily check that

And also

is obviously antisymmetric. Therefore, the part within the major parenthesis is antisymmetric. A continuous antisymmetric function admits at least one root . In addition, because 0 is a root of , you should have . This is easily checked. Both take the value at 0.

**Bibliography**

[1] Capra, R., Lena, S., Santarelli, U., Vescovi, P. (1976). “Cluster Analysis by Moment of Inertia Method”, IBM technical disclosure Bulletin, Vol. 18 No. 8.

[2] Weast R., M. Samuel (1964). “Standard Mathematical Tables”, The Chemical Rubber Co.

[3] Guojun G., M. Chaoqun, W. Jianhong (2007). “Data Clustering, Theory, Algorithms and Applications, ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA.

[4] Hartigan, J. (1975). “Clustering Algorithms”, Toronto, John Wiley & Sons.